FLP-80DOS
FLEXIBLE DISK
OPERATING SYSTEM V2.1
FLP-80DOS Operations Manual

VERSION 2.1
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>SECTION NUMBER</th>
<th>PARAGRAPH NUMBER</th>
<th>TITLE</th>
<th>PAGE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-1</td>
<td>FLP-80DOS GENERAL DESCRIPTION</td>
<td>1-1</td>
</tr>
<tr>
<td></td>
<td>1-4</td>
<td>INTRODUCTION</td>
<td>1-4</td>
</tr>
<tr>
<td></td>
<td>1-5</td>
<td>MONITOR</td>
<td>1-5</td>
</tr>
<tr>
<td></td>
<td>1-6</td>
<td>DESIGNER'S DEVELOPMENT</td>
<td>1-6</td>
</tr>
<tr>
<td></td>
<td>1-7</td>
<td>TOOL-DDT</td>
<td>1-7</td>
</tr>
<tr>
<td></td>
<td>1-8</td>
<td>TEXT EDITOR-EDIT</td>
<td>1-8</td>
</tr>
<tr>
<td></td>
<td>1-9</td>
<td>Z80 ASSEMBLER-ASM</td>
<td>1-9</td>
</tr>
<tr>
<td></td>
<td>1-10</td>
<td>LINKER-LINK</td>
<td>1-10</td>
</tr>
<tr>
<td></td>
<td>1-13</td>
<td>PERIPHERAL INTERCHANGE</td>
<td>1-13</td>
</tr>
<tr>
<td></td>
<td>1-15</td>
<td>PROGRAM-PIP</td>
<td>1-15</td>
</tr>
<tr>
<td></td>
<td>1-16</td>
<td>I/O SYSTEM</td>
<td>1-16</td>
</tr>
<tr>
<td></td>
<td>1-17</td>
<td>OTHER PROGRAMS</td>
<td>1-17</td>
</tr>
<tr>
<td></td>
<td>1-18</td>
<td>REFERENCE DOCUMENTS</td>
<td>1-18</td>
</tr>
<tr>
<td></td>
<td>1-19</td>
<td>DEFINITION OF SYMBOLS USED</td>
<td>1-19</td>
</tr>
<tr>
<td></td>
<td>1-20</td>
<td>IN THIS MANUAL</td>
<td>1-20</td>
</tr>
<tr>
<td></td>
<td>1-21</td>
<td>CONSOLE INTERACTION</td>
<td>1-21</td>
</tr>
<tr>
<td></td>
<td>1-22</td>
<td>ENTERING DATA ON THE</td>
<td>1-22</td>
</tr>
<tr>
<td></td>
<td>1-23</td>
<td>CONSOLE ESCAPE</td>
<td>1-23</td>
</tr>
<tr>
<td></td>
<td>1-24</td>
<td>CONCEPT OF DATASET</td>
<td>1-24</td>
</tr>
<tr>
<td></td>
<td>1-25</td>
<td>CONCEPT OF LOGICAL UNIT</td>
<td>1-25</td>
</tr>
<tr>
<td></td>
<td>1-26</td>
<td>NUMBERS</td>
<td>1-26</td>
</tr>
<tr>
<td></td>
<td>1-27</td>
<td>DATE FEATURE</td>
<td>1-27</td>
</tr>
<tr>
<td></td>
<td>1-28</td>
<td>FLEXIBLE DISK HANDLING</td>
<td>1-28</td>
</tr>
<tr>
<td></td>
<td>1-29</td>
<td>PROCEDURE</td>
<td>1-29</td>
</tr>
<tr>
<td></td>
<td>1-30</td>
<td>START UP PROCEDURES</td>
<td>1-30</td>
</tr>
<tr>
<td></td>
<td>1-31</td>
<td>MEMORY AND I/O SUMMARY</td>
<td>1-31</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS cont.

<table>
<thead>
<tr>
<th>SECTION NUMBER</th>
<th>PARAGRAPH NUMBER</th>
<th>TITLE</th>
<th>PAGE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-42</td>
<td>MEMORY MAP</td>
<td>1-21</td>
<td></td>
</tr>
<tr>
<td>1-44</td>
<td>PORT MAP</td>
<td>1-23</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>MONITOR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-1</td>
<td>INTRODUCTION</td>
<td>2-1</td>
<td></td>
</tr>
<tr>
<td>2-3</td>
<td>OPERATIONS SUMMARY</td>
<td>2-1</td>
<td></td>
</tr>
<tr>
<td>2-4</td>
<td>SYSTEM RESET</td>
<td>2-1</td>
<td></td>
</tr>
<tr>
<td>2-6</td>
<td>POWER UP SEQUENCE</td>
<td>2-2</td>
<td></td>
</tr>
<tr>
<td>2-7</td>
<td>MONITOR COMMAND SUMMARY</td>
<td>2-2</td>
<td></td>
</tr>
<tr>
<td>2-9</td>
<td>IMPLIED RUN COMMAND</td>
<td>2-3</td>
<td></td>
</tr>
<tr>
<td>2-12</td>
<td>COMMAND ENTRY</td>
<td>2-4</td>
<td></td>
</tr>
<tr>
<td>2-13</td>
<td>DEFINITIONS</td>
<td>2-4</td>
<td></td>
</tr>
<tr>
<td>2-14</td>
<td>ASSIGN COMMAND</td>
<td>2-5</td>
<td></td>
</tr>
<tr>
<td>2-17</td>
<td>BEGIN COMMAND</td>
<td>2-6</td>
<td></td>
</tr>
<tr>
<td>2-20</td>
<td>CLEAR COMMAND</td>
<td>2-6</td>
<td></td>
</tr>
<tr>
<td>2-23</td>
<td>DDT COMMAND</td>
<td>2-7</td>
<td></td>
</tr>
<tr>
<td>2-26</td>
<td>DTABLE COMMAND</td>
<td>2-7</td>
<td></td>
</tr>
<tr>
<td>2-29</td>
<td>DUMP COMMAND</td>
<td>2-8</td>
<td></td>
</tr>
<tr>
<td>2-32</td>
<td>GET COMMAND</td>
<td>2-8</td>
<td></td>
</tr>
<tr>
<td>2-35</td>
<td>INIT COMMAND</td>
<td>2-9</td>
<td></td>
</tr>
<tr>
<td>2-38</td>
<td>RTABLE COMMAND</td>
<td>2-9</td>
<td></td>
</tr>
<tr>
<td>2-41</td>
<td>SAVE COMMAND</td>
<td>2-10</td>
<td></td>
</tr>
<tr>
<td>2-45</td>
<td>CONSOLE ESCAPE</td>
<td>2-10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>PERIPHERAL INTERCHANGE PROGRAM (PIP)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-1</td>
<td>INTRODUCTION</td>
<td>3-1</td>
<td></td>
</tr>
<tr>
<td>3-2</td>
<td>ENTERING PIP</td>
<td>3-1</td>
<td></td>
</tr>
</tbody>
</table>
### TABLE OF CONTENTS cont

<table>
<thead>
<tr>
<th>SECTION NUMBER</th>
<th>PARAGRAPH NUMBER</th>
<th>TITLE</th>
<th>PAGE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-5</td>
<td></td>
<td>PIP COMMAND SYNTAX</td>
<td>3-1</td>
</tr>
<tr>
<td>3-9</td>
<td></td>
<td>APPEND COMMAND</td>
<td>3-2</td>
</tr>
<tr>
<td>3-12</td>
<td></td>
<td>COPY COMMAND</td>
<td>3-3</td>
</tr>
<tr>
<td>3-18</td>
<td></td>
<td>DATE COMMAND</td>
<td>3-5</td>
</tr>
<tr>
<td>3-19</td>
<td></td>
<td>DIRECT COMMAND</td>
<td>3-5</td>
</tr>
<tr>
<td>3-23</td>
<td></td>
<td>ERASE COMMAND</td>
<td>3-7</td>
</tr>
<tr>
<td>3-27</td>
<td></td>
<td>FORMAT COMMAND</td>
<td>3-8</td>
</tr>
<tr>
<td>3-34</td>
<td></td>
<td>INIT COMMAND</td>
<td>3-10</td>
</tr>
<tr>
<td>3-37</td>
<td></td>
<td>RENAME COMMAND</td>
<td>3-10</td>
</tr>
<tr>
<td>3-41</td>
<td></td>
<td>STATUS COMMAND</td>
<td>3-11</td>
</tr>
<tr>
<td>3-44</td>
<td></td>
<td>QUIT COMMAND</td>
<td>3-12</td>
</tr>
</tbody>
</table>

### 4 FLP-80DOS TEXT EDITOR (EDIT)

<table>
<thead>
<tr>
<th>PARAGRAPH NUMBER</th>
<th>TITLE</th>
<th>PAGE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-1</td>
<td>INTRODUCTION</td>
<td>4-1</td>
</tr>
<tr>
<td>4-3</td>
<td>CAPABILITIES</td>
<td>4-1</td>
</tr>
<tr>
<td>4-5</td>
<td>SOFTWARE CONFIGURATION</td>
<td>4-1</td>
</tr>
<tr>
<td>4-8</td>
<td>DEFINITIONS</td>
<td>4-2</td>
</tr>
<tr>
<td>4-9</td>
<td>USING THE TEXT EDITOR-</td>
<td>4-5</td>
</tr>
<tr>
<td></td>
<td>CONSOLE INTERACTION</td>
<td></td>
</tr>
<tr>
<td>4-11</td>
<td>USING THE TEXT EDITOR-</td>
<td>4-6</td>
</tr>
<tr>
<td></td>
<td>ENTERING COMMANDS</td>
<td></td>
</tr>
<tr>
<td>4-13</td>
<td>USING THE TEXT EDITOR-FIRST</td>
<td>4-7</td>
</tr>
<tr>
<td></td>
<td>STEPS</td>
<td></td>
</tr>
<tr>
<td>4-15</td>
<td>USING THE TEXT EDITOR-</td>
<td>4-8</td>
</tr>
<tr>
<td></td>
<td>BASIC COMMANDS</td>
<td></td>
</tr>
<tr>
<td>4-16</td>
<td>I - INSERT</td>
<td>4-8</td>
</tr>
<tr>
<td>4-17</td>
<td>An - ADVANCE</td>
<td>4-9</td>
</tr>
<tr>
<td>SECTION NUMBER</td>
<td>PARAGRAPH NUMBER</td>
<td>TITLE</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------</td>
<td>-------</td>
</tr>
<tr>
<td>4-19</td>
<td></td>
<td>Bn - BACKUP</td>
</tr>
<tr>
<td>4-20</td>
<td></td>
<td>Dn - DELETE</td>
</tr>
<tr>
<td>4-21</td>
<td></td>
<td>Ln - GO TO RECORD NUMBER n</td>
</tr>
<tr>
<td>4-22</td>
<td></td>
<td>Vn - VERIFY</td>
</tr>
<tr>
<td>4-23</td>
<td></td>
<td>TEXT EDITOR ADVANCED COMMANDS</td>
</tr>
<tr>
<td>4-24</td>
<td></td>
<td>Cn /string1/string2/ - CHANGE</td>
</tr>
<tr>
<td>4-25</td>
<td></td>
<td>En - EXCHANGE</td>
</tr>
<tr>
<td>4-26</td>
<td></td>
<td>Fn - PRINT FLAG OPTION</td>
</tr>
<tr>
<td>4-27</td>
<td></td>
<td>G dataset - GET RECORDS FROM DATASET</td>
</tr>
<tr>
<td>4-28</td>
<td></td>
<td>Mn-MACRO</td>
</tr>
<tr>
<td>4-29</td>
<td></td>
<td>Pn - dataset - PUT N RECORDS TO DATASET</td>
</tr>
<tr>
<td>4-30</td>
<td></td>
<td>Sn /source image/ - SEARCH</td>
</tr>
<tr>
<td>4-31</td>
<td></td>
<td>T - INSERT AT TOP</td>
</tr>
<tr>
<td>4-32</td>
<td></td>
<td>Wn - WRITE</td>
</tr>
<tr>
<td>4-33</td>
<td></td>
<td>Xn - EXECUTE</td>
</tr>
<tr>
<td>4-34</td>
<td></td>
<td>EDITING LARGE FILES</td>
</tr>
<tr>
<td>4-36</td>
<td></td>
<td>EDITOR MESSAGES</td>
</tr>
<tr>
<td>4-41</td>
<td></td>
<td>SAMPLE EDITING SESSION</td>
</tr>
</tbody>
</table>

5

FLP-80DOS ASSEMBLER (ASM)

5-1 INTRODUCTION
<table>
<thead>
<tr>
<th>SECTION NUMBER</th>
<th>PARAGRAPH NUMBER</th>
<th>TITLE</th>
<th>PAGE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-5</td>
<td></td>
<td>DEFINITIONS</td>
<td>5-1</td>
</tr>
<tr>
<td>5-9</td>
<td></td>
<td>ASSEMBLY LANGUAGE SYNTAX</td>
<td>5-3</td>
</tr>
<tr>
<td>5-11</td>
<td></td>
<td>DELIMITERS</td>
<td>5-3</td>
</tr>
<tr>
<td>5-12</td>
<td></td>
<td>LABELS</td>
<td>5-3</td>
</tr>
<tr>
<td>5-13</td>
<td></td>
<td>OPCODES</td>
<td>5-6</td>
</tr>
<tr>
<td>5-14</td>
<td></td>
<td>PSEUDO-OPS</td>
<td>5-6</td>
</tr>
<tr>
<td>5-15</td>
<td></td>
<td>OPERAND</td>
<td>5-11</td>
</tr>
<tr>
<td>5-16</td>
<td></td>
<td>GENERIC OPERAND</td>
<td>5-11</td>
</tr>
<tr>
<td>5-20</td>
<td></td>
<td>COMMENTS</td>
<td>5-15</td>
</tr>
<tr>
<td>5-21</td>
<td></td>
<td>OBJECT OUTPUT</td>
<td>5-17</td>
</tr>
<tr>
<td>5-23</td>
<td></td>
<td>ASSEMBLY LISTING OUTPUT</td>
<td>5-17</td>
</tr>
<tr>
<td>5-25</td>
<td></td>
<td>ABSOLUTE MODULE RULES</td>
<td>5-17</td>
</tr>
<tr>
<td>5-27</td>
<td></td>
<td>RELOCATABLE MODULE RULES</td>
<td>5-18</td>
</tr>
<tr>
<td>5-33</td>
<td></td>
<td>GLOBAL SYMBOL HANDLING</td>
<td>5-19</td>
</tr>
<tr>
<td>5-36</td>
<td></td>
<td>GLOBAL SYMBOL BASIC RULES</td>
<td>5-21</td>
</tr>
<tr>
<td>5-37</td>
<td></td>
<td>GLOBAL SYMBOL ADVANCED RULES</td>
<td>5-22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>USE OF THE &quot;NAME&quot; PSEUDO-OP</td>
<td>5-23</td>
</tr>
<tr>
<td>5-40</td>
<td></td>
<td>USING THE ASSEMBLER</td>
<td>5-24</td>
</tr>
<tr>
<td>5-42</td>
<td></td>
<td>ASSEMBLER OPTIONS</td>
<td>5-25</td>
</tr>
<tr>
<td>5-44</td>
<td></td>
<td>ERROR MESSAGES</td>
<td>5-26</td>
</tr>
<tr>
<td>5-46</td>
<td></td>
<td>ADVANCED OPERATIONS</td>
<td>5-26</td>
</tr>
<tr>
<td>5-47</td>
<td></td>
<td>PASS 2 ONLY OPERATION</td>
<td>5-26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(SINGLE PASS OPERATION)</td>
<td></td>
</tr>
<tr>
<td>5-49</td>
<td></td>
<td>ASSEMBLING SEVERAL</td>
<td>5-27</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SOURCE MODULES TOGETHER</td>
<td></td>
</tr>
<tr>
<td>5-50</td>
<td></td>
<td>SAMPLE ASSEMBLY SESSION</td>
<td>5-27</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS cont

<table>
<thead>
<tr>
<th>SECTION NUMBER</th>
<th>PARAGRAPH NUMBER</th>
<th>TITLE</th>
<th>PAGE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>6-1</td>
<td>LINKER</td>
<td>6-1</td>
</tr>
<tr>
<td></td>
<td>6-3</td>
<td>LINKER COMMAND</td>
<td>6-1</td>
</tr>
<tr>
<td></td>
<td>6-8</td>
<td>A OPTION</td>
<td>6-2</td>
</tr>
<tr>
<td></td>
<td>6-9</td>
<td>C OPTION</td>
<td>6-3</td>
</tr>
<tr>
<td></td>
<td>6-10</td>
<td>L OPTION</td>
<td>6-3</td>
</tr>
<tr>
<td></td>
<td>6-11</td>
<td>S OPTION</td>
<td>6-4</td>
</tr>
<tr>
<td></td>
<td>6-13</td>
<td>LINKER OPERATION</td>
<td>6-4</td>
</tr>
<tr>
<td></td>
<td>6-15</td>
<td>LINKER RESTRICTIONS</td>
<td>6-5</td>
</tr>
<tr>
<td></td>
<td>6-17</td>
<td>EXAMPLES OF LINK COMMAND</td>
<td>6-6</td>
</tr>
<tr>
<td>7</td>
<td>7-1</td>
<td>DDT-80 DEBUG SYSTEM</td>
<td>7-1</td>
</tr>
<tr>
<td></td>
<td>7-3</td>
<td>INTRODUCTION</td>
<td>7-1</td>
</tr>
<tr>
<td></td>
<td>7-7</td>
<td>SOFTWARE CONFIGURATION</td>
<td>7-1</td>
</tr>
<tr>
<td></td>
<td>7-8</td>
<td>COMMAND SUMMARY</td>
<td>7-6</td>
</tr>
<tr>
<td></td>
<td>7-10</td>
<td>CONVENTIONS</td>
<td>7-6</td>
</tr>
<tr>
<td></td>
<td>7-13</td>
<td>PREPARATION</td>
<td>7-8</td>
</tr>
<tr>
<td></td>
<td>7-14</td>
<td>DESCRIPTION OF DDT COMMANDS</td>
<td>7-8</td>
</tr>
<tr>
<td></td>
<td>7-17</td>
<td>COMMAND FORMAT</td>
<td>7-8</td>
</tr>
<tr>
<td></td>
<td>7-27</td>
<td>OPERANDS</td>
<td>7-9</td>
</tr>
<tr>
<td></td>
<td>7-28</td>
<td>OPERAND EXAMPLES</td>
<td>7-10</td>
</tr>
<tr>
<td></td>
<td>7-30</td>
<td>COMMAND TERMINATORS</td>
<td>7-11</td>
</tr>
<tr>
<td></td>
<td>7-32</td>
<td>SPECIAL KEYS</td>
<td>7-11</td>
</tr>
<tr>
<td></td>
<td>7-34</td>
<td>ERRORS</td>
<td>7-12</td>
</tr>
<tr>
<td></td>
<td>7-42</td>
<td>B-BREAKPOINT COMMAND</td>
<td>7-13</td>
</tr>
<tr>
<td></td>
<td>7-45</td>
<td>C-COPY MEMORY BLOCK COMMAND</td>
<td>7-16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E-EXECUTE COMMAND</td>
<td>7-17</td>
</tr>
</tbody>
</table>
### TABLE OF CONTENTS cont

<table>
<thead>
<tr>
<th>SECTION NUMBER</th>
<th>PARAGRAPH NUMBER</th>
<th>TITLE</th>
<th>PAGE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>48</td>
<td>F-FILL MEMORY COMMAND</td>
<td>7-18</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>H-HEXADECIMAL ARITHMETIC</td>
<td>7-19</td>
</tr>
<tr>
<td>7</td>
<td>54</td>
<td>L-LOCATE 8-BIT PATTERN COMMAND</td>
<td>7-20</td>
</tr>
<tr>
<td>7</td>
<td>57</td>
<td>M-DISPLAY AND UPDATE MEMORY OR REGISTER COMMAND</td>
<td>7-21</td>
</tr>
<tr>
<td>7</td>
<td>68</td>
<td>M-TABULATE MEMORY COMMAND</td>
<td>7-24</td>
</tr>
<tr>
<td>7</td>
<td>71</td>
<td>O-SET OFFSET COMMAND</td>
<td>7-25</td>
</tr>
<tr>
<td>7</td>
<td>74</td>
<td>P-DISPLAY AND UPDATE PORTS COMMAND</td>
<td>7-26</td>
</tr>
<tr>
<td>7</td>
<td>77</td>
<td>Q-QUIT COMMAND</td>
<td>7-27</td>
</tr>
<tr>
<td>7</td>
<td>80</td>
<td>R-DISPLAY CPU REGISTERS COMMAND</td>
<td>7-28</td>
</tr>
<tr>
<td>7</td>
<td>84</td>
<td>V-VERIFY MEMORY COMMAND</td>
<td>7-30</td>
</tr>
<tr>
<td>7</td>
<td>87</td>
<td>W-WALK THROUGH A PROGRAM COMMAND</td>
<td>7-31</td>
</tr>
<tr>
<td>7</td>
<td>91</td>
<td>DEBUGGER ESCAPE (CNTL-C)</td>
<td>7-32</td>
</tr>
</tbody>
</table>

### PART 2

#### TECHNICAL INFORMATION

**RDCHR AND WRCHR SUBROUTINES**

<table>
<thead>
<tr>
<th>SECTION NUMBER</th>
<th>PARAGRAPH NUMBER</th>
<th>TITLE</th>
<th>PAGE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1</td>
<td>INTRODUCTION</td>
<td>8-1</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>RDCHR - READ ONE BYTE</td>
<td>8-1</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>CALLING SEQUENCE</td>
<td>8-1</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>ENTRY PARAMETERS</td>
<td>8-1</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>EXIT PARAMETERS</td>
<td>8-2</td>
</tr>
<tr>
<td>SECTION NUMBER</td>
<td>PARAGRAPH NUMBER</td>
<td>TITLE</td>
<td>PAGE NUMBER</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------</td>
<td>------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>8-7</td>
<td></td>
<td>OPERATION</td>
<td>8-2</td>
</tr>
<tr>
<td>8-9</td>
<td></td>
<td>WRCHR - WRITE ONE BYTE</td>
<td>8-2</td>
</tr>
<tr>
<td>8-10</td>
<td></td>
<td>CALLING SEQUENCE</td>
<td>8-2</td>
</tr>
<tr>
<td>8-11</td>
<td></td>
<td>ENTRY PARAMETERS</td>
<td>8-2</td>
</tr>
<tr>
<td>8-12</td>
<td></td>
<td>EXIT PARAMETERS</td>
<td>8-3</td>
</tr>
<tr>
<td>8-13</td>
<td></td>
<td>OPERATION</td>
<td>8-3</td>
</tr>
<tr>
<td>8-15</td>
<td></td>
<td>DDT OPERATION</td>
<td>8-3</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>INPUT/OUTPUT CONTROL SYSTEM (IOCS)</td>
<td></td>
</tr>
<tr>
<td>9-1</td>
<td></td>
<td>INTRODUCTION</td>
<td>9-1</td>
</tr>
<tr>
<td>9-3</td>
<td></td>
<td>VECTOR DEFINITION</td>
<td>9-1</td>
</tr>
<tr>
<td>9-6</td>
<td></td>
<td>LUNIT</td>
<td>9-4</td>
</tr>
<tr>
<td>9-9</td>
<td></td>
<td>DVCE</td>
<td>9-5</td>
</tr>
<tr>
<td>9-10</td>
<td></td>
<td>UNIT</td>
<td>9-6</td>
</tr>
<tr>
<td>9-11</td>
<td></td>
<td>FNAM</td>
<td>9-6</td>
</tr>
<tr>
<td>9-12</td>
<td></td>
<td>FEXT</td>
<td>9-6</td>
</tr>
<tr>
<td>9-13</td>
<td></td>
<td>VERS</td>
<td>9-7</td>
</tr>
<tr>
<td>9-14</td>
<td></td>
<td>USER</td>
<td>9-7</td>
</tr>
<tr>
<td>9-15</td>
<td></td>
<td>RQST</td>
<td>9-7</td>
</tr>
<tr>
<td>9-16</td>
<td></td>
<td>FMAT</td>
<td>9-8</td>
</tr>
<tr>
<td>9-23</td>
<td></td>
<td>HADDR</td>
<td>9-10</td>
</tr>
<tr>
<td>9-24</td>
<td></td>
<td>ERRA</td>
<td>9-10</td>
</tr>
<tr>
<td>9-25</td>
<td></td>
<td>CFLGS</td>
<td>9-11</td>
</tr>
<tr>
<td>9-32</td>
<td></td>
<td>SFLGS</td>
<td>9-13</td>
</tr>
<tr>
<td>9-33</td>
<td></td>
<td>ERRC</td>
<td>9-13</td>
</tr>
<tr>
<td>9-34</td>
<td></td>
<td>PBFFR</td>
<td>9-13</td>
</tr>
<tr>
<td>9-35</td>
<td></td>
<td>UBFFR</td>
<td>9-13</td>
</tr>
</tbody>
</table>
### TABLE OF CONTENTS cont

<table>
<thead>
<tr>
<th>SECTION NUMBER</th>
<th>PARAGRAPH NUMBER</th>
<th>TITLE</th>
<th>PAGE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>9-36</td>
<td></td>
<td>USIZE</td>
<td>9-13</td>
</tr>
<tr>
<td>9-37</td>
<td></td>
<td>NREC</td>
<td>9-14</td>
</tr>
<tr>
<td>9-38</td>
<td></td>
<td>HSCR</td>
<td>9-14</td>
</tr>
<tr>
<td>9-39</td>
<td></td>
<td>ISCR</td>
<td>9-14</td>
</tr>
<tr>
<td>9-40</td>
<td></td>
<td>HOW TO USE IOCS</td>
<td>9-14</td>
</tr>
<tr>
<td>9-42</td>
<td></td>
<td>SET UP A VECTOR</td>
<td>9-14</td>
</tr>
<tr>
<td>9-43</td>
<td></td>
<td>DEVICE HANDLER REQUIREMENTS</td>
<td>9-16</td>
</tr>
<tr>
<td>9-46</td>
<td></td>
<td>PHYSICAL I/O BUFFERS</td>
<td>9-17</td>
</tr>
<tr>
<td>9-52</td>
<td></td>
<td>SYSTEM INTERRUPT TABLE</td>
<td>9-18</td>
</tr>
<tr>
<td>9-55</td>
<td></td>
<td>IOCS MEMORY MAP</td>
<td>9-19</td>
</tr>
<tr>
<td>9-59</td>
<td></td>
<td>WRITING A DEVICE HANDLER</td>
<td>9-21</td>
</tr>
<tr>
<td>9-60</td>
<td></td>
<td>CHARACTER-ORIENTED DEVICES</td>
<td>9-21</td>
</tr>
<tr>
<td>9-66</td>
<td></td>
<td>RECORD-ORIENTED DEVICES</td>
<td>9-23</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>FLOPPY DISK HANDLER (FDH)</td>
<td></td>
</tr>
<tr>
<td>10-1</td>
<td></td>
<td>INTRODUCTION</td>
<td>10-1</td>
</tr>
<tr>
<td>10-3</td>
<td></td>
<td>COMMUNICATION</td>
<td>10-1</td>
</tr>
<tr>
<td>10-5</td>
<td></td>
<td>DOS RELATED VECTOR PARAMETERS</td>
<td>10-2</td>
</tr>
<tr>
<td>10-6</td>
<td></td>
<td>CALLING CONVENTIONS</td>
<td>10-3</td>
</tr>
<tr>
<td>10-8</td>
<td></td>
<td>GENERAL PURPOSE DISK MACRO</td>
<td>10-3</td>
</tr>
<tr>
<td>10-9</td>
<td></td>
<td>COMPLETE DOS REQUEST CODES</td>
<td>10-6</td>
</tr>
<tr>
<td>10-10</td>
<td></td>
<td>ERROR RETURN</td>
<td>10-10</td>
</tr>
<tr>
<td>10-12</td>
<td></td>
<td>DIRECTORY</td>
<td>10-13</td>
</tr>
<tr>
<td>10-14</td>
<td></td>
<td>DISK FORMAT</td>
<td>10-14</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS cont

<table>
<thead>
<tr>
<th>SECTION NUMBER</th>
<th>PARAGRAPH NUMBER</th>
<th>TITLE</th>
<th>PAGE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-17</td>
<td></td>
<td>SECTOR AND TRACK FORMATS</td>
<td>10-18</td>
</tr>
<tr>
<td>10-18</td>
<td></td>
<td>DISKETTE - IDENTIFICATION</td>
<td>10-18</td>
</tr>
<tr>
<td>10-19</td>
<td></td>
<td>NAME OF DISKETTE AND SPACE ALLOCATION</td>
<td>10-18</td>
</tr>
<tr>
<td>10-20</td>
<td></td>
<td>DATA (FILES)</td>
<td>10-19</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>DISK CONTROLLER FIRMWARE (DCF)</td>
<td></td>
</tr>
<tr>
<td>11-1</td>
<td></td>
<td>INTRODUCTION</td>
<td>11-1</td>
</tr>
<tr>
<td>11-3</td>
<td></td>
<td>SOFTWARE CONFIGURATION</td>
<td>11-1</td>
</tr>
<tr>
<td>11-5</td>
<td></td>
<td>CONTROLLER OVERVIEW</td>
<td>11-1</td>
</tr>
<tr>
<td>11-7</td>
<td></td>
<td>DISK CONTROLLER REQUESTS</td>
<td>11-3</td>
</tr>
<tr>
<td>11-9</td>
<td></td>
<td>DISK CONTROLLER ERROR RETURN CODES</td>
<td>11-4</td>
</tr>
<tr>
<td>11-11</td>
<td></td>
<td>LINKED FILE LOADER</td>
<td>11-5</td>
</tr>
<tr>
<td>11-13</td>
<td></td>
<td>INTERACTIVE BOOT PROCESS</td>
<td>11-5</td>
</tr>
<tr>
<td>11-15</td>
<td></td>
<td>INTERACTIVE SAVE PROCESS</td>
<td>11-6</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>I/O HANDLERS</td>
<td></td>
</tr>
<tr>
<td>12-1</td>
<td></td>
<td>INTRODUCTION</td>
<td>12-1</td>
</tr>
<tr>
<td>12-3</td>
<td></td>
<td>CR - CARD READER</td>
<td>12-2</td>
</tr>
<tr>
<td>12-4</td>
<td></td>
<td>CP - CENTRONICS LINE PRINTER</td>
<td>12-4</td>
</tr>
<tr>
<td>12-5</td>
<td></td>
<td>LP - DATA PRODUCTS LINE PRINTER</td>
<td>12-5</td>
</tr>
<tr>
<td>12-6</td>
<td></td>
<td>PR - PAPER TAPE READER</td>
<td>12-6</td>
</tr>
<tr>
<td>12-7</td>
<td></td>
<td>PP - PAPER TAPE PUNCH</td>
<td>12-7</td>
</tr>
<tr>
<td>12-8</td>
<td></td>
<td>TI - SILENT 700 CASSETTE INPUT</td>
<td>12-8</td>
</tr>
<tr>
<td>SECTION NUMBER</td>
<td>PARAGRAPH NUMBER</td>
<td>TITLE</td>
<td>PAGE NUMBER</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>12-9</td>
<td>TK - KEYBOARD</td>
<td></td>
<td>12-9</td>
</tr>
<tr>
<td>12-10</td>
<td>TT - CONSOLE OUTPUT</td>
<td></td>
<td>12-10</td>
</tr>
<tr>
<td>12-11</td>
<td>TO - SILENT 700 CASSETTE OUTPUT</td>
<td></td>
<td>12-12</td>
</tr>
<tr>
<td>12-12</td>
<td>TR - TELETYPES PAPER TAPE READER</td>
<td></td>
<td>12-13</td>
</tr>
</tbody>
</table>

13 SYSTEM ROUTINES

<table>
<thead>
<tr>
<th>SECTION NUMBER</th>
<th>PARAGRAPH NUMBER</th>
<th>TITLE</th>
<th>PAGE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>13-1</td>
<td>INTRODUCTION</td>
<td></td>
<td>13-1</td>
</tr>
<tr>
<td>13-3</td>
<td>PROM RESIDENT ROUTINES</td>
<td></td>
<td>13-1</td>
</tr>
<tr>
<td>13-6</td>
<td>RAM RESIDENT ROUTINES</td>
<td></td>
<td>13-2</td>
</tr>
<tr>
<td>13-10</td>
<td>ASBIN - CONVERT ASCII DIGIT TO BINARY</td>
<td></td>
<td>13-4</td>
</tr>
<tr>
<td>13-11</td>
<td>ASTCHK-ASTERISK CHECK</td>
<td></td>
<td>13-5</td>
</tr>
<tr>
<td>13-12</td>
<td>CRLF - OUTPUT CARRIAGE RETURN AND LINE FEED</td>
<td></td>
<td>13-6</td>
</tr>
<tr>
<td>13-13</td>
<td>CSI - COMMAND STRING INTERPRETER</td>
<td></td>
<td>13-7</td>
</tr>
<tr>
<td>13-14</td>
<td>CSISYN-JTASK CODE 7</td>
<td></td>
<td>13-8</td>
</tr>
<tr>
<td>13-15</td>
<td>CSIPAR-JTASK CODE 6</td>
<td></td>
<td>13-9</td>
</tr>
<tr>
<td>13-16</td>
<td>RENTRY - DDT-80 RE-ENTRY</td>
<td></td>
<td>13-12</td>
</tr>
<tr>
<td>13-17</td>
<td>ECHO - INPUT AND ECHO A CHARACTER</td>
<td></td>
<td>13-13</td>
</tr>
<tr>
<td>13-18</td>
<td>EH - SYSTEM ERROR HANDLER</td>
<td></td>
<td>13-14</td>
</tr>
<tr>
<td>13-19</td>
<td>GETHL - GET A LINE OF INPUT FROM CONSOLE</td>
<td></td>
<td>13-16</td>
</tr>
<tr>
<td>13-20</td>
<td>GETVEC - GET DEFAULT VECTOR</td>
<td></td>
<td>13-17</td>
</tr>
<tr>
<td>SECTION NUMBER</td>
<td>PARAGRAPH NUMBER</td>
<td>TITLE</td>
<td>PAGE NUMBER</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------</td>
<td>--------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>13-21</td>
<td>ADDRESS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13-22</td>
<td>MINEN - ENABLE MINIMAL LISTENER</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13-23</td>
<td>MRENT - MONITOR RE-ENTRY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13-24</td>
<td>PACC - PRINT ASCII CONTENTS OF THE ACCUMULATOR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13-25</td>
<td>PTXT - PRINT TEXT STRING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13-26</td>
<td>PVECT - PRINT VECTOR DATASET</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13-27</td>
<td>REBOOT - SYSTEM REBOOT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13-28</td>
<td>SCAN - INTERACTIVE SCAN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13-29</td>
<td>SEARCH - FIND DIRECTORY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13-30</td>
<td>SPACE - OUTPUT A SPACE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13-31</td>
<td>SRCHR, SRCHU - SEARCH MNEMONIC TABLES</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>BATCH MODE OPERATION</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14-1</td>
<td>INTRODUCTION</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14-3</td>
<td>PRINCIPLES OF OPERATION</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14-7</td>
<td>BATCH COMMAND SEQUENCE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14-12</td>
<td>EXAMPLE 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14-13</td>
<td>EXAMPLE 2</td>
<td></td>
</tr>
</tbody>
</table>
## TABLE OF CONTENTS cont

<table>
<thead>
<tr>
<th>SECTION NUMBER</th>
<th>PARAGRAPH NUMBER</th>
<th>TITLE</th>
<th>PAGE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>15-1</td>
<td>SYSTEM GENERATION INTRODUCTION</td>
<td>15-1</td>
</tr>
<tr>
<td></td>
<td>15-4</td>
<td>SYSTEM GENERATION PROCEDURE (SYSGEN)</td>
<td>15-1</td>
</tr>
<tr>
<td></td>
<td>15-5</td>
<td>OPERATING SYSTEM MODULES</td>
<td>15-4</td>
</tr>
<tr>
<td></td>
<td>15-6</td>
<td>STANDARD I/O DRIVERS</td>
<td>15-5</td>
</tr>
<tr>
<td></td>
<td>15-10</td>
<td>ADDING NEW I/O DRIVERS</td>
<td>15-6</td>
</tr>
<tr>
<td></td>
<td>15-12</td>
<td>CHANGING THE DEFAULT LOGICAL UNITS</td>
<td>15-7</td>
</tr>
<tr>
<td></td>
<td>15-14</td>
<td>CHANGING THE NUMBER OF DISK UNITS IN THE SYSTEM</td>
<td>15-7</td>
</tr>
<tr>
<td></td>
<td>15-16</td>
<td>SYSTEM GENERATION OF A 64K OPERATING SYSTEM</td>
<td>15-7</td>
</tr>
</tbody>
</table>

### APPENDIX

- **A**  
  Z80 OPCCODES  
- **B**  
  MOSTEKT OBJECT OUTPUT DEFINITION  
- **C**  
  SCRATCHPAD MEMORY MAP  
- **D**  
  TESTING/DIAGNOSTICS  
- **E**  
  FLP-80DOS ERROR DICTIONARY  
- **F**  
  SYSTEM LINKAGES  
- **G**  
  DISK RECOVERY UTILITY
<table>
<thead>
<tr>
<th>FIGURE NUMBER</th>
<th>TITLE</th>
<th>PAGE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>DEVELOPMENT SYSTEM PROGRAMS</td>
<td>1-3</td>
</tr>
<tr>
<td>1-2</td>
<td>FLP-80DOS SYSTEM</td>
<td>1-8</td>
</tr>
<tr>
<td>1-3</td>
<td>INPUT/OUTPUT LOGICAL UNIT NUMBERS</td>
<td>1-15</td>
</tr>
<tr>
<td>1-4</td>
<td>DISKETTE</td>
<td>1-17</td>
</tr>
<tr>
<td>1-5</td>
<td>RELATIONSHIP OF SYSTEM PROGRAMS IN FLP-80DOS</td>
<td>1-20</td>
</tr>
<tr>
<td>1-6</td>
<td>STANDARD FLP-80DOS MEMORY MAP</td>
<td>1-22</td>
</tr>
<tr>
<td>1-7</td>
<td>OEM-80 PORT ALLOCATION</td>
<td>1-24</td>
</tr>
<tr>
<td>1-8</td>
<td>FLP-80DOS COMMAND SUMMARY</td>
<td>1-25</td>
</tr>
<tr>
<td>4-1</td>
<td>EDITOR MEMORY MAP</td>
<td>4-3</td>
</tr>
<tr>
<td>4-2</td>
<td>LOGICAL UNIT NUMBER STRUCTURE</td>
<td>4-4</td>
</tr>
<tr>
<td>5-1</td>
<td>ASSEMBLER MEMORY MAP</td>
<td>5-4</td>
</tr>
<tr>
<td>5-2</td>
<td>LOGICAL UNIT NUMBER STRUCTURE</td>
<td>5-5</td>
</tr>
<tr>
<td>6-1</td>
<td>EXAMPLE LOAD MAP, GLOBAL CROSS REFERENCE, AND GLOBAL SYMBOL TABLE</td>
<td>6-8</td>
</tr>
<tr>
<td>7-1</td>
<td>DDT USER REGISTER MAP</td>
<td>7-3</td>
</tr>
<tr>
<td>7-2</td>
<td>DDT DATA PATHS</td>
<td>7-4</td>
</tr>
<tr>
<td>8-1</td>
<td>EXAMPLE OF RDCHR AND WRCHR</td>
<td>8-5</td>
</tr>
<tr>
<td>9-1</td>
<td>IOCS MEMORY MAP</td>
<td>9-20</td>
</tr>
<tr>
<td>10-1</td>
<td>EFFECTS OF FDH COMMANDS</td>
<td>10-9</td>
</tr>
<tr>
<td>10-2</td>
<td>FLP-80DOS V2.1 DISKETTE FORMAT</td>
<td>10-16</td>
</tr>
<tr>
<td>15-1</td>
<td>SAMPLE SYSTEM GENERATION</td>
<td>15-8</td>
</tr>
<tr>
<td>B-1</td>
<td>EXTERNAL SYMBOL LINK LIST</td>
<td>B-5</td>
</tr>
</tbody>
</table>
# List of Tables

<table>
<thead>
<tr>
<th>Table Number</th>
<th>Title</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-1</td>
<td>Summary of FLP-80DOS Editor Commands</td>
<td>4-22</td>
</tr>
<tr>
<td>5-1</td>
<td>Allowed Characters</td>
<td>5-9</td>
</tr>
<tr>
<td>5-2</td>
<td>Generic Operands</td>
<td>5-13</td>
</tr>
<tr>
<td>5-3</td>
<td>Allowed Operators and Hierarchies in FLP-80DOS Assembler</td>
<td>5-16</td>
</tr>
<tr>
<td>7-1</td>
<td>Mnemonics Recognized by DDT-80</td>
<td>7-5</td>
</tr>
<tr>
<td>7-2</td>
<td>DDT Command Summary</td>
<td>7-7</td>
</tr>
<tr>
<td>9-1</td>
<td>Vector Definition</td>
<td>9-3</td>
</tr>
<tr>
<td>9-2</td>
<td>FLP-80DOS Device Mnemonics</td>
<td>9-5</td>
</tr>
<tr>
<td>9-3</td>
<td>General Purpose Requests</td>
<td>9-7</td>
</tr>
<tr>
<td>9-4</td>
<td>Format Request Codes</td>
<td>9-8</td>
</tr>
<tr>
<td>9-5</td>
<td>Physical Buffer Allocation</td>
<td>9-17</td>
</tr>
</tbody>
</table>
NOTE: Certain sections of this manual refer to specific hardware configurations existing on the MOSTEK AID-80F Development System. In the future, FLP-80DOS will also be implemented on other hardware configurations. Since there will be minor differences in hardware implementation (e.g. I/O port numbers) the user should refer to the appropriate hardware manual for information concerning his system configurations.
PART 1

USER INFORMATION
SECTION 1

FLP-80DOS

GENERAL DESCRIPTION

1-1. INTRODUCTION

NOTE: This section should be read in its entirety. It discusses concepts which are used throughout the system.

1-2. FLP-80DOS is the MOSTEK Disk Operating System for the Z80. It is a software package designed to work with the following minimum hardware configuration:

1. Z80 CPU with a minimum of 16K Bytes of RAM
2. 4K Byte EPROM and a 256 Byte Scratchpad RAM
3. Floppy Disk Interface and 1 to 4 flexible disk units.

1-3. FLP-80DOS consists of development system software and OEM software. The development system programs are diagrammed in Figure 1-1. Each of these programs is discussed in detail in the next 6 sections of this manual. These programs provide state-of-the-art software for developing Z80 programs. The complete FLP-80DOS system is diagrammed in Figure 1-2. The component parts of the system establish a firm basis for OEM products. This diagram is discussed in detail in Sections 8 through 13 of this manual. The following programs are supplied in the FLP-80DOS package:

1. Monitor
2. Debugger
3. Text Editor
4. Z80 Assembler
5. Peripheral Interchange Program
6. Linker
7. A generalized I/O system for peripherals

These programs provide state-of-the-art software for developing Z80 programs as well as establishing a firm basis for OEM products.
FIGURE 1-1. DEVELOPMENT SYSTEM PROGRAMS

- DEBUGGER (DDT)
- TEXT EDITOR (EDIT)
- Z80 ASSEMBLER (ASM)
- PERIPHERAL INTERCHANGE PROGRAM (PIP)
- LINKER (LINK)

FLP-80 DOS MONITOR
1-4. MONITOR. The Monitor provides a user interface from the console to the rest of the software. The user can load and run system programs, such as the Assembler, using one simple command. Programs in binary format can be loaded into and dumped from RAM. All I/O is done via channels which are identified by Logical Unit Numbers. The Monitor allows any software device handler to be assigned to any Logical Unit Number. Thus, the software provides complete flexibility in configuring the system with different peripherals.

1-5. DESIGNER'S DEVELOPMENT TOOL - DDT. The DDT debugger program is supplied in PROM. It provides a complete facility for interactively debugging relative and absolute Z80 programs. Standard commands allow displaying and modifying memory and CPU registers, setting breakpoints, and executing programs. Additional commands allow use of the MOSTEK AIM-80 to interactively debug a target system. Mnemonics are used to represent Z80 registers, thus simplifying the command language.

1-6. TEXT EDITOR - EDIT. The FLP-80DOS Editor permits random access editing of ASCII character strings. The Editor works on blocks of characters which are rolled in from the disk. It can be used as a line or character-oriented editor. Individual characters may be located by position or context. Each edited block is automatically rolled out to disk after editing. Although the Editor is used primarily for creating and modifying Z80 assembly language source statements, it may be applied to any ASCII text delimited by "carriage returns."

1-7. Z80 ASSEMBLER - ASM. The FLP-80DOS Assembler reads Z80 source mnemonics and pseudo-ops and outputs an assembly listing and object code. The assembly listing shows address, machine code, statement number, and source statement. The code is in industry-standard, hexadecimal format modified for relocatable,
linkable assemblies. The Assembler supports conditional assemblies, global symbols, relocatable programs, and a printed symbol table. It can assemble any length program, limited only by a symbol table size which is dependent on available RAM. Expressions involving arithmetic and logical operations are allowed. Although normally used as a two-pass assembler, the Assembler can also be run as a single-pass assembler.

1-8. LINKER-LINK. The Linker provides capability for linking object modules together and creating a binary (RAM image) file on disk. A binary file can be loaded using the Monitor GET or IMPLIED RUN command. Modules are linked together using global symbols for communication between modules. The Linker produces a global symbol table and a global cross-reference table which may be listed on any output device. The Linker also provides a library search option for all global symbols undefined after the specified object modules are processed. If a symbol is undefined, the Linker searches the disk for an object file having the filename of the symbol. If the file is found, it is opened and linked with the main module in an attempt to resolve the undefined symbol.

1-9. PERIPHERAL INTERCHANGE PROGRAM - PIP. The Peripheral Interchange Program provides complete file maintenance facilities for the system. In addition, it can be used to copy information from any device or file to any other device or file. The command language is easy to use and resembles that used on DEC minicomputers.

1-10. I/O SYSTEM. The I/O software, which is the heart of the FLP-80DOS development system, can be used directly in OEM applications. The software consists of two programs which provide a complete disk-handling facility.
1-11. The first package is called the I/O Control system (I/OCS). This is a generalized blocker/deblocker which can interface to any device handler. Input and output can be done via the I/OCS in any of four modes:

1. single byte transfer.
2. line at a time, where the end of a line is defined by carriage return.
3. multibyte transfers, where the number of bytes to be transferred is defined as the logical record length.
4. continuous transfer to end-of-file, which is used for binary (RAM-image) files.

The I/OCS provides easy application of I/O oriented packages to any device. There is one entry point, and all parameters are passed via a vector defined by the calling program. Any given device handler defines the physical attributes of its device which are, in turn, used by the I/OCS to perform blocking and de-blocking.

1-12. The Floppy Disk Handler (FDH) interfaces from the I/OCS to a firmware controller for up to 4 floppy disk units. The FDH provides a sophisticated command structure to handle advanced OEM products. The firmware controller interfaces to MOSTEK's Disk Controller Board. The disk format is soft-sectored. The software directly handles double-sided disks. The FDH has advanced error recovery capability. It supports a bad sector map and an extensive directory which allows multiple users. The file structure is doubly linked to increase data integrity on the disk. A bad file can be recovered from either its start or end.

1-13. OTHER PROGRAMS
MOSTEK offers a number of programs which work with FLP-80DOS. These programs are purchasable options for the Microcomputer. The following programs will be of interest to many users:

**FZCASM**
- The 3870/F8 Cross Assembler allows assembly of all F8 opcodes on the AID-80F. The FLP-80DOS Text Editor and Linker can be used with the Cross Assembler to produce programs which can be debugged.

**ZAIM-72**
- This 3870 family debugger program is to be used with the MOSTEK AIM-72 board for debugging 3870, 3872, or 3876 programs.

**MOSTEK LIBRARY**
- The Library consists of a set of utilities which are used at Mostek. Programs include a word processor, Lawrence Livermore Laboratory BASIC (oriented to controller applications), a disk recovery utility, an 8080 to Z80 source translator, a hexadecimal dump utility, and others. Complete source files are included.

**BASIC**
- MOSTEK BASIC features string and array manipulation, random access disk, and a complete set of standard BASIC commands.

**FORTRAN IV**
- MOSTEK FORTRAN is ANSI X3.3(1966) standard FORTRAN IV. It features an extensive run-time library.

**MACRO-80**
- Powerful Macro Assembler for Z80.

**MACRO-70**
- Powerful Macro Assembler for 3870/F8.
FIGURE 1-2. FLP-80DOS SYSTEM

FLP-80DOS MONITOR

DEBUGGER (DDT)  TEXT EDITOR (EDIT)  Z80 ASSEMBLER (ASM)  PERIPHERAL INTERCHANGE PROGRAM (PIP)  LINKER (LINK)  OEM APPLICATION PROGRAM

I/O CONTROL SYSTEM (IOCS)

CONSOLE DEVICE HANDLER

HARDWARE UART

CONSOLE

LINE PRINTER HANDLER

HARDWARE PIO

LINE PRINTER

FLOPPY DISK HANDLER (FDH)

DISK CONTROLLER FIRMWARE

FLP-80 HARDWARE

OTHER DEVICE HANDLERS

FLOPPY DISK UNITS
1-15. REFERENCE DOCUMENTS

AID-80F Operations Manual                  MK78569
SDB-80 Software Development Board Operations Manual MK78544
SDB-80E (European version)                  MK78548
FLP-80 Hardware Operations Manual          MK78560
FLP-80E (European version)                 MK78561
RAM-80B Operations Manual                  MK78545
RAM-80BE (European Version)                MK78555
DSS-80 Development System Software Program Listing MK78588
   (OEM users only - restricted distribution)
DOPS-80 Disk Operating Software Program Listing MK78589
   (OEM users only - restricted distribution)

1-16. DEFINITION OF SYMBOLS USED IN THIS MANUAL

1-17. The following conventions are used throughout this manual:

All user input from the console device is underlined.
All hexadecimal numbers are identified by a subscript H, except where an example of program input or output is given.
(CR) means carriage return.
aaaa means any hexadecimal number.

1-18. CONSOLE INTERACTION

1-19. ENTERING DATA ON THE CONSOLE. Each line of input from the console is terminated with a carriage return in FLP-80DOS. The maximum length of a line of input is 160 characters. Before ending a line with carriage return, the user can modify the line with the following keys (Note that these standards do not apply to DDT, the debugger):

1. TAB (ASCII 09H) - move the console cursor over mod-8 spaces. Tabs are set every 8 spaces.
2. RUBOUT (ASCII 7FH) - delete the previous character entered. A blackslash is printed on either side of the characters which are deleted.

3. BACKSPACE (ASCII 08H) - delete the previous character. It is erased from the (CRT) screen by overprinting with a blank, and the cursor is moved backward. Backspacing over a tab character will back the cursor to the correct screen position.

4. CNTL-U (ASCII 15H) - delete the current line of input and reprompt for another line.

5. SPACE BAR - used to alternately start and stop listing to console device. This is useful when a long file is being spooled to a CRT screen and the user wishes to view the file a page at a time.

1-20. CONSOLE ESCAPE ("Minimal Listener"). Any executing program in FLP-80DOS can be interrupted from the console device. (This feature is inhibited while DDT, the debugger, is being used.) The following key inputs are allowed:

1. CNTL-X (ASCII 18H) - Monitor Escape. Entering this code from the console keyboard immediately reboots the system software and returns control to the FLP-80DOS Monitor. After a brief delay while the disk is
accessed, the Monitor prompt will appear on the console. The Monitor prompting character is a $. The Monitor escape cannot be used during use of the Debugger (DDT) or the Editor (EDIT).

NOTE: Monitor Escape is designed to provide an immediate reboot of the Monitor without finishing the currently executing program. Any output files which were open when the Monitor Escape was performed will not be closed. This means that those files will have no information stored in them.

2. CNTL-C (ASCII 03H)- Debugger Escape. Entering this code from the console keyboard immediately returns control to the debugger (DDT). The current Z80 registers will be printed on the console, and DDT will wait for a command. To resume execution, enter a dot (.), then the command 'E'. For further details on using DDT as a debugging aid, please see Section 7 of this manual. This escape cannot be used if DDT is called up by the Monitor, or during use of the Editor.

NOTE Debugger Escape is designed to allow a program to be suspended by the user. It also provides a software asynchronous interrupt which is useful in debugging programs. It is not active during usage of DDT, the debugger (i.e., the user cannot use
Debugger Escape when using DDT). It may be used any number of times during the execution of a program.

1-21. CONCEPT OF DATASET

1-22. A dataset is a logical grouping of data associated with an I/O device. Throughout FLP-80DOS a dataset is identified as follows:

DEV:FILENAME.EXT[UIC]

where:

DEV = The device mnemonic consisting of two letters and a decimal digit terminated by a colon. The letters identify the device and the digit identifies the unit (e.g., DK1: is disk unit 1). If no digit is entered, unit 0 is assumed. If the device mnemonic itself does not appear, the system disk (DKO:) is assumed. The following devices can be handled by FLP-80DOS supplied to you:

<table>
<thead>
<tr>
<th>DEVICE NAMES</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP:</td>
<td>Line Printer (Centronics)</td>
</tr>
<tr>
<td>CR:</td>
<td>Documation M300 card reader</td>
</tr>
<tr>
<td>DKO:</td>
<td>System Disk Unit (right hand unit)</td>
</tr>
<tr>
<td>DK1:</td>
<td>User Disk Unit (left hand unit)</td>
</tr>
<tr>
<td>LP:</td>
<td>Line Printer (Data Products)</td>
</tr>
<tr>
<td>PP:</td>
<td>High-Speed Paper Tape Punch</td>
</tr>
<tr>
<td>PR:</td>
<td>High-Speed Paper Tape Reader</td>
</tr>
<tr>
<td>TI:</td>
<td>Silent 700 Cassette Tape Reader Input</td>
</tr>
<tr>
<td>TO:</td>
<td>Silent 700 Cassette Tape Output</td>
</tr>
<tr>
<td>TT:</td>
<td>Teletype Typehead, CRT Screen, or Silent 700 Printer</td>
</tr>
<tr>
<td>TK:</td>
<td>Terminal Keyboard</td>
</tr>
</tbody>
</table>

Additional devices and their corresponding software handlers can be added by the user.
FILENAME = The file name specification consists of one or more letters or digits. The first six letters or digits specify the name. The first character must be a letter. All letters or digits in excess of 6 are ignored. The file name is not used if the device is not a file device (e.g., the line printer).

EXT = The extension specification consists of a period, followed by one or more letters or digits. The first three letters or digits specify the extension. All letters or digits in excess of three are ignored. If an extension does not appear in the dataset, a default extension of 3 blanks is assumed. The extension does not appear if the device is not a file device. The extension 'BIN' is reserved for binary (RAM image) files. The extension 'OBJ' is reserved for object files. The extension 'TMP' is reserved for temporary files by the Editor. The extension 'CRS' is used by the Assembler and the Linker for cross-reference files. The extension 'LST' is used by the Assembler for listing files.

UIC = The user identification code UIC consists of a left square bracket, followed by one to three decimal digits, followed by a right square bracket. The largest valid decimal number is 255. If the user identification code does not appear, a default code of 1 is assumed. The UIC is maintained on all disk files. It can be used to identify files of different users. The UIC does not appear if the device is not a file device.
1-23. CONCEPT OF LOGICAL UNIT NUMBERS

1-24. All FLP-80DOS input and output is done in terms of logical unit numbers, just as in FORTRAN. A Logical Unit Number (LUN) is any number in the range 0 - FFH. Any dataset can be assigned to any Logical Unit Number (LUN) (using the Monitor ASSIGN command). The LUN acts as a channel through which a program performs input and output. This is diagrammed in Figure 1-3.

1-25. Logical Unit Numbers 0-5 are always pre-assigned when the system is powered up or reset. These are all "default" LUN's and they are assigned the following meanings:

<table>
<thead>
<tr>
<th>LUN</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>console input</td>
</tr>
<tr>
<td>1</td>
<td>console output</td>
</tr>
<tr>
<td>2</td>
<td>object input</td>
</tr>
<tr>
<td>3</td>
<td>object output</td>
</tr>
<tr>
<td>4</td>
<td>source input</td>
</tr>
<tr>
<td>5</td>
<td>source output</td>
</tr>
</tbody>
</table>

1-26. LUN 0 and 1 are always assigned to the user console device. LUN's 0-5 have special features which make them useful for writing your own programs (more detail is given in Sections 8 and 9 of this manual). LUN FFH cannot be reassigned to a device. This means that any program using LUN FFH is responsible for making the device assignment. Further detail is given in Section 2 under the Monitor "ASSIGN" command.

1-27. DATE FEATURE

1-28. The date feature in FLP-80DOS V2.1 allows you to record the date of creation or last update of a file. This is done automatically by the system except for binary files.
FIGURE 1-3. INPUT/OUTPUT LOGICAL UNIT NUMBERS

- INPUT DEVICE
- DEVICE HANDLER (SOFTWARE)
- LOGICAL UNIT NUMBER
- APPLICATION PROGRAM OR SYSTEM PROGRAM
- LOGICAL UNIT NUMBER
- DEVICE HANDLER (SOFTWARE)
- OUTPUT DEVICE
1-16

1-29. At power-up time, after system reset, the date can be entered at the system's request. (See start-up procedures in paragraph 1-36 for information on entering the date). Once the date has been entered correctly, it will remain in the system until turned off. A system reset does not destroy the date. In this case the date will appear after the sign-on message and no request to enter it will appear. If the user wishes to change the current date for any reason, it can be done through the DATE command in PIP. (see paragraph 3-18).

1-30. When a new file is created or an old one is updated, for example through the Editor, the current date is stored in its directory entry at the load-address bytes, with the exception of binary files in which case the load-address bytes contain that information and no date is recorded. We recommend that the user create a cross-reference file along with his binary file through the Linker, using option C. (see paragraph 6-9).

1-31. FLEXIBLE DISK HANDLING PROCEDURE

1-32. The 2 diskettes supplied with the system are both system diskettes. That is, each contains all of the FLP-80DOS software. The format is soft-sectored. It is recommended that burnished and qualified diskettes be used with FLP-80 system. New diskettes do not have to be pre-formatted because the system provides formatting capability. Each diskette in the system has all the system software on it. Each has 1964 available sectors of 124 data bytes (243536 bytes total). The capacity is double this for double-sided diskettes.

1-33. Figure 1-4 shows the diskette. The following precautions should be followed in handling the diskettes:
   1. Do not bend or fold the diskette.
FIGURE 1-4. DISKETTE

- LABELS
- RECORDERING SLOT
- SPINDLE HOLE
- INDEX HOLE (USED FOR FORMATTING)
2. Do not touch the exposed recording area of the diskette.

3. Do not place heavy materials on or write on the diskette with other than a felt-tip marker.

4. Do not place the diskette near strong magnetic fields.

1-34. Diskettes are inserted into the drives as follows:
1. Insert the diskette as far as it will go into the disk unit slot. The recording slot should be to the rear and the label should be on the right-hand side.

2. Slowly close the door until it latches.

1-35. Diskettes are removed from the disk unit by depressing the latch button. The disk unit door should spring open and the diskette should be pushed out of the unit.

CAUTION: Do not power up or power down the system with a diskette inserted in a disk unit. Doing so may destroy the integrity of the data on the diskette.

NOTE: It is recommended that all user files be backed up on separate diskettes whenever changes are made. This precaution guards against loss of a file in case a non-recoverable disk error occurs.

1-36. START UP PROCEDURES

1-37. Configure the hardware system as explained in the System Operations Manual. Power up. Insert the FLP-80DOS diskette into the right-hand disk drive, disk unit zero (DK0:), and close the door. Depress the 'carriage return' key on the console device. There should be a slight delay while the system software is read into RAM from disk. Then the Monitor prompt should be
A. PLEASE ENTER DATE (DD-MMM-YY) -->

B. The user enters the date by typing first the day of the month, followed by the first three letters of the month, and then by the last two digits for the year; each item is separated from the next by a hyphen. The entered line can be edited using rub-out, backspace, and control-u. If the user enters an invalid date, a syntax error message is printed, and the date is ignored. If the user does not wish to use the date option he can enter just a carriage return.

Example: PLEASE ENTER DATE (DD-MMM-YY)  7-APR-79 (CR)

1-38. Figure 1-5 shows the relationships among the programs in FLP-80DOS. The user initializes the system by depressing the 'RESET' button on the system and 'carriage return' on his console device. The Boot Procedure reads the system software into RAM from disk and gives control to the Monitor. From the Monitor, any system program can be executed by entering its name (plus any other required information) from the console device.

The Debugger, Text Editor, and Peripheral Interchange Program can be exited by entering 'Q' (for a 'Quit'), at which point control is given back to the Monitor. The Z80 Assembler and Linker return control to the Monitor when their tasks are completed. In the system programs the system can be rebooted by entering CNTL-X (Monitor Escape) except EDIT. The Debugger can be entered
Figure 1-5. Relationship of System Programs in FLP-80DOS

- **Initialize** ('Reset' & 'Carriage Return')
- **Monitor Escape** (CNTL - X)
- **Breakpoint Encountered or Debugger Escape** (CNTL - C)

**Boot Procedure**
- **Debugger**
- **Linker**
- **Monitor**
- **Text Editor**
- **Z80 Assembler**

**Input Files**:
- **Edit (file)**
- **ASM (file)**
- **Q**
- **DDT (Completion)**
- **Link (file)**
- **Q**
- **PIP (Completion)**
- **Peripheral Interchange Program**
1-39. You now have one of the most powerful Z80 development systems at your finger tips. You will probably first wish to create a file on diskette. If so, proceed to Section 4 of this manual.

1-40. If the prompt does not appear on the console, see the troubleshooting section (Appendix D).

1-41. MEMORY SUMMARY

1-42. MEMORY MAP. Figure 1-5 depicts the memory map of the FLP-80DOS software. The standard system is supplied with 32K of RAM starting at address zero, 4-1K PROM's starting at E000H, and 256 bytes of "scratchpad" RAM starting at FF00H.

1-43. The PROM located at ECO0H is the Disk Controller Firmware. It has the responsibility of translating track and sector information into commands to control the FLP-80 board. The three PROM's starting at E000H contain the power up procedure and the DDT debugger. The rest of the system software is read into the upper 9K of RAM from disk. This leaves the first 23K of RAM free for user programs and debugging (in a 32K system). The Editor, Assembler, PIP and the Linker also use this area. The 256 byte "scratchpad" RAM, located at FF00H, is used by the DDT debugger and the Monitor.
FIGURE 1-6. STANDARD FLP-80DOS MEMORY MAP

FFFF  ———  64 K
FF00
EEFF
E000

256 BYTE SCRATCHPAD RAM

4-1K PROMS - BOOTSTRAP SEQUENCE, DEBUGGER, AND DISK CONTROLLER Firmware.

7FFF  ———  32 K

MONITOR, SYSTEM PROGRAMS AND FLEXIBLE DISK HANDLER RESIDE AT TOP OF 32 K RAM MEMORY SPACE.

0

ABOUT 23K RAM AVAILABLE FOR USER PROGRAMS
PORT MAP. Figure 1-6 defines the port allocation on the SDB-80. Ports D0-D7 are the PIO ports that come out to top edge connectors on the SDB-80. Ports D8-DB are the counter timer circuit ports; port D8 is the timer for the UART baud rate. Port DE is used for controlling dataset ready (DSR), clear to send (CTS), and reader step (RS). Also, Port DE is used for sensing the state of data terminal ready (DTR), request to send (RTS), and serial bit string of measuring baud rate (used by the operating system). Ports DC and DD are the UART ports. Ports E2-E7 are the disk controller ports. MOSTEK is reserving ports E8H thru FFH for future expansion of its development system. Ports 7C-7D are also used by the FLP-80DOS Software Version 2.1 and above. It is recommended that the user limit his development system application to ports 00H thru CFH. Of course, for an OEM application all 256 ports are available to the user. In the event any development system add-on peripheral would exceed the assigned number of ports, MOSTEK would start with CFH and work down.
FIGURE 1-7. OEM-80 PORT ALLOCATION

- FF: FUTURE SDB-80 EXPANSION
- E0: FUTURE EXPANSION
- E7: DISK CONTROLLER
- E6: DISK CONTROLLER
- E5: DISK CONTROLLER
- E4: DISK CONTROLLER
- E3: DISK CONTROLLER
- E2: DISK CONTROLLER
- E1: FUTURE EXPANSION
- E0: FUTURE EXPANSION
- DF: DEBUG CONTROL
- DE: SYSTEM CONTROL
- DD: UART CONTROL
- DC: UART DATA
- DB: CTC CHANNEL 3
- DA: CTC CHANNEL 2
- D9: CTC CHANNEL 1
- D8: CTC CHANNEL 0
- D7: PIO-D6 CONTROL
- D6: PIO-D6 DATA
- D5: PIO-D4 CONTROL
- D4: PIO-D4 DATA
- D3: PIO-D2 CONTROL
- D2: PIO-D2 DATA
- D1: PIO-D0 CONTROL
- D0: PIO-D0 DATA
- CF: USER DEFINED PORTS
- 00: USER DEFINED PORTS
FIGURE 1-8. FLP-80DOS COMMAND SUMMARY

POWER UP OR RESET

Depress "CARRIAGE RETURN"

CONSOLE INTERACTION (Except DDT)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEL</td>
<td>delete the previous character</td>
</tr>
<tr>
<td>BACKSPACE</td>
<td>delete the previous character.</td>
</tr>
<tr>
<td>CNTL-U</td>
<td>delete the current line.</td>
</tr>
<tr>
<td>CNTL-C</td>
<td>suspend operation.</td>
</tr>
<tr>
<td>CNTL-X</td>
<td>abort to Monitor and reinitialize.</td>
</tr>
<tr>
<td>RETURN</td>
<td>end of command line.</td>
</tr>
<tr>
<td>CNTL-I</td>
<td>tab over 8 spaces.</td>
</tr>
</tbody>
</table>

MONITOR

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ASSIGN lun, dataset</td>
<td>assign dataset to LUN</td>
</tr>
<tr>
<td>$BEGIN [aaaaa]</td>
<td>start execution at address aaaa.</td>
</tr>
<tr>
<td>$CLEAR lun</td>
<td>clear an assignment in the redirect table.</td>
</tr>
<tr>
<td>$DDT</td>
<td>enter DDT, the debugger.</td>
</tr>
<tr>
<td>$DTABLE</td>
<td>print default LUN table.</td>
</tr>
<tr>
<td>$DUMP aaaa,bbbb,dataset[.OBJ]</td>
<td>dump absolute object module to dataset.</td>
</tr>
<tr>
<td>$GET dataset[.BIN]</td>
<td>load binary file into RAM.</td>
</tr>
<tr>
<td>$INIT</td>
<td>initialize disk.</td>
</tr>
</tbody>
</table>
$RTABLE - print redirect table of LUN's.

$SAVE aaaa,bbbb,dataset[.BIN] - save binary file from RAM.

ASSEMBLER
$ASM source dataset [TO listing dataset[,object dataset]]

OPTIONS

C - Print cross reference listing
K - no listing
L - listing (default)
N - no object output
O - object output (default)
P - pass 2 only
Q - quit - return to Monitor
R - reset symbol table (pass 2 only operation)
S - print symbol table.

DESIGNER'S DEVELOPMENT TOOL
$DDT

NOTE: The console interaction for DDT is slightly different from the rest of FLP-80DOS.
Terminator = Carriage return, ^, /, or dot.
The space between command and operands is printed by the system.

M aaaa,bbbb - Display, update, or tabulate the contents of memory.
P aa - Display and/or update the contents of an I/O port.
E [aaaa] - Transfer control from DDT-80 to a user's program.
H +aaaa+bbbb=...  - Perform 16 bit hexadecimal addition and/or subtraction.
C aaaa,bbbb,cccc  - Copy the contents of a block of memory to another location in memory.
B aaaa  - Insert a breakpoint in the user's program.
R 1,X  - Display the contents of the user registers x=0 short, x=1-long.
O aaaa  - Set the offset constant.
L aaaa,bbbb,cccc  - Locate all occurrences of an 8 or 16 bit data pattern.
F aaaa,bbbb,cc  - Fill memory limits with an 8 bit data pattern.
V aaaa,bbbb,cccc  - Verify that 2 blocks of memory are equal.
W aaaa,nn,xxx  - Software single step (walk) for nn steps.
  xx=HO means print register heading.
Q  - Quit DDT-80 and return to the system Monitor.

EDITOR

$EDIT file

An  Advance n records
Bn  Backup n records.
Cn/string1/string2/ Change n occurrences of string 1 to string 2.
Dn  Delete n records, starting with current record.
En  Exchange n records with inserted records.
Fn  Flag print option: 0=no print, not 0=print
G dataset  Get dataset and insert after current record.
I  Insert records after current record.
Ln  Line: Access record number n.
Mn  Macro: Place command string into alternate command buffer 1 or 2.
Pn dataset
Put n records out to a different dataset (file).

Q
Quit: Save the file on disk and terminate the editor.

Sn/string/
Search for nth occurrence of the string.

T
Top: Insert at top of file before the first record.

Vn
Verify n records on the console device.

Wn
Write n records with record numbers to LUN 5.

Xn
Execute alternate command buffer n (1 or 2).

In all commands, except Fn and Ln, if n is zero or if n is not entered, it is assumed to equal one (1). n can take the form n₁ thru n₂ by entering n₁ - n₂.

LINKER

$LINK dataset 1 ,..,datasetn [TO dataset B [,dataset C]]
where dataset 1 and datasetn are object files, dataset B is binary file, and dataset C is a load map and cross reference listing.

A - enter starting link address.
C - global cross reference table output to dataset C.
L - Library search on a disk unit.
S - global symbol table output to dataset C.
U - print list of undefined global symbols.

PERIPHERAL INTERCHANGE PROGRAM

$PIP
APPEND dataset1 TO dataset 2 -append.
<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPY</td>
<td>dataset2,...,dataset n TO dataset 1</td>
</tr>
<tr>
<td>DATE</td>
<td>- copy.</td>
</tr>
<tr>
<td>DIRECT</td>
<td>dataset 1 [TO dataset 2]</td>
</tr>
<tr>
<td>ERASE</td>
<td>dataset 1 ,...,dataset n</td>
</tr>
<tr>
<td>FORMAT</td>
<td>name</td>
</tr>
<tr>
<td>INIT</td>
<td>- examine/change date.</td>
</tr>
<tr>
<td>RENAME</td>
<td>dataset 1 TO dataset 2</td>
</tr>
<tr>
<td>STATUS</td>
<td>dataset 1 TO dataset 2</td>
</tr>
<tr>
<td>QUIT</td>
<td>- print directory.</td>
</tr>
<tr>
<td></td>
<td>- erase a file.</td>
</tr>
<tr>
<td></td>
<td>- format a disk in disk unit 1.</td>
</tr>
<tr>
<td></td>
<td>- initialize disk units.</td>
</tr>
<tr>
<td></td>
<td>- rename file.</td>
</tr>
<tr>
<td></td>
<td>- print status of disk.</td>
</tr>
<tr>
<td></td>
<td>- return to Monitor</td>
</tr>
</tbody>
</table>
SECTION 2

MONITOR

2-1. INTRODUCTION

2-2. The Monitor provides communication with the user via the console terminal enabling him to load and start execution of either system (e.g., PIP, EDITOR, ASM, LINKER) or user programs. In addition, the Monitor provides utility functions such as reassignment of logical unit devices and the creation of RAM image files. After power up or reset, the system automatically enters the Monitor environment awaiting entry of user commands. The prompting character for the Monitor is a $.

2-3. OPERATIONS SUMMARY

2-4. SYSTEM RESET.

2-5. The FLP-80DOS operating system may be reset by depressing the system RESET switch and then typing a "carriage return" on the console terminal. This starts the system reset sequence which first calculates the terminal baud rate and then loads the operating system into memory from the file OS.BIN[255] and begins execution at its starting address. The Monitor which is the first module in the operating system (See Figure 15-1) begins by initializing the following system parameters.

1. Default logical units 0-5
2. Logical unit redirect table
3. RAM mnemonic table (see Paragraph 15-10).
4. IOCS buffer allocation table (see paragraph 9-46)
5. All disk units containing diskettes (DK0, DK1 and etc.)

After the initialization sequence is completed, the Monitor
prints the system sign on message followed by the date or a prompt to enter the date if the system does not have a valid date stored, (this will always occur after power-up). Then a $ prompt will appear on the console.

2-6. POWER UP SEQUENCE. The power up sequence is identical to reset (See paragraph 2-4).

2-7. MONITOR COMMAND SUMMARY

2-8. Some of the Monitor commands utilize dataset specifications (See para. 1-21). A dataset can consist of device specifications (e.g. PR:) or file specifications (e.g. DK1:BINDEC.OBJ). When entering a monitor command name, only the number of characters required for uniqueness must be entered. These characters are underlined in the command syntax definition. Monitor commands can be divided into the following functional categories.

1. File Creation and Loading
   SAVE - Saves a binary file on disk.
   GET - Loads a binary file into RAM.
   DUMP - Saves an absolute object file.
   BEGIN - Begins execution of a loaded program.

2. Logical Unit Assignment and Table Functions.
   $TABLE - Lists the logical unit default table.
   $ASSIGN - Assigns the redirect of a logical unit.
   $CLEAR - Clears the redirect of a logical unit.
   $RTABLE - Lists the logical unit redirect table.

3. Miscellaneous
   $DDT - Enters DDT environment.
   $INIT - Initialize system for newly inserted diskettes.
2-9. IMPLIED RUN COMMAND. As the user types a command, its characters are entered directly into the command buffer. After a carriage return is entered, the Monitor compares the command name in the buffer with a list of Monitor commands. If a Monitor command is not entered, the Monitor assumes the command name is a binary file (extension = BIN) on the system disk. The system disk which is disk unit 0 (DK0:) is then searched for the specified file. If the file is not found, the following message is printed on the console.

****ERROR 04 FILE NOT FOUND
If the file is found, it is loaded and execution is started at its load address. The implied run command also enables the "minimal listener" which provides a console escape during program execution (see paragraph 2-45).

2-10. The implied run command provides the facility for loading and executing both system programs and user programs. The following commands transfer control from the Monitor to system programs which reside on the system disk (DK0:).

$EDIT - Enter Editor
$PIP - Enter Peripheral Interchange Program
$ASM - Enter Assembler
$LINK - Enter Linker

2-11. A user program can also be executed in an identical manner by entering a program filename. The filename must be a valid dataset (See Paragraph 1-21) and cannot contain imbedded blanks. A binary extension (BIN) or a blank extension which defaults to binary are the only allowed extensions. The file can reside on any supported disk unit (e.g. DK0, DK1). The following examples illustrate execution of user programs using the implied run command.

PROG1
DK1:PROG2.BIN
Upon entry into the user program, the DE register points to the next location (blank or carriage return) in the command buffer after the program name. Using the implied run command, a convenient facility is available for adding either new commands or user extensions to the Monitor.

2-12. COMMAND ENTRY. When entering a command from the terminal the command line may exceed the maximum terminal line length (usually 80 characters). If this occurs, the terminal output driver (TT) will automatically issue a CR and LF to enable continuation of the command on the next line. Since a carriage return input from the keyboard is interpreted by the Monitor to be the terminator of the command string, the user should not enter a carriage return until the entire command has been entered. The maximum command length is set by the command buffer size which is 160 characters.

2-13. DEFINITIONS.

1. DEFAULT TABLE - the default logical unit table. After power up or system reset a default logical unit table consisting of logical units 0 through 5 is created. This provides the user with 6 predefined I/O channels which can be used by application programs. The system subroutines RDCHR and WRCHR (see section 8) can be used for I/O transfers by specifying the logical unit in the E register. After power up or reset, logical unit 0 is always assigned the console input device (TK:) and logical unit 1 is assigned the console output device (TT:). Logical units 2-5 are initialized on power up or reset to values which are defined during the system SYSGEN procedure (See paragraph 15-12). At execution time the default table may be modified if a device is
opened after being redirected by the ASSIGN command (See paragraph 2-14). In this case system reset can be used to initialize the table.

2. REDIRECT TABLE - the logical unit redirect table. If the user wishes to change a logical unit device specification, he can redirect it to a new device using the Assign command. The redirect table consists of a list of all the currently redirected logical units.

3. BINARY FILE - A RAM-image file created by either the SAVE command or the Linker. A binary file generally contains executable machine code but may also contain data. A binary file has the extension BIN.

4. OBJECT FILE - a file created by the object output of either the Assembler or the DUMP command. The object module is in ASCII (See Mostek Object Format, Appendix B). The object module contains data and may also contain relocating and linking information for use by the Linker. An object file has the extension OBJ.

2-14. ASSIGN COMMAND

2-15. SYNTAX: ASSIGN N,Dataset

2-16. The ASSIGN command assigns a dataset to a logical unit number. This reassignment enables the user to change a logical unit device specification at run time. A dataset contains a device specification and a filename if the device is file structured. The logical unit number N is a hexadecimal number between 0 and FE (254 decimal). The ASSIGN command places the logical unit number and dataset into the Redirect Table. After an open
request (See IOCS Section 9) is executed, the assigned dataset is copied into the I/O vector being referenced. All future I/O transfers for the specified logical unit number use the newly assigned dataset.

EXAMPLE 1. Assign logical unit 2 to the paper tape reader device.

$ASSIGN 2,PR:(CR)

EXAMPLE 2. Assign logical unit 0 to a batch input file containing system commands (See Section 14 for batch mode operation).

$ASSIGN 0,DKO:BATCH.CMD(CR)

2-17. BEGIN COMMAND

2-18. SYNTAX: BEGIN [aaaa]

2-19. The BEGIN command starts execution of a previously loaded program. The hexadecimal address aaaa is the starting address which may be specified by the user. If this address is not specified, execution begins at the starting address of the previously loaded program. The program starting or execution address is stored in the user's PC (program counter) register (address FFFEH) after loading a program with the GET command. The BEGIN command also enables the "minimal listener" providing a console escape during program execution (See paragraph 2-45).

EXAMPLE 1. Begin program execution at location 0100H.

$BEGIN 100(CR)

2-20. CLEAR COMMAND

2-21. SYNTAX: CLEAR [N]

2-22. The CLEAR command removes logical unit N from the redirect
table. This cancels any previous reassignment of a logical unit made with the ASSIGN command. If N is not entered, all entries in the Redirect Table are removed.

EXAMPLE 1. Clear logical unit 3.
   $CLEAR 3(CR)

2-23. DDT COMMAND

2-24. SYNTAX: DDT

2-25. The DDT command transfers control to the DDT environment (See Section 7).

2-26. DTABLE COMMAND

2-27. SYNTAX: DTABLE

2-28. The DTABLE command lists the default logical unit table on the console output device. After power up or reset the default logical unit table consisting of logical units 0 through 5 is created. Logical unit 0 is always assigned the console input device (TK:) and logical unit 1 is assigned the console output device (TT:). Default values for logical units 2-5 are defined when the operating system is created using the SYSGEN procedure (See Paragraph 15-12).

EXAMPLE List default logical unit table.
   $DTABLE(CR)
   LU DATASET
   00 TKO:
   01 TTO:
   02 TKO:
   03 CPO:
   04 TKO:
   05 CPO:
2-29. DUMP COMMAND

2-30. SYNTAX: DUMP aaaa,bbbb,Dataset

2-31. The DUMP command outputs the contents of memory in absolute object format (See Appendix B) to the specified output dataset. The hexadecimal address aaaa is the beginning address and bbbb is the ending address of the data in memory. The addresses aaaa and bbbb can be terminated by a comma or a space and any number of spaces may be entered between command elements. The dataset specification can be any supported output device. If the dataset is an output file, the extension must be either OBJ or blank. If the extension is not entered (blank), the Monitor assumes OBJ.

EXAMPLE 1. Create the object file BINDEC which resides between locations 1000 and 1400, then dump it to paper tape.

$DUMP 1000, 1400, BINDEC(CR)
$PIP(CR)
#C BINDEC.OBJ TO PP:(CR)
#Q(CR)

2-32. GET COMMAND

2-33. SYNTAX: GET Dataset

2-34. The GET command loads a binary file specified by the dataset into memory. The program execution address is also loaded into the user's PC (program counter) register. This enables program execution to be initiated using the BEGIN command (See Section 2-17) without specifying the starting address. The execution address of a binary file is the first address or lowest program address in memory. The dataset extension must be either BIN or blank. If the extension is not entered (blank), the Mon-
itor assumes BIN.

EXAMPLE 1. Load the binary file BINDEC from disk unit DKO.
   $GET BINDEC(CR)

EXAMPLE 2. Load the binary file PROG22 from disk unit DK1 and begin execution at the starting address.
   $GET DK1:PROG22.BIN(CR)
   $BEGIN(CR)

2-35. INIT COMMAND

2-36. SYNTAX: INIT

2-37. THE INIT COMMAND MUST BE GIVEN ANYTIME A DISKETTE IS NEWLY INSERTED AND THE USER WISHES TO CONTINUE EXECUTING MONITOR COMMANDS. This guarantees that the proper sector and track maps are in memory during file operations on the newly inserted diskette. If the user fails to give this command, files on the newly inserted diskette may be irretrievably lost. During power up or reset the INIT command is automatically executed by the Monitor. The INIT command may also be given from the PIP environment (See Section 3).

2-38. RTABLE COMMAND

2-39. SYNTAX: RTABLE

2-40. The RTABLE command lists the logical unit redirect table on the console output device. The redirect table contains a list of all the currently redirected logical units.

EXAMPLE List redirected logical units.
   $RTABLE(CR)
   LU DATASET
   02 CRO:
   05 DK1:FILE22.MAC[1]
2-41. SAVE COMMAND

2-42. SYNTAX:  SAVE aaaa,bbbb,Dataset

2-43. The SAVE command outputs the contents of memory in a RAM image form to the disk file specified by the dataset. The hexadecimal address aaaa is the beginning address and bbbb is the ending address of the data in memory. The addresses aaaa and bbbb can be terminated by a comma or a space and any number of spaces may be entered between command elements. The dataset extension must be either BIN or blank. If the extension is not entered (blank), the Monitor assumes BIN.

EXAMPLE 1. Save the memory contents from 0 to 0100 by creating a binary file FILE1.BIN.

$SAVE 0,100,FILE1(CR)

EXAMPLE 2. Create the binary file BINDEC.BIN on disk unit 1.

$SAVE 1000,1400,D1:BINDEC.BIN(CR)

2-44. The SAVE command creates a binary file which can be up to 255 sectors in length. Each sector contains 124 bytes allowing a maximum file length of 31620 decimal or 7B84 hexadecimal bytes. When loading a binary file the GET command loads a fixed number of sectors into memory. A save block size (bbbbb-aaaaa) will not always equal an integral number of sectors. This can cause (worst case) up to 123 extra bytes to be loaded beyond the end address bbbb.

2-45. CONSOLE ESCAPE

2-46. The "Minimal Listener" is a background interrupt processor which detects the console input codes Control-X and Control-C. This provides the facility for a console exit from an executing
program to either the Monitor or DDT. The console escape can be a very useful tool during program debugging. The console input of Control-X suspends execution of a program and reboots the operating system returning control to the Monitor (prompt=$). A console input of Control-C suspends execution and enters DDT (prompt=.). DDT displays the program registers (similar to breakpoint) and execution can be resumed from DDT using the E command. (See Section 7-45).

2-47. The Minimal Listener is enabled only by the BEGIN and IMPLIED RUN commands (See paragraphs 2-9 and 2-17). It is disabled within the Monitor environment, and in the Editor and DDT.
SECTION 3

PERIPHERAL INTERCHANGE PROGRAM (PIP)

3-1. INTRODUCTION

3-2. The transferring of files and data between devices is the primary function of the Peripheral Interchange Program (PIP). PIP uses the device independent features of the I/O control system (IOCS), allowing data to be transferred from any system input device to any output device. In addition, PIP performs utility functions such as listing disk directories, renaming files, and formatting diskettes.

3-3. ENTERING PIP

3-4. The user can enter the PIP environment by typing the file name PIP as a command in the Monitor environment. The Monitor then loads the file PIP.BIN from disk unit DKO and starts its execution. The PIP prompting character is a #. To return to the Monitor the operator enters the QUIT command as illustrated in the following example.

EXAMPLE  $PIP(CR)  ;Enter PIP environment
          #Q(CR)  ;Return to Monitor

3-5. PIP COMMAND SYNTAX

3-6. Each PIP command contains a command name followed by a command operand field. The command names which are up to 6 characters in length denote the function to be performed. Only the first character of each name has to be entered to execute the selected function.

COMMAND NAMES

<table>
<thead>
<tr>
<th>APPEND</th>
<th>DIRECT</th>
<th>INIT</th>
<th>QUIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPY</td>
<td>ERASE</td>
<td>RENAME</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>FORMAT</td>
<td>STATUS</td>
<td></td>
</tr>
</tbody>
</table>
COMMAND SYNTAX

NAME Input Datasets(1...N) TO Output Dataset

3-7. The second part of each command is the command operand field which consists of a single dataset or a series of datasets depending upon the selected command. The keyword 'TO' has special significance in the command operand field. A dataset appearing to the right of 'TO' is defined as an output dataset. A dataset on the left of 'TO' is defined as an input dataset. There can be only one output dataset designation although there can be any number of input datasets (limited only by the command line length of 160 characters). The character '>' can be used in place of the keyword 'TO', performing the identical function.

3-8. A dataset can contain a single device (e.g. PR:) or a device, filename, extension and user number (e.g. DK1:FILE22.MAC [2]) if the device is file structured. The form of a dataset is described in paragraph 1-21. An asterisk can be used to replace the filename, extension or user number in an input dataset, but it is illegal in the output dataset. The asterisk specifies all occurrences of an element.

3-9. APPEND COMMAND

3-10. SYNTAX: APPEND Dataset 1 TO Dataset 2

3-11. The Append command attaches a copy of dataset 1 to the end of dataset 2. Dataset 1 remains unchanged. Both datasets must contain file structured devices (e.g.DK) and neither can be a binary file (Extension = BIN).

EXAMPLE

Append the file F1 on disk unit DK0 to the file F2 on DK0.

#APPEND_F1_TO_F2(CR)
3-12. COPY COMMAND

3-13. SYNTAX: COPY Dataset 2,....Dataset N TO Dataset 1

3-14. The Copy command can be used for a variety of purposes such as listing files, concatenating individual files, or copying all the files from one device (e.g. DKO) to a second device (e.g. DK1). The Copy command copies the contents of the input datasets (Datasets 2,...,N) to the output dataset (Dataset 1). If the file in the output dataset already exists, the following message appears on the console:

    Dataset, ALREADY EXISTS
    ERASE?

If the operator responds by entering a Y (followed by a carriage return) PIP deletes the file in the output dataset. The input datasets are then copied to the output dataset, assuming its name. No action is performed if a response other than Y is given. If a file specified in the input datasets does not exist, the following message is sent to the console:

    Dataset, NO SUCH FILE

3-15. The Copy command does not permit binary (extension = BIN) and non-binary file types to be mixed. If an attempt to copy a binary file to a source file is made, the error message INCOMPATIBLE EXTENSIONS is output to the console.

3-16. If a Copy is executed to a file-structured device with no filename (e.g.DK1), then the filename, extension and user number of the input dataset remains unchanged after transfer to the output device. However, if a filename is specified in an output dataset, the input datasets are concatenated and copied to the output file. In any case the file date of the output file will be the same as in the input file.

3-17. An asterisk can be used to replace the filename, ex-
tension, or user number in a Copy input dataset. The asterisk specifies all occurrences of an element. If an asterisk is specified in an input dataset, PIP automatically prints on the console each input file as it is copied. In order to illustrate the many possible uses of the Copy command, the following examples are given, classified according to output dataset types.

EXAMPLE 1. Copy to a non-file structured output device.

a. Transfer data from the paper tape reader to the paper tape punch. Input data from the paper tape reader is terminated by either an EOF mark of $04H$ or by 50 trailing nulls after the end of data.

   \#COPY PR: TO PP:(CR)

b. List the contents of FILE1 on DK1 to the line printer.

   \#C DK1:FILE1 TO LP:(CR)

EXAMPLE 2. Copy to a file structured device with no filename (e.g. DK1:).

a. Transfer the files F1, F2 and F3 from disk unit DKO to disk unit DK1.

   \#C F1,F2,F3 TO DK1:(CR)

b. Transfer all files from disk unit DKO to disk unit DK1. The diskette in DKO contains 5 files.

   \#C *.*[*] TO DK1:(CR)

   DKO:ASM .SRC[1]
   DKO:ASM .BIN[1]
   DKO:PIP .BIN[1]
   DKO:EDIT .SRC[1]
   DKO:EDIT .BIN[1]

c. Copy all the files with the extension SRC from user number 1 to user number 2.

   \#C *.SRC[1] TO DKO:[2] (CR)
EXAMPLE 3. Copy to a specified filename on a file structured device.

a. Copy FILEA.OBJ on DK1 to FILEB.OBJ on disk unit DKO.

```
#C DK1:FILEA.OBJ TO FILEB.OBJ(CR)
```

b. Concatenate the three source files F1, F2 and F3 and copy them to F123.

```
#C F1,F2,F3 TO F123(CR)
```

3-18. DATE COMMAND

**SYNTAX:** DATE

The DATE command is used to examine and/or modify the system's date. After entering the command, the date on the system will be printed if it exists and the following message will allow you to change it if desired:

```
ENTER DATE (DD-MMM-YY)
```

If only a carriage return is entered then the current system date is retained. Otherwise, type the day of the month first, then the first 3 letters of the month, and then the last 2 digits of the year with each item separated by a dash (-). This date will be stored in the directory of non-binary files when they are created or updated for reference by the user and will be displayed by a Directory command (see DIRECT).

3-19. DIRECT COMMAND

3-20. **SYNTAX:** DIRECT [Dataset 1 TO Dataset 2]
3-21. The DIRECT command is used to list the directory of disk devices. The input dataset (Dataset 1) is used to specify the disk unit (DK0, DK1 and etc.) for which the directory listing will be generated. If the input dataset is omitted, DK0 is assumed. If a filename, extension or user number is specified, only those files with the specified filename, extension and user number will be listed. An asterisk can replace a dataset element (e.g. Filename=*) to specify all or every occurrence of that element (e.g. All Filenames). The output dataset (Dataset 2) is optional and can be used to output the directory listing to any specified device. The default output device is the console.

3-22. The heading of the directory listing contains the disk unit (e.g. DK0) and the Diskette Name which were entered when the disk was formatted (See Paragraph 3-27). A file is identified in the directory by its filename, extension and user number. The directory listing also specifies the number of records used by each file and the starting track and sector location of the file, and the date of creation or last update.

To prevent information from being scrolled off the screen when listing large directories to a video terminal, the listing may be stopped by entering a space from the keyboard. The listing will resume when a second space is entered. The following examples illustrate the DIRECT command.

EXAMPLE 1. List entire directory of system disk on the console device.

```
#D(CR)
DIRECTORY DK0: DISKETTE BACK UP 1
FILENAME EXT USER RECORDS TRK SECT
PIP .BIN 1 25 09H 01H Listed on 8-MAR-79
BINDEC .SRC 1 5 0BH 04H Date 4-MAR-79
```
EXAMPLE 2. List all files of disk unit 1 with the extension OBJ on the line printer.

```
#D DK1:*.*OBJ[1] TO LP:(CR)
```

DIRECTORY DK1: DISKETTE BACK UP 2 On 15-Jun-79
FILENAME EXT USER RECORDS TRK SECT Date
FADD .OBJ 1 3 09H 01H 10-APR-79
FMUL .OBJ 1 3 09H 04H 1 -JUN-79

3-23. ERASE COMMAND

3-24. SYNTAX: ERASE Dataset 1 [, Dataset 2 ,...,Dataset N]

3-25. The Erase command removes the specified file or files from the disk unit and makes the space available for use. A filename must be entered for the ERASE command. The extension and user number if not entered will default to a blank extension and a user number of 1. After the ERASE command is entered, PIP will print the following message on the console:

```
ERASE?
```

If the operator responds by entering a Y (followed by a carriage return) PIP deletes the specified file or files. No action is performed if a response other than Y is given. If the file specified in the dataset does not exist, the following message is sent to the console:

```
Dataset, NO SUCH FILE
```

3-26. An asterisk can be used to replace the filename, extension or user number in the dataset to be erased. The asterisk specifies all occurrences of an element. The following examples
illustrate the ERASE command:
EXAMPLE 1. Erase the files F1 and F2 on the disk in DKO. Note the device defaults to DKO and the user number to 1.

```
#ERASE F1,F2(CR)
```

EXAMPLE 2. Erase an object file from DK1 with a user number of 3.

```
#ERASE DK1:F1.OBJ[3](CR)
```

EXAMPLE 3. ERASE all binary files (EXT=BIN) with a user number of 1 on DK1.

```
#ERASE DK1:*.BIN(CR)
```

EXAMPLE 4. Erase all files on disk DKO.

```
#ERASE *.*[*](CR)
```

3-27. FORMAT COMMAND

3-28. SYNTAX: FORMAT Name

3-29. The Format command formats each track and sector of a diskette in unit DK1 with the information necessary for proper accessing of data from the disk. The operand name used by the Format command gives each formatted disk an identifier for future reference. The name is eleven characters in length and can contain any printable characters. The DIRECT and STATUS commands output this name as a part of their headings to aid in referencing individual diskettes. After the FORMAT command is entered, PIP will print the following message on the console:

```
FORMAT?
```

If the operator responds by entering a Y (followed by a carriage return) PIP formats the diskette in unit DK1. No action is performed if a response other than Y is given.

3-30. To provide additional file protection, it is recommended
that each diskette be formatted with a unique name. The disk operating system prior to an Erase or Close operation verifies that the name of the diskette in a unit (DK0 or DK1) agrees with the name of the last previously initialized diskette in that unit. All disk units are initialized when entering PIP from the Monitor or after execution of the INIT command (See paragraph 3-34).

3-31. Formatting of a diskette initializes all sectors making them available for use (See STATUS paragraph 3-41). A disk must be formatted before it can be used the first time in the system. An unformatted diskette should not be inserted into the system until just prior to execution of the format command. A previously used diskette can be reformatted; however, any files on the diskette will be destroyed.

3-32. The format command requires that an operational system disk is resident in unit DK0. A system disk is defined as a previously formatted disk containing the required operating system programs. The diskette to be formatted is placed in disk unit 1. The system programs are automatically copied to the newly formatted diskette in DK1 during the execution of format.

3-33. The following examples illustrate the Format command:

EXAMPLE 1. Format the disk in unit DK1 giving it a name of BACK UP 1.

    #FORMAT BACK UP 1(CR)

EXAMPLE 2. Format a new disk and also copy the FLP-80DOS Assembler, Editor, Linker and PIP programs to the newly-formatted disk.

    #FORMAT SYS DISK 1(CR)
#C ASM.BIN, EDIT.BIN, LINK.BIN, PIP.BIN TO DK1:(CR)

NOTE: Using the above procedure the user can generate his own system disks containing only the system application programs (E.G. ASM and PIP) which he desires.

3-34. INIT COMMAND

3-35. SYNTAX: INIT

3-36. The Init command should be issued any time a new diskette is inserted and the user wishes to continue executing PIP commands. This guarantees that the proper sector and track maps are in memory during file operations on the newly inserted diskette. When entering PIP from the Monitor, the Init command is automatically executed by PIP.

3-37. RENAME COMMAND

3-38. SYNTAX: RENAME Dataset 1 TO Dataset 2

3-39. The Rename command is used to change the name of a specified file. The filename, extension and user number in Dataset 1 is changed to the filename, extension and user number in Dataset 2. If the file in the output dataset (Dataset 2) already exists, the following message appears on the console:

    Dataset, ALREADY EXISTS
    ERASE?

If the operator responds by entering a Y (followed by a carriage return) PIP deletes the file in Dataset 2. The file in Dataset 1 is then renamed to the name specified in Dataset 2. No action is performed if a response other than Y is given.
3-40. The RENAME command does not permit a binary extension (BIN) to be changed to a nonbinary extension or a nonbinary extension to be changed to a binary extension. The following examples illustrate the Rename command:

EXAMPLE 1. Rename the file FILE1 on disk unit DKO to FILE2.SRC.

#RENAME FILE1 TO FILE2.SRC(CR)

EXAMPLE 2. Rename the file FILEX1.OBJ on disk unit DK1.

#RENAME DK1:FILEX1.OBJ[1] TO DK1:FILEX2.OBJ[3](CR)

3-41. STATUS COMMAND

3-42. SYNTAX: STATUS [Dataset 1 TO Dataset 2]

3-43. The Status command is used to list the diskette name, the total number of sectors available, the number of sectors used and the number of bad sectors. The diskette name which identifies the individual disk is entered when the disk is formatted (See paragraph 3-27). The input dataset (Dataset 1) of the status command identifies the disk unit (DKO or DK1) for which status is desired. The output dataset is optional and can be used to output the status listing to any output device. The default is the console device. The following examples illustrate the STATUS command.

EXAMPLE 1. List the status of disk unit DK1 to the line printer.

#STATUS DK1: TO LP:(CR)
STATUS DK1: DISKETTE BACK UP 2
SECTORS AVAILABLE 1668
SECTORS USED 152
SECTORS BAD 0

EXAMPLE 2. List the status of disk unit DKO. Note if the input dataset is not specified it defaults to DKO.
The diskette name is 'BACK UP 1'

**CR**

STATUS DKO: DISKETTE BACK UP 1
SECTORS AVAILABLE   1020
SECTORS USED         800
SECTORS BAD          0

3-44. QUIT COMMAND

3-45. SYNTAX: QUIT

3-46. The Quit command exits PIP and returns control to the FLP-80DOS Monitor.
SECTION 4

FLP-80DOS TEXT EDITOR (EDIT)

4-1. INTRODUCTION

4-2. The FLP-80DOS Text Editor assists the user in origination and modification of assembly language source programs and English text documentation. The Editor resides on the FLP-80DOS System Diskette. It permits random access editing of ASCII diskette files. The Editor is designed for usage with the MOSTEK FLP-80 system, but it can be adapted to other systems for OEM uses.

4-3. CAPABILITIES

4-5. The FLP-80DOS Text Editor permits random access editing of ASCII diskette files on a line and character basis. Whole lines and character strings embedded within lines can be accessed, changed, deleted, or added to an existing or new diskette file. The size of the file to be edited is limited only by diskette capacity. All I/O operations to the diskette are transparent to the user.

4-5. SOFTWARE CONFIGURATION

4-6. The Editor is resident on diskette. When loaded, it starts at RAM address zero. Figure 4-1 shows the memory map for the Editor. Editor buffers and variables are placed in RAM between the top of the Editor and bottom of the Flexible Disk Handler.

4-7. The Editor uses Logical Unit Numbers 0 and 1 for console interaction and Logical Unit Number 5 for outputting records with
Logical Unit Number 5 is typically assigned to a line printer device. All I/O to the disk is via LUN FFH, which cannot be reassigned via the Monitor 'ASSIGN' command. Figure 4-2 depicts this structure.

4-8. DEFINITIONS

1. SOURCE - ASCII characters comprising a Z80 assembly language program or some other text.

2. RECORD - A single source statement ending with a carriage return.

3. FILE - A diskette file which contains the source.

4. POINTER - the position in the source where the next action of the Editor will be initiated.

5. CURRENT RECORD - the record in the source pointed to by the pointer.

6. RECORD NUMBER - the decimal number of a record, beginning at one (0001) for the first record in a file and increasing sequentially for each record.

7. INSERT - Installation of record(s) in a file immediately following the current record. Inserted records are assigned sequentially increasing line numbers.

8. DELETE - removal of the current record from a file.
FIGURE 4-1. EDITOR MEMORY MAP

- Flexible Disk Handler
- I/O Buffers for IOCS
- Editor Buffers & Variables
- FLP-80 DOS Editor
FIGURE 4-2. LOGICAL UNIT NUMBER STRUCTURE

LUN 0  
CONSOLE INTERACTION  
LUN 1

FLP-80 DOS TEXT EDITOR

LUN FF_H  
FLEXIBLE DISK FILE

LUN 5
SOURCE WITH LINE NUMBERS
4-9. USING THE TEXT EDITOR - CONSOLE INTERACTION

4-10. All user interaction with the EDITOR is via the console device. The Editor issues prompts and messages to direct the user. The user responds by entering commands or data via the console keyboard. Each command or data record is terminated by a carriage return. The user can modify a record before depressing carriage return with the following console keys:

1. **DEL: RUBOUT (ASCII 7FH).** Delete the previous character. Successive characters may be deleted by entering more than one 'rubout'. The characters which are deleted will be printed on the console device between two backslash characters (\).

2. **CNTL-H: BACKSPACE (ASCII 08H).** Performs the same function as RUBOUT, but the backslash is not printed on the console device.

3. **CNTL-U: NEGATIVE ACKNOWLEDGE (ASCII 15H).** Deletes the current line of entered information and reprompts the user for a new record of input.
4-11. USING THE TEXT EDITOR - ENTERING COMMANDS

4-12. When the Editor prompts for a command (>), the user may enter commands via the console. Modification of the input is allowed with RUBOUT, BACKSPACE, and CNTL-U functions. All commands can be entered in lower case as well as upper case. Multiple commands may be entered on one line if they are separated from each other by blanks or commas. A command line is terminated by a carriage return. A command line may have up to 80 characters in it, including carriage return.

EXAMPLE

> I(CR)  
  - insert mode command

> B I(CR)  
  - backup and insert

> b i(CR)  
  - backup and insert

> L10(CR)  
  - go to line number 10.

> L 10, I(CR)  
  - go to line 10 and insert.

Several commands allow an operand n to be entered with the command. The operand may be a decimal number in the range 0-9999. It may be entered immediately following the command or separated from the command by one or more blanks or commas.

EXAMPLE

> L 10(CR)
> L10(CR)  
  - go to line number 10.
Alternatively, the operand may be two decimal numbers separated by a minus sign. In this case, the line number specified by the first number is accessed, then the operation is performed from that line through and including the line specified by the second number. If the first number is greater than the second number, then an error prompt is printed and the command is not done.

EXAMPLE: `V10-20(CR)`
- verify lines numbered 10 through 20 on the user console.

4-13. USING THE TEXT EDITOR - FIRST STEPS

4-14. The FLP-80DOS Text Editor is executed by the following monitor command:

```
$EDIT filename(CR)
```
- where filename is the name of the disk file to be edited.

The Editor responds with the following message:

```
FLP-80DOS EDITOR V2.1
```

If the user does not enter the filename with the EDIT command, then the Editor requests it:

```
ENTER FILE NAME TO BE EDITED>
```

The user then types in the name of the file to be edited. If the file does not exist, then a new one with that name is created.

EXAMPLE: `$EDIT DK1:MYFILE(CR)`

EXAMPLE: `$EDIT NEWFIL.SRC(CR)`
- defaults to device DKO:

EXAMPLE: `$EDIT(CR)`
```
ENTER FILE NAME TO BE EDITED>NEWFILE(CR)
```
The only restriction on the file name is that it cannot have extension 'BIN' or extension 'TMP'. Further, files with extension 'OBJ' are reserved for object files.

If the file does not exist, then the Editor outputs the following message:

-->NEW FILE
0001<

- Editor prompts for insert records (see "INSERT COMMAND").

At the end of Editing, the new file will automatically be created. If the file does exist on disk, then editing of that file will be done. The Editor prompts for a command:

> 

- Editor prompts for a command. See list of commands.

4-15. USING THE TEXT EDITOR - BASIC COMMANDS

4-16. I - INSERT
FORMAT:  >I(CR)
         or
         >i(CR)

This command is used to insert records following the current record or to build new files.

The Editor responds with:

-->INSERT MODE
The user then enters records ending with carriage returns. After each record which is inserted, the Editor reprompts with the next line number. To terminate the insertions, the user enters a sin-
Single carriage return. Note that blank lines must be entered as 'space, carriage return' because a single carriage return terminates the insert mode. If an unprintable character is entered, than a warning message is printed on the console. After the user terminates the insert mode, the Editor prompts for a new command (>).

**EXAMPLE**  
\[>I(CR)\]  
- user selects insert mode.
  
\[-->INSERT MODE\]  
- Editor prompts user.
  
\[0002<THIS IS AN INSERTED LINE (CR)\]  
- user enters record to be inserted.
  
\[0003<(CR)\]  
- user terminates insert mode.
  
\[>\]  
- Editor prompts for another command.

Note that modification of entered records can be done with RUB-OUT, BACKSPACE, and CNTL-U. Inserted records are automatically assigned sequential record numbers. Inserted records can be up to 160 characters long, including the carriage return.

4-17. **An - ADVANCE**

4-18. This command is used to advance the record pointer a specified number of records.

**Format:**  
\[or > An(CR)\]  
\[> an(CR)\]

If n is zero or if n is omitted, the pointer will be positioned to the next record in the file. The record which is accessed is printed on the console after this command.
EXAMPLE > A5(CR) - advance record pointer 5 records.
0015 ANY STATEMENT - the new current record of the file is printed on the console device by the Editor.

EXAMPLE > A(CR) - advance to next record.
0016 NEXT STATEMENT - the next record in the file is printed.

4-19. Bn - BACKUP
FORMAT: or > Bn(CR)
> bn(CR)

This command is used to backup the record pointer a specified number of records.

If n is zero or if n is omitted, then the pointer is position to the previous record in the file. The record which is accessed is printed on the console after this command.

EXAMPLE > B3(CR) - backup record pointer 3 records.
0012 SOME STATEMENT - the new current record of the file is printed on the console device by the Editor.

EXAMPLE > B(CR) - backup to previous record.
0011 A STATEMENT - the previous record in the file is printed.

4-20. Dn - DELETE
FORMAT: or > Dn(CR)
> dn(CR)

This command deletes the specified number of records from the file starting with the current record.
If the constant \( n \) is not entered or if \( n \) is equal to zero, only the current record will be deleted.

**EXAMPLE**

\[ \text{D5(CR)} \] - the current record and the following 4 records will be deleted from the file.

**EXAMPLE**

\[ \text{D(CR)} \] - only the current record will be deleted from the file.

4-21. \( \text{Ln} \) - GO TO RECORD NUMBER \( n \)

**FORMAT:**

\[ \text{or } > \text{Ln(CR)} \]

\[ > \text{Ln(CR)} \]

This command positions the pointer to the record numbered \( n \).

The constant \( n \) must be entered and it must be greater than zero. The record which is accessed is printed on the console device.

**EXAMPLE**

\[ > \text{L10(CR)} \]

0010 LINE NUMBERED 10.

If the record number cannot be found because it is larger than the last record number in the file, then the pointer will be positioned at the last record of the file.

**EXAMPLE**

\[ > \text{L2001(CR)} \]

\[ -->\text{EOF} \]

0943 LAST LINE OF FILE

4-22. \( \text{Vn} \) - VERIFY

**FORMAT:**

\[ \text{or } > \text{Vn(CR)} \]

\[ > \text{vn(CR)} \]
This command prints the specified number of records on the console device. The record pointer is updated to the last record printed. If \( n \) is zero or if \( n \) is not entered, one record (the current record) is printed on the console. Unprintable characters are printed as dots (.) to identify them.

**EXAMPLE**

\[
\begin{align*}
&> \text{V2(CR)}
\end{align*}
\]

0005 CURRENT STATEMENT
0006 NEXT STATEMENT
- two records are verified, i.e., printed on the console device. The current record is number 6.

4-23. TEXT EDITOR ADVANCED COMMANDS

4-24. \( \text{Cn } /\text{string1}/\text{string2}/\) - CHANGE

**FORMAT:**

\[
\begin{align*}
&> \text{Cn } /\text{string 1}/\text{string 2}/(CR)
\end{align*}
\]

or

\[
\begin{align*}
&> \text{cn } /\text{string 1}/\text{string 2}/(CR)
\end{align*}
\]

where \( n \) indicates the number of occurrences to change, \( \text{string 1} \) represents the characters to be changed, \( \text{string 2} \) represents the substitute or new characters, and \( / \) represents a delimiter character which does not appear in either string.

This command changes the next \( n \) occurrences of character \( \text{string 1} \) to \( \text{string 2} \) starting with the current record. Any character which does not appear in either \( \text{string 1} \) or \( \text{string 2} \) may be used as a delimiter. All three delimiters must be identical. If \( n \) is zero or if \( n \) is not entered, then only one occurrence of \( \text{string 1} \) is changed. Each record which is changed will be printed on the console device. If \( \text{string 2} \) is not entered, then \( \text{string 1} \) will be deleted when it is found. The record pointer will be positioned at the record of the last occurrence of the change. If \( n \)
is one or is not entered, then only the current record will be searched for string 1. If string 1 is not present, then a question mark prompt will be printed and the record pointer will remain at the same record:

?>

For n greater than 1, if string 1 is not found before the end of the file, then an end-of-file warning message is printed on the console and the pointer will be positioned at the last record in the file.

**EXAMPLE**

```plaintext
> V(CR)
0010 THIS IS A RECORD.
> C /THIS/THAT/(CR)
0010 THAT IS A RECORD.
> C /IS/WAS/(CR)
0010 THAT WAS A RECORD.
> C /WAS A /(CR)
0010 THAT RECORD.
> C2 /T/V/(CR)
0010 VHAV Record.
```

**EXAMPLE**

```plaintext
> C2/XENON/ARGON/(CR)
--> EOF
```

- The string 'XENON' cannot be found by the Editor.

4-25. *En - Exchange*

**FORMAT:**  

```
> En (CR)
```

or  

```
> en (CR)
```

This command exchanges the specified number of records (starting with the current record) with records to be inserted. It is exactly equivalent to the command sequence:
>Dn (CR) - delete n records.
>B1 (CR) - back up one record.
>I (CR) -->INSERT MODE - enter insert mode.

4-26. Fn - PRINT FLAG OPTION
FORMAT: >F0 (CR) - n=0, inhibit printing after all but
or >f0 (CR) the 'Vn-VERIFY' command.
>Fn (CR) - n not=0, allow printing after all
or access commands.

The Editor normally prints on the console device any record which
is accessed or changed. Thus, the following commands print out a
record: An, Bn, Cn, Ln, Sn, Vn. In order to reduce print out
time on a slower console device (such as a teleytype), this
command can be used to inhibit print out on all of the commands
except Vn - VERIFY.

4-27. G dataset - GET RECORDS FROM DATASET
FORMAT: >G dataset (CR)
or >g dataset (CR)

The command inputs records from a dataset (which must be a disk
file) and inserts then in sequence after the current record. A
carriage return must follow the dataset specification.

EXAMPLE > G FILEX (CR)
-get records from FILEX in DK0: and insert them
after the current record in the file being edited.
4-28. Mn - MACRO

> M1(CR)

or> m1(CR)

> M2(CR)

or> m2(CR)

This command allows a command string to be entered into one of two alternate command buffers (labeled '1' and '2'). The alternate command buffers will accept character strings of 80 characters or less. The Editor responds with the following prompt:

EXAMPLE > M1 (CR)

1>S/OLD/ D1 B1 (CR)

- The user enters into alternate command buffer 1 the commands which:

1. Search for the 1st occurrence of the string 'OLD', starting with the next record.

2. delete that record.

3. backup one record.

4-29. Pn dataset - PUT N RECORDS TO DATASET

FORMAT: > Pn dataset (CR)

or > pn dataset (CR)

This command outputs the specified number of records (starting with the current record) to a dataset which must be a disk file. If n is zero or n is not entered, then only the current record is output. The records which are output are not deleted. If the file being
output to exists, it will be erased before any records are written to it. This command may be used with the G(GET) command to move records around in a file. A carriage return must follow the dataset specification.

EXAMPLE

>P25 XFILE (CR)
- output the next 25 lines in the file being edited to a new file named XFILE on DKO:
>P100-125DK1:FILE1(CR)
- output lines 100 through 125 from the file being edited to file DK1:FILE1.

4-30. Sn /source image/ - SEARCH

FORMAT:  
> Sn /source image/ (CR)  
or> sn /source image/ (CR)

where n is the number of the occurrence, source image represents any set of characters which is to be search for, and / represents a delimiter character which does not appear in the string.

This command searches the file, starting with the next record, for the nth occurrence of the character string between the delimiters. The pointer is then positioned at the record in which the string is found. This command always searches forward in the file. Any character which does not exist in the source image may be used as delimiter. Both the starting and terminating delimiters must be identical. If n is zero or n is not entered, then the first occurrence of the source image will be sought. The record in which the source image is found will be printed on the console. If the string is not encountered before the end of the file, then an end-of-file warning is printed on the console device and the pointer will be positioned at the last record in the file.
EXAMPLE

> S /ORD/ (CR)

0023 SOME RECORD DATA
- Editor searches forward for the character string 'ORD', finds the 1st occurrence, and prints the record on the console.

EXAMPLE

> S10 /9AH/ (CR).

-->EOF

0048 LAST RECORD
- Editor could not find the tenth occurrence of the string '9AH'. A warning is printed indicating end-of-file and the last record in the file is printed.

4-31. T - INSERT AT TOP

FORMAT: >T(CR)

or >t(CR)

This command inserts records at the top of the file before the first record. See the 'I - INSERT' command for proper usage.

4-32. Wn - WRITE

FORMAT: >Wn (CR)

or >wn (CR)

This command performs the same function as the VERIFY command, except that output is directed to LUN 5 which is typically assigned to a line printer device via the following monitor command before the Editor is used:

$ASSIGN 5,LP:(CR)

4-33. Xn - EXECUTE

> X1 (CR)

or > x1 (CR)

> X2 (CR)

or > x2 (CR)
This command executes the commands stored in the alternate command buffer numbered 1 or 2. After an alternate command buffer has been executed, control is returned to the Editor which prints a prompt for a new command (>). The alternate command buffer is not destroyed during the operation. If n is equal to zero or is not entered, then alternate command buffer 1 is selected.

EXAMPLE

    > M1 (CR)
    > S /OLD/ D1 B1 (CR)
    > X1 (CR)

0010 FIRST OCCURRENCE OF OLD.
   - 'OLD' is located and the record is deleted.
0009 LINE NUMBER 9.
   - Backup command prints its record.

NOTE The pseudo-macro command capability is executed by the 'M' and 'X' commands. The user puts his macro command string into alternate buffer 1 or 2 and executes that macro string via the 'X' command.

4-34. EDITING LARGE FILES

4-35. Editing of large files is no different than editing small files. All commands are fully functional. However, diskette access may be required for certain operations and a delay may be apparent before the Editor responds.

4-36. EDITOR MESSAGES

4-37. If the user enters an unrecognizable file name, a syntax error will be indicated and the Editor will reprompt for another file name.

EXAMPLE ENTER FILE NAME TO BE EDITED>LAST=1(CR)
4-19

*****SYNTAX ERROR
ENTER FILE NAME TO EDITED>

4-38. If the user enters an unrecognizable command, then the Editor will print a question mark and another prompt.
EXAMPLE > R20 (CR)
?>

If the user enters the same name for a put file as the name of the file being edited during a PUT command, the Editor will print: -->USE DIFFERENT FILE NAME FOR PUT and it will reprompt for a new command: ?>

4-39. All I/O errors to and from disk result in termination of the Editor with an appropriate error message. The original file should be backed up on another diskette before using the Editor.

4-40. The Editor prompts the user with several messages to the console device.
--> NEW FILE
- indicates that a new file is being created rather than editing of an old file.
--> INSERT MODE
- indicates that records of data are to be entered rather than Editor commands.
--> TOF
- indicates that the top of file (beginning of file) has been encountered.
--> END OF EDITING
- indicates that the Editor has successfully completed. Control is then returned to the FLP-80DOS Monitor.
--> PLEASE WAIT.
- indicates that a long disk access is taking place.
--> END OF WINDOW. USE 'ADVANCE' TO SEE NEXT RECORD.
- occurs only with VERIFY command. Follow the directions.
--IS THE OUTPUT DEVICE READY? (Y/N)
- occurs after the issue of a W command to alert the user that the I/O device assigned to LUN 5 must be configured to his system.

--THERE MAY NOT BE ENOUGH SPACE IN DISK TO EDIT YOUR FILE. DO YOU WISH TO CONTINUE? (Y/N)
- occurs only if at the start of the editing session the free space on the diskette unit of the input file is not at least equal to 125% of the size of the input file. It serves as a warning against the possible loss of that file because of a disk-full error. (Error OB).

4-41. SAMPLE EDITING SESSION

4-42. The user is urged to follow the steps given here to become acquainted with the FLP-80DOS Editor.

$EDIT NEWONE(CR)
- user selects to use FLP-80DOS Editor.

(There will be a slight delay while the Editor is read into RAM from disk.)

FLP-80DOS EDITOR V2.1
- user selects to create a new file on DK0: (disk unit zero), with file name 'NEWONE' and no extension.

--> NEW FILE
--> INSERT MODE

0001 < TITLE ECHO PROGRAM (CR)
- Editor prompts for records to be input via the console. User begins keying in a program.

0002< ; THIS PROGRAM READS A CHARACTER (CR)
0003< ; FROM THE CONSOLE AND ECHOS IT.(CR)
0004< ; CNTL-U RETURNS CONTROL TO THE MONITOR.(CR)
0005< ; (CR)
- user terminates insert mode operation

> B99V20 (CR)
- user goes to beginning of file and verifies 20 records in the file.

--> EOF
- Editor shows that end of file has been encountered.

> L8 (CR)
0008 LOP CALL RDCHR ; READ A CHARACTER
- user verifies line 8 and observes an error.

> C /LOP/LOOP/(CR)
0008 LOOP CALL RDCHR ; READ A CHARACTER
- user modifies line.

> S /7A00/(CR)
0010 JP Z,7A00H ; IF SO, RETURN TO MONITOR
- user searches for the string 7A00.

> C /7A00H/REBOOT/(CR)
0010 JP,Z REBOOT ; IF SO, RETURN TO MONITOR
- user changes the record.

> Q (CR)
- user terminates editing session. The new file will now be on disk unit 0 (DK0) with file name NEWONE.
## TABLE 4-1. SUMMARY OF FLP-80 EDITOR COMMANDS

<table>
<thead>
<tr>
<th>CONSOLE INTERACTION</th>
<th>COMMAND PROMPT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKSPACE</td>
<td>INSERT PROMPT</td>
<td>&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;</td>
<td></td>
</tr>
<tr>
<td>CNTL-U</td>
<td>MESSAGE IDENTIFIER</td>
<td>--&gt;</td>
</tr>
</tbody>
</table>

- **An**  - Advance n records.
- **Bn**  - Backup n records.
- **Cn /string1/string2/**  - Change n occurrences of string 1 to string 2.
- **Dn**  - Delete n records, starting with current record.
- **En**  - Exchange n records with inserted records.
- **Fn**  - Flag print option: 0 = no print, not 0 = print.
- **G dataset**  - Get records from dataset and insert them after current record.
- **I**  - Insert records after current record.
- **Ln**  - Line: Access record number n.
- **Mn**  - Macro: Place command string into alternate command buffer 1 or 2.
- **Pn dataset**  - Put n records out to dataset.
- **Q**  - Quit: Save the file on disk and terminate the editor.
- **Sn /string/**  - Search for nth occurrence of the string.
- **T**  - Top: Insert at top of file before the first record.
- **Vn**  - Verify n records on the console device.
- **Wn**  - Write n records with record numbers to LUN 5.
- **Xn**  - Execute alternate command buffer n (1 or 2).

In all commands, except Fn and Ln, if n is zero or if n is not entered, it is assumed to equal one (1). The operand n may be entered as n1 - n2 which performs the operation on lines n1 through n2.
SECTION 5

FLP-80DOS ASSEMBLER (ASM)

5-1. INTRODUCTION

5-2. The Mostek FLP-80DOS Assembler is provided on flexible diskette. In conjunction with the resident Text Editor and the Linker it provides the means for editing, assembling, and linking Z80 programs. The Assembler reads Z80 source mnemonics and pseudo-ops and outputs an assembly listing and object code. The object code is in industry standard hexadecimal format modified for relocatable, linkable assemblies.

5-3. The Assembler recognizes all standard Z80 source mnemonics. It supports conditional assemblies, global symbols, relocatable programs, and a printed symbol and cross reference table. The Assembler can assemble any length program, limited only by the symbol table size (which is based on available RAM) and available disk space. In a 16K RAM system, the Assembler supports a symbol table size of about 150 symbols. In a 32K RAM system, the size is over 700 symbols.

5-4. Figure 5-2 shows the Assembler with typical device usage. The source module is read from a disk file, the object output is directed to a disk file, and the assembly listing is directed to a line printer. User interaction is via the console device. Note that the Assembler can interact with any dataset.

5-5. DEFINITIONS

1. SOURCE MODULE - the user's source program. Each source module is assembled into one object module by the Assembler. The end of a source module is defined by an EOT
character (04H) on input or an 'END' pseudo-op.

2. OBJECT MODULE - the object output of the Assembler for one source module. The object module contains linking information, address and relocating information, machine code, and checksum information for use by the MOSTEK Linker. The object module is in ASCII. A complete definition of the MOSTEK object format is in Appendix B. The object module is typically output to a disk file with extension 'OBJ'.

3. LOAD MODULE - the binary machine code of one complete program. The load module is defined in RAM as an executable program or on disk as a binary file (extension 'BIN'). It is created by the MOSTEK Linker from one or more object modules (extension 'OBJ').

4. LOCAL SYMBOL - a symbol in a source module which appears in the label field of a source statement.

5. INTERNAL SYMBOL - a symbol in a source (and object) module which is to be made known to all other modules which are loaded with it by the Linker. An internal symbol is also called global, defined, public, or common. Internal symbols are defined by the GLOBAL pseudo-op. An internal symbol must appear in the label field of the same source module. Internal symbols are assumed to be addresses, not constants, and they will be relocated by the Linker.

6. EXTERNAL SYMBOL - a symbol which is used in a source module but which does not appear in the label field of a statement. External symbols are defined by the GLOBAL pseudo-op. External symbols may not appear in an expression which uses operators. An external symbol is a reference to a symbol that exists and is defined as internal in another program module.

7. GLOBAL DEFINITION - both internal and external symbols are defined as "GLOBAL" in a source module. The Assembler determines which are internal and which are external.

8. POSITION INDEPENDENT - a program which can be placed anywhere in memory. It does not require relocating informa-
tion in the object module.

9. ABSOLUTE - a program which has no relocation information in the object module. An absolute program which is not position independent can be loaded only in one place in memory in order to work properly.

10. RELOCATABLE - a program which has extra information in the object module which allows the Linker to place the program anywhere in memory.

11. LINKABLE - a program which has extra information in the object module which defines internal and external symbols. The Linker uses the information to connect, resolve or link, external references to internal symbols.

5-9. ASSEMBLY LANGUAGE SYNTAX

5-10. An assembly language program (source module) consists of labels, opcodes, pseudo-ops, operands, and comments in a sequence which defines the user's program. The assembly language conventions are described below.

5-11. DELIMITERS. Labels, opcodes, operands, and pseudo-ops must be separated from each other by one or more commas, spaces, or tab characters (ASCII 09H). The label may be separated from the opcode by a colon, only, if desired.

5-12. LABELS. A label is composed of one or more characters. If more than 6 characters are used for the label, only the first 6 are recognized by the Assembler. The characters in the label cannot include ' ( ) * + , - 1 = . / : / < > or space. In addition, the first character cannot be a number (0-9). Table 5-1 summarizes the allowed characters in a label or symbol. A label can start in any column if immediately followed by a colon (:). It does not require a colon if started in column one.
FIGURE 5-1. ASSEMBLER MEMORY MAP

- Flexible Disk Handler
- I/O Buffers for I/OCS
- Assembler Symbol Table
- FLP-80 DOS Assembler
- Assembler Variables and Stack
FIGURE 5-2. LOGICAL UNIT NUMBER STRUCTURE
EXAMPLE allowed
LAB
L923
$25
ACCOUNT_PAYABLE

A25E: not allowed

9LAB ; STARTS WITH A NUMBER
L)AB ; ILLEGAL CHARACTER IN LABEL
L:ABC ; ILLEGAL CHARACTER IN LABEL

5-13. OPCODES. There are 74 generic opcodes (such as 'LD'), 25 operand key words (such as 'A'), and 693 legitimate combinations of opcodes and operands in the Z80 instruction set. The full set of these opcodes is documented in the "Z80 CPU TECHNICAL MANUAL" and listed in Appendix A of this manual. The FLP-80DOS Assembler allows one other opcode which is not explicitly shown in the Z80 CPU Technical Manual:

IN F,(C) ; SET THE CONDITION BITS ACCORDING
; TO THE CONTENTS OF THE PORT DEFINED BY THE
C-REGISTER

5-14. PSEUDO-OPS. Pseudo-ops are used to define assembly time parameters. Pseudo-ops appear like Z80 op-codes in the source module. Several pseudo-ops require a label. The following pseudo-ops are recognized by the Assembler:

1. ORG nn -origin - sets the program counter to the value of the expression nn. Each origin statement in a program must be greater than the first origin of the program to assure proper program link-
2. label EQU nn

-equate - sets the value of a label to nn in the program, where nn is an expression; can occur only once for any label.

3. label DEFL nn

-define label - sets the value of a label to nn in the program, where nn is an expression. This may be repeated in the program with different values for the same label. At any point in the program, the label assumes the last previously defined value.

4. DEFM 'aa'

-define message - defines the contents of successive bytes of memory to be the ASCII equivalent code of characters within quotes. Maximum length of the message is 63 characters. The delimiting quote characters are required. A quote character may be placed in a message by a sequence of two quotes (').

5. DEFB n,n,n...

-define byte - defines the contents of bytes located at the current program counter address to be n, where n is any expression.

6. DEFW nn,nn,nn,...-define word - defines the contents of two-byte words to be the value of any expression nn. The least significant byte is located at the current program counter address. The most significant byte is located at the program counter address plus one.
7. **DEFS** nn

-define storage - reserves nn bytes of memory starting at the current program counter, where nn is an expression. When loaded, these bytes are not overwritten, i.e., they will contain what was previously in memory. This pseudo-op cannot be used at the end of a program to reserve storage.

8. **END** nn

-end statement - defines the last line of the program. The 'END' statement is not required. The expression nn is optional and represents the transfer address (starting execution address) of the program. The transfer address defaults to the first address of the program. Note that for binary files the transfer address must be the same as the starting address of the program.

9. **GLOBAL** symbol

-define global symbol - any symbol which is to be made known among several separately assembled modules must appear in this type of statement. The Assembler determines if the symbol is internal (defined as a label in the program), or external (used in the program but not defined as a label).

10. **NAME** symbol

-module name -This pseudo-op defines the name of the program (source and object). The name is placed in the heading of the assembly listing and is placed in the first record of the object module to identify it. This pseudo-op is designed primarily to
TABLE 5-1. ALLOWED CHARACTERS

<table>
<thead>
<tr>
<th>MSD</th>
<th>0000</th>
<th>0001</th>
<th>0100</th>
<th>0110</th>
<th>1000</th>
<th>1001</th>
<th>1100</th>
<th>1110</th>
<th>1111</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NUL</td>
<td>DLE</td>
<td>SPACE</td>
<td>0</td>
<td>@</td>
<td>P</td>
<td>\</td>
<td>p</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SOH</td>
<td>DC1</td>
<td>!</td>
<td>1</td>
<td>A</td>
<td>Q</td>
<td>a</td>
<td>q</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>STX</td>
<td>DC2</td>
<td>&quot;</td>
<td>2</td>
<td>B</td>
<td>R</td>
<td>b</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ETX</td>
<td>DC3</td>
<td>#</td>
<td>3</td>
<td>C</td>
<td>S</td>
<td>c</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>EOT</td>
<td>DC4</td>
<td>$</td>
<td>4</td>
<td>D</td>
<td>T</td>
<td>d</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ENO</td>
<td>NAK</td>
<td>%</td>
<td>5</td>
<td>E</td>
<td>U</td>
<td>e</td>
<td>u</td>
<td></td>
</tr>
</tbody>
</table>

6  | ACK  | SYN  | &    | 6    | F    | V    | f    | v    |      |
7  | BEL  | ETB  | '    | 7    | G    | W    | g    | ,    | w    |
8  | BS   | CAN  | (    | 8    | H    | X    | h    | x    |      |
9  | HT   | EM   | )    | 9    | I    | Y    | i    | y    |      |
A   | LF   | SUB  | *    |      | J    | Z    | j    | z    |      |
B   | VT   | ESC  | +    |      | K    | (    | k    | (    |      |
C   | FF   | FS   | ,    |      | L    | \   | l    | |     |
D   | CR   | GS   | -    |      | M    | )    | m    | )    |      |
E   | SO   | RS   | >    |      | N    | \   | n    | -    |      |
F   | SI   | US   | ?    |      | O    | _   | o    | DEL  |

**NOT ALLOWED**

**ADDITIONAL CHARACTERS NOT ALLOWED AS FIRST CHARACTER**
facilitate future compiler design. The name of the module defaults to 6 blanks.

11. PSECT op

-program section - This pseudo-op may appear only once at the start of a source module. It defines the program module attributes for the following operands:

   REL - relocatable program (defaults).
   ABS - absolute program. No relocating information is generated in the object module by the Assembler. The module will be loaded where it is origined.

12. IF nn

-conditional assembly - If the expression nn is true (non-zero), the IF pseudo-op is ignored. If the expression is false (zero), the assembly of subsequent statements is disabled. 'IF' pseudo-ops cannot be nested.

13. ENDIF

-end of conditional assembly - re-enables assembly of subsequent statements.

14. COND nn

-same function as IF pseudo-op.

15. ENDC

-same function as ENDIF pseudo-op.

16. INCLUDE dataset-include source from another dataset - allows source statements from another dataset to be included within the body of the given program. The file is searched for first on DKO:; then on DKL:; If the dataset cannot be opened properly, then assembly is aborted.
The source module must not end with an 'END' pseudo-op (otherwise, assembly would be terminated). The source module must end with an EOT character (04), which is true for all FLP-80DOS ASCII datasets. The INCLUDE pseudo-op cannot be nested, but it can be chained. The means that an included dataset can have an INCLUDE pseudo-op at the end of it. At the end of the last included dataset, assembly continues in the original module.

Note: The INCLUDE pseudo-op cannot be followed by a comment on the same line.

LIST - turn listing on.
NLIST - turn listing off.
EJECT - eject a page of listing.
TITLE S - print title 'S' at top of each page of listing.
    'S' may be up to 32 characters long.

5-15. OPERAND. There may be zero, one, or more operands in a statement depending on the opcode or pseudo-op used. Operands in the Assembler may take the following forms:

5-16. GENERIC OPERAND. Such as the letter 'A', which stands for the Accumulator. Table 5-2 summarizes these operands and their meanings.

5-17. Constant. The constant must be in the range 0 through OFFFFH. It can be in the following forms:

1. Decimal - this is the default mode of the Assembler. Any number may be denoted as decimal by following it with the letter 'D'. E.g., 35, 249D.
2. Hexadecimal - must begin with a number (0-9) and end with the letter 'H'. E.g., 0AF1H.

3. Octal - must end with the letter 'O' or 'Q'. E.g., 377Q, 2770.

4. Binary - must end with the letter 'B'. E.g., 0110111B.

5. ASCII - letters enclosed in quote marks will be converted to their ASCII equivalent value. E.g., 'A' = 41H.

5-18. A LABEL which appears elsewhere in the program. Note that labels cannot be defined by labels which have not yet appeared in the user program (this is an inherent limitation of a two-pass assembler).

    not allowed
L EQU H
H EQU I

I EQU 7
allowed
I EQU 7
H EQU I
L EQU H
<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A register (accumulator)</td>
</tr>
<tr>
<td>B</td>
<td>B register</td>
</tr>
<tr>
<td>C</td>
<td>C register</td>
</tr>
<tr>
<td>D</td>
<td>D register</td>
</tr>
<tr>
<td>E</td>
<td>E register</td>
</tr>
<tr>
<td>F</td>
<td>F register</td>
</tr>
<tr>
<td>H</td>
<td>H register</td>
</tr>
<tr>
<td>L</td>
<td>L register</td>
</tr>
<tr>
<td>AF</td>
<td>AF register pair</td>
</tr>
<tr>
<td>AF'</td>
<td>AF' register pair</td>
</tr>
<tr>
<td>BC</td>
<td>BC register pair</td>
</tr>
<tr>
<td>DE</td>
<td>DE register pair</td>
</tr>
<tr>
<td>HL</td>
<td>HL register pair</td>
</tr>
<tr>
<td>SP</td>
<td>SP Stack Pointer register</td>
</tr>
<tr>
<td>$</td>
<td>Program Counter</td>
</tr>
<tr>
<td>I</td>
<td>I register (interrupt vector MS byte)</td>
</tr>
<tr>
<td>R</td>
<td>Refresh register</td>
</tr>
<tr>
<td>IX</td>
<td>IX index register</td>
</tr>
<tr>
<td>IY</td>
<td>IY index register</td>
</tr>
<tr>
<td>NZ</td>
<td>Not zero</td>
</tr>
<tr>
<td>Z</td>
<td>Zero</td>
</tr>
<tr>
<td>NC</td>
<td>Not Carry</td>
</tr>
<tr>
<td>C</td>
<td>Carry</td>
</tr>
<tr>
<td>PO</td>
<td>Parity odd/not overflow</td>
</tr>
<tr>
<td>PE</td>
<td>Parity even/overflow</td>
</tr>
<tr>
<td>P</td>
<td>Sign positive</td>
</tr>
<tr>
<td>M</td>
<td>Sign negative</td>
</tr>
</tbody>
</table>
5-19. AN EXPRESSION—the MOSTEK FLP-8000S Assembler accepts a wide range of expressions in the operand field of a statement. All expressions are evaluated left to right constrained by the hierarchies shown in Table 5-3. Parentheses may be used to ensure correct expression evaluation. Table 5-3 shows the allowed operators and their hierarchies. The symbol '$' is used to represent the value of the program counter of the current instruction. Note that enclosing an expression wholly in parentheses indicates a memory address. The contents of the memory address equivalent to the expression value will be used as the operand value. Integer two's complement arithmetic is used throughout. The negative (2's complement) of an expression or quantity may be formed by preceding it with a minus sign. The one's complement of an expression may be formed by preceding it with the '.NOT.' operator.

In doing relative addressing, the current value of the program counter must be subtracted from the label if a branch is to be made to that label address.

EXAMPLE:

JR LOOP-$

...will jump relative to 'LOOP'.

The allowed range of an expression depends on the context of its use. An error message will be generated if this range is exceeded during its evaluation. In general, the limits on the range of an expression are 0 through OFFFH. The limits on the range of a relative jump ('JR' or 'DJNZ') are -126 bytes and +129 bytes. The Assembler monitors the number of items in an expression. If an expression is too long, an error message will be output. This limit will probably never be reached by a typical program. For relocatable programs, the Assembler will output relocation information in the object module for those addresses which are to be relocated by the Linker. Expressions are determined to be relocatable addresses or non-relocatable constants
according to the following rules:

(constant) (operation) (constant) = (constant)
(constant) (operation) (relocatable) = (relocatable)
(relocatable) (operation) (constant) = (relocatable)
(relocatable) (operation) (relocatable) = (constant)

EXAMPLE
I EQU 1 ;CONSTANT DEFINITION
DEFW I ;CONSTANT WHICH WILL NOT BE RELOCATED
LAB EQU $ ;RELOCATABLE DEFINITION
  
  
  
JP LAB ;RELOCATABLE OPERAND
JR LAB-$ ;CONSTANT OPERAND
JR +5+(I) ;CONSTANT OPERAND

For a further discussion of relocatable values, see paragraph 5-27.

5-20. COMMENTS. A comment is defined as any characters following a semicolon in a line. A semicolon which appears in quotes in an operand is treated as an expression rather than a comment starter. Comments are ignored by the Assembler, but they are printed in the assembly listing. Comments can begin in any column. Note also that the Assembler ignores any statements which have an asterisk (*) in column one.
TABLE 5-3. ALLOWED OPERATORS AND HIERARCHIES IN FLP-80DOS ASSEMBLER

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>.RES.</td>
<td>-reset overflow. Anytime the .RES. operator is found, the overflow indicator will be unconditionally reset after the expression is evaluated. This can be used to prevent overflow errors in certain arithmetic expressions.</td>
<td>0</td>
</tr>
<tr>
<td>Unary plus</td>
<td>(+)</td>
<td>1</td>
</tr>
<tr>
<td>Unary minus</td>
<td>(-) (2's complement)</td>
<td>1</td>
</tr>
<tr>
<td>Logical NOT</td>
<td>(.NOT.) (1's complement)</td>
<td>1</td>
</tr>
<tr>
<td>Multiplication</td>
<td>(*)</td>
<td>2</td>
</tr>
<tr>
<td>Division</td>
<td>(/)</td>
<td>2</td>
</tr>
<tr>
<td>Addition</td>
<td>(+)</td>
<td>3</td>
</tr>
<tr>
<td>Subtraction</td>
<td>(-)</td>
<td>3</td>
</tr>
<tr>
<td>Logical AND</td>
<td>(.AND.)</td>
<td>4</td>
</tr>
<tr>
<td>Logical OR</td>
<td>(.OR.)</td>
<td>4</td>
</tr>
<tr>
<td>Logical XOR</td>
<td>(.XOR.)</td>
<td>4</td>
</tr>
<tr>
<td>Logical shift right</td>
<td>(.SHR.)</td>
<td>4</td>
</tr>
<tr>
<td>Logical shift left</td>
<td>(.SHL.)</td>
<td>4</td>
</tr>
<tr>
<td>Shift right 8</td>
<td>(.)</td>
<td>4</td>
</tr>
</tbody>
</table>

The shift operators (.SHR. and SHL.) shift their first argument right or left by the number of bit positions given in their second argument. Zeros are shifted into the high-order or low-order bits respectively. The dot operator (.) may be placed at the end of an expression. Its effect is to shift a 16 bit value right by 8 bits so the most significant byte can be accessed. Zeros are shifted into the higher order bits.
5-21. **OBJECT OUTPUT**

5-22. The object module of the Assembler can be loaded by an Intel hexadecimal loader for non-linkable programs. Extra information is inserted into the object module for linkable and relocatable programs for using the MOSTEK Linker. For a complete discussion of the object format, see Appendix B.

5-23. **ASSEMBLY LISTING OUTPUT**

5-24. The user must insert tabs in the source to obtain columns in the assembly listing. The value of each equated symbol will be printed with a pointer (> ) next to it. Any address which is relocatable will be identified with a quote ( ' ) character. The statement number and page number are printed in decimal. Listing control pseudo-ops do not appear in the listing but they are assigned statement numbers. If the listing option is not selected, errors will be output to the console device.

5-25. **ABSOLUTE MODULE RULES**

5-26. The pseudo-op 'PSECT ABS' defines a module to be absolute. The program will be loaded in the exact addresses at which it is assembled. This is useful for constants, a common block of global symbols, or a software driver whose position must be known. This method can also be used to define a list of global constants.

```assembly
EXAMPLE

PSECT ABS
GLOBAL AA
AA EQU 0
GLOBAL AB

;ABSOLUTE ASSEMBLY
```
All symbols in the above module will assume constant values which may be used by any other program.

5-27. RELOCATABLE MODULE RULES

5-28. The following rules apply to relocatable programs.
   1. Programs default to relocatable if the 'PSECT ABS' pseudo-op is not used or if 'PSECT REL' is specified.
   2. Only those values which are 16-bit address values will be relocated. 16-bit constants will not be relocated (internal symbols are exceptions).

   **EXAMPLE**
   ```
   AA EQU OA13H ;ABSOLUTE VALUE
   LD A,(AA) ;AA NOT RELOCATED
   AR EQU $ ;RELOCATABLE VALUE
   LD A,(AR) ;AR WILL BE RELOCATED UPON LOADING
   ```

5-29. Relocatable quantities may not be used as 8-bit operands. This restriction exists because only 16-bit operands are relocated by the Linker.

   **EXAMPLE**
   ```
   LAB EQU $ ;RELOCATABLE DEFINITION
   DEFB LAB ;NOT ALLOWED
   LD A,LAB ;NOT ALLOWED
   LD A,(LAB) ;ALLOWED
   LD HL,LAB ;ALLOWED
   ```
5-30. Labels equated to labels which are constants will be treated as constants. Labels equated to labels which are relocatable values will relocated. Internal symbols are exceptions.

```
EXAMPLE  B8  EQU  20H   ;ABSOLUTE VALUE
    C8  EQU  B8       ;ABSOLUTE VALUE
      LD  A,(C8)   ;C8 WILL NOT BE RELOCATED
    AR  EQU  $       ;RELOCATABLE VALUE
    BR  EQU  AR      ;RELOCATABLE VALUE
      LD  A,(BR)   ;BR WILL BE RELOCATED
```

5-31. Internal symbols will always be marked relocatable. This point is important because an internal symbol will be relocated even though it looks like a constant. This point is discussed further, below.

5-32. External symbols will always be marked relocatable, except for the first usage in the program.

5-33. GLOBAL SYMBOL HANDLING

5-34. A global symbol is a symbol which is known by more than one module. A global symbol has its value defined in one module. It can be used by that module and any other module. A global symbol is defined as such by the GLOBAL pseudo-op. For example:

```
GLOBAL SYM1
```

- SYM1 is a symbol which is defined as "global".

An internal symbol is one which is defined as global and also appears in the label field of a statement in the same program.
EXAMPLE  GLOBAL SYM1
CALL    SYM1
.
.
.
END

-SYM1 is an external symbol

EXAMPLE
GLOBAL  SYM1
SYM1    EQU  $
LD      A,(SYM1)
.
.
.
.
END

-SYM1 is an internal symbol. Its value is the address of the LD instruction.

If these two programs were linked by the MOSTEK Linker, all global symbol references would be "resolved". This means that each address in which an external symbol was used would be modified to the value of the corresponding internal symbol. The loaded programs would be equivalent (using our example) to one program written as follows.

EXAMPLE  CALL    SYM1
.
.
.
.
SYM1    EQU  $
5-35. Global symbols are used to allow large programs to be broken up into smaller modules. The smaller modules are used to ease programming, facilitate changes or allow programming by different members of the same team. The Assembler has several rules which apply to global symbols. The examples in the following paragraphs should be studied carefully.

5-36. GLOBAL SYMBOL BASIC RULES. Both passes of the Assembler must be done in their entirety if global symbols are used. This restriction exists because symbols are defined as global during pass 1, and an external reference link list is built up during pass 2.

1. Global symbols follow the same syntax rules as labels. They may not start with a number (0-9) or a restricted character. They may not contain restricted characters.

   EXAMPLE allowed
   
   GLOBAL SYM1
   GLOBAL A&&
   GLOBAL $BB

   not allowed
   
   GLOBAL 1AB ;STARTS WITH A NUMBER
   GLOBAL A=B ;CONTAINS A RESTRICTED CHARACTER

2. An external symbol may not appear in an expression.

   EXAMPLE GLOBAL SYM1 ;EXTERNAL SYMBOL
   CALL SYM1 ;OK
LD HL, (SYM1) ;OK
LD HL, SYM1+25H ;NOT ALLOWED
JP SYM1+2 ;NOT ALLOWED

3. An external symbol is always considered to be a 16-bit address. Therefore, an external symbol may not appear in an instruction requiring an 8-bit operand. It may not be used for a displacement or an 8-bit constant.

EXAMPLE
GLOBAL SYM1 ;EXTERNAL SYMBOL
CALL SYM1 ;OK
LD A, (SYM1) ;OK
LD A, SYM1 ;NOT ALLOWED
LD (IX+SYM1), A ;NOT ALLOWED
BIT SYM1, A ;NOT ALLOWED

4. In relocatable assembly, a global symbol is always considered to be a relocatable 16-bit address. This applies to both internal and external symbols. It does not apply to absolute assemblies (PSECT ABS).

5. By definition, an external symbol cannot also be an internal symbol.

6. For a set of modules to be linked, no duplication of internal symbol names is allowed. That is, an internal symbol can be defined only once in a set of modules to be linked together.

5-37. GLOBAL SYMBOL ADVANCED RULES.

1. An external symbol cannot appear in the operand field of a 'EQU' or 'DEFL' pseudo-op. Thus, an external symbol must be explicitly defined as global.

EXAMPLE
GLOBAL SYM1 ;EXTERNAL SYMBOL
SYM2 EQU SYM1 ;NOT ALLOWED
SYM3 DEFL SYM1 ;NOT ALLOWED
2. All references to an external symbol are marked relocatable, except the first reference in a program. The object code for these references is actually a backward link list, terminating in the constant $0FFF_{16}$. (See definition of object format in Appendix B) (This rule does not apply to absolute assemblies).

3. An internal symbol is always marked relocatable, except for absolute assemblies. This point is important, because an internal symbol will be relocated even though it looks like a constant.

**EXAMPLE**
```
PSECT REL ; RELOCATABLE MODULE
GLOBAL YY ; INTERNAL SYMBOL
YY EQU OAF3H ; YY WILL ALWAYS BE MARKED RELOCATABLE
LD A,(YY) ; YY WILL BE RELOCATED WHEN LOADED.
```

; THE ABOVE INSTRUCTION LOADS THE CONTENTS OF THE ADDRESS YY, RELOCATED, INTO THE A-REGISTER.

**EXAMPLE**
```
PSECT ABS ; ABSOLUTE ASSEMBLY
GLOBAL YY ; INTERNAL SYMBOL
YY EQU OAF3H ; YY IS AN ABSOLUTE VALUE
LD A,(YY) ; THIS LOADS THE CONTENTS OF ADDRESS OAF3H INTO THE A-REGISTER
```

4. All other rules that apply to local symbols also apply to internal symbols.

5-38. USE OF THE "NAME" PSEUDO-OP.

5-39. The NAME pseudo-op can be used to identify both a source module and an object module. The name of the module being assembled can be assigned by the NAME pseudo-op. The name is placed in the heading of the assembly listing. The name is also placed in the first record of the object module. The first record is the module definition record (record type 05), and it is described in Appendix B. The name of a module follows the same rules as a local symbol.
5-40. USING THE ASSEMBLER

5-41. The FLP-80DOS Assembler is resident on the FLP-80DOS system flexible diskette. The user first prepares his source modules using the FLP-80DOS Editor. Then the source file may be assembled. The command to invoke the Assembler is:

```bash
$ASM dataset1 [TO datasetL [,datasetO]](CR)
```

where
- `dataset1` = source input dataset.
- `datasetL` = assembly listing output dataset (optional).
- `datasetO` = object output dataset (optional).

The Assembler can interact with any dataset. `dataset1` must be a disk file. `datasetL` and a `datasetO` are optional in the command. `datasetL` defaults to the same unit and filename as `dataset1` with an extension of 'LST'; `datasetO` defaults to the same unit and filename as `dataset1` with an extension of 'OBJ'. `datasetL` and `datasetO` can be specified in the command. If `datasetO` is a disk file, it must have an extension of 'OBJ' or a blank extension which defaults to 'OBJ'. `dataset1` and `datasetL` may not have the following extensions: OBJ, BIN, or CRS. The Assembler then outputs the following message to the console output device:

```
MOSTEK FLP-80DOS ASSEMBLER V2.1. OPTIONS?
```

Options are described in paragraph 5-67. If no options are to be entered, the user enters "carriage return". The Assembler then reads the source module for pass 1. During pass 1, the symbol table and external references are defined. The name of the module is defined, and the external symbol link list is built. At the end of reading, the source dataset is rewound, and the following message is printed on the console device:

```
PASS 1 DONE
```

The Assembler proceeds into pass 2 automatically. During pass 2,
the assembly listing and object module are output. At the end of pass 2, the following message is output on the console output device:

ERRORS = nnnn

where nnnn is the total number of errors (in decimal) which were found by the Assembler. Control is then returned to the FLP-80DOS Monitor.

5-42. ASSEMBLER OPTIONS

5-43. The Assembler allows the user to select the following options from the console. When the Assembler outputs the message:

MOSTEK FLP-80DOS ASSEMBLER V2.1. OPTIONS?

The user may enter any of the following codes. A carriage return terminates the options. Normal editing of a line is allowed.

C-Cross Reference Listing. This option prints a symbol cross reference table at the end of the assembly listing.
K-No listing. This suppresses the assembly listing output. All errors will be output to the console device.
L-Listing (default). The assembly listing is normally output.
N-No object output. This suppresses object output from the Assembler.
O-Object output (default). The object output is normally output.
P-Pass 2 only. This selects and runs only pass 2 of the Assembler.
Q-Quit. This returns control to the FLP-80DOS Monitor.
R-Reset the symbol table. This option clears the symbol table of all previous symbol references. This operation is automatically done for pass 1. It is used primarily for single pass operations (described in paragraph 5-78).
S-Symbol table. The symbol table is normally not output by the Assembler. This option prints a symbol table at the end of the assembly listing.

EXAMPLE

```
OPTIONS? NS(CR)
```

- the user has selected no object output and a printed symbol table.

5-44. ERROR MESSAGES

5-45. Any error which is found is denoted in the assembly listing. A message is printed immediately after the statement which is in error. Appendix E defines all Assembler error codes and messages.

EXAMPLE

```
H2: LC A,B
*****ERROR 41 INVALID_OPCODE
```

Several errors abort the Assembler when they are encountered. These are noted in Appendix E. Abort error messages are output only to the console output device. Control is immediately returned to the FLP-80DOS Monitor. Abort errors may occur during pass 1 or pass 2.

5-46. ADVANCED OPERATIONS

5-47. PASS 2 ONLY OPERATION (SINGLE PASS OPERATION). The FLP-80DOS Assembler can be used as a single pass assembler under the following restrictions:

1. No GLOBAL symbols are defined.
2. No forward symbol references occur.
3. The NAME pseudo-op is not in the source.

The Assembler will correctly assemble Z80 programs under the
above restrictions during pass 2. This is useful for assembling data tables and certain types of programs. The Assembler symbol table should be initialized to assure proper operation in this mode. This may be done by using the 'R' option to reset the symbol table prior to assembling using pass 2 only as follows:

$ASM MYFILE(CR)
MOSTEK FLP-80 ASSEMBLER V2.1. Options? PR(CR)
-user selects pass 2 only operation and resets the symbol table prior to assembly.

.

.

.

The symbol table initialization described above only has to be done after power up and after symbols are left in the table from a previous assembly.

5-49. ASSEMBLING SEVERAL SOURCE MODULES TOGETHER. Several source modules may be assembled together to form one object module. The 'INCLUDE' pseudo-op may be used several times in one module to properly sequence a set of source modules.

EXAMPLE

NAME MYFILE ;name of final object module
INCLUDE FILE1
INCLUDE FILE2
INCLUDE FILE3
END

-the object module named 'MYFILE' will be built by the assembly of FILE1 + FILE2 + FILE3.

5-50. SAMPLE ASSEMBLY SESSION
5-51. Assume that the file to be assembled is named PROG1. The diskette on which PROG1 exists is in disk unit 1 (DK1). The object output of the Assembler is to be directed to file PROG1.OBJ on disk unit 1. The assembly listing is to be directed to a line printer (LP:). A printed symbol table is to be obtained. The following sequence will perform the assembly:

EXAMPLE

$ASM DK1:PROG1 TO LP: (CR)
MOSTEK FLP-80 ASSEMBLER V2.1. OPTIONS? S(CR)
   -user selects a printed symbol table.
   
   

ERROR = 0000
   - indication of zero assembly errors
$
   - indication that assembly is done, and control is returned to the Monitor.
SECTION 6

LINKER

6-1. INTRODUCTION

6-2. The Linker program provides the capability for linking object files together and creating a binary (EXT=BIN) or RAM image file. The Linker concatenates modules together and resolves global symbol references which provide communication between modules. A starting link address may be entered to position a linked module anywhere in the memory map. The Monitor GET or Implied Run command can be used to load binary files allowing fast access of linked modules.

6-3. LINKER COMMAND

6-4. SYNTAX: LINK Dataset 1,..... Dataset N TO Dataset B [,Dataset C](CR)

6-5. The input datasets (Dataset 1,.....Dataset N) are object files produced by either the Assembler or the Monitor DUMP command. The object files must be on a supported disk unit (e.g. DKO or DK1). In the Linker command the object input datasets must have an extension of OBJ or blank. If a blank extension is entered the Linker will assume an extension of OBJ. Dataset B is the binary output file which is created by the linker. Specification of Dataset B by the user is optional. If Dataset B is not specified it automatically defaults to a file having an extension of BIN and a filename of Dataset 1 which is the first input dataset. If Dataset B is specified it must be on a supported disk unit (e.g. DKO, DK1) and must have an extension of BIN or blank. If a blank extension is entered, the Linker will assume an extension of BIN. Dataset C is the output file for
6-2

the global cross reference table and symbol table when the C and S options are specified (See Paragraph 6-9 and 6-11). Dataset C can be any supported output device (e.g. LP:, TT:). Specification of Dataset C is optional. If Dataset C is not specified it automatically defaults to a file having the extension of CRS and the filename of Dataset B.

6-6. When entering the Linker command if a large number of input datasets are specified the command line may exceed the maximum terminal line length (usually 80 characters). If this occurs, the terminal output driver (TT) will automatically issue a CR and LF to enable continuation of the command on the next line. Since a carriage return input from the keyboard is interpreted by the Linker to be the terminator of the command string, the user should not enter a carriage return until the entire Linker command has been entered. The maximum length of the Linker command string is 160 characters, however, the library search option (See Paragraph 6-10) may be used if the user wishes to link additional datasets.

6-7. After a valid command is entered the Linker outputs the following message on the console.

OPTIONS?
The user can then enter any of the supported Linker options (A,C,L,U,S). A carriage return terminates the options list.

6-8. A OPTION. The A option enables the user to enter a starting link address. After the A option is entered the following message is output to the console.

ENTER STARTING LINK ADDRESS >
The user may then specify the starting link address for the first object module. The beginning load address of the first relocatable module is the starting link address plus the module starting address defined by the Assembler ORG pseudo-op. If the
ORG pseudo-op is omitted or its address is 0, then the starting link address equals the beginning load address. If an object module is absolute the A option is ignored and the module is always loaded at its starting address as defined by the ORG pseudo-op. The PSECT pseudo-op of the Assembler defines a module as either relocatable or absolute. If the A option is not specified the Linker assumes a starting link address of 0. The beginning and ending address of each module is printed on the console by the Linker during Pass 2.

6-9. C OPTION. The C option causes the global cross reference table (See Figure 6-1) to be generated and output to the device specified in Dataset C. The global cross reference table contains the symbol name, definition address and reference addresses. A global symbol can be defined only once but can be referenced many times. A symbol is defined by a module if it occurs in the label field of the module and is specified by the GLOBAL pseudo-op. A global symbol is referenced within a module when it occurs in the operand field. When the C option is specified a load map is also output which specifies the object input files linked and their beginning and ending addresses.

6-10. L OPTION. The L option enables the user to perform a library search for undefined global symbols. If any symbols are undefined after linking the input datasets (Dataset 1, Dataset 2, ..., Dataset N) during Pass 1, the Linker prints out the number of undefined symbols. (The U option prints out a list of undefined symbols.) If the L option has been selected the Linker prints the following message on the console.

SEARCH DISK UNIT 1/0?

The user may then initiate a library search by entering a 1 or 0 followed by a carriage return. Any other response terminates the search and Pass 2 execution is started. If a library search has been requested the Linker searches the disk unit specified for
an object file having the filename of the first undefined symbol. If the file is found, it is linked into the binary output file and any global references which are defined are resolved. This process is repeated for each undefined symbol in the original list. After the search has been completed for the first list of symbols, the sequence can be repeated for a new list if any symbols remain undefined. After the original list has been searched more undefined symbols might actually exist if a file from the previous list contains additional undefined symbols. Each time the search is repeated either disk unit may be searched. Disks should not be removed or inserted between library searches. The library search option may be used to minimize the number of input files that must be typed in the Link command. This can be done by giving an object file the same name as a global symbol definition within the module.

6-11. S OPTION. The S option causes the global symbol table (See Figure 6-1) to be generated and outputted to the device specified in Dataset C. The global symbol table contains the symbol name and definition address. A symbol is defined by a module if it occurs in the label field of the module and is specified by the GLOBAL pseudo-op. If a global symbol is referenced but not defined it is marked undefined (UNDEF=****). A global symbol is referenced within a module when it occurs in the operand field. When the S option is specified a load map is also output which specifies the object input files linked and their beginning and ending addresses.

6-12. U OPTION. The U option prints out a list of undefined global symbols after the Linker has completed Pass 1.

6-13. LINKER OPERATION

6-14. During Pass 1 the Linker reads the specified object files
and places the global symbol definitions in the symbol table. In Pass 2 the global symbols are defined and a binary or ramimage output file is produced. As each object module is read in Pass 2 its beginning and ending address in memory is printed on the console. The module type is also listed as either absolute or relocatable (ABS/REL). Absolute modules are always positioned at their starting address in memory as defined by the ORG pseudo-op. Relocatable modules are positioned at the next location after the end address of the previous module. If the first input module is relocatable, it is positioned by the starting link address (See Para. 6-8). If the starting link address is not specified by the A option it assumes a value of 0.

6-15. LINKER RESTRICTIONS

6-16. When absolute modules are being linked together, the files in the LINK command must appear in sequential order according to their starting addresses in memory. If an absolute module is encountered having a starting address lower in memory than a previous module the following error message is printed on the console.

****ERROR 35 MODULE SEQUENCE ERROR

The maximum size allowed for an individual object input module is limited by the linker buffer size which is dynamically allocated depending upon the size of the memory. On the standard system having 32K of RAM, it is 18K bytes in length and on the minimum system having 16K of RAM it is 4.5K bytes. There is no restriction on the length of the binary output file.

When loading a binary file using the Monitor GET or Implied Run commands the entire memory space is available except for 48 bytes in scratchpad RAM starting at OFF60H. This space is reserved for the Monitor I/O vector and cannot be overlayed during a load sequence.
6-17. EXAMPLES OF LINK COMMAND

EXAMPLE 1. Link the relocatable object modules MAIN1.OBJ, SUB1.OBJ, SUB2.OBJ, and SUB3.OBJ together starting at 2000H and produce the binary file TEST.BIN. Also generate a symbol table, cross reference table and load map and store them in the file TEST.CRS. This file may be printed using the PIP copy command (See Figure 6-1).

$\textbf{LINK MAIN1, SUB1, SUB2, SUB3 TO TEST(CR)}$

\begin{verbatim}
OPTIONS? A C S(CR)
ENTER STARTING LINK ADDRESS 2000
DKO:MAINI .OBJ[1]
DKO:SUB1 .OBJ[1]
DKO:SUB2 .OBJ[1]
DKO:SUB3 .OBJ[1]
UNDEFINED SYMBOLS 00
PASS 2
DKO:SUB1 .OBJ[1] REL BEG ADDR 2034 END ADDR 20DB
DKO:SUB2 .OBJ[1] REL BEG ADDR 20DC END ADDR 20F6
DKO:SUB3 .OBJ[1] REL BEG ADDR 20F7 END ADDR 2120

$
\end{verbatim}

EXAMPLE 2. Link the absolute file MAIN.OBJ and the relocatable subroutines SUB1.OBJ, SUB2.OBJ, SUB3.OBJ together producing the binary file MAIN.BIN. Access the object files DKO:SUB1.OBJ, DKO:SUB2.OBJ and DK1:SUB3.OBJ using the library search option.

$\textbf{LINK MAIN (CR)}$

\begin{verbatim}
OPTIONS? L U (CR)
DKO:MAIN .OBJ[1]
MODNO MSGBEG MSGEND MSGMAI PRINT
SUB1 SUB2 SUB3
UNDEFINED SYMBOLS 08
\end{verbatim}
SEARCH DISK UNIT 1/0 ? 0 (CR)
DKO:SUB1 .OBJ[1]
DKO:SUB2 .OBJ[1]
MODNO SUB3
UNDEFINED SYMBOLS 02
SEARCH DISK UNIT 1/0 ? 1 (CR)
DK1:SUB3 .OBJ[1]
UNDEFINED SYMBOLS 00
PASS 2
DKO:MAIN .OBJ[1] ABS BEG ADDR 1000 END ADDR 1025
DKO:SUB1 .OBJ[1] REL BEG ADDR 1026 END ADDR 10CD
DKO:SUB2 .OBJ[1] REL BEG ADDR 10CE END ADDR 10E8
DK1:SUB3 .OBJ[1] REL BEG ADDR 10E9 END ADDR 1115
FIGURE 6-1. EXAMPLES OF LOAD MAP, GLOBAL CROSS REFERENCE, AND GLOBAL SYMBOL TABLE

LOAD MAP

DKO:SUB1 .OBJ[1] REL BEG ADDR 2034 END ADDR 20DB
DKO:SUB2 .OBJ[1] REL BEG ADDR 20DC END ADDR 20F6
DKO:SUB3 .OBJ[1] REL BEG ADDR 20F7 END ADDR 2120

GLOBAL CROSS REFERENCE TABLE

SYMBOL ADDR REFERENCES
CRLF 2030 211A 20F4
MAIN 2000
MODNO 2109 20E2 20DF 203A 2037 2011 200E
MSGBEG 204D 2006
MSGEND 2073 2023
MSGMAI 2098 2014
MSGMOD 2000 210F
MSGSB2 20A3 20E5
MSGSB3 20A9 2100
PRINT 20EE 2103 204A 2040 2026 2017 2009
PTEST 2046 2106 20EB
SUB1 2034 201A
SUB123 211D
SUB2 20DC 201D
SUB3 20F7 2020

GLOBAL SYMBOL TABLE

CRLF 2030 MAIN 2000 MODNO 2109 MSGBEG 204D
MSGEND 2073 MSGMAI 2098 MSGMOD 20D0 MSGSB2 20A3
MSGSB3 20A9 PRINT 20EE PTEST 2046 SUB1 2034
SUB123 211D SUB2 20DC SUB3 20F7
SECTION 7

DDT-80 DEBUG SYSTEM

7-1. INTRODUCTION

7-2. This section describes the functions and operation of DDT-80 (Designer's Development Tool 80) resident in the FLP-80DOS system. The DDT software provides a complete facility for interactively debugging relative and absolute Z80 programs. Standard commands allow displaying and modifying memory and CPU registers, setting breakpoints, and executing programs. Additional commands allow use of the MOSTEK AIM-80 to interactively debug a target system. Mnemonics are used to represent Z80 registers, thus simplifying the command language.

7-3. SOFTWARE CONFIGURATION

7-4. DDT-80 is a program that resides in PROM (located from E000H to EFFFH) on the SDB-80 board. In addition to the PROM, DDT uses 256x8 of RAM for scratch RAM and temporary storage. This RAM resides at locations FFOOH - FFFFH.

7-5. The 256x8 Scratchpad RAM is used by the DDT for temporary storage and a push down stack (for return address, etc.). This RAM also holds an image (or map) of all the user's internal CPU registers. Figure 7-1 is a detailed memory map of the 256x8 Scratchpad RAM.

7-6. An important concept in DDT is preservation of the user's internal CPU registers. The state of the CPU is described by the contents of the registers. To preserve the state of the CPU for a user's program while debugging, DDT keeps an image or map of all the user's registers. This image or map is referred to as the
User Register Map throughout this documentation. DDT installs or makes the CPU registers equal to the user register map when control is transferred from DDT to a user program (as in the E command discussed in paragraph 7-45). DDT-80 saves the user register map when DDT is commanded (breakpoint command discussed in paragraph 7-34) to interrupt a user program. DDT allows modification to this register map with the display and/or update memory command (M command, discussed in paragraph 7-57). The user register map resides in the 256x8 Scratchpad, locations FFE6H thru FFFFH, as shown in Figure 7-1. Figure 7-2 shows the data paths between the user register map and the CPU registers. Also shown is the modification path between DDT and the User Register Map.
FIGURE 7-1. DDT USER REGISTER MAP

<table>
<thead>
<tr>
<th>MEMORY LOCATION</th>
<th>USER REGISTER</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FFFF</td>
<td>PC</td>
<td>PROGRAM</td>
<td>MSB</td>
</tr>
<tr>
<td>FFFE</td>
<td>COUNTER</td>
<td></td>
<td>LSB</td>
</tr>
<tr>
<td>FFFD</td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFFC</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFFB</td>
<td>I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFFA</td>
<td>IF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFF9</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFF8</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFF7</td>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFF6</td>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFF5</td>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFF4</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFF3</td>
<td>A'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFF2</td>
<td>F'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFF1</td>
<td>B'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFF0</td>
<td>C'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFEF</td>
<td>D'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFEE</td>
<td>E'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFEED</td>
<td>H'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFECE</td>
<td>L'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFE8</td>
<td>IX</td>
<td></td>
<td>MSB</td>
</tr>
<tr>
<td>FFEA</td>
<td></td>
<td></td>
<td>LSB</td>
</tr>
<tr>
<td>FFE9</td>
<td>IY</td>
<td></td>
<td>MSB</td>
</tr>
<tr>
<td>FFE8</td>
<td></td>
<td></td>
<td>LSB</td>
</tr>
<tr>
<td>FFE7</td>
<td>SP</td>
<td></td>
<td>MSB</td>
</tr>
<tr>
<td>FFE6</td>
<td>POINTER</td>
<td></td>
<td>LSB</td>
</tr>
</tbody>
</table>
FIGURE 7-2. DDT DATA PATHS

NOTE: During 'W' & 'S' command, the registers are saved and reloaded after every instruction step.
TABLE 7-1. MNEMONICS RECOGNIZED BY DDT-80

Unrecognized mnemonics are resolved with a value of zero.

<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>ADDRESS REPRESENTED</th>
<th>DATA SAVED AT THAT ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>:PC*</td>
<td>FFFFE</td>
<td>User's PC Register</td>
</tr>
<tr>
<td>:A</td>
<td>FFFFD</td>
<td>User's A Register</td>
</tr>
<tr>
<td>:F</td>
<td>FFFFC</td>
<td>User's F Register</td>
</tr>
<tr>
<td>:I</td>
<td>FFFFB</td>
<td>User's I Register</td>
</tr>
<tr>
<td>:IF</td>
<td>FFFFA</td>
<td>User's IFF Register</td>
</tr>
<tr>
<td>:B</td>
<td>FFF9</td>
<td>User's B Register</td>
</tr>
<tr>
<td>:C</td>
<td>FFF8</td>
<td>User's C Register</td>
</tr>
<tr>
<td>:D</td>
<td>FFF7</td>
<td>User's D Register</td>
</tr>
<tr>
<td>:E</td>
<td>FFF6</td>
<td>User's E Register</td>
</tr>
<tr>
<td>:H</td>
<td>FFF5</td>
<td>User's H Register</td>
</tr>
<tr>
<td>:L</td>
<td>FFF4</td>
<td>User's L Register</td>
</tr>
<tr>
<td>:A'</td>
<td>FFF3</td>
<td>User's A' Register</td>
</tr>
<tr>
<td>:F'</td>
<td>FFF2</td>
<td>User's F' Register</td>
</tr>
<tr>
<td>:B'</td>
<td>FFF1</td>
<td>User's B' Register</td>
</tr>
<tr>
<td>:C'</td>
<td>FFF0</td>
<td>User's C' Register</td>
</tr>
<tr>
<td>:D'</td>
<td>FFEF</td>
<td>User's D' Register</td>
</tr>
<tr>
<td>:E'</td>
<td>FFEE</td>
<td>User's E' Register</td>
</tr>
<tr>
<td>:H'</td>
<td>FFED</td>
<td>User's H' Register</td>
</tr>
<tr>
<td>:L'</td>
<td>FFEC</td>
<td>User's L' Register</td>
</tr>
<tr>
<td>:IX*</td>
<td>FFEA</td>
<td>User's IX Register</td>
</tr>
<tr>
<td>:IY*</td>
<td>FFE8</td>
<td>User's IY Register</td>
</tr>
<tr>
<td>:SP*</td>
<td>FFE6</td>
<td>User's SP Register</td>
</tr>
</tbody>
</table>

* = 2 byte mnemonics
7-7. COMMAND SUMMARY

Table 7-2 lists all the DDT commands for reference.

7-8. CONVENTIONS

7-9. Hexadecimal numbers are denoted by the number followed by a subscript H. E.g., AF3_H. In a command sequence user input is underlined. (CR) means carriage return. Bracketed items [ ] in a command line are optional. Items in a command line which must be entered exactly as they appear are shown as upper case. Items in a command line which are variables are shown as lower case.
TABLE 7-2. DDT COMMAND SUMMARY

TO INVOKE DDT:

$DDT(CR)

CONSOLE INTERACTION:

- prompt character
- (CR) terminate a command
- or cnt1-U abort

COMMANDS:

B aaaa insert a breakpoint in user's program.
C aaaa,bbbb,cccc copy memory aaaa thru bbbb to cccc and above
E aaaa execute user's program
F aaaa,bbbb,cc fill memory aaaa thru thru bbbb with data cc.
H ... hexadecimal arithmetic.
L aaaa,bbbb,cccc locate all occurrences of data cccc in memory aaaa thru bbbb.
M aaaa,bbbb display, update, or tabulate memory or registers.
O aaaa set offset constant for relocatable programs.
P aa display and update port.
Q quit - return to Monitor.
R a,bb display user registers.
W aaaa ,bb single step starting at address aaaa for bb steps.
V aaaa,bbbb,cccc verify that two blocks of memory are identical.
7-10. PREPARATION

7-11. Create, assemble, and link your Z80 program as described in Section 4, 5, and 6 of this manual.

7-12. You should now be ready to debug a binary file which has your Z80 program on it. To debug the program, use the Monitor GET command to load the program into RAM:

`$GET file(CR)`

where file is the name of the binary file created by the LINK process.

Then execute DDT:

`$DDT(CR)`

The dot (.) indicates that DDT is ready to accept commands.

7-13. DESCRIPTION OF DDT COMMANDS

7-14. COMMAND FORMAT.

7-15. DDT recognizes commands which consist of three parts:

1. A single letter command.
2. An operand or operands separated by commas or blanks.
3. A terminator to either abort the command or cause it to be executed.

    EXAMPLE

    `.M 100,102(CR)`

1. 2. 3.

7-16. In the command mode DDT prompts on the user console with a dot (.). The user may enter any single letter command. A space is then printed on the console. The user may then enter any required operands and a terminator. Operands are separated from each other by a space or a comma. The terminator may be a
carriage return, dot (.) or control-U. Carriage return causes execution of the command. A dot or control-U aborts the command, and the user is prompted again.

NOTE The format of entering commands in DDT differs from FLP-80DOS Monitor commands in that DDT automatically inserts a space after a command to separate it from the operands.

7-17. OPERANDS

7-18. Operands are separated from each other by a space or comma. An operand may take any one of the following forms.

7-19. Hexadecimal number. Leading zeros need not be entered. The last four digits are used for the value entered for address values. The last two digits are used for data values.

7-20. ASCII literal value. Any characters preceded by the letter "L" are converted to their ASCII equivalent value. E.G., LA(=41H), LAB(=4142H).

7-21. Relative Address. A hexadecimal number preceded by the character "R" causes the offset specified by the O command to be added to the number. A relative address is identified by an apostrophe next to it. E.g., (assuming offset = 100H) R0(=100H), R4FF(=5FFH).

7-22. The offset and relative address functions are useful when debugging modules of a program which have been relocated by the Linker.

7-23. Program Counter. The character "$" is used to represent the current address. It is used with the M command to calculate relative branch displacements.
7-24. Added or subtracted numbers. Hexadecimal numbers may be added to or subtracted from each other to represent an operand. E.g., A + A (=14H), 5A + A - 10 (=54H).

7-25. Equal Sign. An equal sign (=) may be entered at any time to display the current value of an operand as 4 hexadecimal digits. E.g., 5A + A -10 = 0054, LAC = 4143.

7-26. Mnemonic. A mnemonic consists of one or two characters following a colon (:). Mnemonics are used to represent Z80 CPU registers. Table 7-1 lists all the allowed mnemonics in DDT and their meanings.

7-27. OPERAND EXAMPLES

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4F7F</td>
<td>The operand value is equal to 4F7FH.</td>
</tr>
<tr>
<td>:PC</td>
<td>The mnemonic PC is equivalent to the save location of the user's program counter.</td>
</tr>
<tr>
<td>5038-5000</td>
<td>The operand value is 38H,</td>
</tr>
<tr>
<td>5038-5000=0038</td>
<td>The same as above except &quot;=&quot; was entered to display the operand value.</td>
</tr>
<tr>
<td>5038-$</td>
<td>If current address = 5000H, then $=5002H and the operand value equals 36H for relative jump instructions.</td>
</tr>
<tr>
<td>5038-$=0036</td>
<td>The same as above except the equal sign was entered.</td>
</tr>
<tr>
<td>305038</td>
<td>More than 4 digits entered, therefore only the last 4 have meaning. Operand value = 5038H.</td>
</tr>
<tr>
<td>305038=5038</td>
<td>The same as above except the equal sign was entered.</td>
</tr>
<tr>
<td>LAB=4142</td>
<td>Operand is equal to the ASCII value of &quot;AB&quot;.</td>
</tr>
<tr>
<td>LA=2041</td>
<td>Operand is equal (LSB) to ASCII value of 'A'.</td>
</tr>
<tr>
<td>R100=1100</td>
<td>Assumes offset = 1000.</td>
</tr>
</tbody>
</table>
7-28. COMMAND TERMINATORS

7-29. The command terminator immediately follows the operand(s) and signals DDT that the command has been entered. Depending on the terminator, DDT will do one of the following:

<table>
<thead>
<tr>
<th>Terminator</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>(CR)</td>
<td>Carriage return. DDT executes the entered command.</td>
</tr>
<tr>
<td>. OR CNTL-U</td>
<td>Period or CNTL-U. DDT aborts the command. The user is prompted for another command.</td>
</tr>
<tr>
<td>^</td>
<td>Carat or up arrow. This terminator is valid only for the M and P commands. When updating a memory location (M) or a port (P), it signals DDT to display the contents of the location or port just updated, or if the location was not updated, the previous location.</td>
</tr>
<tr>
<td>/</td>
<td>Slash. This terminator is valid only for the M command. This causes the data entered to replace the old data and then return to the command mode. If no data was entered, it is treated as a period.</td>
</tr>
</tbody>
</table>

7-30. SPECIAL KEYS

7-31. Several keys have special meaning in DDT:
- period (.) memory printouts on the console (L, M, or V commands) may be aborted by entering a period. Single stepping (W Command) may also be aborted this way. DDT then enters the command mode.
Space bar  The space bar may be used to start and stop single stepping (W command).

7-32. ERRORS

7-33. Any time erroneous input is detected, a question mark (?) is printed and DDT returns to the command mode.
7-34. B COMMAND, BREAKPOINT COMMAND

7-35. Format:

\[ .B \text{ aaaa(CR)} \quad \text{Set breakpoint at memory address aaaa.} \]

\[ .B \text{ (CR)} \quad \text{Clear previous breakpoint.} \]

7-36. Overview. When the breakpoint command is used, a "trap" which consists of three bytes is placed into the user's program. The original program bytes are automatically saved.

7-37. The user then uses the E (execute) command to start execution of the program. When the trap is encountered, DDT is signalled and execution is stopped. The registers from the CPU are then transferred to DDT and printed out on the user console. To resume execution of the program, the user must use the E (execute) command again or the W (single step) command.

7-38. Description. The user types the command identifier B followed by the address where it is desired to place a breakpoint "trap". DDT proceeds to remove any pre-existing breakpoint, extracts and saves 3 bytes of the user's program at the breakpoint address, and places a 3 byte trap into the address. DDT then returns to the command mode. The user may start program execution via the E(execute) command. When the breakpoint trap is encountered, execution is stopped and control is transferred back to DDT. DDT then restores the three bytes of user code at the breakpoint address, reads all the target CPU registers and prints them out(see R-register command).

7-39. DDT then waits for the user to enter one of the following characters:

1. Period (.) returns DDT to the command mode.
2. Carriage return causes one program instruction to be
stepped. After the instruction is executed, the target registers will be printed again and DDT will again wait for user input.

3. Line feed has the same effect as carriage return, but a heading to identify the registers will be printed out.

4. Space bar starts automatically single stepping. Single stepping will continue for 256 steps or until the space bar is pressed again. The user can thus start and stop single stepping of his target program. (See W-Step command).

NOTE: The contents of the registers reflect the effect of the last instruction before the breakpoint was encountered.

7-40. One breakpoint can be set at a time before execution is begun. A breakpoint can be reset by entering the B command with no operands. A breakpoint at a specific address can be cleared by executing that address.

7-41. There are certain characteristics of the DDT breakpoint facility which the user should be aware of during debugging:

1. The trap sequence used by DDT-80 is as follows:
   JP DDT      Jump to DDT Breakpoint Processor

2. Since DDT replaces three bytes of the user program, a breakpoint should be set such that when the user program is executed, control can only be transferred to the first byte of the trap sequence. In addition, the breakpoint must reference the first byte of an instruction. For example in the following sequence:
   LI JR NZ,L3-$
L2 LD A,0
L3 LD B,OFH

A breakpoint should not be set at L2 because if the branch condition at L1 is met, control would be transferred to the third byte of the trap sequence.

3. No error indication is given if one attempts to set a breakpoint in ROM.

4. After a breakpoint has been set, it can be changed simply by entering a new breakpoint. The act of entering a new breakpoint automatically clears the previous breakpoint.

5. When a breakpoint is encountered in a user program, DDT-80 saves the state of interrupts (through IFF) in the :IF register. The state of interrupts is restored or set according to the content of :IF when control is transferred to the user program.

6. Breakpoint will not work in areas where executable code is modified by the program.

EXAMPLE

.B 24E(CR)
   -Set a breakpoint at location 24EH.
.O 100(CR)
   -Set offset.
.B R4F3(CR)
   -Set breakpoint at relative address 4F3H (=5F3H absolute).
7-42. C-COPY MEMORY BLOCKS COMMAND

7-43. Format.

```
.C aaaa,bbbb,cccc(CR)
```

Copy locations aaaa through bbbb inclusive to the memory block starting at address cccc.

7-44. Description. The user enters the command identifier C followed by the starting address aaaa and ending address bbbb of the block to be moved, followed by the starting address cccc of the block receiving the data. The operands may be absolute or relative and are separated by commas or blanks. Upon terminating with a carriage return, DDT performs the requested copy operation, and returns to the command mode. The copy command permits any block of memory data to be moved to any area of memory. The move may be forward or backward and the new block may or may not overlap with the original memory block. Entire programs or subroutines may be moved around in this way. Care should be taken to copy complete instructions on both ends of the block when copying programs, and any relative jump instructions contained within a block to be moved should not jump outside the block. If the second operand entered (bbbb) is smaller than the first (aaaa), a question mark (?) is printed and control returns to the command mode.

EXAMPLE.

```
.C 100,200,1200(CR)  Copy memory locations 100H through 200H inclusive to locations 1200H through 1300H.
.C 100,200,150(CR)  Copy memory locations, 100H through 200H inclusive to locations 150H through 250H. (overlapping copy)
.O 100(CR)  Set relative offset to 100H.
.C RO,R100,R50(CR)  This would be the same as the previous example.
```
7-45. **E-EXECUTE COMMAND**

7-46. Format.

- **E aaaa(CR)** Transfer control to the program starting at address aaaa.

- **E (CR)** Transfer control to the address specified by register:PC.

7-47. Description. To cause execution of a program the user types the identifier E followed by the desired entry address of his program. Upon typing carriage return DDT loads the Z80 CPU registers and then transfers control to the program entry point. The contents of the register map reflect the effect of the last instruction before the breakpoint was encountered. If no entry address is specified after the E command, DDT will transfer control to the address specified by the :PC register (program counter).

Example.

- **E 1200(CR)** Execute the program starting at location 1200H.

To return control to DDT the user's program must encounter a breakpoint (see B-Breakpoint Command).

- **M :PC(CR)** Examine user's program counter (PC).
- **:PC 62FF 1220(CR)** Set user's PC to 1220H.
- **E (CR)** Execute program starting at location 1220H.

The execute command may be used together with the breakpoint command to execute portions of programs while debugging.
7-18

7-48. F-FILL MEMORY COMMAND

7-49. Format:

\[ .F \text{ aaaa,bbbb,cc(CR)} \]

Fill memory locations aaaa through bbbb inclusive with cc.

7-50. Description. the user enters the command identifier F followed by the starting address aaaa and ending address bbbb, followed by the data cc. The operands are separated by commas or blanks. Upon terminating with a carriage return, DDT performs the requested fill operation and then prints a "," to indicate that DDT is ready to accept another command.

Example

\[ .F 100,1FF,5A \text{(CR)} \]

Insert a 5A in every memory location from \(100_H\) through \(1FF_H\).

\[ .O 100\text{(CR)} \]

Set relative offset to \(100_H\).

\[ .F R0,RFF,5A\text{(CR)} \]

Fill same addresses as first example.

DDT waiting for next command.
7-51. H-HEXADECIMAL ARITHMETIC

7-52. Format.

:H +aaaa-bbbb+...+yyyy=zzzz(CR) Perform hexadecimal arithmetic.

7-53. Description. The user enters the command identifier and then enters the arithmetic expression. Only + and - are legal operations. If the sign of the first operand is omitted, it is assumed +. The equal sign causes the 4 digit (least significant 4 digits) result to be displayed. When the terminator is entered DDT returns to accept another command.

EXAMPLES.

:H 5000-4FFF=0001(CR)   Subtract 4FFFH from 5000H.
:H 5000+4FFF=9FFF(CR)    Add 4FFFH to 5000H.

The equal sign caused the 4 digit result to be printed.

DDT waiting for next command.
7-54. **L-LOCATE 8-BIT DATA PATTERN COMMAND**

7-55. **Format.**

```
.L aaaa,bbbb,cccc(CR)
```

Locate and print the address of every occurrence of cccc from aaaa to and including bbbb.

7-56. **Description.** The user enters the command identifier **L** followed by the starting address aaaa and ending address bbbb, followed by the data cccc to be located. Upon terminating with a carriage return, DDT prints every address between aaaa and bbbb which contains cccc. If cccc is less than 100\(_H\), then a one byte comparison is made. If cccc is greater than or equal to 100\(_H\), then a two byte comparison is made. The data to be located should be entered with the most significant two digits of data first followed by the least significant two digits of data (if location 1000\(_H\) contained 13 and location 1001\(_H\) contained 92, the user would enter 9213 as the data to locate).

**EXAMPLE:**

```
.L 0,750,35(CR)
0052  35
00F3  35
0542  35
0750  35

.L 750,35FF(CR)
00F3  35
0542  35
```

Locate every occurrence of 35\(_H\) between address 0 and 750\(_H\).
Every location containing 35 is printed between (and including) 0 and 750\(_H\).
Locate every occurrence of the 2 byte value FF35\(_H\) between address 0 and 750\(_H\).
Every address where 35FF is found is printed out. The location previous to the location printed out contains the least significant two digits.
7-57. **M-DISPLAY AND UPDATE MEMORY OR REGISTER COMMAND**

7-58. **Format:**

```
.M aaaa(CR)
```

7-59. **Description.** The user enters the command identifier M and the operand aaaa followed by a carriage return. DDT prints the memory address or mnemonic on the next line, followed by the contents of that particular address in hexadecimal. If the content is to be changed, the new value is entered. Any number of digits may be entered, but only the least significant two (or four) digits are accepted.

7-60. **Terminators.** When the user is examining and/or modifying a register or memory location, the accompanying terminator signals the action DDT is to take. The possible operand (new value entered) and terminator combinations are:

<table>
<thead>
<tr>
<th>Terminator</th>
<th>Meanings</th>
</tr>
</thead>
<tbody>
<tr>
<td>(CR)</td>
<td>No operand entered, display next address or register.</td>
</tr>
<tr>
<td>^</td>
<td>No operand entered, display previous address or register.</td>
</tr>
<tr>
<td>/</td>
<td>No operand entered, display next address or register.</td>
</tr>
<tr>
<td>aaaa</td>
<td>Operand aaaa entered but &quot;.&quot; aborts command with no change to value at address.</td>
</tr>
<tr>
<td>aaaa(CR)</td>
<td>Operand aaaa entered, change value at address to aaaa and step to next address.</td>
</tr>
<tr>
<td>aaaa^</td>
<td>Operand aaaa entered, change value at address to aaaa and display same address with the new value aaaa displayed.</td>
</tr>
<tr>
<td>aaaa/</td>
<td>Operand entered, change value at address to aaaa then exit to command mode.</td>
</tr>
</tbody>
</table>
7-61. Memory display. Memory locations are accessed as follows:

M 16A(CR)  Examine memory location 016AH.
016A 3F(CR)  It contains 3FH do not change, step to next location.
016B 92 ▲  Next location contains 92H, do not change, go back to previous location.
016A 3F 34FF■  Change contents of 016A to FFH and display same location. Note that only the last 2 digits typed are stored in 016A (the entry 34 was in error).
016A FF(CR)  New contents displayed, step to next.
016B 92 ●  DDT waiting for next command.

7-62. When accessing relative memory locations, the user sets the offset with the "O" command and uses the "R" prefix with the memory address. Assuming the offset was set to 1000:

M RO(CR)  The relative address, absolute address and data are printed out.
0000 1000 xx  DDT waiting for next command.

7-63. Register display. The user may examine and change his CPU registers. They may be initialized, for example, prior to program execution, or after a breakpoint has been encountered in the program to be debugged. The contents of the user's registers may be accessed through the use of the mnemonics discussed in paragraph 7-26.

M :A(CR)  Examine user's accumulator.
:PC 0010  User's PC Register, return to command mode.
:M :PC(CR)  Examine user's PC (program counter) register.
:PC 0010 .  Return to command mode.
.  DDT waiting for next command.

7-64. When resuming execution of the user's program, these new values will be inserted into the user's Z80 CPU registers.

7-65. Relative branches. A special feature of DDT allows the user to conveniently compute relative addresses used in relative branch instructions. The value of the symbol "$" is defined as the value of the current location and only has meaning during display and update commands.

7-66. This example shows the entering of a jump relative instruction at location $0H to branch to location $38H.

   .M 0(CR)  Examine location 0H.
   0000 20 18(CR)  Insert First byte of jump (JR 38H-$)
   0001 F8 38-$=0036\triangleleft  Compute and display relative displacement for branch from 0H to 38H.
   0001 36 .  Branch displacement of 36 shown.
.  DDT waiting for next command.

7-67. It should be noted that the maximum allowed displacement value for forward branches is $7FH and for backward is $80H. It is simple to determine if the relative branch is within its range by examining the most significant two digits of the computed displacement. For forward branches, the most significant two digits should be $00H and for backward branches, the most significant two digits should be $FFH.
7-68. **M-TABULATE MEMORY COMMAND**

7-69. Format

```
.M aaaa,bbbb(CR)  Display memory location aaaa through bbbb.
```

7-70. Description. The user enters the command identifier M followed by the starting (aaaa) and ending (bbbb) addresses of the memory block. Upon terminating with a carriage return DDT prints a line feed, and then prints the contents of aaaaH to bbbbH inclusive with up to 16 values per line. DDT then returns to the command mode. The tabulation may be stopped at any time by entering "." on the console. When the 'R' prefix is used, the relative address is printed before absolute.

**EXAMPLE**

```
.M 4100,4127(CR)  display memory locations 4100H through 4127H inclusive
```

```
4100 2B 90 12 20 00 B7 A5 21 10 94 04 20  CA B7 44 18
4110 81 11 34 21 07 94 17 45 12 55 A5 18  21 80 C5 55
4120 90 0C A5 81 09 21 40 22
```

**:Q 4100(CR) set offset to 4100.
```
.M RO,R27(CR)
'0000 4100 2B 90 12 20 00 B7 A5 21 10 94 04 20  CA B7 44 18
'0010 4110 81 11 34 21 07 94 17 45 12 55 A5 18  21 80 C5 55
'0020 4120 90 0C A5 81 09 21 40 22
```
7-71. O-SET OFFSET CONSTANT COMMAND

7-72. Format:

\[ .O \text{ aaaa(CR)} \]

Set offset equal to aaaa.

7-73. Description. The user enters the command identifier O followed by the offset aaaa. Upon terminating with a carriage return, DDT saves the 16 bit offset. After the offset has been set, both relative and absolute addresses are printed any time addresses are displayed and until the offset is cleared. The offset can be cleared by entering the O command with no operands.

EXAMPLE

\[ .O \text{ 200(CR)} \]
\[ .H \text{ RO=0200(CR)} \]
\[ . \text{ DDT waiting for next command.} \]
7-74. P-DISPLAY AND UPDATE PORTS COMMAND

7-75. Format.

\[ .P \text{ aa(CR)} \]

7-76. Description. The user enters the command identifier \( P \) followed by the port address \( \text{aa} \) and a carriage return. DDT responds by printing the port address and the value at that port. If the value at that port is to be changed, the user enters the new value. The new value entered is a 2 hexadecimal digit operand. When the user is examining and/or modifying a port, the terminator signals the action DDT is to take. The possible operand (new value entered) and terminator combinations are:

<table>
<thead>
<tr>
<th>Terminator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>(CR)</td>
<td>No operand entered, display next port.</td>
</tr>
<tr>
<td></td>
<td>No operand entered, display previous port.</td>
</tr>
<tr>
<td>.</td>
<td>No operand entered, return to command mode.</td>
</tr>
<tr>
<td>aa</td>
<td>Operand ( \text{aa} ) entered, but &quot;.&quot; aborts command with no change to the port.</td>
</tr>
<tr>
<td>aa(CR)</td>
<td>Operand ( \text{aa} ) entered, change the port value to ( \text{aa} ) and step to display the value at the next port.</td>
</tr>
</tbody>
</table>

EXAMPLE

\[ .P \ E2(CR) \] User displays port \( E2_H \).
\[ E2 \ 00 \ 12(CR) \] User changes value to \( 12_H \).
\[ E3 \ 15 . \] Return to command mode.
\[ . \] DDT waiting for next command.
7-77. **Q-QUIT COMMAND**

7-78. **Format**

```
.Q CR
```

7-79. **Description.** The user enters Q to exit DDT and return to the FLP-80DOS Monitor. The Monitor prints $ upon entry. **EXAMPLE.**

```
.Q(CR) exit DDT.
$ enter Monitor (Monitor prompts $)
```
7-80. R-DISPLAY CPU REGISTERS COMMAND

7-81. Formats.

- **.R (CR)**
  - Print the contents of the CPU registers.
- **.R 1(CR)**
  - Print a heading to label the CPU registers on one line, on the next line print the contents of the CPU registers.
- **.R 1,aa(CR)**
  - Print a heading to label the CPU registers and set the long/short flag as follows. `aa=0` SHORT, `aa=1` LONG. Long causes all registers to be printed after breakpoint and single step. Short causes only PC and AF to be printed. The LONG/SHORT FLAG remains set until changed by the 'R' command.

7-82. Description. The user enters the comma command identifier `R`. If the user wants a heading to be printed that labels the register contents, an operand of 1 is entered. If no heading is desired, then no operand is entered. If the '0' command has been used to set an offset, the relative PC is also printed (PC'). The second operand is optional and has the following meaning:

  - **aa=0** - short form: only the Z80 program counter and AF register will be displayed.
  - **aa=1** - long form. All CPU registers will be displayed.
7-83. Note that aa remains set to the value entered during all following commands until it is reset.

Examples.

.R (CR)
A000 0100 0104 CFB3 C09A FFEE EDF6 9C3E C3DC FE9B D6ED F1BE FFB4

.R 1 (CR)
PC AF IIF BC DE HL A'F' B'C' D'E' H'L' IX IY SP
A000 0181 0104 CFB3 0010 C09A FFEE EDF6 C3DC FE9B D6EC F1BE FFB4

bit
PC contains A000H 7 0
A contains 01H F = 1 0 0 0 0 0 0 1
F contains 81H S Z X H X P/V N C
I contains 01H
IF contains 04 (Bit 3 = 1 implies IFF = 1)

S = sign flag
Z = zero flag
X = indeterminate flag
H = half carry (for BCD operations)

SP contains FFB4H P/V = parity or overflow flag
N = BCD add/subtract flag
C = carry flag
7-84. **V-VERIFY MEMORY COMMAND**

7-85. Format.

\[ .V \text{aaaa,bbbb,cccc(CR)} \]

Compare memory location aaaa to bbbb with the memory starting at cccc.

7-86. Description. The user enters command identifier V followed by the starting address aaaa and ending address bbbb, followed by the starting address cccc of the second memory block. The operands are separated by commas or blanks. Upon terminating with a carriage return, every address from aaaa to bbbb is compared with the corresponding address starting at cccc. Any discrepancies are printed on the console. ("address data address data"). When the comparison is complete, DDT is ready to accept another command. Printing of addresses may be aborted by entering a period (.) from the user console at anytime.

Example.

\[ .V \text{0,FF,1000(CR)} \]

Compare every location from 0 to FFH inclusive.

\[ .0 \text{100(CR)} \]

Set offset.

\[ .V \text{RO,RFF,R1000(CR)} \]

Relative and absolute address on non-matches.

\[ '0000 0100 \text{ BC '1000 1100 CC} \]
7-87. W-WALK THROUGH A PROGRAM COMMAND

The walk command, also known as software single-step, allows stepping through a program which is contained in RAM. The user's registers are saved and displayed after each step.

7-88. Format.

.W aaaa,nn,xxx(CR) Begin software single-step at address aaaa, for nnH steps, xxx = HD requests register heading, xxx = DIS requests disassembly (AIM-80 required for DIS).

.W Raaaa,nn,xxx(CR) Relative address.

7-89. Description. The user enters the command identifier W followed by the starting address aaaa, the number of steps to take nn, and the options operand xxx. The operands are separated by commas or spaces. Upon terminating with a carriage return, the DDT begins "walking" through the user's program (RAM resident). After each step the user's registers are displayed (See 'R' command). When nn steps have been taken, DDT waits for the user to enter a carriage return, line feed, space, or ".". A carriage return causes the next instruction to be executed and wait again for input. A line feed causes the register heading to be printed before executing the next instruction. A space causes single stepping to continue for 256 instructions or until another space is entered to stop stepping. If nn is omitted, the default is 1. If aaaa is omitted, the last value of the user's program counter (:PC) is used to begin "walking". The stepping may always be stopped by entering any of the characters described above. When the address entered is relative, the 'PC is also printed (relative PC).
7-90. Restrictions to W Command.
   1. Only operates with programs in RAM.
   2. Cannot CALL or RESTART to an address one or two locations before the CALL or RESTART.
   3. Walking through self modifying code is not allowed.

7-91. DEBUGGER ESCAPE (CNTL-C)

7-92. During normal use of DDT the Debugger Escape is not enabled because the minimal listener is not enabled. However, if execution of the user program is begun with the Monitor Implied Run Command or by the Monitor BEGIN command, the minimal listener is enabled. Debugger Escape can be used to trap out of the executing program as if a breakpoint had been encountered. The CPU registers will be saved and all DDT commands can be used. In this mode, Debugger Escape can be used any number of times.

EXAMPLE

$FILE1(CR)
   - user uses Implied RUN command to load and execute his program from disk file FILE1.

(cntl-C)
   - user depresses cntl-C to cause Debugger Escape.

A000 0100 0103 CFB3 C09A FFEE EDF6 9C3E C3DC FE9B D6ED F1BE FFB4
   - DDT is entered as if a breakpoint had been encountered.
PART 2

TECHNICAL INFORMATION
SECTION 8

RDCHR AND WRCHR SUBROUTINES

8-1. INTRODUCTION
NOTE: These two routines allow the simplest way of performing device I/O on the FLP-80DOS system. It is suggested that the example shown in this section be programmed to acquaint the user with this system.

8-2. RDCHR and WRCHR are two subroutines which allow simplified byte I/O to any of the 6 default Logical Unit Numbers. RDCHR returns one byte from a device via LUN 0, 2, or 4. WRCHR writes one character to a device via LUN 1, 3, or 5. Each subroutine assumes that the selected Logical Unit Number has been assigned to a device handler via the Monitor $ASSIGN command. The following paragraphs define entry and exit parameters. Users of DDT-80 V1.3 and ASMB-80 from the SDB-80 paper-tape system will recognize that this protocol is exactly the same as RDCHR and WRCHR in that software package. This allows current paper tape users to easily upgrade to the FLP-80DOS software.

8-3. RDCHR - READ ONE BYTE

8-4. CALLING SEQUENCE.
CALL RDCHR ;RDCHR Address is specified in Appendix F.

8-5. ENTRY PARAMETERS.
E register:
Bits 0-2 = LUN (0-5).
Bits 3 = 1 to initialize or open the device.
Bits 4, 5 = reserved.
Bit 7 = 1 for immediate return.
8-6. EXIT PARAMETERS.
A register and D register = byte which was read (ASCII).
E register:
  Bit 3  reset after initialization.
  Bit 6 = 1 if error occurred on input.
  Bit 7  reset if operation was performed.
All other registers are maintained.

8-7. OPERATION. The driver uses LUN 0, 2, 4 or input. Lun's 1, 3 and 5 are modified to 0, 2, 4, respectively, within the subroutine. If the initialize bit (3) is set, OPENR request will be performed. Each READ request will return one byte (Byte Format I/O). Upon encountering 04H (EOT), the close request will be performed. Bit 6 will indicate if an I/O error occurred.

8-8. If bit 7 is set upon entry, the device status is read, but no read operation is initiated unless the device is ready. However control is always returned to the caller whether or not the operation was performed. This feature is not available with the disk.

8-9. WRCHR - WRITE ONE BYTE

8-10. CALLING SEQUENCE
CALL WRCHR ;WRCHR Address is specified in Appendix F.

8-11. ENTRY PARAMETERS
E register:
  Bits 0-2 = LUN (0-5).
  Bits 3  = 1 for initialize.
  Bits 4,5 - reserved.
  Bits 7  = 1 for immediate return.
D register = byte to be output (ASCII).
8-12. EXIT PARAMETERS
A register - changed.
E register:
    Bit 3 reset after initialization.
    Bit 6 = 1 if error occurred on output.
    Bit 7 reset if operation was performed.
All other registers are maintained.

8-13. OPERATION. The driver uses LUN 1,3 or 5 for output. LUN's 0,2, and 4 default to 1,3,5 respectively within the subroutine. If the initialize bit is set, OPENW request will be performed. If the unit is a disk unit and if the file exists, it will be erased and reopened. Each WRITE request outputs one byte (Byte Format I/O). If the byte is 04H (EOT), it will be output and a close request will be performed. Bit 6 indicates if an error occurred. The error number will be in the default vector for the correct LUN.

8-14. If bit 7 is set upon entry, the status port will be read, but no write operation is initiated unless the device is ready. However, control is always returned to the caller whether or not the operation was performed. This feature is not available with the disk.

8-15. DDT OPERATION

8-16. During execution of DDT (debugger) all I/O is directed to the console drivers without using the IOCS facilities. This allows the user to use all of available RAM and facilitates the AIM-80 memory map and operation. This mode can be forced by the programmer by setting location FF12H to the value 2.
EXAMPLE - See Figure 8-1.

CAUTION: When using RDCHR, the last character of a file, which
will be EOT (04H), must be read in order to properly close the file. When using WRCHR, the last character output must be EOT (04H) in order to properly close the file.

NOTE The calling addresses for RDCHR and WRCHR will not change in future versions of FLP-80DOS.
EXAMPLE OF RDCHR & WRCHR

NAME FIG8_1

0002 0003 ;
0004 ; THIS PROGRAM READS CHARACTERS INTO A BUFFER UNTIL A CARRIAGE RETURN IS ENCOUNTERED. THEN THE BUFFER IS PRINTED OUT ON THE CONSOLE DEVICE.
0005 ;
0006 ;
0007 ;
0008 ; THIS PROGRAM MUST BE LINKED WITH 'SYSLNK' IN ORDER TO RESOLVE THE EXTERNAL REFERENCES.
0009 ; E.G.: SLINK FIG8D1,SYSLNK
0010 ;
0011 ;
0012 ; EXTERNAL LINKAGES TO SYSTEM Routines
0013 ;
0014   GLOBAL JTASK
0015   GLOBAL RDCHR
0016   GLOBAL WRCHR
0017 ;
0000 212200' 0018   LD  HL,BUF ;GET BUFFER ADDRESS
0003 1E00 0019   LD  E,0 ;CONSOLE LOGICAL UNIT NUMBER
0020 ;
0005 CDFFFF 0021 LOOP  CALL  RDCHR ;READ ONE CHARACTER FROM CONSOLE
0008 77 0022   LD  (HL),A ;PLACE IT INTO THE BUFFER
0009 23 0023 INC  HL ;INCREMENT BUFFER POINTER
000A FE0D 0024   CP  ODH ;CHECK FOR CARRIAGE RETURN
000C 20F7 0025   JR  NZ,LOOP-S ;IF NOT, LOOP FOR MORE
000E 212200' 0026 ;
0011 1E01 0027   LD  HL,BUF ;REINITIALIZE BUFFER POINTER
0028 ;
0029 ;
0013 56 0030 LOOP2  LD  D,(HL) ;GET CHARACTER FROM BUFFER
0014 CDFFFF 0031 CALL  WRCHR ;WRITE IT OUT TO CONSOLE LUN
0017 23 0032 INC  HL ;INCREMENT BUFFER POINTER
0018 7A 0033   LD  A,D ;GET CHARACTER INTO A-REG
0019 FE0D 0034   CP  ODH ;CHECK FOR CARRIAGE RETURN
001B 20F6 0035   JR  NZ,LOOP2-S ;IF NOT, LOOP FOR MORE
0036 ;
001D 3E01 0037   LD  A,1 ;
001F C3FFF 0038   JP  JTASK ;ELSE RETURN TO MONITOR
0039 ;
0040 ; INPUT/OUTPUT BUFFER
0041 ;
0022 0042 BUF  DEFS  128 ;DEFs CANNOT TERMINATE A Module
0042 0043 DEFB  0
0044 END

ERRORS=0000
SECTION 9

INPUT/OUTPUT CONTROL SYSTEM (IOCS)

9-1. INTRODUCTION

9-2. The Input/Output Control System (IOCS) provides a general purpose means of accessing all types of I/O devices. It makes any differences between devices as transparent as possible to the user. IOCS may be used to access data from a device or write data to a device. This may be achieved in a user program by filling a vector within the user program with information regarding the type of I/O action required and calling IOCS. IOCS not only uses the information contained in the vector, but also returns information to the user in the vector. Several system routines exist to aid the user in working with IOCS and are described in Section 13.

9-3. VECTOR DEFINITION

9-4. IOCS requires that a 48 byte (30H) vector be filled with information regarding the type of I/O action to be performed and where that action is to take place. The vector may be filled within the user program or by using the $ASSIGN command previous to entering the program (see section 2 of this manual). If the $ASSIGN command is used, IOCS fills the vector pointed to by the IY register when an OPEN request is made (see Section 9-15). When a user makes a request to IOCS, the IY register must point to the first address of the vector being used. Bytes 0-29 of the vector are the user interface to IOCS. Bytes 30-39 are reserved for I/O device handler usage. Bytes 40-47 are reserved
for IOCS usage. Table 9-1 lists the sections of the vector and assigns a name to each section for easy reference. Each vector name contained in table 9-1 will be discussed in detail. The user may reference the sample program in section 9-71 to see how the vector and IOCS are used.
### TABLE 9-1. VECTOR DEFINITION

<table>
<thead>
<tr>
<th>IELD</th>
<th>#BYTES</th>
<th>OFFSET</th>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>FORM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>*(IY+0)</td>
<td>LUNIT</td>
<td>Logical Unit Number</td>
<td>(Binary)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>*(IY+1)</td>
<td>DVCE</td>
<td>Device Mnemonic</td>
<td>(ASCII)</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>*(IY+3)</td>
<td>UNIT</td>
<td>Unit Number</td>
<td>(ASCII)</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>*(IY+4)</td>
<td>FNAM</td>
<td>File Name</td>
<td>(ASCII)</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>*(IY+10)</td>
<td>FEXT</td>
<td>File Extension</td>
<td>(ASCII)</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>*(IY+13)</td>
<td>VERS</td>
<td>File Version</td>
<td>(Binary)</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>*(IY+14)</td>
<td>USER</td>
<td>User Number</td>
<td>(Binary)</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>*(IY+15)</td>
<td>RQST</td>
<td>Request Code</td>
<td>(Binary)</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>*(IY+16)</td>
<td>FMAT</td>
<td>I/O Format</td>
<td>(Binary)</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>*(IY+17)</td>
<td>HADDR</td>
<td>Device Handler Address</td>
<td>(Binary)</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>*(IY+19)</td>
<td>ERRA</td>
<td>User Specified Error Return Address</td>
<td>(Binary)</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>*(IY+21)</td>
<td>CFLGS</td>
<td>Control Flags</td>
<td>(Binary)</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>*(IY+22)</td>
<td>SFLGS</td>
<td>Status Flags</td>
<td>(Binary)</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>*(IY+23)</td>
<td>ERRC</td>
<td>Error Code</td>
<td>(Binary)</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>*(IY+24)</td>
<td>PBFFR</td>
<td>Physical Buffer Number</td>
<td>(Binary)</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>*(IY+25)</td>
<td>UBFFR</td>
<td>User's Buffer Address</td>
<td>(Binary)</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>*(IY+27)</td>
<td>USIZE</td>
<td>User's Buffer Size</td>
<td>(Binary)</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>*(IY+29)</td>
<td>NREC</td>
<td>Number of Records</td>
<td>(Binary)</td>
</tr>
<tr>
<td>19</td>
<td>10</td>
<td>*(IY+30)</td>
<td>HSCR</td>
<td>Device Handler Scratch</td>
<td>(Binary)</td>
</tr>
<tr>
<td>20</td>
<td>8</td>
<td>*(IY+40)</td>
<td>ISCR</td>
<td>IOCS Scratch</td>
<td>(Binary)</td>
</tr>
</tbody>
</table>

where * appears indicates the parameter is to be set up by the user prior to calling IOCS.
9-5. The following paragraphs describe each field in the IOCS vector.

9-6. LUNIT. The LUNIT field in the vector is the Logical Unit Number. There may be as many as 256 logical units, numbered 0-FFH. The number stored in the LUNIT field corresponds to the logical unit number used in the Monitor $ASSIGN command (See Section 2). When an OPEN request is made in IOCS, the REDIRECT TABLE is searched for a logical unit number which has been redirected via the $ASSIGN command corresponding to the number stored in the LUNIT field of the vector. LUN FFH is never redirected. If a match is found, the data found in the REDIRECT TABLE is stored in the user vector and the requested operation is performed. Logical unit numbers 0 - 5 are the default logical units and are assigned by FLP-80DOS at power up and when FLP-80DOS is booted from disk into RAM (See Section 2 of this manual). Vectors for the default logical units already exist in RAM and the user need not set up additional vectors for them. The addresses of the default vectors may be accessed by loading the D-reg. with the default logical unit number and calling GETVEC (see Section 13). These vectors are used by FLP-80DOS utility programs, and they may also be used by user application programs. Lun's 0 and 1 are always assigned to the console input and console output devices respectively. All other LUN's require that memory space be allocated for the 48-byte vector by the program using the LUN.

9-7. Any LUN may be assigned to a device handler by setting up the device information in the vector. (See below). Any LUN (except FFH) may be redirected to any device by the Monitor $ASSIGN command (See Section 2). LUN FFH is never redirected: the device information placed in the vector is the information used by IOCS. In addition when LUN 0 and 1 are reassigned in the
Monitor they are closed and reopened immediately to facilitate batch mode operation (See Section 14).

9-8. The same LUN may be used in any number of different vectors. This can facilitate a multi-user system in which several different programs use a LUN with a separate vector for each program. Further, LUN FF₁ can be used for any number of different vectors within the same program. The FLP-80DOS Text Editor uses this feature.

NOTE An LUN is redirected to a different device by using the Monitor $ASSIGN command. However, the redirection does not take place until the LUN is opened. (Except for LUN 0 and LUN 1). Section 2 describes this in more detail.

9-9. DVCE. The DVCE field is composed of two ASCII character mnemonic which represents an I/O device. IOCS calls an external routine which searches for the mnemonics in a table. The Mnemonic Lookup Table also contains the corresponding address of the device handler. FLP-80DOS provides an expandable Mnemonic Lookup Table with a number of pre-assigned device mnemonics in it. The list of available FLP-80DOS device mnemonics is shown in Table 9-2.

<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP</td>
<td>Line Printer (Centronics compatible)</td>
</tr>
<tr>
<td>CR</td>
<td>Card Reader (Documation M200)</td>
</tr>
<tr>
<td>DK</td>
<td>Flexible Disk</td>
</tr>
<tr>
<td>LP</td>
<td>Line Printer (Data products compatible)</td>
</tr>
<tr>
<td>PP</td>
<td>Paper Tape Punch</td>
</tr>
</tbody>
</table>
PR    Paper Tape Reader
TI    Silent 700 digital cassette reader (ADC is required)
TK    Terminal Keyboard
TO    Silent 700 digital cassette write (ADC is required)
TR    Teletype paper tape reader (step control is required)
TT    Teletype Printer or CRT screen, or Silent 700 printer.

9-10. UNIT. The UNIT field specifies one of a number of devices having the mnemonic specified in DVCE. For example if the DVCE was 'DK' (Flexible Disk), the Unit field would specify which disk unit the I/O operation is directed to. The device handler is responsible for decoding and using the UNIT field. In FLP-80DOS, all supplied handlers access one device (UNIT=0) except the Flexible Disk Handler (FDH).

9-11. FNAM. The FNAM (Filename) field is used only when accessing file structured devices. The six (6) ASCII bytes of the filename to be accessed are filled in by the user in the user program previous to calling IOCS or by use of the $ASSIGN command (See Section 2). In FLP-80DOS, the filename starts at the beginning of the field and is padded with blanks.

9-12. FEXT. The FEXT is an extension on a filename. In FLP-80DOS the following system extensions are reserved:

OBJ    ASCII hexadecimal object format
BIN    Binary RAM Image format
CRS    Linker Cross Reference file
TMP    Editor or Assembler temporary file
LST    Assembler listing file
The user may define and use other extensions as required. If the $ASSIGN command was used to enter the filename, the extension defaults to three (3) blanks.

9-13. VERS. The VERS field (version) is another extension on the filename. FLP-80DOS system programs do not support the version number. However, IOCS and the Floppy Disk Handler (FDH) do support it, but it is used for the date implementation in version 2.1 of FLP-80DOS.

9-14. USER. The USER field can be used to further identify a file. FLP-80DOS system programs support the USER field, but they do not support a multi-user environment. OEM users may wish to use this facility to develop a multi-user system. The default user number is one.

9-15. RQST. The RQST field is the request code. This field defines which type of action will be performed by IOCS. How a device handler interfaces to these request codes is described later in this section. The FLP-80DOS Flexible Disk Handler (FDH) supports an extended range of request codes which may be passed to IOCS. These codes are described in Section 10 of this manual.

<table>
<thead>
<tr>
<th>RQST CODE (HEX)</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>OPENR</td>
<td>OPEN this unit for READING</td>
</tr>
<tr>
<td>01</td>
<td>OPENW</td>
<td>OPEN this unit for WRITING</td>
</tr>
<tr>
<td>02</td>
<td>CLOSE</td>
<td>Close this logical unit</td>
</tr>
<tr>
<td>03</td>
<td>READ</td>
<td>Read data from this unit</td>
</tr>
<tr>
<td>04</td>
<td>WRITE</td>
<td>Write data to this unit</td>
</tr>
<tr>
<td>05</td>
<td>REWIND</td>
<td>Go to beginning of input/file</td>
</tr>
<tr>
<td>06</td>
<td>INIT</td>
<td>Initialize all units of this device type</td>
</tr>
<tr>
<td>07</td>
<td>ERASE</td>
<td>Erase this file</td>
</tr>
</tbody>
</table>
9-16. FMAT. The FMAT field in the vector describes the I/O format selected by the user (high order 4 bits of the FMAT field) as well as the number of physical records to be allocated by the physical buffer allocator when the unit is opened (low order 4 bits (x) of the FMAT field). The user must select the format code best suited for the type of action required and the type of file being used.

**TABLE 9-4. FORMAT REQUEST CODES**

<table>
<thead>
<tr>
<th>FMAT CODE (HEX)</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0X</td>
<td>Byte I/O</td>
<td>Pass single bytes through A-REG.</td>
</tr>
<tr>
<td>1X</td>
<td>ASCII Line</td>
<td>Read/Write until carriage return.</td>
</tr>
<tr>
<td>2X</td>
<td>Logical Buffer</td>
<td>Read/Write number of bytes specified by USIZE.</td>
</tr>
<tr>
<td>3X</td>
<td>Binary ram image</td>
<td>RAM IMAGE to/from disk for binary save or load.</td>
</tr>
</tbody>
</table>

9-17. In all formats except Binary Format, double buffering takes place. That is, when a READ or WRITE request is made, data is placed in a buffer at the top of available RAM (the address of the buffer is determined by the physical buffer allocator). When a READ request is made to IOCS, data is retrieved from the buffer rather than the disk file. When a WRITE request is made, data is placed into the buffer until the buffer is filled before outputting the data to a disk file. IOCS handles all blocking/deblocking functions.

9-18. The size of the buffer used for storing data is controlled by the user in the low order 4 bits (x) of the FMAT field. This
number (0–FH) corresponds to the number of physical records to be allocated. For example, if the user selected to read data from a file and selected to store 4 records of data in the buffer, the buffer size would be 496 bytes in length (4 records 124 bytes per record). The user must select the best trade-off for his particular application. If the user chooses a small number of records to be allocated, more memory will be available for user programs in RAM. However, disk access time may be greater. A large number of allocated records will cause disk access time to be reduced but user RAM will be reduced also.

9-19. In Byte I/O Format, a single character may be written to a device. The character to be written is passed to IOCS in the A-register. When reading, the byte read is passed back in the A-register.

9-20. In ASCII Line Format, data may be written to a device or read from a device on a line-at-a-time basis. If reading from a device, UBFFR (IY+25) contains the address (least significant byte first) where the line is to be stored in RAM. If writing to a device, UBFFR contains the address in RAM where the ASCII line to be written begins. Action on each line continues until a carriage return/line feed is encountered. The contents of UBFFR are not destroyed after the request is completed.

9-21. In Logical Buffer Format, the user can control the number of bytes to read or write with the USIZE (IY+27) parameter. To read data from a device, the user should load the UBFFR (IY+25) parameter with the address of the beginning area in RAM where data is to be stored. The USIZE parameter should be filled with the number of bytes to read. IOCS will read data from the device specified, store the data in RAM beginning at the address contained in UBFFR, and continue this operation until the USIZE
parameter is satisfied. To write data to a device, the user should load UBFFR with the address of the beginning area in RAM where data is to be written from. IOCS will begin reading data from RAM pointed to by UBFFR and writing the data to a device until the USIZE parameter is satisfied. If writing to a disk file, USIZE must be less than or equal to 'X' times 124, where 'X' is the number of physical records allocated as specified in the FMAT (IY+16) field.

9-22. Binary Format is reserved for binary disk files. When an OPENR (open for reading) request is made, the load address is read from the directory and placed in the UBFFR (IY+25) parameter. UBFFR determines where the contents of the binary file are to be loaded in RAM. The user may alter the address in UBFFR previous to making READ request to IOCS to load the data in a different area in RAM. The binary file will be read and stored in RAM beginning at the address contained in UBFFR and continue until end-of-file is encountered. When an OPENW (open for writing) request is made, the address contained in UBFFR is stored in the directory. The USIZE (IY+27) parameter specifies the number of bytes to be saved. This will be rounded mod-124 in FLP-80DOS. When a WRITE request is made to IOCS, data will be read from RAM beginning at the address contained in UBFFR and stored on a disk file. This action will continue until the USIZE parameter has been satisfied.

9-23. HADDR. The HADDR field is the address of the device handler. This field is filled in by the IOCS when the logical unit is opened. (OPENR or OPENW request).

9-24. ERRA. The ERRA field is a user-specified error return
address, least significant byte first. If the field is left zero, then IOCS will return without calling the return address. If bit 4 of CFLGS (See Section 9-28) is set, the system error handler will print a message on the device assigned to default Logical Unit 1.

9-25. CFLGS. The CFLGS field specifies various user specified I/O options as listed in the following table:

<table>
<thead>
<tr>
<th>BIT #</th>
<th>FLAG DESCRIPTION</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>&quot;MOUNT&quot;/&quot;DISMOUNT&quot; Upon Open/Close</td>
<td>MOUNT</td>
</tr>
<tr>
<td>1</td>
<td>Auto Echo Serial Device</td>
<td>ECHO</td>
</tr>
<tr>
<td>2</td>
<td>Immediate Return</td>
<td>IRET</td>
</tr>
<tr>
<td>3</td>
<td>Read after Write requested</td>
<td>RDWR</td>
</tr>
<tr>
<td>4</td>
<td>Error Print Request</td>
<td>ERRPR</td>
</tr>
<tr>
<td>5</td>
<td>Strip Parity</td>
<td>NPAR</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

9-26. If the MOUNT bit is set in the CFLGS Field, then IOCS will print the following message for OPEN and CLOSE requests:

for OPENR or OPENW
  MOUNT XXY, TYPE C WHEN READY:
for CLOSE
  DISMOUNT XXY, TYPE C WHEN READY:

where XX is device mnemonic
and Y is unit

This allows the user to output a message to ensure the device he is trying to access is made ready before execution.

9-27. If the ECHO bit is set, in ASCII line input, each
character read in is echoed to the console output device (as specified in default Logical Unit 1). Additional editing is performed on the line (Backspace, Rubout, Control-U, Tab). The following conventions are used:

BACKSPACE (ASCII 08H) - delete character from the buffer. The cursor movement is backspace, overprint with a blank, and backspace again.

RUBOUT (ASCII 7FH) - delete previous character from the buffer. A backslash is printed on either side of the characters which are deleted.

CONTROL-U (ASCII 15H) - delete line.

TAB (ASCII 09H) - the tab character is entered into the buffer and the cursor is moved over mod - 8 spaces.

9-28. If the IRET bit is set, then any device handler which supports IRET will return immediately to the caller regardless of the status of the device. The device handler interrogates the device status. If the device is not ready, IRET flag set will be returned to caller. If the device is ready, the I/O operation will be performed and IRET flag reset will be returned to caller. This facility can be used by OEM users in a multitasking environment for handling I/O devices. Immediate Return can be used to check for time out on certain devices.

9-29. If the RDWR bit is set, then those handlers which support this facility will perform a read and verification after write. The FLP-80DOS Floppy Disk Handler (FDH) supports this facility.

9-30. If the ERRPR bit is set, then any error generated by a device handler or IOCS will be printed on the console device by IOCS. Appendix E shows the format of the messages.

9-31. If the NPAR bit is set, then bit 7 of every byte of I/O will be unconditionally reset by IOCS.
9-32. SFLGS. The SFLGS field contains flags used by IOCS to keep track of the status of a logical unit. This field must be cleared (00H) by user before opening a logical unit.

<table>
<thead>
<tr>
<th>BIT #</th>
<th>FLAG DESCRIPTION</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unit open</td>
<td>UNOP</td>
</tr>
<tr>
<td>1</td>
<td>Unit open for write</td>
<td>UNOPW</td>
</tr>
<tr>
<td>2</td>
<td>Unit on</td>
<td>UNON</td>
</tr>
<tr>
<td>3</td>
<td>End of File Detected</td>
<td>EOF</td>
</tr>
</tbody>
</table>

9-33. ERRC. The ERRC is a system error code inserted by IOCS or a device handler upon detection of an error. ERRC should be interrogated after each call to IOCS by the application program. Appendix E lists all the error codes for FLP-80DOS.

9-34. PBFFR. The PBFFR field is used by IOCS when assigning a physical buffer for an open logical unit. The user must not change this field.

9-35. UBFFR. The UBFFR (user buffer) field is specified by the user to direct IOCS where to locate the I/O data. This field is left unchanged by IOCS except in I/O Format 3X, in which case it is changed by IOCS to point to the last byte transferred +1. The buffer address is entered least significant byte first. The user should refer to the section regarding the type of format being used.

9-36. USIZE. The USIZE field is the user's buffer size (in
bytes), least significant byte first. In I/O Format 2X (LOGICAL BUFFER I/O), the IOCS fills the entire buffer on a read and outputs the entire buffer on a write. If the end of file is reached for format 2X on read operation before the UBFFR is filled, then USIZE is changed by IOCS to the actual number of bytes read. In I/O Format 3X (BINARY RAM IMAGE I/O), the USIZE parameter specifies the number of bytes to be saved (rounded mod-124). The user should refer to the section regarding the type of format being used.

9-37. NREC. The NREC field tells the device handler the number of physical records to read, write or skip. This field is used by IOCS.

9-38. HSCR. The HSCR field is available to the device handler to use for scratch variables associated with logical unit.

9-39. ISCR. The ISCR is reserved for IOCS to use as scratch variables.

9-40. HOW TO USE IOCS

9-41. When a user wishes to access an I/O device via IOCS, the following procedure should be followed.

9-42. SET UP A VECTOR. The vector should be first initialized to zeros, then appropriate data should be placed into the vector. In FLP-80DOS, the default vectors 0-5 are available for use by an application program but 0 and 1 are reserved for the console device. Recall that the vectors for LUN's 0-5 already exist; their starting addresses are defined via GETVEC (See Section 13). All other LUN's require that the application program provide the vector space (48 bytes). The following fields should
be preset by the user program: LUN, DVCE, UNIT, FNAME, FEXT, VERS, and USER, if file structured device is used; RQST, FMAT, ERRA (if used) CFLGS; and UBFFR and USIZE if ASCII Line Format, Logical Buffer Format, or Binary Format is used.

1. SET IY equal to the address of the first byte of the vector.

2. OPEN the device. Insert an OPENR (open for read) or OPENW (open for write) request code into the RQST field of the vector, then call IOCS: CALL JIOCS ;the address of JIOCS is shown in Appendix F.

NOTE: The calling address of IOCS (=JIOCS) will not change in further versions of FLP-80DOS.

3. The READ/WRITE request is placed into the RQST field and IOCS is called once for each I/O operation.

4. CLOSE THE DEVICE. The CLOSE request is placed into the RQST field of the vector and IOCS is called when no more I/O is to be done. FLP-80DOS uses 04H as end-of-file indicator for ASCII files.

5. After each call to IOCS, the ERRC field should be checked for errors. If it is zero, then no errors were encountered. Some errors are fatal or non-recoverable, such as DISK I/O ERROR. Others are merely indicators, such as END OF FILE.

Idiosyncracies of the Flexible Disk Handler are described in Section 10 this manual.
9-43. DEVICE HANDLER REQUIREMENTS

9-44. Each device handler must begin with a displacement table for each of the supported IOCS requests. If a function is supported, the displacement is added to the table address to determine the handler entry point for a given function. If a function is not supported, then IOCS generates an error code and returns to caller. The following is an example of paper tape device handler.

```
PTAPE  DEFB  3  ; The largest request code supported
      DEFB  PTOPEN-$ ; Displacement for OPENR (RQST 0)
      DEFB  0     ; OPENW is not supported (RQST 1)
      DEFB  PTCLOSE-$ ; Displacement for CLOSE (RQST 2)
      DEFB  PTREAD-$ ; Displacement for READ (RQST 3)
PTOPEN -- ; Initialize Paper Tape RDR
PTCLOSE -- ; Disable Paper Tape Reader
PTREAD  -- ; Read a Byte
RET
```

9-45. The first byte of the handler specifies that the largest request supported is 3. Any request code between 0 and 3 must have a zero displacement if it is not supported. When a device handler is opened, it must pass the physical buffer size back to IOCS in the BC register. If the .BIN data type is supported by a device, the handler must generate and/or strip off all non-data bytes such as sync characters and CRC. For devices that do not support REWIND, IOCS will print the following message on the console when REWIND is requested:

"REWIND XXY, ENTER C WHEN READY:"

Where XX is the device mnemonic and Y is the unit number.

NOTE I/O Device Handlers must not destroy the alternate register set or the main set of registers.
9-46. PHYSICAL I/O BUFFERS

9-47. When the user opens a file for use with I/O format 0, 1 or 2 (Byte I/O, ASCII line, or logical record I/O), then IOCS allocates a physical record buffer for the device. When the handler returns control to IOCS after an OPENR or OPENW, the BC register contains the physical record size (in bytes) for the device. IOCS then allocates that number (IF >1) of bytes and assigns a physical buffer number to PBFFFR in the vector. IOCS maintains a physical buffer allocation table and can allocate up to 16 concurrent buffers.

9-48. The allocation table contains the start address for each physical buffer which is shown in following table:

<table>
<thead>
<tr>
<th>TABLE 9-5. PHYSICAL BUFFER ALLOCATION TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFFRO DEFS 2     ; Present location of I/O Buffer #0.</td>
</tr>
<tr>
<td>BUFFR1 DEFS 2     ; Present location of I/O Buffer #1.</td>
</tr>
<tr>
<td>BUFFR2 DEFS 2     ; Present location of I/O Buffer #2.</td>
</tr>
<tr>
<td>BUFFR3 DEFS 2     ; Present location of I/O Buffer #3.</td>
</tr>
<tr>
<td>BUFFR4 DEFS 2     ; Present location of I/O Buffer #4.</td>
</tr>
<tr>
<td>BUFFR5 DEFS 2     ; Present location of I/O Buffer #5.</td>
</tr>
<tr>
<td>BUFFR6 DEFS 2     ; Present location of I/O Buffer #6.</td>
</tr>
<tr>
<td>BUFFR7 DEFS 2     ; Present location of I/O Buffer #7.</td>
</tr>
<tr>
<td>BUFFR8 DEFS 2     ; Present location of I/O Buffer #8.</td>
</tr>
<tr>
<td>BUFFR9 DEFS 2     ; Present location of I/O Buffer #9.</td>
</tr>
<tr>
<td>BuffRA DEFS 2     ; Present location of I/O Buffer #A.</td>
</tr>
<tr>
<td>BuffRB DEFS 2     ; Present location of I/O Buffer #B.</td>
</tr>
<tr>
<td>BuffRC DEFS 2     ; Present location of I/O Buffer #C.</td>
</tr>
<tr>
<td>BuffRD DEFS 2     ; Present location of I/O Buffer #D.</td>
</tr>
<tr>
<td>BuffRE DEFS 2     ; Present location of I/O Buffer #E.</td>
</tr>
<tr>
<td>BuffRF DEFS 2     ; Present location of I/O Buffer #F.</td>
</tr>
</tbody>
</table>

9-49. IOCS allocates the first buffer with a buffer number of 0.
This number is placed in the PBFFR field of the VECTOR. The buffer number placed in the vector is $\text{FF}_H$ for byte oriented devices (physical buffersize = 1).

9-50. The actual physical buffers contain the number of bytes specified by $(BC)$ after an OPENR or OPENW plus eight bytes for deblocking and de-allocaton as follows:

(start of buffer)
DEFS 2 ; Size of Buffer (not including first 8 bytes)
DEFS 2 ; Temporary Buffer Pointer
DEFS 2 ; The physical record size = $(BC)$ after OPENR or OPENW
DEFS 2 ; Last address transferred after a read

9-51. When a logical unit which had a physical buffer assigned to it is closed, IOCS de-allocates the buffer space and compresses the buffers, removing any holes in the buffer block.

9-52. SYSTEM INTERRUPT TABLE

9-53. The top 32 bytes in the user RAM space are reserved for the system Interrupt table. The program module DK reserves a 32 byte buffer for this purpose so the end address of OS.BIN [255] can be positioned at the top of RAM (see SYSGEN Section 15). During the system boot sequence the Monitor automatically calculates the top of RAM memory and stores that value in TOR (OFFOOH). The following displacements from TOR have been reserved for system devices.

<table>
<thead>
<tr>
<th>TOR DISPLACEMENT</th>
<th>DEVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Operating System Minimal Listener</td>
</tr>
<tr>
<td>7</td>
<td>LP:</td>
</tr>
<tr>
<td>9</td>
<td>PR:</td>
</tr>
<tr>
<td>11</td>
<td>PP:</td>
</tr>
<tr>
<td>13</td>
<td>CR:</td>
</tr>
<tr>
<td>21-31</td>
<td>Reserved for User Interrupt Devices</td>
</tr>
</tbody>
</table>
9-54. The Open routine within a device handler may use the value in TOR (FF00H) and its designated displacement (see above table) to calculate the position of its interrupt vector. The open routine should place the MSB of the Interrupt vector into the I register and output the LSB to the designated PIO. The open routine should also place the address of the device interrupt service routine into the interrupt vector in the interrupt table. (See Paragraph 9-63).

9-55. IOCS MEMORY MAP

9-56. The Default Logical Unit Table, the Logical Unit Redirect table and the IOCS buffer allocation table are included in the program module IOCS. IOCS is an operating system module which is linked into OS.BIN [255] during the SYSGEN procedure (See Section 15). IOCS physical I/O buffers are allocated dynamically downward from the operating system as outlined in figure 9-1.

9-57. The Logical Unit Redirect Table contains the assignment of device handlers to logical unit numbers by the Monitor ASSIGN command. Each item in the table is 15 bytes long. These 15 bytes correspond exactly with the first 15 bytes of the IOCS vector (See Section 9-4). Up to 6 items can be placed into the redirect table. The redirect table is terminated by a logical unit number (1st byte of an item) of FFH (Recall that this is the Logical Unit Number which cannot be redirected).

9-58. Bottom of Allocated RAM (BALR) is a pointer to the bottom of the system routines less any physical buffers allocated (dynamically) by IOCS. The BALR pointer is maintained in scratchpad locations FF02-FF03H and is updated by IOCS as it allocates and de-allocates physical buffers.
FIGURE 9-1. IOCS MEMORY MAP

- INTERRUPT TABLE (32 BYTES)
- OS.BIN[255]
- IOCS PHYSICAL BUFFERS
- BALR
- USER RAM
9-59. WRITING A DEVICE HANDLER

9-60. CHARACTER-ORIENTED DEVICES

9-61. Introduction. Device handlers for character oriented devices are rather straightforward in their design. The paper tape reader for FLP-80DOS is included in Section 12 of this manual. The following discussion examines the design in detail.

9-62. Design Criteria. The handler is to input one character at a time. It will be interrupt driven. Control and I/O will be done via a Z80 PIO, which takes two sequential port addresses (in this case, DOH for control and D1H for data). The control port number is contained in a byte in the handler.

1. Disable interrupts while the Z80 PIO is programmed. The reader is directed to the "Z80 PIO Technical Manual" for details of programming the device.
2. Access the control port number. The least significant bit is used as a ready flag.
3. Access item number in Interrupt Pointer Table (C-reg = 0, the first item).
4. Access the interrupt handler address (RINT).
5. This address is place into the first items of the Interrupt Pointer Table.
6. Program the Z80 PIO for proper operation.
7. Initialize the status bit to zero (not ready).
8. Program the interrupt handler vector into Z80 PIO (LS byte) and into the Z80 I-register (MS byte). Z80 Interrupt Mode 3 is used throughout FLP-80DOS. The reader is referred to the "Z80 CPU Technical Manual" for further discussion.
9. Set up a physical buffer size of one for one byte transfers (BC-reg = 1).

10. Perform first I/O operation to start reader.

11. Enable interrupts and return to caller.

9-64. Close Process. No operation is performed; return to caller.

   1. Access port number and strip off status bit (bit 0).
   2. Set up an initial time out of about 250 msec.
   3. Enable interrupts.
   4. Check the status flag. The status flag is set in the RINT routine when an interrupt occurs.
   5. If the status flag is not set (not ready), then check for immediate return. If immediate return is set, then return to the caller (IOCS) without performing any input operation. Otherwise check for time out. If time out occurs, call the system Error Handler(EH) (Described in Section 13) with the time out error code in the A-reg. Then reinitialize the timeout counter and loop on status. Thus the time out error message will be output periodically until the system is reset or the device goes ready.
   6. If the status flag is set (device ready, data is available), then read the data from the data port. Reset the status flag and the immediate return flag. Complement the data and return it in the A-register. The complement operation is dependent upon the interface to the device.
9-66. RECORD ORIENTED DEVICES.

9-67. Introduction. Device Handlers which operate on a physical record basis must meet additional requirements for IOCS. The handler must place bytes directly into the IOCS buffer rather than passing them via the A-register. The handler must also properly process multiple record requests by IOCS. An optional Card Reader Driver is shown in Section 12 of this manual. The Card Reader Driver is supplied on the FLP-80DOS diskette in source and relocatable object format, but it is not integrated into the system. The following discussion examines the design in detail.

9-68. Design Criteria. The handler is to input one card at a time. The physical buffer size is 80 bytes plus 2 more for carriage return and line feed. Control and I/O will be done via a Z80 PIO which takes 4 sequential port addresses (starting at 69H in this case). The first port number is taken from a byte in the handler. The handler uses interrupts where each interrupt corresponds to one card column read. Thus, after card pick, the handler must process 80 fast, sequential interrupts. The handler must read as many cards as are requested by IOCS.

9-69. Open Process. Interrupts are disabled. The card reader interrupt handler address (CRDRDR) is placed into the Interrupt Pointer Table. The least significant byte of the interrupt vector is programmed into the Z80 PIO. The most significant byte is loaded into the Z80 I-register (Interrupt Mode 3 is used). The PIO is programmed for handshake (See the Z80 PIO Technical Manual for full details). A physical buffer size of 82 is returned to IOCS via the BC-register.

9-70. Close Process. No operation is performed; return to caller.

   1. The number of records (NREC) being requested by IOCS
is accessed and saved in the handler scratch area (HSCR) of the IOCS vector. Then NREC is set to zero. NREC becomes the counter of the actual number of records (cards) read by the handler.

2. The IOCS physical buffer address is accessed. This is the starting address where the handler is to place data which is read. Recall that this buffer was dynamically allocated by IOCS when the device was opened.

3. The card reader is tested for ready condition. If it is not ready after 4 seconds, then a time out error message is issued. The time out is reprogrammed and loop on status. Note that immediate return is allowed here (IRET bit).

4. When the card reader goes ready, PIO local interrupts are enabled and a card pick is forced. CPU interrupts are enabled.

5. A loop is entered until 80 columns have been read. The interrupt handler (CRDRDR) has the responsibility of reading the data and incrementing the column counter (A-reg).

6. Interrupt Handler. (CRDRDR). The interrupt handler reads data from the PIO ports after each interrupt. One interrupt corresponds to one card column. The data is converted from hollerith image to ASCII via the HOLTAB table. The data is then stored into the physical buffer, pointed to by the DE register. The DE-register is then incremented, as is the column counter (A-register). Return from interrupt is done after reenabling interrupts.

7. After all 80 columns of a card have been read CPU and local PIO interrupts are disabled. The number of records (NREC) is incremented.
8. The first column of the card is accessed in the physical buffer. If the byte is EOT (ASCII 04H, punch 9-7), then this is the end of file indicator. Upon end of file, the end of file error code is placed in the IOCS vector, the buffer pointer is updated, and return is made to caller.

9. If end of file was not found, then trailing blanks are compressed in the physical buffer. Carriage return and line feed are appended to the card image.

10. The number of records read is checked. If all have been read, then the IOCS buffer pointer is updated and return to caller. Otherwise, another card pick and read is initiated.
; THIS PROGRAM IS TO DEMONSTRATE SOME OF THE USES
; OF IOCS. THE PROGRAM READS A LINE OF TEXT FROM
; A FILE ON DISK UNIT 0 IN BYTE I/O FORMAT. A COUNTER
; IS KEPT TO IDENTIFY EACH LINE AND IS PLACED AT THE
; BEGINNING OF EACH LINE. THE NEW LINE WITH THE LINE
; NUMBER IS THEN OUTPUT TO ANOTHER FILE ON DISK UNIT
; 0 IN ASCII LINE FORMAT. THE FILE BEING READ IS
; CALLED 'PROGRAM.INP'. THE NEW FILE IS CALLED 'PROGRAM
; OUT'. THE USER MAY USE THIS PROGRAM AS A GUIDE TO
; SETTING UP VECTORS AND FOR USING IOCS TO PERFORM
; VARIOUS FUNCTIONS. THE PROGRAM USES GLOBAL REFERENCES
; AND MUST BE LINKED WITH SYSLNK.OBJ (SHIPPED ON
; THE SYSTEM DISKETTE).

; GLOBAL JTASK
GLOBAL JIOCS
GLOBAL PTXT

; THIS SECTION CLEARS THE INPUT AND OUTPUT VECTORS

LD HL, INVEC ; HL -> INPUT VECTOR
LD DE, INVEC ; DE -> INPUT VECTOR + 1
INC DE
XOR A
LD (HL), A ; LOAD INITIAL 0 IN VECTOR
SET UP LOOP COUNT TO ...
... ZERO BOTH VECTORS.

; THIS SECTION STUFFS THE INPUT VECTOR AND Prepares
; TO OPEN THE INPUT FILE FOR READING.

LD IY, INVEC ; IY -> VECTOR ADDRESS
LD (IY+0), OFFH ; SET LUN = FF
LD (IY+1), 'D' ; SET DEVICE TO DK:
LD (IY+2), 'K'
LD (IY+3), 'O'
LD (IY+4), 'P'
LD (IY+5), 'R'
LD (IY+6), 'G'
LD (IY+7), 'R'
LD (IY+8), 'M'
LD (IY+9), 'I'
LD (IY+10), 'N'
LD (IY+11), 'P'
LD (IY+12), 'P'
LD (IY+13), 0 ; SET VERSION TO 0
LD (IY+14), 1 ; SET USER # TO 1
LD (IY+15), 0 ; REQUEST TO OPEN FOR READ
LD (IY+16), 4 ; FORMAT TO BYTE I/O, 4 REC !
LD (IY+17), 0 ; CLEAR ERROR RETURN ADDR
LD (IY+18), 0
LD (IY+19), 0
LD (IY+20), 0
LD (IY+21), 10H ; SET CFLAGS TO PRINT ERRORS
LD (IY+22), 0 ; CLEAR STATUS FLAGS
LD (IY+23), 0 ; SET USIZE TO 124 (7CH)
LD (IY+24), 0
LD (IY+25), 0
LD (IY+26), 0
LD (IY+27), 0 ; OPEN INPUT FILE
CALL JIOCS
LD A, (IY+28) ; TEST FOR ERRORS
NAME
'MOSTEK FLP-80 ASSEMBLER V2.0 PAGE 00C

DATAFILE = DKO:EXAM

ADDR  OBJECT  ST #  SOURCE STATEMENT

0074  A7      0059  AND  A  0060  JP  NZ,ERMSG  ;IF FOUND, PRINT MSG.
0075  C23F01' 0061  ;
0076  ; THIS SECTION STUFFS THE OUTPUT VECTOR AND PREPARES
0077  ; TO OPEN THE OUTPUT FILE FOR WRITE.
0078  FD21CA01' 0064  LD  IY,OUTVEC  ;IY -> VECTOR ADDRESS
0079  FD360FF  0065  LD  (IY+0),OFFH  ;SET LUN = FF
0080  FD360144  0066  LD  (IY+1),'D'  ;SET DEVICE TO D:
0081  FD36024B  0067  LD  (IY+2),'K'
0082  FD360330  0068  LD  (IY+3),'O'
0083  FD360450  0069  LD  (IY+4),'F'
0084  FD360552  0070  LD  (IY+5),'R'
0085  FD36054F  0071  LD  (IY+6),'O'
0086  FD360747  0072  LD  (IY+7),'G'
0087  FD360852  0073  LD  (IY+8),'R'
0088  FD36094D  0074  LD  (IY+9),'M'
0089  FD360A4F  0075  LD  (IY+10),'O'
008A  FD360B55  0076  LD  (IY+11),'U'
008B  FD360C54  0077  LD  (IY+12),'T'
008C  FD360D00  0078  LD  (IY+13),0  ;SET VERSION TO 0
008D  FD360E01  0079  LD  (IY+14),0  ;SET USER # TO 1
008E  FD360F01  0080  LD  (IY+15),1  ;REQUEST TO OPEN FOR REF
008F  FD361014  0081  LD  (IY+16),14H  ;FORMAT=ASCII LINE, 4 RE
0090  FD361300  0082  LD  (IY+19),0  ;CLEAR ERROR RETURN ADDR
0091  FD361400  0083  LD  (IY+20),0
0092  FD361510  0084  LD  (IY+21),10H  ;SET CFLAGS TO PRINT ERR
0093  FD361600  0085  LD  (IY+22),0  ;CLEAR STATUS FLAGS
0094  FD36177C  0086  LD  (IY+27),07CH  ;SET USIZE TO 124 (7CH)
0095  FD361C00  0087  LD  (IY+28),0
0096  CD5F00'  0088  CALL  JIOCS  ;OPEN INPUT FILE
0097  FD7E17  0089  LD  A,(IY+23)  ;TEST FOR ERRORS
0098  A7      0090  AND  A
0099  C23F01'  0091  JP  NZ,ERMSG  ;IF FOUND, PRINT MSG.
009A  ; THIS SECTION READS DATA FROM THE INPUT FILE,
009B  ; ADDS THE LINE # TO THE BEGINNING OF THE LINE,
009C  ; AND OUTPUTS THE NEW LINE TO THE OUTPUT FILE.
009D  FD219A01'  0097  READ  LD  IY,INVEC  ;IY -> INPUT VECTOR
009E  FD360F03  0098  LD  (IY+15),3  ;REQUEST FOR READ
009F  21FD01'  0099  LD  HL,INBUF  ;HL -> BUFFER
00A0  CDD900'  0100  INLOOP  CALL  JIOCS  ;READ 1 BYTE FROM FILE
00A1  57      0101  LD  D,A  ;STORE CHAR IN D REG
00A2  FD7E17  0102  LD  A,(IY+23)  ;TEST FOR ERROR
00A3  A7      0103  AND  A
00A4  C23F01'  0104  JP  NZ,ERMSG  ;RESTORE CHAR IN A
00A5  7A      0105  LD  A,D  ;TEST FOR END OF FILE
00A6  FE04  0106  CP  04H  ;EXIT IF FOUND.
00A7  CA4701'  0107  JP  Z,EXIT  ;EXIT IF FOUND.
00A8  77      0108  LD  (HL),A  ;STORE CHAR IN BUFFER
00A9  23      0109  INC  HL  ;INC BUFFER POINTER
00AA  FD3A  0110  CP  0AH  ;TEST FOR LF
00AB  20E9  0111  JR  NZ,INLOOP-S  ;NO, CONTINUE READING
00AC  0112  ;
00AD  3A4E02'  0113  LD  A,(LINE)  ;GET CURRENT LINE NUMBER
00AE  3C      0114  INC  A  ;INC NUMBER
00AF  324E02'  0115  LD  (LINE),A  ;STORE NEW NUMBER
00B0  21FA01'  0116  LD  HL,OUTBUF  ;HL -> OUTPUT BUFFER
DDT

03JECR ST # SOURCE STATEMENT

DATASET = DKO:EXAM

; GET UPPER DIGIT OF LINE #

; CONVERT DIGIT TO ASCII

; STORE ASCII CHAR IN BUFFER

; INC BUFFER POINTER

; GET LOWER DIGIT

; CONVERT TO ASCII

; STORE ASCII CHAR IN BUFFER

; STORE SPACE AFTER LINE #

; IY -> OUTPUT VECTOR

; STORE ADDRESS OF BUFFER...

; ... IN BUFFER FOR WRITING.

; TURN OFF ERROR PRINT

; OUTPUT NEW LINE

; TEST FOR ERROR

; NO ERROR, GET NEXT LINE

; THIS SECTION PRINTS AN ERROR MESSAGE

; AND EXITS AFTER CLOSING THE FILES.

; HL -> MESSAGE TO PRINT

; SET FOR CONSOLE DEVICE

; PRINT MESSAGE

; IY -> INPUT VECTOR

; IY -> OUTPUT VECTOR

; PREPARE TO WRITE 04H AT...

; ... END OF OUTPUT FILE.

; REQUEST TO WRITE

; OUTPUT 04

; CLOSE REQUEST

; CLOSE OUTPUT FILE

; RETURN TO MONITOR

; ROUTINE TO CONVERT 4 BIT HEX INTO ASCII

; ASCII AND 0FH

; ADD A, 90H

; DAA

; ADC A, 40H

; DAA

; RET

; ERROR FOUND DURING EXECUTION
ADDX | OBJECT | ST | # | SOURCE STATEMENT | DATASET = DK0:EXAM

5220464F
554E4420
4455524E
47204558
45435554
494F4E

'0199 03 0175 DEFB 03H
0176 ;

'>019A 0177 INVEC DEFS 48

'>01CA 0178 OUTVEC DEFS 48

'>01FA 0179 OUTBUF DEFS 3

'>01FD 0180 INBUF DEFS 80

'024D 00 0181 DEFB 0

'024E 00 0182 LINE DEFB 0

0183 END

ERRORS=0000
10-1. INTRODUCTION

10-2. All calls for communication with the disk will be through the Floppy Disk Handler. Because a disk is not a character oriented device, all calls will be for a file whose minimum length is 1 record of 124 bytes. By maintaining a directory in the first two tracks of the disk, file operations may take place independent of the physical location of the data on the disk. The Disk Handler System not only provides file reading and writing capability but special pointer manipulation, record deletion and insertion, and directory manipulations such as file creation, renaming, and deletion. The FDH outlined here can serve as a building block for a file maintenance system, a disk based Assembler and Text Editor, BASIC and other high level languages.

10-3. COMMUNICATION

10-4. The FDH can be communicated with by a calling vector (equivalent to the IOCS calling vector pointed to by IY) which contains all parameter information with each parameter having a fixed displacement from the vector pointer. This vector has been appended to the I/O Control System vector. The purpose of the IOCS is to generalize all calls to the peripheral devices so as to dissolve any device dependence of data structure. However, because the disk is a file oriented device as opposed to being a character oriented device, much additional calling information is required. The required entries into the 48 byte IOCS defined vector are listed as follows.
## 10-5. DOS RELATED VECTOR PARAMETERS

<table>
<thead>
<tr>
<th>FIELD</th>
<th>#BYTES</th>
<th>OFFSET</th>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>FORM</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>(IY+3)</td>
<td>UNIT</td>
<td>UNIT number</td>
<td>(ASCII)</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>(IY+4)</td>
<td>FNAM</td>
<td>Filename</td>
<td>(ASCII)</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>(IY+10)</td>
<td>FEXT</td>
<td>File extension</td>
<td>(ASCII)</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>(IY+13)</td>
<td>VERS</td>
<td>File Version</td>
<td>(Binary)</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>(IY+14)</td>
<td>USER</td>
<td>User number</td>
<td>(Binary)</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>(IY+15)</td>
<td>RQST</td>
<td>Request Code</td>
<td>(Binary)</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>(IY+23)</td>
<td>ERRC</td>
<td>Error Code</td>
<td>(Binary)</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>(IY+25)</td>
<td>UBFFR</td>
<td>User's Buffer Address</td>
<td>(Binary)</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>(IY+29)</td>
<td>NREC</td>
<td>Number of records to be transferred</td>
<td>(Binary)</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>(IY+30)</td>
<td>SCTR</td>
<td>Current sector pointer</td>
<td>(Binary)</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>(IY+31)</td>
<td>TRCK</td>
<td>Current track pointer</td>
<td>(Binary)</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>(IY+32)</td>
<td>LSCTR</td>
<td>Last Sector</td>
<td>(Binary)</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>(IY+33)</td>
<td>LTRK</td>
<td>Last Track</td>
<td>(Binary)</td>
</tr>
</tbody>
</table>
10-6. CALLING CONVENTIONS

10-7. There are three ways for a user to communicate with the Disk System. The user may make calls through the IOCS defined general purpose request codes 0-7H. These request codes are converted to a set of Macros of request codes made up from the complete set of DOS request codes. This permits the disk system to be used as if it were any standard character type device. The second way to communicate is through the complete set of disk request codes. This allows use of more complex but more powerful requests that would be used by sophisticated environments such as the Text Editor. The third communication technique is through direct disk controller commands. See Section 11 for more information.

10-8. GENERAL PURPOSE IOCS DISK MACRO REQUESTS

CALLING SEQUENCE - LD A,0 ;FDH JTASK CODE
                    CALL JTASK ;CALL FDH VIA JTASK

<table>
<thead>
<tr>
<th>RQST CODE</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>02H</td>
<td>CLOSE</td>
<td>The Close command will store off all linkage information into the directory and update the sector and track maps of the diskette containing the file.</td>
</tr>
<tr>
<td>03H</td>
<td>READ</td>
<td>Read Next N Records - Reads the next</td>
</tr>
<tr>
<td>RQST CODE</td>
<td>NAME</td>
<td>DESCRIPTION (CONT.)</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>04H</td>
<td>WRITE</td>
<td>Insert N Records - Allocates and writes N records from memory starting at the Data Transfer Address, (UBFFR) with the first record written after the current one. The pointer will be left positioned at the last record written.</td>
</tr>
<tr>
<td>05H</td>
<td>REWIND</td>
<td>The Rewind command positions the pointer back to the directory entry for the file. All records will now be written before any existing records, or the first record may now be read.</td>
</tr>
<tr>
<td>06H</td>
<td>INIT</td>
<td>Initialize - Reads sector and track maps from all disks which are ready and clears active file table of the FDH.</td>
</tr>
<tr>
<td>07H</td>
<td>ERASE</td>
<td>Erase File - Writes reformatted directory entry over the entry for the file, de-allocates all re-</td>
</tr>
</tbody>
</table>
cords in the file, removes the active file entry from the table, and rewrites the updated sector map. Any records following one not readable will not be reallocated. The file must be opened before it can be erased.
10-9. COMPLETE DOS REQUEST CODES

REQUEST CODE   DESCRIPTION

18H  Initialize - Reads sector and track maps from all diskettes which are ready and clears active file table of the FDH. This is equivalent to request code 06H.

1CH  Open File - Finds file in directory and creates an entry in the active file table; pointer remains on the directory but the number of records in file is placed in NREC. If file has a BIN extension, UBFFR is set to the binary load address.

1EH  Create File - Creates directory entry for a file and creates entry in active file table. Error is returned if file already exists and the operation is aborted. Pointer is positioned to the directory entry for the file.

20H  Close File - Writes updated directory entry back to the Disk Directory, removes the active file table entry, and rewrites the updated sector map. This is equivalent to request code 02H.

22H  Erase File - Writes reformatted directory entry over the entry for the file, de-allocates all records in the file, removes the active file entry from the table, and rewrites the updated sector map. Any records following one not readable will not be reallocated. This is equivalent to re-
COMPLETE DOS REQUEST CODES (CONT.)

REQUEST CODE  OPERATIONS

quest code 07H.

24H Rename File - Takes a second filename and filetype starting in the second parameter vector (IY+48) and verifies that it does not exist or takes error exit. The directory entry for the first filename is replaced by the one for the second. Two contiguous I/O vectors must be defined. The first is a complete 48 byte I/O vector and contains the current name of the file (which must be open). The second contains the new name of the file and may consist of only the first 15 bytes of the standard I/O vector (contains only the new filename).

26H Rewind File - Repositions the pointer for the file to the directory entry with the next record pointing to the first record to be read by Read Next Record. This is equivalent to request code 05H.

28H Read Next N Records - Reads the next N records, where N is in (IY+29), into memory starting at transfer address given in (UBFFR). The pointer will be positioned on the last record read and if error exit is required, NREC contains actual number of records transferred. This is equivalent to request code 03H.
COMPLETE DOS REQUEST CODES (CONT.)

REQUEST CODE  OPERATIONS

2AH  Read Current Record - Reads the single current record into memory starting at the transfer address. The pointer will not be moved.

2CH  Read Previous Record - Reads the single record previous to the current one into memory starting at the transfer address given. The pointer will be positioned on this record.

2EH  Skip Forward N Records - Moves pointer N records forward but no data will be transferred.

30H  Skip Backward N Records - Moves the pointer N records backward but no data will be transferred.

32H  Replace (Rewrite) Current Record - Rewrites the single current record from memory starting at the Data Transfer Address. The pointer is not moved.

34H  Insert N Records - Allocates and writes N records from memory starting at the Data Transfer Address, (UBFFR) with the first record coming after the current one. The pointer will be left positioned at the last record written. This is equivalent to request code 04H.

36H  Delete N Records - The current record and the next N-1 records are de-allocated and removed from the file.
### Figure 10-1. Effects of FDH Commands

<table>
<thead>
<tr>
<th>REQUEST</th>
<th>FILENAME, EXT, VERS, USER</th>
<th>NREC</th>
<th>UBFFR</th>
<th>SCTR/TRK</th>
</tr>
</thead>
<tbody>
<tr>
<td>00H OPENR</td>
<td>-</td>
<td>File length</td>
<td>Load Address</td>
<td>Directory</td>
</tr>
<tr>
<td>01H OPENW</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>Directory</td>
</tr>
<tr>
<td>02H CLOSE</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>Unknown</td>
</tr>
<tr>
<td>03H READ</td>
<td>-</td>
<td>Number sectors read</td>
<td>(UBFFR) + N * 124</td>
<td>Last sector read</td>
</tr>
<tr>
<td>04H WRITE</td>
<td>-</td>
<td>Number sectors written</td>
<td>(UBFFR) + N * 124</td>
<td>Last sector written</td>
</tr>
<tr>
<td>05H REWIND</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>Directory</td>
</tr>
<tr>
<td>06H INIT</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>Unknown</td>
</tr>
<tr>
<td>07H ERASE</td>
<td>-</td>
<td>File length</td>
<td>-</td>
<td>Unknown, File closed</td>
</tr>
<tr>
<td>1CH OPEN</td>
<td>-</td>
<td>File length</td>
<td>Load address</td>
<td>Directory</td>
</tr>
<tr>
<td>1EH CREATE</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>Directory</td>
</tr>
<tr>
<td>24H RENAME</td>
<td>Moved from vector following this one.</td>
<td>0</td>
<td>-</td>
<td>Directory</td>
</tr>
<tr>
<td>2AH RDCURR</td>
<td>-</td>
<td>0</td>
<td>(UBFFR)+124</td>
<td>Previous sector</td>
</tr>
<tr>
<td>2CH RDPRVR</td>
<td>-</td>
<td>1</td>
<td>(UBFFR)+124</td>
<td>Previous sector</td>
</tr>
<tr>
<td>2EH SKIPFWD</td>
<td>-</td>
<td>Number successful skips</td>
<td>-</td>
<td>Last sector read</td>
</tr>
<tr>
<td>30H SKIPBKD</td>
<td>-</td>
<td>Number successful skips</td>
<td>-</td>
<td>Last sector read</td>
</tr>
<tr>
<td>32H RPCURR</td>
<td>-</td>
<td>0</td>
<td>(UBFFR)+124</td>
<td>Previous sector</td>
</tr>
<tr>
<td>36H DELETE</td>
<td>-</td>
<td>Number records deleted</td>
<td>-</td>
<td>Previous sector</td>
</tr>
<tr>
<td>3CH JUMP</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3EH DISKID</td>
<td>Disk id (11 characters)</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>40H STATUS</td>
<td>Sectors available, used and bad (2 bytes each)</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
3CH Jump - Go to sector/track defined by SCTR (IY+30) and TRK (IY+31). No data is transferred.

3EH Read Disk Id - Loads disk name (11 bytes) into filename, extension and version fields of the I/O vector.

40H Read Status - Loads available, used and bad sector counts into filename field of the vector. 2 bytes each (total of 6 bytes).

10-10. ERROR RETURN

10-11. The error parameter is in (IY+23) and is returned at the end of a DOS operation the contents of (IY+23) is also in the accumulator. A 0 return indicates that no error has occurred. The error return codes are:

<table>
<thead>
<tr>
<th>ERROR CODE</th>
<th>INTERPRETATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>01H</td>
<td>Invalid Operation - A request word was specified which is not a valid DOS request.</td>
</tr>
<tr>
<td>02H</td>
<td>Duplicate File - An attempt was made to create a directory entry for a file that already exists. Can occur only on create or rename. In the case of OPENW, the file is opened but this error is reported only as a flag.</td>
</tr>
<tr>
<td>03H</td>
<td>Active File Table Full - An attempt was made to insert another entry in the active file table when it is full. Can occur only on open or create. A maximum of 7 files may be open at any time.</td>
</tr>
<tr>
<td>ERROR CODE</td>
<td>INTERPRETATION</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
</tr>
<tr>
<td>05H</td>
<td>Directory Full - There is no more space to insert another directory entry.</td>
</tr>
<tr>
<td>06H</td>
<td>Write Protect - Diskette is write protected and an attempt has been made to write on it.</td>
</tr>
<tr>
<td>08H</td>
<td>File Not Open - An attempt was made to close or perform some record operation on a file which had not been opened. Can occur on any operation except initialize, open, or create.</td>
</tr>
<tr>
<td>09H</td>
<td>End of File - An attempt was made to advance the pointer beyond the last record in the file. The error can occur on any read, delete or skip operation. In the case of delete it indicates an attempted delete operation on the directory.</td>
</tr>
<tr>
<td>0AH</td>
<td>Disk Error - A disk I/O error occurred during the operation. Data may have been lost. Can occur on any operation except rewind.</td>
</tr>
<tr>
<td>0BH</td>
<td>Disk Full - Diskette is full and will not allow the allocation of another record. Can occur only on insert. The number of records successfully transferred is left in NREC. The file must be explicitly closed or erased</td>
</tr>
<tr>
<td>ERROR CODE</td>
<td>INTERPRETATION</td>
</tr>
<tr>
<td>------------</td>
<td>---------------</td>
</tr>
<tr>
<td>OCH</td>
<td>Pointer Error - The pointers read do not agree with the next or previous record. Can occur on any record operation except rewind. Pointer errors occur because a sector is not readable or because an application program has written on a disk without initializing the handler first, or two diskettes were used with the same Disk ID.</td>
</tr>
<tr>
<td>ODH</td>
<td>Directory or map transfer error. A read or write error occurred during operations involving the disk directory or sector and track maps. If operation occurred during a close or erase, directory or maps could be destroyed.</td>
</tr>
<tr>
<td>OEH</td>
<td>File Already Open - An attempt was made to open or create a file which is currently active.</td>
</tr>
<tr>
<td>OFH</td>
<td>Disk Not Ready - Can occur on any operation when a diskette is not fully inserted and door closed.</td>
</tr>
<tr>
<td>10H</td>
<td>Wrong Disk - A file is being accessed on a disk whose ID is different from the one currently in memory. This can occur if disks are changed during operations without initializing. Can occur only on close, open and erase. Error can be avoided by initializing diskette before operations begin.</td>
</tr>
<tr>
<td>11H</td>
<td>Non-Existent Disk - A unit number has been specified which is not supported by the FDH. Typically, units DK2 or DK3. See Section 15 for details on how to SYSGEN a system to handle more</td>
</tr>
</tbody>
</table>
than two disk drives.

1AH
Beginning Of File - An attempt was made to move the pointer backwards past the beginning of the file. Can occur on read previous record, skip backward, read current record, or rewrite current record.

1BH
Invalid drive, track, or sector. Controller has received invalid drive number, or sector and track out of normal range. Can occur on jump or as a result of some fatal FDH error.

1CH
Controller not able to locate track during seek, read, or write operation.

1DH
Sector not found - Sector address marks not readable.

1EH
CRC Operation - incorrect data has been flagged by CRC check during reading.

20H
Data lost - hardware problem causing data overrun in reading or writing.

10-12. DIRECTORY

10-13. Associated with each diskette is a 4K block of storage divided into 32 sectors which contain the Directory information: track 0, sector 1-26, track 1, sector 1-6. Each sector contains 6 entries of 20 bytes/entry. Each file name will be entered into the Directory or accessed from the Directory by a hash function
for the Filename. This facilitates searches for Directory entries and reduces RAM requirements for the Directory buffer. The format for each Directory entry is the following:

<table>
<thead>
<tr>
<th>BYTE</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-5</td>
<td>Filename, left justified, blank filled</td>
</tr>
<tr>
<td>6-8</td>
<td>Extension</td>
</tr>
<tr>
<td>9</td>
<td>Version - Reserved by Mostek for future use</td>
</tr>
<tr>
<td>10</td>
<td>User</td>
</tr>
<tr>
<td>11</td>
<td>Key - Reserved by Mostek for future use</td>
</tr>
<tr>
<td>12-13</td>
<td>Number of records in file</td>
</tr>
<tr>
<td>14</td>
<td>Sector - Location for first record in file</td>
</tr>
<tr>
<td>15</td>
<td>Track</td>
</tr>
<tr>
<td>16</td>
<td>Sector - Location for last record in file</td>
</tr>
<tr>
<td>17</td>
<td>Track</td>
</tr>
<tr>
<td>18</td>
<td>LSB - Address for Load Location for Binary File,</td>
</tr>
<tr>
<td>19</td>
<td>MSB - or file-date storage if non-Binary File</td>
</tr>
</tbody>
</table>

Each file is composed of one or more records with each record containing trailer information consisting of a forward and backward pointer to locate the next and preceding records respectively. A null pointer (FFH) is used to indicate no next record or no previous record.

10-14. DISK FORMAT

10-15. Should any of the file structures become disjoint by extended periods of erasing and inserting of new and different length files, the operation of backing up a disk (copying) will optimize the actual file structure on the new disk. The FDH treats the disk as a continuous string of 1964 sectors. Every other sector is written on each track and a 5 sector shift is used between starting sectors of contiguous tracks to allow for
head motion. This allows a complete track to be read or written in 2.2 revolutions. The sector allocator looks for the first string of available sectors which is large enough for the file being stored (defined by NREC) when inserts are done.
FIGURE 10-2. FLP-80DOS V2.1 DISKETTE FORMAT

DIRECTORY

SECTOR MAP
Track 1 Sector B
4 byte group

FORMAT: 1 BYTE

TRK1 SCTR7 thru 0 0 0
4 byte group

TRACK 0 - 1 8 9 16 17 24 25,26

SECTOR NUMBER

TRACK 1 - 1 8 9 16 17 24 25,26

First Side

TRACK 76 1 8 9 16 17 24

TRACK 77
Second Side

(AAll Zeros for single-sided Diskette)

TRACK 153 PHYSICAL

0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1

SECTOR B

Last 4 bytes of TRK 1 SECTOR B is all "11"s. The "11" pattern is not a required pattern.
NAME OF DISKETTE AND SPACE ALLOCATION

TRACK 1 SECTOR C

First 73 bytes (bytes 0 thru 72) of TRK1 SCTR C are "11'S.

Diskette name is 11 sequential ASCII bytes starting in byte 73.

Available space on disk is number of sectors. Quantity is located in two hex bytes, least significant byte first, in bytes 84 & 85.

Used space is in bytes 86 & 87, same format.

Number of bad sectors is in bytes 88 & 89, same format.

Diskette number is in bytes 90 & 91 (random number given by the system).

The rest of sector C is not used.

DATA (FILES)
Data begins in TRK1 Sector D.

Double sided disk uses same format.

Track 77 is on second side opposite Track 0, Track 153 is on second side opposite track 76.
10-17. SECTOR AND TRACK FORMATS

10-18. The sector map is stored in track-sector location 1-7 through 1-B. Each bit of each byte in the sector map represents one sector. A bit is set for its respective sector if:

1) The sector has been linked into the doubly linked list of the file structure.

or

2) The operating system has tried without success to store information in the sector and has therefore made this sector not available.

The sector map resides in memory along with FDH and is changed when any file is being altered by erasure, deletion, or insertion. The map is stored off when these operations are complete. Bad sector locations will be de-allocated as if they were in use.

10-18. DISKETTE - IDENTIFICATION

10-19. NAME OF DISKETTE AND SPACE ALLOCATION

Diskette identification and space allocation information reside on track 1 sector C. The first 73 bytes of this sector are 11H -this is not a required pattern. The diskette name is contained in the following sequential bytes (73 through 83). The available space on the diskette (in sectors) is contained in bytes 84 and 85, most significant byte last. The number of used sectors is contained in bytes 86 and 87; the number of bad sectors in bytes 88 and 89. The diskette number is in bytes 90 & 91. This number is randomly assigned at format time. The rest of sector C is not used.
10-20. DATA (FILES)

Data is stored beginning on track 1 sector D. A double-sided disk uses the format described above, except that track 77 is on the opposite side from track 0 and track 153 is on the opposite side from track 76.
SECTION 11

DISK CONTROLLER Firmware (DCF)

11-1. INTRODUCTION

11-2. The Disk Controller Firmware (DCF) interfaces from the Flexible Disk Handler (FDH) to the Mostek FLP-80 Disk Controller Board. Input to the DCF consists of request code, unit number, track number, and sector number. Control of the hardware is exercised via 6 parallel I/O ports which are decoded on the FLP-80 board. A bootstrap sequence is included in the DCF which is used to boot binary files from disk into RAM. Interactive boot and save sequences are also available.

11-3. SOFTWARE CONFIGURATION

11-4. The DCF resides on the SDB-80 in one 2708 PROM located at address ECOOH. It is approximately 1K bytes long.

11-5. CONTROLLER OVERVIEW

11-6. The calling address for the DCF is ECOOH. All requests are made via the 48-byte IOCS parameter vector. See Section 9 of this manual for a complete definition of the vector. After each request is processed and the operation is completed, return is made to the caller. This is not an interrupt driven program; rather, the operation must be completed before further processing can take place. All I/O to the disk is done via a hardware FIFO. A complete sector (128 bytes) is buffered in the FIFO before transfer from/to the DCF takes place. All registers except the flags are preserved by the DCF. After an operation takes place,
the zero flag is set if no error occurred. The zero flag is re-
set if any errors occurred during the operation. If any error
occurred, then bit zero of the vector ERRC parameter is also set.
The Unit number is assumed to be in the vector UNIT parameter,
the track number in TRK, and the sector number in SCTR. The re-
quest code must be in RQST. The unit may be 0-3. The track may be
0-76 for single-sided drives or 0-153 for double sided drives.
The type of drive is indicated by bit 0 of port E2H; if set, a
double-sided drive is indicated. The sector may be 1-26. The re-
ader is referred to the Disk Drive Controller Hardware Manual for
his hardware configuration. A complete software listing of the
controller is given in 'DOPS-80 Program Source Listing', MK78589,
which is available only to OEM users. The following IOCS vector
parameters must be set up; IY must contain the first address of
the vector. Numbers enclosed in parenthesis indicate
displacement from the beginning of the vector.

UNIT (3) - disk unit number (either binary or ASCII)
RQST (15) - request code, described in paragraph 11-7.
UBFFR (25) - transfer address for data for read or write
operation.
SCTR (30) - sector number
TRK (31) - track number

The following parameters are returned:

ERRC (23) - bit 0 set if an error occurred. The error
code is saved in location FF09H. Note that
bit 0 only is set or reset. The rest of the
byte is left unchanged.
SCTR (30) - not changed
TRK (31) - not changed
LSCTR (32) - last sector pointer
LTRK (33) - last track pointer
NSCTR (34) - next sector pointer
NTRK (35) - next track pointer

NOTE: OFFH in LSCTR and LTRK indicate the current record is
the first record in the file. OFFH in NSCTR and NTRK indicate
end of file.

11-7. DISK CONTROLLER REQUESTS

11-8. On the following controller operations, request codes are
placed in RQST, sector and track into SCTR and TRK, and transfer
address into UBFFR. On exit, UBFFR is incremented by 124 if data
is transferred. Only one sector is transferred per call.

<table>
<thead>
<tr>
<th>COMMAND CODE</th>
<th>COMMAND</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>10H</td>
<td>Status</td>
<td>Returns disk drive status of disk drive not ready, disk drive not safe, disk drive write protected (see 11-9 for status code format).</td>
</tr>
<tr>
<td>11H</td>
<td>Read</td>
<td>Transfers a sector of data to host-specified buffer area.</td>
</tr>
<tr>
<td>12H</td>
<td>Write</td>
<td>Write a sector of data with address marks and CRC from specified host buffer.</td>
</tr>
<tr>
<td>13H</td>
<td>Seek</td>
<td>Positions head to track location specified in TRK.</td>
</tr>
</tbody>
</table>
11-4

14H Restore Initializes the disk unit and position the head to track 0 (outermost track).

15H Read ID Reads next available sector ID and track, and places it into a two byte read ID buffer. Byte 0 is the sector and Byte 1 is the track.

16H Write Deleted Identical to write command except that a deleted data address mark replaces regular data address mark.

17H Format Formats track specified in TRK to IBM 3740 specification.

NOTE that this formatting operation is not the same as the PIP formatting operation (see section 3). While this format command causes sector address and timing marks to be copied from a user created buffer to the disk being formatted, the PIP format command formats and also builds a file directory on the disk. A 4992 byte buffer is required (pointed to by UBFFR) which contains timing marks and other formatting information. Use of this command is not recommended.

11-9. DISK CONTROLLER ERROR RETURN CODES

11-10. Upon encountering an error, Bit 0 of the ERRC parameter in the IOCS vector is set. An error code is placed into location FF09H to indicate the type of error:

BIT	ERROR IF SET
7	Invalid drive, track or sector
6  Disk unit not ready
5  Track seek error
4  Sector not found
3  CRC error
2  Data lost
1  Disk is write protected
0  Attempt to read a deleted sector

The Z flag is set if no error was detected otherwise it is reset.

11-11. LINKED FILE LOADER

11-12. The Linked File Loader is a part of the DCF PROM. It accesses the disk at a given track and sector and loads information from the disk until the last sector in the linked structure is found. The Unit, Track/Sector address and load address are passed via an IOCS vector which is pointed to by the IY-register. 10 retries are performed. The IOCS vector is set up as shown for the DCF, described above. Entry address is ECO3H. No request code is required.

11-13. INTERACTIVE BOOT PROCESS

NOTE: This procedure is used only to load programs into areas different than the load address defined in the directory.

11-14. This DCF program allows the user to specify the starting track and sector number of a file to be loaded directly into RAM. All interaction is via the console device. The FLP-80DOS system must be in RAM because IOCS is used. The information from disk is loaded sector by sector. The linked structure on the disk is followed until the last sector in the file is loaded.

To use this process, perform the following command sequence:

$DDT(CR)
.E ECO9(CR)  - user executes the starting address of the interactive boot process.
LOAD ADR: aaaa(CR) - user enters RAM starting load address (in hexadecimal) for information from the disk. Console interaction at this point is the same as DDT (See Sections 7-12 and 7-18).

UNIT, TRK, SCTR: u, t, s(CR)
- user enters disk unit number (0, 1, 2, 3), starting track number and starting sector number of information to be loaded from disk. All three numbers are entered in hexadecimal.
- after loading is complete, the DDT prompt is issued.

If any errors occurred during the load process, then the following message will be printed on the console:

DSK ERR

If the FLP-80DOS system is not in RAM, then a small section of code which performs the following instructions must be executed to bypass usage of IOCS for console interaction:

LD A, 2
LD (OFF12H), A
JP ECO9H

11-15. INTERACTIVE SAVE PROCESS

NOTE This procedure may be used only for modifying the directory or Track/Sector maps. Improper use can destroy files.

11-16. This DCF program allows the user to save a section of RAM on disk as a set of sequential sectors. The doubly linked structure is maintained on disk, but tracks and sectors are not allocated as in the Disk Handler. The sectors are allocated sequentially and without regard to the disk directory. All
interaction is via the console device. The FLP-80DOS system must be in RAM because IOCS is used for console I/O. To use this process, perform the following command sequence:

\$DDT(CR)
\$.EC06(CR) - user executes the starting address of the interactive save process.

SAVE ADR,#SCTRS: aaaa,bb(CR)
- user enters the starting address of the information to be saved on disk and the number of sectors to be saved. Each sector is 124 bytes long, and up to FFH sectors may be saved (31744 bytes). Console interaction is the same as DDT. The two parameters are entered in hexadecimal.

UNIT,TRK,SCTR: u,t,s(CR)
- user enters disk unit number (0,1,2 or 3), starting track number and starting sector number for information to be saved on disk. Sectors and tracks are allocated sequentially increasing. All three numbers are entered in hexadecimal.

- after saving is finished, the DDT prompt is issued.

If any errors occurred during the save process, then the following message will be printed on the console:

DSK ERR

If the FLP-80DOS system is not in RAM then a small section of code which performs the following instructions must be executed to bypass usage of IOCS for console interaction:

LD A,2
LD (OFF12H),A ;SET DEBUG FLAG
JP OEC06H
SECTION 12

I/O HANDLERS

12-1. INTRODUCTION

12-2. This section describes the I/O handlers supplied with FLP-80DOS. In addition, listings of these handlers are included here to aid the user in writing his own handlers for his own devices. The system that is shipped to you contains only TK (keyboard), TT (console output), and CP (Centronics line printer) handlers linked into it. The other handlers are supplied as source and relocatable object modules. In order to use them in your system, you must perform a SYSGEN (System Generation); Hardware configurations are documented in the appropriate system Manual.
12-3. CR - CARD READER

DESCRIPTION - I/O handler. This handler interfaces a Documation M200 Card Reader to the FLP-80 system via two PIO ports. It is callable by IOCS. This is an interrupt driven driver. Immediate return is supported.

OPERATION -

OPEN. Interrupts are disabled. The address of the card reader interrupt handler is entered into the FLP-80 Interrupt Handler Address Table. The interrupt handler address is also programmed in to Port A control. Port A is then programmed for mode 2, and local interrupts are disable. The least significant byte of the interrupt handler address is also programmed into Port B control. Port B is then programmed. Finally, the BC register is set to physical record size of 82 (80 card columns plus carriage return and line feed). Interrupts are re-enabled.

CLOSE. No operation is performed in the handler.

READ. Initialize. The number of physical records to be read (NREC) is recorded, then zeroed. The assigned buffer area is noted. The card reader is checked to see if it is ready. Initial time out is 4 seconds. Immediate return is supported at this point. Additional time out counts are 20 seconds each. When the reader goes ready, a card pick is
initiated.

Card Input. Each column of data on the card causes an interrupt which is monitored by 'CRDRDR'. The interrupts are counted by the A - register until 80 interrupts are registered. During reading, conversion of the card EBCDIC data is done in 'CRDRDR' via table 'HOLTAB'.

Card Massaging. After the card has been read into the IOCS buffer, interrupts are disabled in the CPU and locally. If an EOT (ASCII 04H) exists in column 1 of the last card, an end-of-file sequence is initiated (discussed below). Trailing blanks on the card are compressed. A carriage return and line feed are appended to the resultant card image. NREC is incremented and checked to see if all cards requested were read. If not, another card is read. Otherwise, the IOCS buffer pointer is updated to the byte following the last card image and the subroutine returns to caller.

End-of-file. Upon end-of-file (04H in card column 1), the EOF error code (9) is placed in the 'ERRC' parameter of the vector. The IOCS buffer pointer is updated and return is made to caller.
12-4. **CP-CENTRONICS LINE PRINTER**

**DESCRIPTION** - I/O handler. This handler interfaces to any Centronics line printer. Immediate return is not supported. I/O timeout is checked. Tab (09H) and form feed (OCH) are decoded and the appropriate horizontal and vertical spacing is done.

**OPERATION** -

**OPEN** The ports are initialized, the horizontal and vertical counters are initialized, and a physical record size of one is returned.

**CLOSE** A form feed is issued to eject the paper from the printer at the end of an operation. The form feed is translated into a series of line feeds as described below.

**WRITE** The character to be written is checked. IF it is a tab, then it is translated to spaces mod-8. If it is a line feed then the vertical counter is incremented. If it is a form feed, then the page is ejected by issuing a series of line feeds. Users with form feed option may wish to delete this function. If it is a carriage return, then the horizontal counter is initialized. The line width is checked to truncate each line to 'LWIDTH' characters. Status is checked. If not ready, then the timeout is checked. If time out has occurred, then an error message is output and a new time out is set up. If ready, the character is output with the appropriate interface signals.
12-5. LP-DATA PRODUCTS LINE PRINTER

DESCRIPTION - I/O handler. This handler interfaces to any Data Products line printer. The handler is interrupt driven; one character at a time is output. Immediate return is supported. I/O timeout is checked.

OPERATION -

OPEN - The port is initialized. The line printer interrupt handler address is stored in the IOCS Interrupt Address Table. The vector address is programmed to the PIO. The tab count is initialized. A physical record size of one is returned.

CLOSE - No action.

WRITE - An initial 3 second time out is set up. The ready bit of the status (bit 0 of LPST) is checked. Immediate return is supported. If the device does not go ready, an error message is issued and the timeout is reprogrammed to 20 seconds. When the device goes ready, the ready bit is reset and the character is checked. If the character is not a tab, then it is output to the device. If the character is a tab then, it is expanded into spaces mod-8.
12-6. PR - PAPER TAPE READER

DESCRIPTION - I/O handler. This handler interfaces a paper tape reader to FLP-80DOS via a PIO port. This handler is called by IOCS. It is interrupt driven. One character at a time is input. Immediate return is supported. I/O timeout is checked.

OPERATION -

OPEN. The port is initialized. The paper tape reader interrupt handler address is stored in the IOCS interrupt handler address table. The first read Operation is initiated. A physical record size of 1 is returned to IOCS.

CLOSE. No action is performed.

READ. Upon reception of an interrupt, Bit 0 of 'PRST' is set to indicate that the reader is ready with another character.

An initial timeout of 250 msec is programmed. The status flag located in the LS bit of address PRST is checked. If it is set, then an interrupt has occurred. This indicates that a character is ready. The character is read and complemented and return is made to caller.
12-7. PP-PAPER TAPE PUNCH

DESCRIPTION - I/O handler. This handler interfaces a paper tape punch to FLP-80DOS. It is interrupt driven and immediate return is supported. One character at a time is output. I/O timeout is checked. The operation of this handler is similar to LP -Data Products Line Printer handler except that tabs are not expanded.
12-8. **TI-SILENT 700 CASSETTE INPUT**

**DESCRIPTION** - I/O handler. This handler interfaces a Silent 700 digital cassette for input to FLP-80DOS via the serial ASCII port. Thus, the Silent 700 is also the system terminal. The handler is not interrupt driven and immediate return is not supported. This handler will read tapes recorded in LINE or CONTINUOUS mode. The handler is compatible with other MOSTEK Systems.

**HARDWARE** - ADC option is required (this is a Texas Instruments field-installable option). The handler will work if RDC is installed, but not all functions of the RDC option will be used. The option to allow printing on the Silent 700 printer must be enabled. This handler will work for 300 or 1200 baud rate.

**OPERATION** -

**OPEN.** - Buffer count and null count are initialized to zero. The "Minimal Listener" is disabled to prevent false triggering of the "Debugger Escape". A physical record size of one is returned to caller.

**CLOSE.** - A DC3 (13H) character is issued to the Silent 700 to assure that the tape transport is turned off. The buffer count is reinitialized and the "Minimal Listener" is reenabled.

**READ.** - The read function reads one record from the cassette tape into a buffer and deblocks that buffer one byte at a time. When the buffer is empty, another record is read. End of record is defined by DC3 (13H). End of file on the tape is defined by EOT (04H), a sequence or 127 nulls, or a time out condition greater than 2 seconds.
12-9. **TK-KEYBOARD**

**DESCRIPTION** - I/O handler. This handler interfaces the terminal keyboard for input to the FLP-80DOS via the serial ASCII port. This handler is called by IOCS. It is not interrupt driven. One character at a time is input. Immediate return is supported. I/O timeout is not checked.

**HARDWARE** - Any serial terminal with ASCII keyboard. Allowed baud rates are 110, 300, 600, 1200, 2400, 4800 and 9600. RS-232 and 20mA current loop interfaces are supported.

**OPERATION** -

**OPEN** - A physical record size of 1 is returned to IOCS.

**CLOSE** - No action is performed.

**READ** - If a character was entered via the "Minimal Listener", it is taken as the keyed-in character. Otherwise the Status of the UART is checked. Immediate return is supported. When the UART goes ready, a character is read. Parity bit is cleared and the Minimal Listener holding register is cleared. If the Minimal Listener is enabled, then a test is made for CNTL-C (Debugger Escape) or CNTL-X (reboot). A positive test branches to the appropriate routine. If the Minimal Listener is not enabled, then return is made to caller.
12-10. TT - CONSOLE OUTPUT

DESCRIPTION - I/O handler. This handler is used for all output to the console device. It will support the following terminals depending on the baud rate.

<table>
<thead>
<tr>
<th>BAUD RATE</th>
<th>TERMINAL TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>Teletype or CRT</td>
</tr>
<tr>
<td>300</td>
<td>Silent 700 or CRT</td>
</tr>
<tr>
<td>600</td>
<td>CRT</td>
</tr>
<tr>
<td>1200</td>
<td>Silent 700 or CRT</td>
</tr>
<tr>
<td>2400-9600</td>
<td>CRT</td>
</tr>
</tbody>
</table>

Tabs are expanded by the handler, and an automatic carriage return/line feed is issued when the right side of the screen is reached. Immediate return is not supported.

HARDWARE - Any terminal with RS-232 or 20mA current loop interface.

OPERATION -

OPEN - A physical record size of one byte is returned.

CLOSE - No action.

WRITE - The character to be output is checked. If it is a tab (ASCII 09H), then the required number of spaces to position the print head or cursor mod-8 is output. If the character is a backspace, then the position counter is decremented and the backspace is output. For any character other than a carriage return (0D_H) or form feed (0C_H), the width of the current line is checked. If the
cursor is at the right side of the screen specified by 'LWIDTH'), then a carriage return and line feed are output. The position counter is then updated and the UART status is checked. When ready, the character is output to the device. If the device is a TTY or Silent 700, then a form feed (0CH) is translated to 6 line feeds to prevent uncontrolled paper scrolling. If the baud rate is 1200 baud for a Silent 700, then a 32 msec delay is executed after each character output. If the character is a carriage return and the baud rate is 300 or 1200, then an extra 210 msec delay is executed to allow full return of the print head. After each carriage return to output (ODH) the keyboard status byte, (TKST) in the scratchpad, is checked and if it contains a space (020H) then it is cleared and checked again in a loop until the next space is input from TK for release to continue output. This allows pausing the listing of a file to the console device by pressing the space bar once, and continuing the listing by pressing the space bar once again.
12-11. TO - SILENT 700 CASSETTE OUTPUT

DESCRIPTION - I/O handler. This handler interfaces a Silent 700 digital cassette for output to the FLP-80DOS system via the serial port. Thus, the Silent 700 is also the system terminal. This handler is not interrupt driven. Immediate return is not supported. This handler will record tapes in LINE or CONTINUOUS mode. It is compatible with other MOSTEK products.

HARDWARE - See description for 'TI'.

OPERATION -

OPEN - A buffer pointer is initialized and a physical record size of one of returned to caller.

CLOSE - A DC4 (14H) is issued to the Silent 700 to assure that the tape transport is off.

WRITE - Characters are blocked into a buffer one at a time until an end of record is encountered. An end of record is defined as a line feed character. When the end of record is encountered, the buffer is output to the device. The record format is: data, CR, LF, DC3, RUBOUT. If an end of file (EOT=04H) is to be output, then any bytes in the buffer are output. Then the EOT is output followed by a carriage return (ODH) to terminate LINE mode. A series of null characters is written to the device to assure that this last record is written to the tape in CONTINUOUS mode.
12-12. TR - TELETYPE PAPER TAPE READER

DESCRIPTION - I/O handler. This handler interfaces a teletype paper tape reader to FLP-80DS via the serial I/O port. This handler is called by IOCS. It is not interrupt driven. One character at a time is output. Immediate return is not supported. I/O time out is 250 milliseconds and returns to caller.

HARDWARE - Reader step control is required on the teletype.

OPERATION -

OPEN - The 'Minimal Listener' is turned off. A physical record size of 1 is returned to IOCS.

CLOSE - The 'Minimal Listener' is turned on and returns to caller.

WRITE - The reader is turned on. The UART is checked. A timeout of 250 milliseconds is initiated. If the UART does not go ready, return is made to caller. Otherwise, the reader is turned off the the character is read via TKREAD.
THIS IS THE INTERFACE FOR PRINTERS WHICH REQUIRE
A PULSE INSTEAD OF AN EDGE FOR DATA TRANSFER. FOR
EACH CHARACTER TRANSFERED, A 7.6 US. PULSE WILL
BE SENT 16.4 US. AFTER DATA IS SENT TO THE PRINTER.
BUSY IS USED TO INDICATE THAT THE BUFFER IS FULL
OR A RETURN OR LINE FEED HAS BEEN SENT.
100 US./CHAR IS THE FASTEST RATE THAT THE DRIVER
CAN OUTPUT DATA.
Both BITS 4 AND 5 MUST BE LOW FROM THE PRINTER
FOR DATA TO BE TRANSFERED. THE 7402 ON PORT
D2 INVERTS THE DATA, THEREFORE BOTH BITS MUST
BE HIGH IN THE ACC AFTER THE INPUT INSTRUCTION.
AFTER SCANNING FOR 1 SEC IF BOTH BITS ARE NOT HIGH
A TIMEOUT MESSAGE WILL BE PRINTED BY THE DRIVER.

BESIDES THE NORMAL PRINTABLE ASCII CHARACTERS, THIS
DRIVER RESPONDS TO 2 ASCII CONTROL CHARACTERS. THESE
CONTROL CHARACTERS ARE DECODED BY THE DRIVER AND ARE
TRANSLATED CHARACTERS WHICH EVERY PRINTER CAN USE.
THEN ARE: TAB (09H) AND FORM FEED (OCH).

GLOBAL CP
GLOBAL EH
LPCSTB EQU 0 ; STROBE FOR CENTRONICS TYPE
LPPE EQU 4 ; PAPER EMPTY
LPBSY EQU 5 ; PRINTER BUSY
LPDPO EQU 0D0H ;DATA PORT
LPDC EQU 0D1H ;CONTROL PORT
LPSP EQU 0D2H ;STROBE/BUSY PORT
LPSC EQU 0D3H ;STROBE/BUSY CONTROL PORT
TIMOUT EQU 07 ;TIMEOUT ERROR CODE
PAGE EQU 66 ;PAGE LENGTH
LWIDTH EQU 80 ;MAXIMUM LINE WIDTH

CP DEFB 4 ;MAX REQUEST
DEFB 0 ;OPENR
DEFB LPOPEN-S
DEFB LPCLOS-S
DEFB 0 ;READ
DEFB LPWRIT-S ;WRITE
DEFB 0 ; COLUMN COUNTER
DEFB 1 ;LINE COUNTER
LD A,0FH ;PORT A MODE 0
OUT (LPDC),A
LD A,0CFH ;PORT B MODE 3
OUT (LPSC),A
<table>
<thead>
<tr>
<th>CLP</th>
<th>OBJECT</th>
<th>ST</th>
<th>SOURCE STATEMENT</th>
<th>DATASET = DKO:LPC .SRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>'0010 3EFO</td>
<td>0060</td>
<td>LD A,OF0H</td>
<td>;HIGH HALF FOR INPUTS</td>
<td></td>
</tr>
<tr>
<td>'0012 D3D3</td>
<td>0061</td>
<td>OUT (LPSC),A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0014 3E03</td>
<td>0062</td>
<td>LD A,3</td>
<td>;DISABLE INTFS</td>
<td></td>
</tr>
<tr>
<td>'0016 D3D1</td>
<td>0063</td>
<td>OUT (LPDC),A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0018 3E11</td>
<td>0064</td>
<td>LD A,11H</td>
<td>;SELECT WITH DC1</td>
<td></td>
</tr>
<tr>
<td>'001A D3D0</td>
<td>0065</td>
<td>OUT (LPDP),A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'001C DBD2</td>
<td>0066</td>
<td>IN A,(LPSP)</td>
<td>;RESET LP CSTROBE</td>
<td></td>
</tr>
<tr>
<td>'001E CBC7</td>
<td>0067</td>
<td>SET LPCSTB,A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0020 0602</td>
<td>0068</td>
<td>LD B,2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0022 D3D2</td>
<td>0069</td>
<td>OUT (LPSP),A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0024 EE01</td>
<td>0070</td>
<td>XOR 1</td>
<td>;2**LPCSTB</td>
<td></td>
</tr>
<tr>
<td>'0026 10FA</td>
<td>0071</td>
<td>LD A,11H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0028 AF</td>
<td>0072</td>
<td>XOR A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0029 320600'</td>
<td>0073</td>
<td>LD (HCNTR),A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'002C 320700'</td>
<td>0074</td>
<td>LD (VCNTR),A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'002F 010100</td>
<td>0075</td>
<td>LD BC,1</td>
<td>;PHYSICAL RECORD SIZE =</td>
<td></td>
</tr>
<tr>
<td>'0032 C9</td>
<td>0076</td>
<td>RET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0033 3E0D</td>
<td>0078</td>
<td>LD A,ODH</td>
<td>;OUTPUT CARRIAGE RETURN</td>
<td></td>
</tr>
<tr>
<td>'0035 CD3A00'</td>
<td>0079</td>
<td>CALL LPWRIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0038 3E0C</td>
<td>0080</td>
<td>LD A,OCH</td>
<td>;OUTPUT FORM FEED</td>
<td></td>
</tr>
<tr>
<td>'003A FE09</td>
<td>0083</td>
<td>CP 9</td>
<td>;TAB?</td>
<td></td>
</tr>
<tr>
<td>'003C 2015</td>
<td>0084</td>
<td>JR NZ,LP2A-S</td>
<td>;NO</td>
<td></td>
</tr>
<tr>
<td>'003E C5</td>
<td>0085</td>
<td>PUSH BC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'003F 3A0600'</td>
<td>0086</td>
<td>LD A,(HCNTR)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0042 47</td>
<td>0087</td>
<td>LD B,A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0043 E6F8</td>
<td>0088</td>
<td>AND OF8H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0045 C608</td>
<td>0089</td>
<td>ADD A,8</td>
<td>;NEXT TAB LOC</td>
<td></td>
</tr>
<tr>
<td>'0047 0E20</td>
<td>0090</td>
<td>LD C,'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0049 90</td>
<td>0091</td>
<td>LD C,'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'004A 47</td>
<td>0092</td>
<td>LD B,A</td>
<td>;#SPACES</td>
<td></td>
</tr>
<tr>
<td>'004B 79</td>
<td>0093</td>
<td>LD A,C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'004C CD6200'</td>
<td>0094</td>
<td>CALL LP2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'004F 10FA</td>
<td>0095</td>
<td>DJNZ LP3-S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0051 C1</td>
<td>0096</td>
<td>POP BC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0052 C9</td>
<td>0097</td>
<td>RET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0053 FE0C</td>
<td>0098</td>
<td>CP OCH</td>
<td>;FORM FEED?</td>
<td></td>
</tr>
<tr>
<td>'0055 200B</td>
<td>0099</td>
<td>JR NZ,LP2-S</td>
<td>;NOTE: THIS LOGIC GENERATES</td>
<td></td>
</tr>
<tr>
<td>'0057 C5</td>
<td>0100</td>
<td>PUSH BC</td>
<td>;TO EJECT PAGE. IF LINE</td>
<td></td>
</tr>
<tr>
<td>'0058 3A0700'</td>
<td>0101</td>
<td>LD A,(VCNTR)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'005B 47</td>
<td>0102</td>
<td>LD B,A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'005C 3E42</td>
<td>0103</td>
<td>LD A,PAGE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'005E 0E0A</td>
<td>0104</td>
<td>LD C,OAH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0060 18E7</td>
<td>0105</td>
<td>JR LP3A-S</td>
<td>;LINE FEED OUT</td>
<td></td>
</tr>
<tr>
<td>'0062 F5</td>
<td>0106</td>
<td>LJNZ LP3A-S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0063 FE0A</td>
<td>0107</td>
<td>CP OAH</td>
<td>;SAVE CHARACTER</td>
<td></td>
</tr>
<tr>
<td>'0065 200E</td>
<td>0108</td>
<td>JR NZ,LP5-S</td>
<td>;LINE FEED?</td>
<td></td>
</tr>
<tr>
<td>'0067 3A0700'</td>
<td>0109</td>
<td>JR NZ,LP4-S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0069 3C</td>
<td>0110</td>
<td>INC A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'006A FE42</td>
<td>0111</td>
<td>CP PAGE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'006B 2001</td>
<td>0112</td>
<td>JR NZ,LP4-S</td>
<td>;TO ZERO AFTER MAX PAGE</td>
<td></td>
</tr>
<tr>
<td>'006F AF</td>
<td>0113</td>
<td>XR A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0070 320700'</td>
<td>0114</td>
<td>LD (VCNTR),A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0073 1813</td>
<td>0115</td>
<td>JR LP20-S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0077</td>
<td>0116</td>
<td>JR LP20-S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0079</td>
<td>0117</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IDDR OBJECT ST # SOURCE STATEMENT DATASET = DKO:LPC .SRC

0075 FE0D 0118 LP5 CP ODH ;IF CARRAGE RET
0077 3E00 0119 LD A,0 ;ZERO HORIZONTAL CTR.
0079 280A 0120 JR Z,LP10-S
0121 ;
007B 3A0600' 0122 LP6 LD A,(HCNTR) ;FETCH HORIZONTAL CTR
007E FE50 0123 CP LWIDTH ;IF MAX LINE WIDTH
0080 2002 0124 JR NZ,LP8-S ;IS REACHED THEN RETURN.
0082 F1 0125 POP AF
0083 C9 0126 RET
0084 3C 0127 LP8 INC A
0085 320600' 0128 LP10 LD (HCNTR),A ;UPDATE HORIZONTAL CTR
0129 ;
0088 C5 0130 LP20 PUSH BC ;SAVE BC
0089 01C409 0131 LD BC,2500 ;2.5 SECOND DELAY COUNT
008C C5 0132 LP22 PUSH BC
008D 062F 0133 LD B,47 ;MSEC COUNTER
0134 ;
008F DBD2 0135 LP24 INC A,(LPSP) ;EXIT TO PRINT CHARACTER
0091 E630 0136 AND O30H ;IF BOTH STATUS BITS 4 & 5
0093 FE30 0137 CP O30H ;ARE SET INDICATING PAPER
0095 2813 0138 JR Z,LP30-S ;IS NOT EMPTY AND PRINTER
0097 0B 0139 DEC BC ;IS NOT BUSY
0098 10F5 0140 DJNZ LP24- ;LOOP FOR 1 MSEC
0141 ;
009A C1 0142 POP BC ;DECREMENT COUNT
009B 0B 0143 DEC BC
009C 78 0144 LD A,B
009D B1 0145 OR C
009E 20EC 0146 JR NZ,LP22-S
0147 ;
00A0 3E07 0148 LD A,TIMOUT ;PRINT TIMEOUT ERROR
00A2 C0FF0F 0149 CALL EH ;NEW TIME OUT
00A5 01204E 0150 LD BC,20000
00A8 18E2 0151 JR LP22-S
0152 ;
00AA C1 0153 LP30 POP BC ;ADJUST STACK
00AB C1 0154 POP BC ;RESTORE BC
00AC F1 0155 POP AF ;GET CHAR
00AD D3D0 0156 OUT (LPDP),A ;OUTPUT CHAP
00AF F5 0157 PUSH AF ;SAVE CHAR
00B0 DBD2 0158 IN A,(LPSP) ;RESET STROBE
00B2 CB87 0159 RES LPCSTB,A ;GENERATING PULSE.
00B4 D3D2 0160 OUT (LPSP),A
00B6 C5C7 0161 SET LPCSTB,A
00B8 D3D2 0162 OUT (LPSP),A
00BA F1 0163 POP AF ;RESTORE CHAP
00BB C9 0164 RET
0165 ;
0166 END

ERRORS=0000
NAME LPDATA

DATA PRODUCTS LINE PRINTER HANDLER

GLOBAL LP
GLOBAL EH

TOR EQU OFF00H
IRET EQU 2
CFLGS EQU 21
TIMOUT EQU 7

LP

LPST DEFB OAAH

LPDIS EQU 7 ;VECTOR DISPLACEMENT FROM TOR

LPOPEN-$
LPCLOS-$
LPWRIT-$
OAAH

A

HL, (TOR) ;ACCESS INTERRUPT TABLE
DE,LPDIS
OR A
HL, DE

LD HL, (TOR) ;ACCESS INTERRUPT TABLE
LD DE,LPDIS
OR A
HL, DE

PUSH HL ;SAVE VECTOR ADDR

LD DE,LINT ;GET INTERRUPT HANDLER ADDRESS
LD (HL),E ;SAVE IN VECTOR
INC HL

LD (HL),D

LD HL,LPST ;HL -> STATUS BYTE
LD HL,LPST

SET 0,(HL) ;SET READY BIT
LD C,(HL) ;GET PORT FOR CONTROL
LD A,0FH ;OUTPUT CONTROL
OUT (C),A
OUT (C),E ;OUTPUT VECTOR LSBYTE
LD A,83H
OUT (C),A

LD A,D ;SET UP VECTOR MSBYTE
LD I,A
LD A,0CH ;OUTPUT FORM FEED TO INITIALIZE

CALL LPWRIT
LD A,0DH ;AND A CR
CALL LPWRIT
LD A,8 ;INITIALIZE TAB COUNT
LD (CNT),A
LD BC,1

EI ;ENABLE

RET ;RETURN TO CALLER

PUSH HL ;SAVE REGS
PUSH BC
"0042 F5 0060 PUSH AF ;SAVE BYTE TO OUTPUT
"0043 210600' 0061 LD HL,LPST ;HL -> STATUS BYTE
"0046 01B80B 0062 LD BC,3000 ;3 SECOND TIME OUT
"0049 C5 0063 LPA PUSH BC ;SAVE
"004A 0629 0064 LD B,41 ;MSEC COUNTER
"004C FB 0065 LPL EI ;ENABLE INTPS
"004D CB46 0066 BIT 0,(HL) ;CHECK FOR READY
"004F 201D 0067 JR NZ,LPR-S ;YES, SKIP OUT
"0051 FDCB1556 0068 BIT IRET,(IY+CFLGS) ;CHECK IMMED RETURN
"0055 2012 0069 JR NZ,LPI-S ;YES, SKIP OUT
"0057 10F3 0070 DJNZ LPL-S ;LOOP FOR TIMEOUT
"0059 C1 0071 POP BC ;DECENENT COUNT
"005A 03 0072 DEC BC
"005B 78 0073 LD A,B
"005C B1 0074 OR C
"005D 20EA 0075 JR NZ,LPA-S ;LOOP FOR TIMEOUT
"005F 3E07 0077 LD A,TIMOUT ;TIME OUT ERRCR
"0061 CDFFFF 0078 CALL EI ;OUTPUT IT
"0064 01204E 0079 LD BC,20000 ;NEW TIMEOUT
"0067 18E0 0080 JR LPA-S
"0069 C1 0082 LPI POP BC
"006A F1 0083 POP AF
"006B C1 0084 POP BC
"006C E1 0085 POP HL
"006D C9 0086 RET
"006E C1 0088 LPR POP BC ;RESTORE STACK
"006F E1 0089 POP AF ;GET BYTE
"0070 CB86 0090 RES 0,(HL) ;RESET READY BIT
"0072 4E 0091 LD C,(HL) ;GET DATA PORT NR
"0073 FE09 0092 CP 9 ;IS THIS A TAB CHARACTER?
"0075 2016 0093 JR NZ,LPR2-S ;NO, SKIP
"0077 3E20 0094 LD A,,' ;IF TAB OUTPUT A BLANK
"0079 ED79 0095 OUT (C),A
"007B 3AAFF00' 0096 LD A,(CNT) ;DECREMENT COUNT
"007E FE08 0097 CP 8 ;CHECK IF AT END OF TAB SPACE
"0080 2819 0098 JR Z,LPR4-S ;IF SO, SKIP OUT
"0082 3D 0099 DEC A ;UNTIL IT TURNS TO ZERO
"0083 32AF00' 0100 LD (CNT),A
"0086 3E09 0101 LD A,9 ;REINITIALIZE CHARACTER=TAB
"0088 F5 0102 PUSH AF
"0089 20BE 0103 JR NZ,LPA-S ;IF NOT DONE, OUTPUT MORE
"008B 180C 0104 JR LPR3-S ELSE REINIT TAB COUNTER
"008D ED79 0106 LPR2 OUT (C),A ;OUTPUT NON-TAB CHARACTER
"008F FE0D 0107 CP ODH ;IF CARRIAGE RETURN
"0091 2806 0108 JR Z,LPR3-S ;GO REINIT TAB COUNTER
"0093 3AAFF00' 0109 LD A,(CNT) ;DECREMENT COUNTER
"0096 3D 0110 DEC A
"0097 2002 0111 JR NZ,LPR4-S ;IF NOT ZERO, SKIP
"0099 3E08 0113 LPR3 LD A,8 ;REINIT TAB COUNTER
"009B 32AF00' 0114 LPR4 LD (CNT),A ;SET TAB COUNTER
"009E FDCB1596 0115 RES IRET,(IY+CFLGS) ;RESET IMMED RETURN
"00A2 C1 0116 POP BC
"00A3 E1 0117 POP HL

Note: DESTROYS A-REG
DATASET = $\text{DKO:LPD .SRC}$

<table>
<thead>
<tr>
<th>ADDR</th>
<th>OBJEC</th>
<th>ST #</th>
<th>SOURCE STATEMENT</th>
<th>DATASET = $\text{DKO:LPD .SRC}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>'00A4</td>
<td>C9</td>
<td>0118</td>
<td>RET</td>
<td>RETURN TO CALLER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0119</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0120</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'00A5</td>
<td>E5</td>
<td>0121</td>
<td>LINT PUSH HL</td>
<td>LINE PRINTER INTERRUPT HANDLER</td>
</tr>
<tr>
<td>'00A6</td>
<td>210600'</td>
<td>0122</td>
<td>LD HL,LPST</td>
<td></td>
</tr>
<tr>
<td>'00A9</td>
<td>CBC6</td>
<td>0123</td>
<td>SET 0,(HL)</td>
<td>SET READY BIT</td>
</tr>
<tr>
<td>'00AB</td>
<td>E1</td>
<td>0124</td>
<td>POP HL</td>
<td></td>
</tr>
<tr>
<td>'00AC</td>
<td>FB</td>
<td>0125</td>
<td>EI</td>
<td></td>
</tr>
<tr>
<td>'00AD</td>
<td>ED4D</td>
<td>0126</td>
<td>RETI</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0127</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'00AF</td>
<td>00</td>
<td>0128</td>
<td>CNT DEFB 0</td>
<td>TAB COUNTER</td>
</tr>
</tbody>
</table>

ERRORS=0000
NAME CR

;TITLE: CARD READER DRIVER FOR FLP-80

*ID: ZCR80 V2.0 27MAY78

*TYPE: SUBROUTINE

*SYSTEM: AID-80F WITH FLP-80DOS

*DESCRIPTION: THIS DRIVER INTERFACES A DOCUMENTATION

*STACK USAGE: MAX 10 ENTRIES

*CALLED ROUTINES: EH

*ID: ZCR80 V2.0 27MAY78

*PROGRAMMER: D. LEITCH

P. FORMANIAK

EXTERNAL SYMBOLS

GLOBAL EH

SCRATCHPAD EQUATES

;USER BUFFER OFFSET IN VECTOR

UBFFR EQU 25

CFFGS EQU 21

ERRC EQU 23

HSCR EQU 30

IRET EQU 2

NREC EQU 29

LOCAL EQUATES

EOT EQU 4

TIMOUT EQU 7

EOFERR EQU 9

GLOBAL CR

CR EQU S

DEBF 3

DEBF CROPEN-S ;OPEN FOR READ

DEBF 0 ;OPEN FOR WRITE

DEBF CRCLOS-S ;CLOSE

DEBF CRREAD-S ;READ

CRPORT DEFB OADH ;PORT FOR CARD READER

CRDIS EQU 0DH ;INTP VECTOR DISPLACEMENT FROM TO

CROPEN DI ;OPEN CARD READER

LD HL,(TOR) ;GET VECTOR ADDRESS

LD DE,CRDIS ;OFFSET OF VECTOR FROM TO

OR A
TIMEOUT ERROR. OUTPUT THE ERROR TO CONSOLE. THEN LOOP...
0118 ; UNTIL DEVICE GOES READY.
0121 CDFFFF CALL EH ;OUTPUT THE ERROR
0123 21204E LD HL,20000 ;20 SECOND TIMEOUT FROM HERE:
0126 18E1 JR CRDYL-S ;AND LOOP FOR MORE
0129 ;
0130 0D CRGO DEC C ;ADJUST TO A CNTL
0131 3E83 LD A,83H ;ENABLE INTERRUPTS
0132 ED79 OUT (C),A
0133 AF XOR A ;CLEAR A
0134 3E83 DEC C ;ADJUST TO A DATA
0135 ED79 OUT (C),A ;FORCE A PICK
0136 C5 PUSH BC ;SAVE BC
0137 FB EI
0138 CBZ1 EQU S ;GO READ THE CARD VIA INTFS
0139 FE50 CP 80 ;A=80 => FINISHED
0140 20FC JR NZ,CBZ1-S
0141 F3 DI
0142 C1 POP BC ;RESTORE BC
0143 0D DEC C ;ADJUST TO A CNTL
0144 3E03 LD A,3 ;DISABLE I/O INTERRUPTS
0145 ED79 OUT (C),A
0146 ;
0147 ; CHECK FOR EOT (04H) IN COLUMN 1
0148 ;
0149 D5 PUSH DE ;DE INTO HL
0150 E1 POP HL
0151 C5 POP BC ;SAVE BC-REG
0152 FD41D INC (IY+NREC) ;INCREMENT NBR OF CARDS READ
0153 015000 LD BC,80 ;ACCESS FIRST CHARACTER OF CARD
0154 B7 OR A
0155 ED42 SBC HL,BC
0156 7E LD A,(HL) ;GET CHARACTER IN COLUMN 1
0157 C1 POP BC
0158 FE04 CP EOT ;CHECK FOR END OF FILE INDICATOR
0159 2006 JR NZ,NEOT-S ;NOT EOT, SKIP
0160 FD361709 LD (IY+ERRC),EOFERR ;SET UP END OF FILE
0161 1817 JR CREOT-S ;AND SKIP OUT
0162 ;
0163 ; NOT EOT, COMPRESS TRAILING BLANKS ON CARD
0164 ;
0165 B0 NEOT DEC DE ;DECREMENT POINTER
0166 1A LD A,(DE) ;GET CHARACTER
0167 FE20 CP 20H ;BLANK?
0168 28FA JR Z,NEOT-S ;YES, KEEP COMPRESSING
0169 13 INC DE ;CORRECT POINTER
0170 ;
0171 EB EX DE,HL ;HL -> END OF CARD BUFFER
0172 360D LD (HL),0DH ;STUFF A CR
0173 BA 23 INC HL
0174 360A LD (HL),0AH
0175 BD 23 INC HL
ADDR  OBJECT ST # SOURCE STATEMENT DATASET = DKO:CR .SRC

'00BE  EB  0176  EX  DE,HL ; DE -> CARD BUFFER
'00BF  FD7E1D  0177  LD  A, (IY+NREC) ; CHECK FOR ALL CARDS RE!
'00C2  FD961E  0178  SUB  (IY+HSCR) ; THAT WERE REQUESTED
'00C5  208F  0179  JR  NZ, CRLOOP-S ; NOT DONE, LOOP FOR NEXT:

'00C7  FD7319  0180  ; ;
'00CA  FD721A  0181  CREOT  LD  (IY+UBFFR), E ; UPDATE BUFFER POINTER I
'00CD  C1  0182  POP  BC ; RESTORE BC REG
'00CE  D1  0183  POP  DE
'00CF  E1  0184  POP  HL
'00D0  FDCB1596  0185  RES  IRET, (IY+CFLGS) ; RESET IMMEDIATE RETURN
'00D4  FB  0186  EI
'00D5  C9  0187  RET

'00D6  C9  0191  CRCLOSE RET ; CLOSE

'00D7  F5  0192  CRDRDR  PUSH  AF ; SAVE AF AND BC
'00D8  0607  0193  LD  B, 7
'00DA  3A0500'  0194  LD  A, (CRPORT) ; GET CARD READER PORT
'00DD  4F  0195  LD  C, A
'00DE  0D  0196  DEC  C  ; ADJUST TO A DATA
'00DF  ED78  0197  IN  A, (C)  ; INPUT A DATA
'00E1  2F  0198  CPL  ; COM DATA FROM A
'00E2  6F  0199  LD  L, A  ; SAVE A DATA
'00E3  0C  0200  INC  C  ; ADJUST TO B DATA
'00E4  0C  0201  INC  C
'00E5  ED78  0202  IN  A, (C)  ; B DATA
'00E6  E6F0  0203  AND  0F0H  ; MASK OFF LS 4BITS
'00E9  CB7D  0204  BIT  7, L  ; MOVE BIT 7 FROM A
'00EB  2802  0205  JR  Z, CRD1-S ; TO BIT3 OF B
'00ED  F608  0206  OR  8
'00EF  CBBD  0207  LD  A, (CRDPORT) ; GET CARD READER PORT
'00F1  CB25  0208  LD  B, 7
'00F3  FAF800'  0209  JP  M, CRD3
'00F6  10F9  0210  DJNZ  CRD2-S

'00F8  80  0211  ADD  A, B  ; LS 3 BITS OF DISPLACE-
'00F9  4F  0212  LD  C, A  ; MENT ADDED IN
'00FA  0600  0213  LD  B, 0  ; BC=TOTAL DISPLACEMENT
'00FC  210801'  0214  LD  HL, HOLTAB ; HL=HOLLERITH TABLE
'00FF  09  0215  ADD  HL, BC  ; GET ADDRESS OF CHAR
'0100  7E  0216  LD  A, (HL)  ; GET CHARACTER
'0101  12  0217  LD  (DE), A  ; STORE INTO BUFFER
'0102  13  0218  INC  DE  ; INCR PTR
'0103  F1  0219  POP  AF
'0104  3C  0220  INC  A  ; COUNT INTERRUPTS
'0105  FB  0221  EI
'0106  ED4D  0222  RETI

'0108  20  0223  ADD  A, B  ; LS 3 BITS OF DISPLACE-
'0109  31  0224  ADD  A, B  ; MENT ADDED IN
'010A  32  0225  ADD  A, B  ; BC=TOTAL DISPLACEMENT
'010B  33  0226  ADD  A, B  ; HL=HOLLERITH TABLE
'010C  34  0227  ADD  A, B  ; GET ADDRESS OF CHAR
'010D  35  0228  ADD  A, B  ; GET CHARACTER
'010E  36  0229  ADD  A, B  ; STORE INTO BUFFER
'010F  37  0230  ADD  A, B  ; INCR PTR
'0110  38  0231  ADD  A, B  ; COUNT INTERRUPTS

'0108  20  0228  HOLTAB  DEFB  ' ' ; BLANK
'0109  31  0229  HOLTAB  DEFB  '1' ; 1
'010A  32  0230  HOLTAB  DEFB  '2' ; 2
'010B  33  0231  HOLTAB  DEFB  '3' ; 3
'010C  34  0232  HOLTAB  DEFB  '4' ; 4
'010D  35  0233  HOLTAB  DEFB  '5' ; 5
<table>
<thead>
<tr>
<th>ADDR</th>
<th>OBJ</th>
<th>ST</th>
<th>SRC</th>
<th>STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>010E</td>
<td>36</td>
<td>0234</td>
<td>DEFB</td>
<td>'6'</td>
</tr>
<tr>
<td>010F</td>
<td>37</td>
<td>0235</td>
<td>DEFB</td>
<td>'7'</td>
</tr>
<tr>
<td>0110</td>
<td>38</td>
<td>0236</td>
<td>DEFB</td>
<td>'8'</td>
</tr>
<tr>
<td>0111</td>
<td>60</td>
<td>0237</td>
<td>DEFB</td>
<td>60H</td>
</tr>
<tr>
<td>0112</td>
<td>3A</td>
<td>0238</td>
<td>DEFB</td>
<td>'.'</td>
</tr>
<tr>
<td>0113</td>
<td>23</td>
<td>0239</td>
<td>DEFB</td>
<td>'#.'</td>
</tr>
<tr>
<td>0114</td>
<td>40</td>
<td>0240</td>
<td>DEFB</td>
<td>'M'</td>
</tr>
<tr>
<td>0115</td>
<td>27</td>
<td>0241</td>
<td>DEFB</td>
<td>27H</td>
</tr>
<tr>
<td>0116</td>
<td>3D</td>
<td>0242</td>
<td>DEFB</td>
<td>'='</td>
</tr>
<tr>
<td>0117</td>
<td>22</td>
<td>0243</td>
<td>DEFB</td>
<td>'*'</td>
</tr>
<tr>
<td>0118</td>
<td>39</td>
<td>0244</td>
<td>DEFB</td>
<td>'9'</td>
</tr>
<tr>
<td>0119</td>
<td>00</td>
<td>0245</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>011A</td>
<td>16</td>
<td>0246</td>
<td>DEFB</td>
<td>16H</td>
</tr>
<tr>
<td>011B</td>
<td>00</td>
<td>0247</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>011C</td>
<td>00</td>
<td>0248</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>011D</td>
<td>00</td>
<td>0249</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>011E</td>
<td>00</td>
<td>0250</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>011F</td>
<td>04</td>
<td>0251</td>
<td>DEFB</td>
<td>04H</td>
</tr>
<tr>
<td>0120</td>
<td>00</td>
<td>0252</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>0121</td>
<td>00</td>
<td>0253</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>0122</td>
<td>00</td>
<td>0254</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>0123</td>
<td>00</td>
<td>0255</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>0124</td>
<td>14</td>
<td>0256</td>
<td>DEFB</td>
<td>14H</td>
</tr>
<tr>
<td>0125</td>
<td>15</td>
<td>0257</td>
<td>DEFB</td>
<td>15H</td>
</tr>
<tr>
<td>0126</td>
<td>00</td>
<td>0258</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>0127</td>
<td>1A</td>
<td>0259</td>
<td>DEFB</td>
<td>1AH</td>
</tr>
<tr>
<td>0128</td>
<td>30</td>
<td>0260</td>
<td>DEFB</td>
<td>'0.'</td>
</tr>
<tr>
<td>0129</td>
<td>2F</td>
<td>0261</td>
<td>DEFB</td>
<td>'/'</td>
</tr>
<tr>
<td>012A</td>
<td>53</td>
<td>0262</td>
<td>DEFB</td>
<td>'S.'</td>
</tr>
<tr>
<td>012B</td>
<td>54</td>
<td>0263</td>
<td>DEFB</td>
<td>'T.'</td>
</tr>
<tr>
<td>012C</td>
<td>55</td>
<td>0264</td>
<td>DEFB</td>
<td>'U.'</td>
</tr>
<tr>
<td>012D</td>
<td>56</td>
<td>0265</td>
<td>DEFB</td>
<td>'V.'</td>
</tr>
<tr>
<td>012E</td>
<td>57</td>
<td>0266</td>
<td>DEFB</td>
<td>'W.'</td>
</tr>
<tr>
<td>012F</td>
<td>58</td>
<td>0267</td>
<td>DEFB</td>
<td>'X.'</td>
</tr>
<tr>
<td>0130</td>
<td>59</td>
<td>0268</td>
<td>DEFB</td>
<td>'Y.'</td>
</tr>
<tr>
<td>0131</td>
<td>00</td>
<td>0269</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>0132</td>
<td>5D</td>
<td>0270</td>
<td>DEFB</td>
<td>5DH</td>
</tr>
<tr>
<td>0133</td>
<td>2C</td>
<td>0271</td>
<td>DEFB</td>
<td>','</td>
</tr>
<tr>
<td>0134</td>
<td>25</td>
<td>0272</td>
<td>DEFB</td>
<td>','</td>
</tr>
<tr>
<td>0135</td>
<td>5F</td>
<td>0273</td>
<td>DEFB</td>
<td>5FH</td>
</tr>
<tr>
<td>0136</td>
<td>3E</td>
<td>0274</td>
<td>DEFB</td>
<td>'&gt;'</td>
</tr>
<tr>
<td>0137</td>
<td>3F</td>
<td>0275</td>
<td>DEFB</td>
<td>'?'</td>
</tr>
<tr>
<td>0138</td>
<td>5A</td>
<td>0276</td>
<td>DEFB</td>
<td>'Z.'</td>
</tr>
<tr>
<td>0139</td>
<td>00</td>
<td>0277</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>013A</td>
<td>00</td>
<td>0278</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>013B</td>
<td>00</td>
<td>0279</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>013C</td>
<td>00</td>
<td>0280</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>013D</td>
<td>0A</td>
<td>0281</td>
<td>DEFB</td>
<td>0AH</td>
</tr>
<tr>
<td>013E</td>
<td>17</td>
<td>0282</td>
<td>DEFB</td>
<td>017H</td>
</tr>
<tr>
<td>013F</td>
<td>1B</td>
<td>0283</td>
<td>DEFB</td>
<td>1BH</td>
</tr>
<tr>
<td>0140</td>
<td>00</td>
<td>0284</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>0141</td>
<td>00</td>
<td>0285</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>0142</td>
<td>00</td>
<td>0286</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>0143</td>
<td>00</td>
<td>0287</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>0144</td>
<td>00</td>
<td>0288</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>0145</td>
<td>05</td>
<td>0289</td>
<td>DEFB</td>
<td>05H</td>
</tr>
<tr>
<td>0146</td>
<td>06</td>
<td>0290</td>
<td>DEFB</td>
<td>06H</td>
</tr>
<tr>
<td>0147</td>
<td>07</td>
<td>0291</td>
<td>DEFB</td>
<td>07H</td>
</tr>
<tr>
<td>ADDR</td>
<td>OBJECT</td>
<td>ST</td>
<td>#</td>
<td>SOURCE</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>----</td>
<td>---</td>
<td>--------</td>
</tr>
<tr>
<td>'0148</td>
<td>2D</td>
<td>0292</td>
<td>DEFB</td>
<td>' - '</td>
</tr>
<tr>
<td>'0149</td>
<td>4A</td>
<td>0293</td>
<td>DEFB</td>
<td>' J '</td>
</tr>
<tr>
<td>'014A</td>
<td>4B</td>
<td>0294</td>
<td>DEFB</td>
<td>' K '</td>
</tr>
<tr>
<td>'014B</td>
<td>4C</td>
<td>0295</td>
<td>DEFB</td>
<td>' L '</td>
</tr>
<tr>
<td>'014C</td>
<td>4D</td>
<td>0296</td>
<td>DEFB</td>
<td>' M '</td>
</tr>
<tr>
<td>'014D</td>
<td>4E</td>
<td>0297</td>
<td>DEFB</td>
<td>' N '</td>
</tr>
<tr>
<td>'014E</td>
<td>4F</td>
<td>0298</td>
<td>DEFB</td>
<td>' O '</td>
</tr>
<tr>
<td>'014F</td>
<td>50</td>
<td>0299</td>
<td>DEFB</td>
<td>' P '</td>
</tr>
<tr>
<td>'0150</td>
<td>51</td>
<td>0300</td>
<td>DEFB</td>
<td>' Q '</td>
</tr>
<tr>
<td>'0151</td>
<td>00</td>
<td>0301</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'0152</td>
<td>21</td>
<td>0302</td>
<td>DEFB</td>
<td>' ! '</td>
</tr>
<tr>
<td>'0153</td>
<td>24</td>
<td>0303</td>
<td>DEFB</td>
<td>' S '</td>
</tr>
<tr>
<td>'0154</td>
<td>2A</td>
<td>0304</td>
<td>DEFB</td>
<td>' * '</td>
</tr>
<tr>
<td>'0155</td>
<td>29</td>
<td>0305</td>
<td>DEFB</td>
<td>' ) '</td>
</tr>
<tr>
<td>'0156</td>
<td>3B</td>
<td>0306</td>
<td>DEFB</td>
<td>' ; '</td>
</tr>
<tr>
<td>'0157</td>
<td>5C</td>
<td>0307</td>
<td>DEFB</td>
<td>5CH</td>
</tr>
<tr>
<td>'0158</td>
<td>52</td>
<td>0308</td>
<td>DEFB</td>
<td>' R '</td>
</tr>
<tr>
<td>'0159</td>
<td>11</td>
<td>0309</td>
<td>DEFB</td>
<td>11H</td>
</tr>
<tr>
<td>'015A</td>
<td>12</td>
<td>0310</td>
<td>DEFB</td>
<td>12H</td>
</tr>
<tr>
<td>'015B</td>
<td>13</td>
<td>0311</td>
<td>DEFB</td>
<td>13H</td>
</tr>
<tr>
<td>'015C</td>
<td>00</td>
<td>0312</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'015D</td>
<td>00</td>
<td>0313</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'015E</td>
<td>08</td>
<td>0314</td>
<td>DEFB</td>
<td>08H</td>
</tr>
<tr>
<td>'015F</td>
<td>00</td>
<td>0315</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'0160</td>
<td>18</td>
<td>0316</td>
<td>DEFB</td>
<td>18H</td>
</tr>
<tr>
<td>'0161</td>
<td>13</td>
<td>0317</td>
<td>DEFB</td>
<td>19</td>
</tr>
<tr>
<td>'0162</td>
<td>00</td>
<td>0318</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'0163</td>
<td>00</td>
<td>0319</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'0164</td>
<td>1C</td>
<td>0320</td>
<td>DEFB</td>
<td>1CH</td>
</tr>
<tr>
<td>'0165</td>
<td>1D</td>
<td>0321</td>
<td>DEFB</td>
<td>1DH</td>
</tr>
<tr>
<td>'0166</td>
<td>1E</td>
<td>0322</td>
<td>DEFB</td>
<td>1EH</td>
</tr>
<tr>
<td>'0167</td>
<td>1F</td>
<td>0323</td>
<td>DEFB</td>
<td>1FH</td>
</tr>
<tr>
<td>'0168</td>
<td>7D</td>
<td>0324</td>
<td>DEFB</td>
<td>7DH</td>
</tr>
<tr>
<td>'0169</td>
<td>7E</td>
<td>0325</td>
<td>DEFB</td>
<td>7EH</td>
</tr>
<tr>
<td>'016A</td>
<td>73</td>
<td>0326</td>
<td>DEFB</td>
<td>73H</td>
</tr>
<tr>
<td>'016B</td>
<td>74</td>
<td>0327</td>
<td>DEFB</td>
<td>74H</td>
</tr>
<tr>
<td>'016C</td>
<td>75</td>
<td>0328</td>
<td>DEFB</td>
<td>75H</td>
</tr>
<tr>
<td>'016D</td>
<td>76</td>
<td>0329</td>
<td>DEFB</td>
<td>76H</td>
</tr>
<tr>
<td>'016E</td>
<td>77</td>
<td>0330</td>
<td>DEFB</td>
<td>77H</td>
</tr>
<tr>
<td>'016F</td>
<td>78</td>
<td>0331</td>
<td>DEFB</td>
<td>78H</td>
</tr>
<tr>
<td>'0170</td>
<td>79</td>
<td>0332</td>
<td>DEFB</td>
<td>79H</td>
</tr>
<tr>
<td>'0171</td>
<td>00</td>
<td>0333</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'0172</td>
<td>00</td>
<td>0334</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'0173</td>
<td>00</td>
<td>0335</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'0174</td>
<td>00</td>
<td>0336</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'0175</td>
<td>00</td>
<td>0337</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'0176</td>
<td>00</td>
<td>0338</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'0177</td>
<td>00</td>
<td>0339</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'0178</td>
<td>7A</td>
<td>0340</td>
<td>DEFB</td>
<td>7AH</td>
</tr>
<tr>
<td>'0179</td>
<td>00</td>
<td>0341</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'017A</td>
<td>00</td>
<td>0342</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'017B</td>
<td>00</td>
<td>0343</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'017C</td>
<td>00</td>
<td>0344</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'017D</td>
<td>00</td>
<td>0345</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'017E</td>
<td>00</td>
<td>0346</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'017F</td>
<td>00</td>
<td>0347</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'0180</td>
<td>00</td>
<td>0348</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>'0181</td>
<td>00</td>
<td>0349</td>
<td>DEFB</td>
<td>0</td>
</tr>
<tr>
<td>ADDR</td>
<td>OBJECT</td>
<td>ST #</td>
<td>SOURCE STATEMENT</td>
<td>DATASET = DKO:CR .SRC</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>------</td>
<td>------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>'0182 00</td>
<td>0350</td>
<td>DEFB 0</td>
<td>;11-0-9-8-2</td>
<td></td>
</tr>
<tr>
<td>'0183 00</td>
<td>0351</td>
<td>DEFB 0</td>
<td>;11-0-9-8-3</td>
<td></td>
</tr>
<tr>
<td>'0184 00</td>
<td>0352</td>
<td>DEFB 0</td>
<td>;11-0-9-8-4</td>
<td></td>
</tr>
<tr>
<td>'0185 00</td>
<td>0353</td>
<td>DEFB 0</td>
<td>;11-0-9-8-5</td>
<td></td>
</tr>
<tr>
<td>'0186 00</td>
<td>0354</td>
<td>DEFB 0</td>
<td>;11-0-9-8-6</td>
<td></td>
</tr>
<tr>
<td>'0187 00</td>
<td>0355</td>
<td>DEFB 0</td>
<td>;11-0-9-8-7</td>
<td></td>
</tr>
<tr>
<td>'0188 26</td>
<td>0356</td>
<td>DEFB 26H</td>
<td>;12</td>
<td></td>
</tr>
<tr>
<td>'0189 41</td>
<td>0357</td>
<td>DEFB 'A'</td>
<td>;12-1</td>
<td></td>
</tr>
<tr>
<td>'018A 42</td>
<td>0358</td>
<td>DEFB 'B'</td>
<td>;12-2</td>
<td></td>
</tr>
<tr>
<td>'018B 43</td>
<td>0359</td>
<td>DEFB 'C'</td>
<td>;12-3</td>
<td></td>
</tr>
<tr>
<td>'018C 44</td>
<td>0360</td>
<td>DEFB 'D'</td>
<td>;12-4</td>
<td></td>
</tr>
<tr>
<td>'018D 45</td>
<td>0361</td>
<td>DEFB 'E'</td>
<td>;12-5</td>
<td></td>
</tr>
<tr>
<td>'018E 46</td>
<td>0362</td>
<td>DEFB 'F'</td>
<td>;12-6</td>
<td></td>
</tr>
<tr>
<td>'018F 47</td>
<td>0363</td>
<td>DEFB 'G'</td>
<td>;12-7</td>
<td></td>
</tr>
<tr>
<td>'0190 48</td>
<td>0364</td>
<td>DEFB 'H'</td>
<td>;12-8</td>
<td></td>
</tr>
<tr>
<td>'0191 00</td>
<td>0365</td>
<td>DEFB 0</td>
<td>;12-8-1</td>
<td></td>
</tr>
<tr>
<td>'0192 5B</td>
<td>0366</td>
<td>DEFB 5BH</td>
<td>;12-8-2</td>
<td></td>
</tr>
<tr>
<td>'0193 2E</td>
<td>0367</td>
<td>DEFB '.'</td>
<td>;12-8-3</td>
<td></td>
</tr>
<tr>
<td>'0194 3C</td>
<td>0368</td>
<td>DEFB '&lt;'</td>
<td>;12-8-4</td>
<td></td>
</tr>
<tr>
<td>'0195 28</td>
<td>0369</td>
<td>DEFB '('</td>
<td>;12-8-5</td>
<td></td>
</tr>
<tr>
<td>'0196 2B</td>
<td>0370</td>
<td>DEFB '+'</td>
<td>;12-8-6</td>
<td></td>
</tr>
<tr>
<td>'0197 5E</td>
<td>0371</td>
<td>DEFB 5EH</td>
<td>;12-8-7</td>
<td></td>
</tr>
<tr>
<td>'0198 49</td>
<td>0372</td>
<td>DEFB 'I'</td>
<td>;12-9-</td>
<td></td>
</tr>
<tr>
<td>'0199 01</td>
<td>0373</td>
<td>DEFB 01H</td>
<td>;12-9-1</td>
<td></td>
</tr>
<tr>
<td>'019A 02</td>
<td>0374</td>
<td>DEFB 02H</td>
<td>;12-9-2</td>
<td></td>
</tr>
<tr>
<td>'019B 03</td>
<td>0375</td>
<td>DEFB 03H</td>
<td>;12-9-3</td>
<td></td>
</tr>
<tr>
<td>'019C 00</td>
<td>0376</td>
<td>DEFB 0</td>
<td>;12-9-4</td>
<td></td>
</tr>
<tr>
<td>'019D 09</td>
<td>0377</td>
<td>DEFB 09H</td>
<td>;12-9-5</td>
<td></td>
</tr>
<tr>
<td>'019E 08</td>
<td>0378</td>
<td>DEFB 0</td>
<td>;12-9-6</td>
<td></td>
</tr>
<tr>
<td>'019F 7F</td>
<td>0379</td>
<td>DEFB 7FH</td>
<td>;12-9-7</td>
<td></td>
</tr>
<tr>
<td>'01A0 00</td>
<td>0380</td>
<td>DEFB 0</td>
<td>;12-9-8</td>
<td></td>
</tr>
<tr>
<td>'01A1 00</td>
<td>0381</td>
<td>DEFB 0</td>
<td>;12-9-9-1</td>
<td></td>
</tr>
<tr>
<td>'01A2 00</td>
<td>0382</td>
<td>DEFB 0</td>
<td>;12-9-8-2</td>
<td></td>
</tr>
<tr>
<td>'01A3 0B</td>
<td>0383</td>
<td>DEFB 0BH</td>
<td>;12-9-8-3</td>
<td></td>
</tr>
<tr>
<td>'01A4 0C</td>
<td>0384</td>
<td>DEFB 0CH</td>
<td>;12-9-8-4</td>
<td></td>
</tr>
<tr>
<td>'01A5 0D</td>
<td>0385</td>
<td>DEFB 0DH</td>
<td>;12-9-8-5</td>
<td></td>
</tr>
<tr>
<td>'01A6 0E</td>
<td>0386</td>
<td>DEFB 0EH</td>
<td>;12-9-8-6</td>
<td></td>
</tr>
<tr>
<td>'01A7 0F</td>
<td>0387</td>
<td>DEFB 0FH</td>
<td>;12-9-8-7</td>
<td></td>
</tr>
<tr>
<td>'01A8 7B</td>
<td>0388</td>
<td>DEFB 7BH</td>
<td>;12-0-</td>
<td></td>
</tr>
<tr>
<td>'01A9 61</td>
<td>0389</td>
<td>DEFB 61H</td>
<td>;12-0-1</td>
<td></td>
</tr>
<tr>
<td>'01AA 62</td>
<td>0390</td>
<td>DEFB 62H</td>
<td>;12-0-2</td>
<td></td>
</tr>
<tr>
<td>'01AB 63</td>
<td>0391</td>
<td>DEFB 63H</td>
<td>;12-0-3</td>
<td></td>
</tr>
<tr>
<td>'01AC 64</td>
<td>0392</td>
<td>DEFB 64H</td>
<td>;12-0-4</td>
<td></td>
</tr>
<tr>
<td>'01AD 65</td>
<td>0393</td>
<td>DEFB 65H</td>
<td>;12-0-5</td>
<td></td>
</tr>
<tr>
<td>'01AE 66</td>
<td>0394</td>
<td>DEFB 66H</td>
<td>;12-0-6</td>
<td></td>
</tr>
<tr>
<td>'01AF 67</td>
<td>0395</td>
<td>DEFB 67H</td>
<td>;12-0-7</td>
<td></td>
</tr>
<tr>
<td>'01B0 68</td>
<td>0396</td>
<td>DEFB 68H</td>
<td>;12-0-8</td>
<td></td>
</tr>
<tr>
<td>'01B1 00</td>
<td>0397</td>
<td>DEFB 0</td>
<td>;12-0-8-1</td>
<td></td>
</tr>
<tr>
<td>'01B2 00</td>
<td>0398</td>
<td>DEFB 0</td>
<td>;12-0-8-2</td>
<td></td>
</tr>
<tr>
<td>'01B3 00</td>
<td>0399</td>
<td>DEFB 0</td>
<td>;12-0-8-3</td>
<td></td>
</tr>
<tr>
<td>'01B4 00</td>
<td>0400</td>
<td>DEFB 0</td>
<td>;12-0-8-4</td>
<td></td>
</tr>
<tr>
<td>'01B5 00</td>
<td>0401</td>
<td>DEFB 0</td>
<td>;12-0-8-5</td>
<td></td>
</tr>
<tr>
<td>'01B6 00</td>
<td>0402</td>
<td>DEFB 0</td>
<td>;12-0-8-6</td>
<td></td>
</tr>
<tr>
<td>'01B7 00</td>
<td>0403</td>
<td>DEFB 0</td>
<td>;12-0-8-7</td>
<td></td>
</tr>
<tr>
<td>'01B8 69</td>
<td>0404</td>
<td>DEFB 69H</td>
<td>;12-0-9</td>
<td></td>
</tr>
<tr>
<td>'01B9 00</td>
<td>0405</td>
<td>DEFB 0</td>
<td>;12-0-9-1</td>
<td></td>
</tr>
<tr>
<td>'01BA 00</td>
<td>0406</td>
<td>DEFB 0</td>
<td>;12-0-9-2</td>
<td></td>
</tr>
<tr>
<td>'01BB 00</td>
<td>0407</td>
<td>DEFB 0</td>
<td>;12-0-9-3</td>
<td></td>
</tr>
<tr>
<td>ADDR</td>
<td>ST #</td>
<td>SOURCE STATEMENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01BC 00</td>
<td>0408</td>
<td>DEFB 0 ;12-0-9-4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01BD 00</td>
<td>0409</td>
<td>DEFB 0 ;12-0-9-5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01BE 00</td>
<td>0410</td>
<td>DEFB 0 ;12-0-9-6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01BF 00</td>
<td>0411</td>
<td>DEFB 0 ;12-0-9-7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01C0 00</td>
<td>0412</td>
<td>DEFB 0 ;12-0-9-8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01C1 00</td>
<td>0413</td>
<td>DEFB 0 ;12-0-9-8-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01C2 00</td>
<td>0414</td>
<td>DEFB 0 ;12-0-9-8-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01C3 00</td>
<td>0415</td>
<td>DEFB 0 ;12-0-9-8-3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01C4 00</td>
<td>0416</td>
<td>DEFB 0 ;12-0-9-8-4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01C5 00</td>
<td>0417</td>
<td>DEFB 0 ;12-0-9-8-5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01C6 00</td>
<td>0418</td>
<td>DEFB 0 ;12-0-9-8-6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01C7 00</td>
<td>0419</td>
<td>DEFB 0 ;12-0-9-8-7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01C8 7C</td>
<td>0420</td>
<td>DEFB 7CH ;12-11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01C9 6A</td>
<td>0421</td>
<td>DEFB 6AH ;12-11-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01CA 6B</td>
<td>0422</td>
<td>DEFB 6BH ;12-11-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01CB 6C</td>
<td>0423</td>
<td>DEFB 6CH ;12-11-3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01CC 6D</td>
<td>0424</td>
<td>DEFB 6DH ;12-11-4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01CD 6E</td>
<td>0425</td>
<td>DEFB 6EH ;12-11-5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01CE 6F</td>
<td>0426</td>
<td>DEFB 6FH ;12-11-6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01CF 70</td>
<td>0427</td>
<td>DEFB 70H ;12-11-7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01D0 71</td>
<td>0428</td>
<td>DEFB 71H ;12-11-8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01D1 00</td>
<td>0429</td>
<td>DEFB 0 ;12-11-8-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01D2 00</td>
<td>0430</td>
<td>DEFB 0 ;12-11-8-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01D3 00</td>
<td>0431</td>
<td>DEFB 0 ;12-11-8-3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01D4 00</td>
<td>0432</td>
<td>DEFB 0 ;12-11-8-4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01D5 00</td>
<td>0433</td>
<td>DEFB 0 ;12-11-8-5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01D6 00</td>
<td>0434</td>
<td>DEFB 0 ;12-11-8-6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01D7 00</td>
<td>0435</td>
<td>DEFB 0 ;12-11-8-7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01D8 72</td>
<td>0436</td>
<td>DEFB 72H ;12-11-9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01D9 00</td>
<td>0437</td>
<td>DEFB 0 ;12-11-9-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01DA 00</td>
<td>0438</td>
<td>DEFB 0 ;12-11-9-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01DB 00</td>
<td>0439</td>
<td>DEFB 0 ;12-11-9-3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01DC 00</td>
<td>0440</td>
<td>DEFB 0 ;12-11-9-4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01DD 00</td>
<td>0441</td>
<td>DEFB 0 ;12-11-9-5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01DE 00</td>
<td>0442</td>
<td>DEFB 0 ;12-11-9-6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01DF 00</td>
<td>0443</td>
<td>DEFB 0 ;12-11-9-7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01E0 00</td>
<td>0444</td>
<td>DEFB 0 ;12-11-9-8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01E1 10</td>
<td>0445</td>
<td>DEFB 10H ;12-11-9-8-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01E2 00</td>
<td>0446</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01E3 00</td>
<td>0447</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01E4 00</td>
<td>0448</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01E5 00</td>
<td>0449</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01E6 00</td>
<td>0450</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01E7 00</td>
<td>0451</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01E8 00</td>
<td>0452</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01E9 00</td>
<td>0453</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01EA 00</td>
<td>0454</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01EB 00</td>
<td>0455</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01EC 00</td>
<td>0456</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01ED 00</td>
<td>0457</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01EE 00</td>
<td>0458</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01EF 00</td>
<td>0459</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01F0 00</td>
<td>0460</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01F1 00</td>
<td>0461</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01F2 00</td>
<td>0462</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01F3 00</td>
<td>0463</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01F4 00</td>
<td>0464</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01F5 00</td>
<td>0465</td>
<td>DEFB 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDR</td>
<td>OBJECT</td>
<td>ST#</td>
<td>SOURCE</td>
<td>STATEMENT</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>-----</td>
<td>--------</td>
<td>-----------</td>
</tr>
<tr>
<td>'01F6 00</td>
<td>0466</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01F7 00</td>
<td>0467</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01F8 00</td>
<td>0468</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01F9 00</td>
<td>0469</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01FA 00</td>
<td>0470</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01FB 00</td>
<td>0471</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01FC 00</td>
<td>0472</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01FD 00</td>
<td>0473</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01FE 00</td>
<td>0474</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01FF 00</td>
<td>0475</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0200 00</td>
<td>0476</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0201 00</td>
<td>0477</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0202 00</td>
<td>0478</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0203 00</td>
<td>0479</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0204 00</td>
<td>0480</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0205 00</td>
<td>0481</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0206 00</td>
<td>0482</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0207 00</td>
<td>0483</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0208 00</td>
<td>0484</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0209 00</td>
<td>0485</td>
<td>DEF B 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0486</td>
<td>END</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ERRORS=0000
NAME PP

'; PAPER TAPE PUNCH DRIVER FOR FLP-80DOS V2.0

0006 GLOBAL PP
0007 GLOBAL EH
0008

>FF00 0009 TOR EQU 0FF00H
>0002 0010 IRET EQU 2
>0015 0011 CFLGS EQU 21
>0007 0012 TIMOUT EQU 7
0013 ;
0014 ;
0000 0015 PP EQU $
0000 04 0016 DEFB 4 ; MAX RQST
0001 00 0017 DEFB 0
0002 05 0018 DEFB PPOPEN-S
0003 2D 0019 DEFB PPCLOS-S
0004 00 0020 DEFB 0
0005 2C 0021 DEFB PPWRIT-S
0022 ;
0006 AA 0023 PPST DEFB OA0AH ; PAPER TAPE PUNCH PORT
>000B 0024 PPDIS EQU OBH ; OFFSET FROM TOR FOR VECTOR
0025 ;
0007 F3 0026 PPOPEN DI ; OPEN DEVICE
0008 2A00FF 0027 LD HL,(TOR) ; ACCESS INTERRUPT TABLE
000B 110B00 0028 LD DE,PPDIS ; VECTOR OFFSET FROM TOR
000E B7 0029 OR A
000F ED52 0030 SBC HL,DE
0011 E5 0031 PUSH HL
0012 116D00' 0032 LD DE,PINT ; DE -> INTERRUPT HANDLER
0015 73 0033 LD (HL),E ; SAVE VECTOR ADDRESS
0016 23 0034 INC HL
0017 72 0035 LD (HL),D
0018 D1 0036 POP DE ; DE = VECTOR ADDRESS
0019 210600' 0037 LD HL,PPST ; HL -> STATUS BYTE
001C CBC6 0038 SET 0,(HL) ; SET READY BIT
001E 4E 0039 LD C,(HL) ; GET PORT FOR CONTROL
001F 3E0F 0040 LD A,0FH ; OUTPUT CONTROL
0021 ED79 0041 OUT (C),A
0023 ED59 0042 OUT (C),E ; OUTPUT INT P VECTOR LSBYTE
0025 3E83 0043 LD A,83H ; OUTPUT CONTROL
0027 ED79 0044 OUT (C),A
0029 7A 0045 LD A,D ; SET VECTOR MSBYTE
002A ED47 0046 LD I,A
002C 010100 0047 LD BC,1 ; PHYSICAL RECORD SIZE
002F FB 0048 EI
0030 C9 0049 PPCLOSE RET ; RETURN TO CALLER
0050 ;
0051 ;
0031 E5 0052 PPWRIT PUSH HL
0032 C5 0053 PUSH BC
0033 F5 0054 PUSH AF ; SAVE BYTE TO OUTPUT
0034 210600' 0055 LD HL,PPST ; HL -> STATUS BYTE
0037 01D007 0056 LD BC,2000 ; 2000 MSEC TIME OUT COUNT
003A C5 0057 PPA PUSH BC
003B 0629 0058 LD B,41 ; MSEC COUNTER
003D FB 0059 PPL EI ; ENABLE INTPS
'003E CB46 0060 BIT 0, (HL) ; CHECK FOR READY
'0040 201D 0061 JR NZ, PPR-S ; YES, SKIP
'0042 FDCB1556 0062 BIT IRET, (IY+CFLGS) ; CHECK IMMED RETURN
'0046 2012 0063 JR NZ, PPI-S ; YES, SKIP OUT
'0048 10F3 0064 DJNZ PPL-S ; LOOP FOR TIMEOUT
'004A C1 0065 POP BC
'004B 0B 0066 DEC BC ; DECREMENT COUNT
'004C 78 0067 LD A, B
'004D B1 0068 OR C
'004E 20EA 0069 JR NZ, PPA-S ; LOOP FOR TIMEOUT
'0050 3E07 0070 LD A, TIMOUT ; TIMEOUT ERROR MESSAGE
'0052 CDFFFF 0071 CALL EH ; OUTPUT THE MESSAGE
'0055 01204E 0072 LD BC, 20000 ; NEW TIMEOUT
'0058 18E0 0073 JR PPA-S ; LOOP AGAIN
'005A C1 0075 PPI POP BC ; RESTORE STACK
'005B F1 0076 POP AF ; RESTORE BYTE
'005C C1 0077 POP BC ; RESTORE REGS
'005D E1 0078 POP HL
'005E C9 0079 RET ; RETURN TO CALLER
'0060 E1 0082 POP AF ; GET BYTE
'0061 CB86 0083 RES 0, (HL) ; RESET READY BIT
'0063 4E 0084 LD C, (HL) ; GET PORT NUMBER
'0064 ED79 0085 OUT (C), A ; OUTPUT DATA TO PP
'0066 FDCB1596 0086 RES IRET, (IY+CFLGS) ; RESET IMMED RETURN BIT
'006A C1 0087 POP BC ; RESTORE REGS
'006B E1 0088 POP HL
'006C C9 0089 RET ; RETURN TO CALLER
'006D E5 0092 PINT PUSH HL ; PAPER TAPE PUNCH INT HANDLER
'006E 210600' 0093 LD HL, PPST
'0071 CBC6 0094 SET 0, (HL)
'0073 E1 0095 POP HL
'0074 FB 0096 EI
'0075 ED4D 0097 RETI

ERRORS=0000
NAME PR

; PAPER TAPE READER DRIVER FOR FLP-80DOS V2.0

GLOBAL PR

GLOBAL EH

TIMOUT EQU 7
TOR EQU OFF00H
IRET EQU 2
CFLGS EQU 21

PR EQU $

DEFB 3 ; MAX REQST
DEFB PROPEN-$
DEFB OPENW
DEFB PRCLOS-$
DEFB PRREAD-$

PRST DEFB 0A8H ; READER PORT NUMBER

PRDIS EQU 09 ; VECTOR OFFSET FROM TOR

PROPEN DI ; DISABLE INTPS
LD HL,(TOR) ; ACCESS INTERRUPT TABLE
DE,PRDIS
OR A
ED52 SBC HL,DE ; ACCESS START OF TABLE
E5 Push HL ; SAVE IT
116F00' LD DE,RINT ; PR INTERRUPT IS FIRST ENTRY
73 LD (HL),E ; SAVE HANDLER ADDRESS
23 INC HL ; IN INTP TABLE
72 LD (HL),D
D1 POP DE ; DE = VECTOR ADDRESS
210500' LD HL,PRST ; HL = STATUS BYTE
C8C6 SET 0,(HL) ; SET FOR CONTROL
4E LD C,(HL) ; GET PORT NUMBER
3E4F LD A,4FH ; OUTPUT CONTROL
ED79 OUT (C),A
ED59 OUT (C),E ; OUTPUT VECTOR LSBYTE
3E83 LD A,33H ; OUTPUT CONTROL
ED79 OUT (C),A
7A LD A,D ; SET UP VECTOR MSBYTE
ED47 LD I,A
C886 RES 0,(HL) ; INIT STATUS BIT
4E LD C,(HL) ; GET PORT
ED70 IN F,(C) ; READ PORT TO INITIALIZE OPERATIC
010100 LD BC,1 ; PHYSICAL RECORD SIZE = 1 BYTE
FB EI ; ENABLE INTPS
C9 PRCLOS RET ; RETURN TO CALLER

PRREAD PUSH HL
PRDIS PUSH BC
LD HL,PRST ; HL -> STATUS BYTE
LD BC,250 ; TIMEOUT = 250 MSEC
PRA PUSH BC ; SAVE
ADDR | OBJECT | ST # | SOURCE | STATEMENT | DATASET = DKO:PR .SRC

'003E 0629 | 0060 | LD B,41 ;MSEC COUNTER
'0040 FB | 0061 PRL | EI ;ENABLE INTPS
'0041 CB46 | 0062 | BIT 0,(HL) ;CHECK IF READY
'0043 2018 | 0063 | JR NZ,PRR-S ;YES, SKIP
'0045 FDCB1556 | 0064 | BIT IRET,(IY+CFLGS) ;CHECK FOR IMMED RETURN
'0049 2020 | 0065 | JR NZ,PRI-S ;IF SO, SKIP OUT
'004B 10F3 | 0066 | DJNZ PRL-$ ;LOOP FOR TIMEOUT
'004D C1 | 0067 | POP BC
'004E 05 | 0068 | DEC BC ;DECREMENT COUNTER
'0050 78 | 0069 | LD A,B ;CHECK COUNT
'0051 B1 | 0070 OR C
'0053 3E07 | 0072 | LD A,TIMOUT ;TIME OUT ERROR CODE
'0055 CDFFFF | 0073 | CALL EH
'0058 01204E | 0074 | LD BC,20000 ;NEW TIME OUT COUNT
'005B 18E0 | 0075 | JR PRA-$ 0076 ;
'005D C1 | 0077 PRR | POP BC
'005E CB86 | 0078 | RES 0,(HL) ;ZERO DATA AVAILABLE FLAG
'0060 4E | 0079 | LD C,(HL) ;GET PORT FOR DATA
'0061 ED78 | 0080 | IN A,(C) ;GET DATA
'0063 2F | 0081 | CPL ;COMPLEMENT THE DATA
'0064 FDCB1596 | 0082 | RES IRET,(IY+CFLGS) ;RESET IMMED RETURN
'0068 C1 | 0083 | POP BC
'0069 E1 | 0084 | POP HL
'006A C9 | 0085 | RET ;RETURN TO CALLER 0086 ;
'006B C1 | 0087 PRI | POP BC
'006C C1 | 0088 | POP BC
'006D E1 | 0089 | POP HL
'006E C9 | 0090 | RET ;RETURN TO CALLER 0091 ;
'0092 ;

'006F E5 | 0093 RINT | PUSH HL ;READER INTERRUPT HANDLER
'0070 210500' | 0094 | LD HL,PRST
'0073 CBC6 | 0095 | SET 0,(HL) ;SET READY BIT
'0075 E1 | 0096 | POP HL
'0076 FB | 0097 | EI
'0077 ED4D | 0098 | RETI

ERRORS=0000
NAME TI
SILENT TAPE INPUT HANDLER FOR FLP-80DCS V2.0
COMPATIBLE WITH PREVIOUS SYSTEMS
HSCR DC1 DC3
TIOPEN
TIREAD
GLOBAL MINDIS
GLOBAL MINEN
GLOBAL TI

0005 FD361E00 0023 TIOPEN LD (IY+HSCR),0 ;ZERO BUFFER COUNTER
0009 FD362000 0024 LD (IY+HSCR+2),0 ;ZERO NULL COUNTER
000D CDFFFF 0025 CALL MINDIS ;DISABLE MINIMAL LISTENER
0010 010100 0026 LD BC,1 ;PHYSICAL RECORD SIZE
0013 C9 0027 RET
0014 E5 0029 TIREAD PUSH HL
0015 C5 0030 PUSH BC
0016 FD7E1E 0031 LD A,(IY+HSCR) ;GET BUFFER COUNTER
0019 A7 0032 AND A ;CHECK IT
001A 2051 0033 JR NZ,TIB-S ;IF NOT ZERO, SKIP
0034 ;READ A RECORD FROM TAPE INTO THE BUFFER
0036 ;START THE TRANSPORT
003E CD0D01 0038 CALL S700P
0021 218000 0039 LD HL,TIBUF ;HL -> BUFFER
0024 01D007 0040 TI1 LD BC,2000 ;SECOND TIMEOUT
0027 C5 0041 TI1A PUSH BC
0028 0630 0042 LD B,48 ;MSEC COUNT
002A DBDD 0043 TI2 IN A,(ODDH) ;CHECK THE UART STATUS
002C CB77 0044 BIT 6,A
002E 200A 0045 JR NZ,TI3-S ;IF READY, SKIP
0030 10F8 0046 DJNZ TI2-S ;LOOP FOR MSECOND
0032 C1 0047 POP BC
0033 0B 0048 DEC BC ;DECREMENT BC COUNTER
0034 78 0049 LD A,B ;CHECK TIMEOUT COUNTER
0035 B1 0050 OR C
0036 20EF 0051 JR NZ,TI1A-S ;IF NOT TIMEOUT, LOOP
0038 1811 0052 JR TI3A-S ;ELSE FAKE AN END OF FILE
003A C1 0054 TI3 POP BC
003B DBDC 0055 IN A,(ODCH) ;GET DATA BYTE
003D E57F 0056 AND 7FH ;REMOVE PARITY
003E 200E 0057 JR NZ,TI4-S ;IF NOT NULL, SKIP
0058 ;NULL FOUND, COUNT IT. IF UP TO 127 NULLS,
0059 ;FORCE EOT = 04H.
ERRORS=0000
0002    NAME    TK
0003 ;**************************************************************
0004 ;*KEYBOARD INPUT DRIVER AND * 
0005 ;*MINIMAL LISTNER SERVICE ROUTINE.* 
0006 ;* ID: TK *  
0007 ;* PROGRAMMER: JOHN BATES * 
0008 ;* M. FREEMAN *  
0009 ;* DATE: 6/1/78 *  
0010 ;**************************************************************
0011 ;** INTERNAL GLOBAL VARIABLES** 
0012 
0013 
0014 
0015 
0016 GLOBA L TK 
0017 GLOBA L MINLIS 
0018 
0019 
0020 
0021 
0022 
0023 
0024 
0025 
0026 CTC1 EQU 0D9H 
0027 CFLGS EQU 21 
0028 IRET EQU 2 
0029 TKST EQU OFF25H 
0030 MINFLG EQU OFF24H 
0031 COUNT EQU OFF06H 
0032 LONG EQU OFF13H 
0033 UCTL EQU ODCH ;UAR T CONTROL PORT 
0034 UDATA EQU ODDH 
0035 ETX EQU 03 
0036 CAN EQU 18H 
0037 ;**************************************************************
0038 ;* TK INPUT DRIVER * 
0039 ;**************************************************************
0040 TK EQU S 
0041 DEFB 3 ;MAX REQUEST 
0042 DEFB TKOPEN-S ;OPENR 
0043 DEFB 0 ;OPENW 
0044 DEFB TKCLOS-S ;CLOSE 
0045 DEFB TKREAD-S ;READ 
0046 ;
0047 TKOPEN EQU S 
0048 LD A,3 ;TURN ON CTS 
0049 TKO1 OUT (0DEH),A 
0050 LD BC,1 ;PHYS REC SIZE 
0051 TKCLOS RET ;RETURN TO CALLER 
0052 ;
0053 ;
0054 TKREAD EQU S 
0055 TTI LD A,(TKST) ;FROM ESCAPE TEST 
0056 OR A ;IF NZ 
0057 JR NZ,TTID1A-S 
0058 TTIDO IN A,(ODDH) ;CHECK UART STATUS 
0059 BIT 6,A
'0017 2008 0060  JR  NZ,TTID1-S ;READY, SKIP
'0019 FDCCB1556 0061  BIT  IRET,(IY+CFLGS) ;IMMED RETURN?
'001D 28EE 0062  JR  Z,TTI-S ;NO, LOOP
'001F BF 0063  CP  A
'0020 C9 0064  RET  ; YES, EXIT

'0021 0066  TTID1 EQU $2008
'0022 DBDC 0067  IN  A,(ODCH) ;GET DATA
'0023 FDCCB1596 0068  TTID1A RES  IRET,(IY+CFLGS) ;CLEAR IMMED RET BIT
'0027 CBBF 0069  RES  7,A ;CLEAR PARITY
'0029 F5 0070  PUSH AF
'002A AF 0071  XOR A
'002B 3225FF 0072  LD  (TKST),A ; CLEAR HOLD REG.
'002E 3A24FF 0073  LD  A,(MINFLG) ;MINIMAL LISTNER ENABLE
'0031 B7 0074  OR A
'0032 2002 0075  JR  NZ,TT1D1B-S ;IF YES, TEST FOR TRAPS
'0034 F1 0076  POP AF
'0035 C9 0077  RET
'0036 F1 0078  TT1D1B POP AF
'0037 FE18 0079  CP  CAN
'0039 280E 0080  JR  Z,TTICAN-S
'003B FE03 0081  CP  ETX
'003D C0 0082  RET  NZ ;NORMAL DATA
'003E 3E01 0083  LD  A,1 ;EXIT TO DDT
'0040 3213FF 0084  LD  (LONG),A
'0043 3206FF 0085  LD  (COUNT),A
'0046 C3FFFE 0086  JP  ENTRY ;JUMP TO DDT BREAK PT
'0049 3E01 0087  TTICAN LD  A,1
'004B D3D9 0088  OUT  (0D9H),A ;KILL MIN. LIST.
'004D C3FFFE 0089  JP  REBOOT

0090 ;*******************************
0091 ;*
0092 ;* MINIMAL LISTNER INTERRUPT  *
0093 ;* SERVICE ROUTINE   *
0094 ;* 
0095 ;*******************************

'0050 F5 0096  MINLIS PUSH AF ;SAVE CHARACTER
'0051 DBDD 0097  IN  A,(UCTL) ;DATA READY ?
'0053 CB77 0098  BIT  6,A
'0055 281F 0099  JR  Z,MLIS1-S
'0057 DBDC 0100  IN  A,(UDATA) ;GET A CHAR
'0059 E67F 0101  AND  7FH
'005B FE18 0102  CP  CAN ;CNTL X ?
'005D 281B 0103  JR  Z,MLIS3-S
'005F FE03 0104  CP  ETX ;CNTL C
'0061 2010 0105  JR  NZ,MLIS2-S ;EXIT
'0063 3E01 0106  LD  A,1
'0065 3206FF 0107  LD  (COUNT),A
'0068 3213FF 0108  LD  (LONG),A
'006B F1 0109  POP  AF ;GOTO DDT
'006C E5 0110  PUSH HL
'006D 214700' 0111  LD  HL,ENTRY;CTL C TRAP TO DDT
'0070 E3 0112  MLIS4  EX  (SP),HL
'0071 1804 0113  JR  MLISO-S
'0073 3225FF 0114  MLIS2  LD  (TKST),A ;SAVE FOR BACKGROUND
'0076 F1 0115  MLIS1 POP AF ;SAVE AF
'0077 FB 0116  MLISO EI
'0078 ED4D 0117  RETI ;EXIT
IDDR OBJECT ST # SOURCE STATEMENT DATASET = DKO:TK .SRC

07A 3E01 0118 MLIS3 LD A, 1
07C D3D9 0119 OUT (CTC1), A ; TURN OFF MINIMAL LISTNER
07E 214E00 ' 0120 LD HL, REBOOT ; CTL X TRAP TO B001
081 18ED 0121 JR MLIS4-S
 0122 ;
 0123 END

RRORS=0000
NAME TO 0002
0003 ; SILENT 700 TAPE OUTPUT HANDLER
0004 ; COMPATIBLE WITH PREVIOUS SYSTEMS
0005 ; FOR FLP-80DOS V2.0
0006 ;
0007 ;
0008 ; GLOBAL TO
0009 ;
>0010 HSCR EQU 30
>0011 DC1 EQU 11H
>0012 DC2 EQU 12H
>0013 DC3 EQU 13H
>0014 DC4 EQU 14H
0015 ;
0016 ;
>0017 TO EOF EQU S
0018 DEFB 4
0019 DEFB 0
0020 DEFB TOOPEN-S
0021 DEFB TOCLOS-S
0022 DEFB 0
0023 DEFB TOWRIT-S
0024 ;
0025 ;
'0006 FD361E00 0026 TOOPEN LD (IY+HSCR),O ;ZERO POINTER
'000A 010100 0027 LD BC,1 ;PHYS RECORD SIZE
'000D C9 0028 RET ;RETURN TO CALLER
0029 ;
0030 ;
'000E C5 0031 TOWRIT PUSH BC
'000F E5 0032 PUSH HL
'0010 FD4E1E 0033 LD C,(IY+HSCR) ;GET BUFFER COUNT
'0013 0500 0034 LD B,O
'0015 217A00' 0035 LD HL,TOTBUF ;HL -> BLOCKING BUFFER
'0018 09 0036 ADD HL,BC ;GET TO POINT IN BUFFER
'0019 77 0037 LD (HL),A ;PUT CHAR INTO BUFFER
'001A FD341E 0038 INC (IY+HSCR) ;INCREMENT POINTER
'001D FE0A 0039 CP 0AH ;CHECK FOR LF
'001F 2804 0040 JR Z,T02-S ;YES, SKIP OUT
'0021 FE04 0041 CP 4 ;CHECK FOR END OF FILE
'0023 2048 0042 JR NZ,T05-S
0043 ;
0044 ; WRITE OUT BUFFER TO DEVICE
0045 ;
'0025 3E12 0046 TOB LD A,DC2 ;START RECORD OPERATION
'0027 CDFA00' 0047 CALL S700P
'002A 217A00' 0048 LD HL,TOTBUF ;HL -> BUFFER
0049 ;
'002D 7E 0050 TO2 LD A,(HL) ;GET CHARACTER FROM BUFFER
'002E 23 0051 INC HL
'002F FE7F 0052 CP 7FH ;IGNORE RUBOUT
'0031 28FA 0053 JR Z,T02-S
'0033 FE11 0054 CP DC1 ;IGNORE DC1 - DC4
'0035 3804 0055 JR C,T03-S
'0037 FE15 0056 CP DC4+1
'0039 38F2 0057 JR C,T02-S
0058 ;
'003B CDFA00' 0059 TO3 CALL S700P ;OUTPUT THE CHARACTER
TO COPYRIGHT 1978 MOSTEK CORP MOSTEK FLP-80 ASSEMBLER V2.0 PAGE 00

ADDR OBJECT ST # SOURCE STATEMENT DATASET = DKO:STO .SRC

'003E FE04 0060 CP 4 ;CHECK FOR END OF FILE
'0040 200F 0061 JR NZ,T03A-$ ;NO, SKIP
0062 ; OUTPUT 86 NULLS TO FLUSH BUFFER TO
0063 ; TERMINATE CONTINUOUS MODE

'0042 0656 0064 LD B,86
'0044 AF 0065 T03L XOR A
'0045 CDFA00' 0066 CALL S700P
'0048 10FA 0067 DJNZ T03L-S
0068 ; OUTPUT A CARRIAGE RETURN AFTER THE EOT

'004A 3E0D 0069 LD A,0DH ;FOR LINE MODE TERMINATION
'004C CDFA00' 0070 CALL S700P
'004F 1804 0071 JR TO4-$ ;AND SKIP OUT
0072 ;

'0051 FE0A 0073 T03A CP OAH ;CHECK FOR LF
'0053 20D8 0074 JR NZ,T02-S ;IF NOT, LOOP FOR MORE

'0055 3E13 0075 T04 LD A,DC3 ;OUTPUT CONTROL CHARACTERS

'0057 CDFA00' 0076 T04 CALL S700P ;AT END OF RECORD

'0059 3E7F 0077 LD A,7FH
'005C CDFA00' 0078 CALL S700P
'005F 3E14 0079 LD A,DC4

'0061 CDFA00' 0080 CALL S700P

'0064 3E7F 0081 CALL S700P
'0066 CDFA00' 0082 LD A,7FH

'0069 FD361E00 0083 CALL S700P

'006D E1 0084 T05 POP HL

'006E C1 0085 POP BC
'006F C9 0086 RET ;RETURN TO CALLER
0088 ;
0089 ;

'0070 3E14 0090 TOCLOS LD A,DC4 ;ASSURE TAPE IS OFF
'0072 CDFA00' 0091 CALL S700P
'0075 FD361E00 0092 LD (IY+HSCR),0 ;REINIT BUFFER POINTER

'0079 C9 0093 RET

'>007A 0095 TOBUF DEFS 128

'00FA F5 0096 S700P PUSH AF ;SAVE BYTE
'00FB DBDD 0097 S700R IN A,(ODDH) ;CHECK UART STATUS
'00FD CB7F 0098 S700R BIT 7,A
'00FF 28FA 0100 JR Z,S700R-S
'0101 F1 0101 POP AF ;OUTPUT THE BYTE

'0102 D3DC 0102 OUT (ODCH),A

'0104 C9 0103 RET
0104 ;

ERRORS=0000
NAME TR

TITLE: DRIVER FOR TELETYPewriter TAPE READER *

ID: PR VERSION 2.0 *

PROGRAMMER: JOHN BATES *

DATE: 6/20/78 *

************************************************

SYSTEM EQUATES

EQU EQU EQU EQU EQU EQU EQU EQU EQU GLOBAL EQU EQU EQU EQU

DC1 DC2 DC3 DC4 ERC CFLGS TIMOUT MS250 IRET TROPEN-$

DCB,1 DCB,1 DCB,1 DCB,1 DCB,1 DCB,1 DCB,1 DCB,1 DCB,1 DCB,1 DCB,1 DCB,1

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

EQU EQU EQU EQU EQU EQU EQU EQU EQU EQU EQU EQU EQU EQU EQU EQU EQU EQU EQU

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB

CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL CALL

DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB DEFB
0027 C1 0060 POP BC ; ERROR OUT
0028 C9 0061 RET
0029 3E03 0063 TRD2 LD A,3 ; TURN OFF READER
002B D3DE 0064 OUT (ODEH),A
002D C1 0065 POP BC
002E 180B 0066 JR TKREAD-S ; GET CHAR
0030 F5 0068 TTTWRIT PUSH AF ; SAVE CHAR
0031 DBDD 0069 TTOD0 IN A,(ODDH) ; CHECK UART STATUS
0033 CB7F 0070 BIT 7,A
0035 28FA 0071 JR Z,TTOD0-S ; IF NOT READY, LOOP
0037 F1 0072 TTOD1 POP AF ; RESTORE CHARACTER
0038 D3DC 0073 OUT (ODCH),A ; OUTPUT IT
003A C9 0074 RET
003B DBDD 0076 TKREAD IN A,(ODDH) ; CHECK UART STATUS
003D CB77 0077 BIT 6,A
003F 2008 0078 JR NZ,TTID1-S ; READ, SKIP
0041 FDCB1556 0079 BIT IRET,(IY+CFLGS); IMMED RETURN?
0045 28F4 0080 JR Z,TKREAD-S ; NO, LOOP
0047 BF 0081 CP A
0048 C9 0082 RET ; YRES, EXIT
0049 DBDC 0084 TTID1 IN A,(ODCH) ; GET DATA
004B FDCB1596 0085 TTID1A RES IRET,(IY+CFLGS); CLEAR IMMED RET BIT
004F CB8F 0086 RES 7,A ; CLEAR PARITY
0051 C9 0087 RET
0088 ;
0089 END

ERRORS=0000
GLOBAL EQU TELETYPE, S700

BRATE LWIDTH HCTR

TTOPEN TTCLOS WRITE

DEFB EQU DEFB DEFB DEFB DEFB DEFB

LD RET

OFFEOH 80 iBAUD RATE VARIABLE

iTERMINAL LINE WIDTH

iHORIZONTAL COLUMN COUNTER

BC,1 iPHYSICAL RECORD SIZE = 1

NZ,WRITE1-$ iBACKSPACE

iDECREMENT CHARACTER COUNTER

NZ,TT6A-$ iBACKSPACE = OFH

A,(HCTR) iCHAR = TAB?

FOR
'0032 FE0D 0060 TT6A CP ODH ;DO NOT INCREMENT HCTR
'0034 3819 0061 JR C,TT20-$ ;FOR LF=OA OR FF=OC.
'0036 2811 0062 JR Z,TT14-$ ;IF CHAR=CR CLEAR HCTR
'0038 3A0600' 0063 LD A,(HCTR)
'003B FE50 0064 CP LWIDTH ;END OF LINE REACHED?
'003D 200C 0065 JR NZ,TT16-$ ;IF NOT INCREMENT HCTR
'003F 3E0D 0066 LD A,0DH ;IF END OF LINE IS
'0041 CD2400' 0067 CALL WRITE1 ;REACHED THEN AUTOMATICAL
'0044 3E0A 0068 LD A,0AH ;OUTPUT A CR AND LF.
'0046 CD2400' 0069 CALL WRITE1
'0049 3EFF 0070 TT14 LD A,0FFH
'004B 3C 0071 TT16 INC A
'004C 320600' 0072 LD (HCTR),A
'004F 3AE0FF 0073 TT20 LD A,(BRATE)
'0052 FE10 0074 CP 010H ;600 BAUD?
'0054 2804 0075 JR Z,CRT-S
'0056 FE08 0076 JR O8H ;110, 300, 1200 BAUD?
'0058 300F 0077 JR NC,TTF7-S
'0078 ;
'0079 ;
'0080 ;
'005A F1 0081 CRT POP AF ;RESTORE CHARACTER
'005B FE04 0082 TTWRIT CP 04 ;IGNORE 04
'005D C8 0083 RET Z
'005E F5 0084 PUSH AF ;SAVE CHAR
'005F DBDD 0085 TT100 IN A,(ODDH) ;CHECK UART STATUS
'0061 CB7F 0086 BIT 7,A
'0063 28FA 0087 JR Z,TT100-S ;IF NOT READY, LOOP
'0065 F1 0088 POP AF ;RESTORE CHAR
'0066 D3DC 0089 OUT (ODCH),A ;OUTPUT IT
'0068 C9 0090 RET

'0069 F1 0091 TTFF POP AF ;RESTORE CHARACTER
'006A FE0C 0092 CP OCH ;FORM FEED?
'006C 200B 0093 JR NZ,STWRIT-S
'006E C5 0094 PUSH BC
'006F 0605 0095 LD B,5 ;IF FORM FEED THEN OUTPUT
'0071 3E0A 0096 LD A,0AH ;6 LINE FEEDS
'0073 CD7900' 0097 STO CALL STWRIT
'0076 10FB 0098 DJNZ STO-S
'0078 C1 0099 POP BC ;RESTORE BC
'0079 F5 0100 STWRIT PUSH AF ;SAVE CHAR
'007A CD5B00' 0101 CALL TTWRIT ;OUTPUT CHARACTER
'007D 3AE0FF 0102 LD A,(BRATE)
'0080 FE57 0103 CP 57H ;110 BAUD?
'0082 280C 0104 JR Z,TTRET-S
'0084 FE08 0105 CP O8H ;1200 BAUD?
'0086 CC9200' 0106 CALL Z,DEL32 ;DELAY 32 MSEC IF 1200 B.
'0089 F1 0107 POP AF ;RESTORE CHAR
'008A FE0D 0108 CP OOH
'008C CC9700' 0109 CALL Z,DEL210 ;DELAY 210 MSEC IF CHAR=C
'008F C9 0110 RET
'0090 F1 0111 TTRET POP AF ;RESTORE CHAR
1091 C9 0118 RET
1092 C5 0120 DEL32 PUSH BC ; DELAY 32 MSEC
1093 0E20 0121 LD C,32
1095 1803 0122 JR DELAY-S
1097 C5 0123 DEL210 PUSH BC ; DELAY 210 MSEC
1098 0ED2 0124 LD C,210
109A 06BF 0125 DELAY LD B,191
109C 10FE 0126 DEL1 DJNZ DEL1-S ; 1 MSEC DELAY
109E 0D 0127 DEC C
109F 20F9 0128 JR NZ,DELAY-S
10A1 C1 0129 POP BC
10A2 C9 0130 RET
1031 ;
SECTION 13
SYSTEM ROUTINES

13-1. INTRODUCTION

13-2. Many subroutines in FLP-80DOS are accessable to the user. The following pages describe these routines, which fall into two major categories: PROM resident routines and RAM resident routines (within the RAM portion of the operating system).

13-3. PROM RESIDENT ROUTINES

13-4. Since the routines located in PROM reside at fixed addresses, they may be called directly. The usual method of calling one of these routines is to declare the name of that routine as a GLOBAL symbol. The routine may then be called just as if it resided within the calling program. To actually resolve the calling address of the routine, the file SYSLNK.OBJ must be included when linking the program.

13-5. Example. Suppose that the System Error Handler (EH) is to be called with an error number held in variable "ERRCOD."

```
GLOBAL EH ;SYSTEM ERROR HANDLER
.
.
.
ERROR LD A,(ERRCOD) ;GET ERROR CODE
CALL EH ;
.
.
.
```
When the program is linked, SYSLNK.OBJ would be included. $LINK PROG,SYSLNK TO PROG(CR)

13-6. RAM RESIDENT ROUTINES

13-7. User callable system subroutines that reside within the RAM-based portion of the operating system may not be accessed in the same manner as the PROM resident routines. With the SYSGEN Feature in FLP-80DOS, the user is given the option to position the operating system at any location in RAM. This positioning causes the addresses of the callable routines within the operating system to change depending on where the current operating system was positioned during the SYSGEN procedure (See Section 15).

13-8. This problem is solved in the following manner. A routine called JTASK is located in scratchpad RAM and has a fixed address. JTASK contains a mechanism for locating all RAM resident callable routines. Each of these routines has been assigned a number which is placed into register A just prior to calling JTASK. JTASK then jumps to the appropriate routine (all other calling parameters are as described for that routine later in this section). These codes are listed below. Individual routines not reserved for system use are described in greater detail later in this section.

<table>
<thead>
<tr>
<th>Code</th>
<th>Routine</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>FDH (Floppy Disk Handler). Described in Section 10.</td>
</tr>
<tr>
<td>1</td>
<td>MRENT (Monitor Reentry Point).</td>
</tr>
<tr>
<td>2</td>
<td>IOCS RDC (Read Character), reserved for system use.</td>
</tr>
<tr>
<td>3</td>
<td>IOCS WRC (Write Character), reserved for system use.</td>
</tr>
</tbody>
</table>
13-3

4 PVECT (Print Vector Contents).
5 GETLIN (Get Line Into Monitor Command Buffer).
   Reserved for system use.
6 CSIPAR (Parse Dataset Specifications Into Vector).
7 CSISYN (Check Syntax of Dataset Specifications).
8 ASTCHK (Check For Asterisk In I/O Vector).
9 GETHL (Get Line From Console Into Buffer).
10 GETVEC (Get Address of Default LUN Buffer).
11 SEARCH (Get Directory entry for a given file).

13-9. The following is an example of the calling sequence used
   to access these RAM resident routines.

   GLOBAL JTASK ;SYSTEM LINKAGE ROUTINE
   GETVEC EQU 10 ;GETVEC JTASK CODE
   MRENT EQU 1 ;MRENT JTASK CODE

   LD D,1 ;CONSOLE OUTPUT LUN
   LD A,GETVEC ;GETVEC JTASK CODE
   CALL JTASK ;CALL GETVEC

   LD A,MRENT ;MRENT JTASK CODE
   JP JTASK ;JP MRENT
   ;END OF PROGRAM
   ;SO DON'T CALL
   END ;JTASK-JUST JUMP
13-10. **ASBIN** - CONVERT ASCII DIGIT TO BINARY
- PROM RESIDENT

DESCRIPTION - Convert ASCII representation of a hex digit to binary. No error checking is done, so the binary "equivalent" of any ASCII character can be found using ASBIN.

ENTRY PARAMETERS - A - reg contains the ASCII character to be converted (8-bits).

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>00000001B</td>
</tr>
<tr>
<td>32</td>
<td>00000010B</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>39</td>
<td>00001001B</td>
</tr>
<tr>
<td>41</td>
<td>00001010B</td>
</tr>
<tr>
<td>42</td>
<td>00001011B</td>
</tr>
<tr>
<td>43</td>
<td>00001100B</td>
</tr>
<tr>
<td>44</td>
<td>00001101B</td>
</tr>
<tr>
<td>45</td>
<td>00001110B</td>
</tr>
<tr>
<td>46</td>
<td>00001111B</td>
</tr>
</tbody>
</table>

EXIT PARAMETERS - A - reg contains the corresponding binary value of the ASCII character.

CALLING SEQUENCE - CALL ASBIN

EXAMPLE - GLOBAL ASBIN
LD A,'A' ;CONVERT ASCII 'A' TO
CALL ASBIN ;BINARY
;A = 00001010B = AH
13-11. ASTCHK - ASTERISK CHECK
- RAM RESIDENT
- JTASK CODE 8

DESCRIPTION - This routine checks for asterisk (*) in an IOCS vector. If an asterisk is found in the device code, filename, extension, or user identification code, then zero flag is set. This routine is called after a CSI routine.

ENTRY PARAMETERS - IY reg points to start of an IOCS vector to be checked.

EXIT PARAMETER -
Z flag = 1 if asterisk found in string.
Z flag = 0 if no asterisk found in string.

CALLING SEQUENCE - LD A,8
CALL JTASK

EXAMPLE -
GLOBAL JTASK
LD IY,VECTOR ;IY = VECTOR ADDRESS
LD A,8 ;ASTCHK JTASK CODE
CALL JTASK
JP Z,ASTFND ;IF ASTERISK, JUMP
; NO ASTERISKS FOUND - CONTINUE
13-12. CRLF - OUTPUT CARRIAGE RETURN AND LINE FEED
- PROM RESIDENT

DESCRIPTION - Output a carriage return (ODH) and line feed (OAH).

ENTRY PARAMETERS - E - reg. designates LUN as in WRCHR (see Section 8).

EXIT PARAMETERS - A - reg is destroyed.
D - reg contains line feed (OAH).

CALLING SEQUENCE - CALL CRLF

EXAMPLE - GLOBAL CRLF
LD E,1 ;CONSOLE OUT LUN
CALL CRLF ;OUTPUT CARRIAGE RETURN
;AND LINE FEED TO
;CONSOLE
13-13. CSI - COMMAND STRING INTERPRETER
   - RAM RESIDENT
   - JTASK CODES 6 (CSIPAR) AND 7 (CSISYN)

DESCRIPTION - The Command String Interpreter is a system routine which reads command strings containing dataset specifications. CSI is used extensively by FLP-80DOS system programs (MONITOR, PIP, ASM, etc.) but is also available for use in application programs. CSI assumes that the HL register points to a command string containing datasets which is terminated by a carriage return. A dataset (See paragraph 1-21) is defined as follows:

   DEV:FILENAME.EXT[UIC]

The command string interpreter contains the following subroutines:

<table>
<thead>
<tr>
<th>NAME</th>
<th>FUNCTION</th>
</tr>
</thead>
</table>
| CSISYN | Checks the syntax of a command string contain-
|        | ing datasets.                                 |
| CSIPAR | Parses a single dataset and places dataset    |
|        | specifications in I/O vector.                 |
13-14. CSISYN - JTASK CODE 7

CALLING SEQUENCE - LD A,7
CALL JTASK

ENTRY PARAMETERS

1. HL points to the first character or a blank preceding the first character of the dataset portion of the command string. The end of the string must be terminated by a carriage return.

EXIT PARAMETERS

1. REGISTER A
   0 - Indicates Valid Dataset Specifications (no Syntax Errors). Zero flag is set.
   2 - Invalid Dataset Specifications (Syntax Error). Zero flag is cleared.

2. Other Registers Modified: None
13-15. CSIPAR - JTASK CODE 6

CALLING SEQUENCE - LD A,6
CALL JTASK

EXIT PARAMETERS
1. HL points to the first character or a blank preceding the first character of the dataset portion of the command string.
2. IY points to I/O vector.

13-14. On Exit From CSIPAR
1. REGISTER A
   0 - Indicates Dataset Found andParsed. Zero flag is set.
   1 - Dataset Not Found. End of line (carriage return) was encountered. Zero flag is cleared.
   2 - Syntax Error (Note CSIPAR does partial but not complete syntax check. For complete check call CSISYN). Zero flag is cleared.
2. REGISTER C
   Register C contains the character that terminates the dataset.

DATASET TERMINATOR C REGISTER ON EXIT
, '0'
CARRIAGE RETURN UD 'T'
10 'T'
> 'T'

NOTE: > is equivalent to TO.
3. **HL REGISTER**
   If a valid dataset is found \((A=0)\) then HL points to the next character after the dataset.

4. **I/O Vector**
   If a dataset is found, then the device, filename, extension and user number are placed in the I/O vector (See para. 9-3). The following default conditions are assumed if the dataset element is not specified.

<table>
<thead>
<tr>
<th>ELEMENT</th>
<th>DEFAULT NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>2 blanks</td>
</tr>
<tr>
<td>Unit No.</td>
<td>0</td>
</tr>
<tr>
<td>Filename</td>
<td>6 blanks</td>
</tr>
<tr>
<td>Extension</td>
<td>3 blanks</td>
</tr>
<tr>
<td>User Code</td>
<td>1</td>
</tr>
</tbody>
</table>

5. **REGISTER D'**
   1 - If user number was entered.
   0 - If user number was not entered.

6. **Other Registers Modified: A'**

   **EXAMPLE** - Upon entry to a program from the Monitor, the DE-register points to the rest of the command buffer after the program name. For example, the command:
   ```
   $MYPROG DK1:FILE1(CR)
   ```
   loads and executes the file 'MYPROG.BIN'. Upon entry to MYPROG, the DE-register points to the blank after 'MYPROG' in the command line. To syntax check and parse the dataset specification into its I/O vector, the following sequence of code may be used.

   ```
   GLOBAL JTASK
   CSIPAR EQU 6
   CSISYN EQU 7
   ```
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MYPROG</td>
<td>PUSH DE ;MOVE POINTER</td>
</tr>
<tr>
<td></td>
<td>POP HL ;TO HL</td>
</tr>
<tr>
<td></td>
<td>LD A,CSISYN ;CHECK SYNTAX</td>
</tr>
<tr>
<td></td>
<td>CALL JTASK ;OF DATASET</td>
</tr>
<tr>
<td></td>
<td>JP NZ,ERR ;IF SYNTAX ERROR, SKIP</td>
</tr>
<tr>
<td></td>
<td>LD IY,VECT ;GET VECTOR ADDRESS</td>
</tr>
<tr>
<td></td>
<td>LD A,CSIPAR ;PARSE DATASET</td>
</tr>
<tr>
<td></td>
<td>CALL JTASK ;INTO VECTOR</td>
</tr>
<tr>
<td></td>
<td>JP NZ,ERR ;IF ERROR, SKIP</td>
</tr>
<tr>
<td></td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
</tr>
</tbody>
</table>
13-16. RENTRY - DDT-80 RE-ENTRY
- PROM RESIDENT

DESCRIPTION - Entry address to DDT. This address should be jumped to, not called. DDT will print a carriage return, line feed, and a period (.) prompt. The user register map is not saved when jumping to RENTRY. DDT is then ready to accept a command.
13-17. ECHO - INPUT AND ECHO A CHARACTER
- PROM RESIDENT

DESCRIPTION - Read and write a character through the same LUN pair. Input LUN is 0, 2, or 4. Output LUN is 1, 3, or 5. Valid LUN pairs are (0,1), (2,3), (4,5).

ENTRY PARAMETERS -
E - reg designates the LUN as in RDCHR and WRCHR. Immediate return is not valid when calling ECHO.

EXIT PARAMETERS -
A - reg is destroyed
D - reg contains the character read and printed

CALLING SEQUENCE - CALL ECHO

EXAMPLE -
GLOBAL ECHO
LD E,0 ;READ AND WRITE
;CHARACTER TO
CALL ECHO ;CONSOLE
13-18. EH - SYSTEM ERROR HANDLER
- PROM RESIDENT

DESCRIPTION - Print error message in the following format:

```
***** ERROR nn (message) (dataset specification)
```

where nn is the error code in hexadecimal, the
message is obtained from a lookup table within
EH, and the dataset is the one defined by IY.

FLP-80DOS all I/O error messages (numbers 1-1FH)
are cataloged in EH. If an error code not as­
sociated with a message is input, then the output
is:

```
***** ERROR nn
```

Output is directed via the DDT console output handle­
ter (thus bypassing IOCS).

Error messages for FLP-80DOS are shown in Appendix E.

ENTRY PARAMETERS -
A - reg = error code (8 bits). If A = 1 through 1FH
then the standard message format will be output.
IY - reg = vector address containing a dataset specifica­
tion of the dataset for which the error occurred.

EXIT PARAMETERS -
All registers remain unchanged.

CALLING SEQUENCE - CALL EH
EXAMPLE -

GLOBAL EH
GLOBAL JIOCS
GLOBAL JTASK

LD IY,VECTOR ;IY = VECTOR ADDRESS
LD (IY+RQST),OPENR ;OPEN READ REQUEST
CALL JIOCS ;OPEN THE FILE
LO A,(IY+ERRC) ;GET ERROR CODE
AND A ;CHECK FOR ERRORS
JR Z,CONT-$ ;IF NONE, SKIP
CALL EH ;ELSE PRINT ERROR
LD A,1 ;MRENT CODE
JP JTASK ;RETURN TO MONITOR

CONT -----------
13-19. GETHL - GET LINE FROM THE CONSOLE DEVICE
- RAM RESIDENT
- JTASK CODE 9

DESCRIPTION - GETHL inputs a line of data from the console device into the buffer pointed to by HL. All line editing functions are active: tab, backspace, rub-out, and line delete (CNTL-U). Return is made to caller upon carriage return.

ENTRY PARAMETERS - HL-reg pair points to input buffer.
D - reg contains reprompt character for line delete function (see above). This character is displayed on the console whenever a line is deleted via CNTL-U.

EXIT PARAMETERS - Data is placed into buffer. All registers are saved.

CALLING SEQUENCE -
LD A,9
CALL JTASK

EXAMPLE -
GLOBAL JTASK
LD HL,INBUF ;INPUT BUFFER POINTER
LD D,"$" ;RE_PROMPT CHARACTER
LD A,9 ;GETHL CODE
CALL JTASK

INBUF DEFS 160 ;MAXIMUM SIZE = 160 BYTES
13-20. GETVEC - GET DEFAULT VECTOR ADDRESS
  - RAM RESIDENT
  - JTASK CODE 10

DESCRIPTION - This routine calculates the default vector address for LUN's 0-5.

ENTRY PARAMETERS -
  D-reg contains default vector number (0 through 5).

EXIT PARAMETERS -
  IY reg points to start of IOCS default vector address. Carry bit set if A-reg > 5 upon entry, otherwise carry is reset.

CALLING SEQUENCE -
  LD   A,10
  CALL  JTASK

EXAMPLE -
  GLOBAL  JTASK
  LD   D,0  ;GET VECTOR ADDRESS
  LD   A,10  ;FOR LUN 0
  CALL  JTASK
  ;IY points to default vector for LUN 0
MINDIS - DISABLE MINIMAL LISTENER
- PROM RESIDENT

DESCRIPTION - This subroutine turns off the minimal listener function to disable Console Escape (control-X) and Debugger Escape (control-C).

ENTRY PARAMETERS - None

EXIT PARAMETERS - None

CALLING SEQUENCE - CALL MINDIS
13-22. MINEN - ENABLE MINIMAL LISTENER
- PROM RESIDENT

DESCRIPTION - This subroutine turns on the Minimal Listener function to enable Console Escape (control-X) and Debugger Escape (control-C).

ENTRY PARAMETERS - None

EXIT PARAMETERS - None

CALLING SEQUENCE - CALL MINEN
13-23. MRENT - MONITOR RE-ENTRY
- RAM RESIDENT
- JTASK CODE 1

DESCRIPTION - This is the normal re-entry address to the Monitor. Program exits should return to the Monitor via a jump to this address if the system software has not been overlayed.

CALLING SEQUENCE -
LD A,1
JP JTASK
13-24. **PACC - PRINT ASCII CONTENTS OF THE ACCUMULATOR**  
- PROM RESIDENT

**DESCRIPTION** - Print the contents of the A - register in ASCII equivalent.

**ENTRY PARAMETERS -**
- **E** - reg designates LUN as for WRCHR (see Section 8). Immediate return is not valid when calling PACC.
- **A** - reg contains the binary equivalent of the 2 hexadecimal digits to be printed in ASCII.

**EXIT PARAMETERS -**
- **E** - reg used as in WRCHR.
- **A** - reg is destroyed.

**CALLING SEQUENCE - CALL PACC**

**EXAMPLE -**

```
LD A,25H ;A=25
LD E,1 ;SELECT CONSOLE LUN
CALL PACC ;PRINT THE
           ;CHARACTERS
;'25' ON CONSOLE
: DEVICE
```
13-25. PTXT - PRINT TEXT STRING
- PROM RESIDENT

DESCRIPTION - Print a text string. The string terminates with 
ETX (03H), which is not output.

ENTRY PARAMETERS -
E - reg designates LUN as in WRCHR (see Section 8). Im-
mediate return is not valid when calling PTXT.
HL - reg pair contains the beginning address where the text 
string is stored in memory. The text string must ter-
minate with ETX (03H).

EXIT PARAMETERS -
A - reg is destroyed
D - reg contains ETX (03H)
HL - reg pair contains address in memory where the ETX ter-
minator is stored.

CALLING SEQUENCE - CALL PTXT

EXAMPLE - LD HL,MSG ;GET MESSAGE ADDRESS
             LD E,1 ;SELECT CONSOLE LUN
             CALL PTXT ;PRINT MESSAGE
             .
             .
             .
             MSG DEFM 'THIS IS A MESSAGE'
             DEFB 3 ;ETX
13-26. PVECT - PRINT VECTOR DATASET
   - PROM RESIDENT

DESCRIPTION - This routine prints out a dataset specification
  from an IOCS vector on the device specified by the
  console output LUN (LUN1).

ENTRY PARAMETERS -
  IY reg points to start of IOCS vector.

EXIT PARAMETERS - None.

CALLING SEQUENCE - CALL PVECT

EXAMPLE - LD IY,VECTOR ;IY POINTS TO
  CALL PVECT ;START OF VECTOR
13-27. REBOOT - SYSTEM REBOOT SEQUENCE
- PROM RESIDENT

DESCRIPTION - Reboot System. This is the beginning of the initialization sequence after the terminal baud rate is determined. The system software is booted in RAM from OS.BIN[255] and the Monitor prompt ($) is issued to the console.

This location should be jumped to, not called. It is the entry point for Monitor Escape (CNTL-X).

CALLING SEQUENCE - JP REBOOT
13-28. SCAN - INTERACTIVE SCAN
- PROM RESIDENT

DESCRIPTION - This routine is the interactive scan routine used in DDT. It can be called to return up to 3 parameters from the user terminal in the interactive mode described for DDT. The hexadecimal operands are converted from ASCII into 16-bit binary. Up to 3 operands may be entered, separated by commas or blanks. If more than three operands are entered, then the third operand is updated to the last one entered.

ENTRY PARAMETERS - None.

EXIT PARAMETERS -
OPFLG = FF1AH - number of a operands entered, 0, 1, 2, or 3.
OPR1 = FF14H - first operand (16 bits).
OPR2 = FF16H - second operand (16 bits).
OPR3 = FF18H - third operand (16 bits).
NXTCHR = FF1B - last character processed by the SCAN routine.

CALLING SEQUENCE - CALL SCAN
13-29. SEARCH - FIND DIRECTORY ENTRY OF A FILE
   - RAM RESIDENT
   - JTASK CODE 11

DESCRIPTION - This routine finds the directory entry for the file specified in the IOCS vector.

ENTRY PARAMETERS -
   IY reg points to the file vector.

EXIT PARAMETERS -
   DE reg has the directory address
   C reg has the unit number
   The Z flag is set if found
   The NZ flag is set if not found.

CALLING SEQUENCE - LD A,11
   CALL JTASK

EXAMPLE - GLOBAL JTASK
   LD IY, VINP ; POINT TO VECTOR
   LD A,11
   CALL JTASK
13-30. SPACE - OUTPUT A SPACE
  - PROM RESIDENT

DESCRIPTION - Output a blank (20H).

ENTRY PARAMETERS -
  E - reg designates LUN as in WRCHR.

EXIT PARAMETERS -
  A - reg is destroyed.
  B - reg contains blank (20H).

CALLING SEQUENCE - CALL SPACE

EXAMPLE - LD E,1 ;CONSOLE LUN
           CALL SPACE ;OUTPUT A SPACE
13-31. SRCHR, SRCHU - SEARCH MNEMONIC TABLES
- PROM RESIDENT

DESCRIPTION - Search resident mnemonic table (SRCHR) or search user mnemonic table (SRCHU) for a match. The re­sident mnemonic table contains the user registers and their save locations accessed by DDT. This table exists in PROM. The user mnemonic table con­tains the device handlers and their addresses. The user mnemonic table in part of the SYSGEN FILE (RAM resi­dent).

ENTRY PARAMETERS -
HL - reg pair points to 2 character mnemonic to be searched for. The first character goes into L, the second goes into H.

EXIT PARAMETERS -
Zero flag reset if no match.
Zero flag set if match found and HL reg pair equals 16 bit address associated with the mnemonic.

CALLING SEQUENCE - CALL SRCHR
CALL SRCHU

EXAMPLE - LD H,'P'
LD L,'L'
CALL SRCHU
;GET ADDRESS OF
;LP = HANDLER
;HL = ADDRESS OF
;HANDLER ON EXIT
14-1. INTRODUCTION

14-2. FLP-80DOS directly supports batch mode operation in configurations with more than 16K of RAM. In batch mode operation, all commands are entered via a batch input device. The batch input device is specified by the dataset assigned to logical unit 0 and may be any input device such as a card reader, paper tape reader, or a disk file. All responses by the system to the batch input device may be directed to any other output dataset. In batch mode operation, all input from an input dataset corresponds exactly to what the user would normally type in via the terminal keyboard. There is no difference between commands entered via the console or in batch mode. Batch mode operation can be applied to all programs in FLP-80DOS, except DDT, the debugger. Insert mode in the Editor also cannot be activated in batch mode from a disk file. User programs which interface to the console device via IOCS may also be used directly in batch mode operation.

14-3. PRINCIPLES OF OPERATION

14-4. The key to batch mode operation in FLP-80DOS is the system's ability to reassign the console channels (Logical Unit Numbers 0 and 1). LUN 0 is used for all console input. LUN 1 is used for all console output. These LUN's may be reassigned to any other dataset via the Monitor ASSIGN command. When the Monitor makes an assignment of a dataset to LUN 0 or 1, the Monitor automatically closes the currently assigned dataset. Then it opens the new dataset. This operation is different from
the other LUN assignments in which the Monitor does not automatically open the new dataset.

14-5. When an assignment is made to LUN 0 (Console In), the assigned dataset referred to as the batch input device is opened and input is automatically started by the Monitor. Commands input from the dataset are called the Batch Command Sequence (BCS), and they control the system operation. Reassignment back to the original user terminal is then the responsibility of the batch command sequence from the dataset.

14-6. When an assignment is made to LUN 1 (Console Out), the new dataset is opened and all output which would normally appear on the user terminal is directed to the new dataset. Such an assignment, if it is to be done, should be done by the first statement of a batch command sequence (BCS).

14-7. BATCH COMMAND SEQUENCE SYNTAX

14-8. The syntax of a batch command sequence (BCS) is exactly like the user input from the terminal. In this manual, all user input is underlined. Each line of input in a BCS is terminated with a carriage return. A BCS can be built on a disk file by using the FLP-80DOS Text Editor. If a card reader is interfaced to the system, the BCS can be on cards.

14-9. If the console output is to be directed to a non-console dataset, the assignment should be the first record of the BCS:

ASSIGN 1,LP:

14-10. The last record of the BCS should be assignment of LUN's 0 and 1 back to the original console datasets:

ASSIGN 1,TT:
ASSIGN 0,TK:
14-11. During Batch Mode Operation, no initialization of the disk units is performed by the system. This means that diskettes cannot be switched during batch mode. This restriction is necessary because during initialization the disk handler's active file table is cleared. This action would clear the BCS disk file, and further BCS records could not be accessed.

14-12. EXAMPLE 1. Build a BCS on a disk file called "BATCH" which accesses PIP and prints out the directory and status of each disk unit on the line printer. The following commands are entered from the user terminal (interactive mode) to build the BATCH file:

    $EDIT BATCH(CR)
    FLP-80DOS EDITOR V2.0
    -- > NEW FILE
    -- > INSERT MODE
    0001 < PIP(CR)
    0002 < D TO LP:(CR)
    0003 < S TO LP:(CR)
    0004 < D DK1: TO LP:(CR)
    0005 < S DK1: TO LP:(CR)
    0006 < Q(CR)
    0007 < ASSIGN O,TK:(CR)
    0008 < (CR)
    >Q(CR)

To execute the batch file, the following command should be entered:

    $ASSIGN O, BATCH(CR)

The BATCH file will be executed, command by command. The total command sequence will be printed on the terminal. The directory and status listings will be directed to the line printer.
EXAMPLE 2. Assemble two files in batch mode, directing all printable output to the line printer. The BCS to be built up as a file (named 'BSC1') is:

```
ASSIGN 1, LP:
ASM FILE 1 TO LP:
S
  -this is the "option" input to the Assembler.
ASM FILE2 TO LP:
S
ASSIGN 1,TT:
ASSIGN 0,TK:
The BCS is executed by entering the following Monitor command:
$ASSIGN 0,BSC1(CR)
```
15-1. INTRODUCTION

15-2. After reset or power up the system boot routine resident in PROM loads the operating system from the file OS.BIN into memory and starts execution at its beginning address. The system generation or SYSGEN procedure can be used to link operating system object modules together to generate a modified OS.BIN if desired. The following parameters are defined during SYSGEN.
   1. Operating System starting address.
   2. Number of disk drives (1-4)
   3. I/O drivers linked into system (E.G., LP,CR and etc.)
   4. Default I/O vectors for logical units 2-5

15-3. The standard system as shipped from the factory contains 32K of RAM (see Figure 15-1) and contains the I/O drivers TK:, TT: and CP:. The SYSGEN procedure which may be used to modify the operating system is performed as outlined below and is also illustrated in Figure 15-1. All system object files are on the MOSTEK supplied system disk.

15-4. SYSTEM GENERATION PROCEDURE (SYSGEN)
   STEP 1. Place a Version 2.0 system diskette containing the operating system object files in disk unit DKO. Depress reset and the carriage return key to boot up the system. If a change in the number of disk drives to be supported needs to be made, follow the instructions in paragraph 15-15. If the user wishes to change the system device
table for purposes of adding a mnemonic for a new I/O driver, he should follow the instructions starting at paragraph 15-10. If modifications to the default logical units are required see paragraph 15-13.

STEP 2. Use the LINKER to create a test operating system file. The Linker A option is used to specify the operating system beginning address.

EXAMPLE:

$LINK MONITOR,IOCS,SYSGEN,CSI,TASK,TK,TT,LPC,DKUNIT,DKTAB,DK, SYSLNK TO TEST.BIN(CR)

OPTIONS? A_U_C(CR)
ENTER STARTING LINK ADDRESS >5A00(CR)

NOTES

1). The user may arbitrarily choose the starting address. The LINKER generates a load map listing the beginning and ending addresses of each module (see Figure 15-1). Step 2 may be repeated a second time in order to position the operating system at the top of the user's RAM space, thereby maximizing the amount of RAM available for the user.

2). When entering the Linker command from the terminal the command line may exceed the maximum terminal line length (usually 80 characters). If this occurs, the terminal output driver will automatically issue a CR and LF to enable continuation of the command on the next line. Since a carriage return input from the
keyboard is interpreted by the Linker to be the terminator of the command string, the user should not enter a carriage return until the entire Linker command has been entered. Maximum command line length is 160 characters.

3). The terminal I/O drivers TK and TT must always be linked into the system. Additional I/O drivers (E., G., LPC) may also be linked into the system.

4). The order in which the system modules are linked must be maintained as shown in the table in paragraph 15-5. The Monitor must be the first module and SYSLINK must be the last module. Additional I/O drivers should be added after the TT driver.

5). The Linker C option may be used to save a copy of the new operating system load map (See figure 15-1) and the global cross reference table for future reference. The Cross reference output defaults to the file TEST.CRS unless another output device is specified (See LINKER Section 6).

6). The Linker U option is used to list all of the I/O drivers which are not linked into the new operating system but are in the System Device Table (See Paragraph 15-7). The linker load map specifies all the I/O drivers which have been linked into the system.

STEP 3. Place the diskette on which the new operating system is to be copied into the disk unit DK1. Enter PIP and copy the new operating system to OS.BIN 255 on DK1 as shown below. Other system programs such as PIP, LINK, EDIT and ASM may also be copied to the diskette DK1 should they
they already not be on that diskette.

$\text{PIP(CR)}$

\texttt{#C TEST.BIN TO DK1:OS.BIN 255 (CR)}

\texttt{#C PIP.BIN,LINK.BIN,EDIT.BIN,ASM.BIN TO DK1:(CR)}

\textbf{NOTE:} The user may also copy the file TEST.CRS which contains the operating system load map to DK1:OS.CRS[255] which may be listed using PIP.

\textbf{STEP 4.} Move the diskette with the modified OS.BIN 255 operating system from disk unit DK1 to DK0. Depress reset and carriage return on the terminal and verify that the modified operating system responds with sign on message:

\begin{center}
MOSTEK FLP-80DOS V2.0
\end{center}

The sign on message should be followed by a $\$ \$ indicating that the user is in the monitor environment and that the new operating system has been created successfully. The environments EDIT,ASM,LINK and PIP may be entered next to verify that all system programs are operational.

This completes the System Generation Procedure.

\textbf{15-5. OPERATING SYSTEM MODULES}

The following is a list of the system object modules in the order in which they must be linked into the operating system during the SYSGEN procedure. (See STEP 2).

\begin{tabular}{|c|c|}
\hline
MODULE & DESCRIPTION \\
\hline
1. MONITOR & System Monitor \\
2. IOCS & I/O Control system \\
\hline
\end{tabular}
3. SYSGEN  See description on next page
4. CSI      Command String Interpreter
5. TASK     Task selector for system subroutines
6. TK       Terminal Input Driver
7. TT       Terminal Output Driver
8. I/O DRIVERS  See description below
9. DKUNIT   Specifies Number of Disk Units
10. DKTAB   Table or buffer space for disk drives
11. DK      Disk handler
12. SYSLNK   Linkages to system software in PROM (E000-EFFF)

15-6. STANDARD I/O DRIVERS

15-7. The user may link up to a maximum of 12 I/O drivers into his system at one time using the SYSGEN procedure. The following is a list of the standard I/O devices which are in the system device table (See Paragraph 15-9) and are also supplied with the system diskette.

<table>
<thead>
<tr>
<th>DRIVER</th>
<th>FILE NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>TT:</td>
<td>TT</td>
<td>Terminal Output Device</td>
</tr>
<tr>
<td>TK:</td>
<td>TK</td>
<td>Terminal Keyboard</td>
</tr>
<tr>
<td>LP:</td>
<td>LPD</td>
<td>Data Products Line Printer</td>
</tr>
<tr>
<td>CP:</td>
<td>LPC</td>
<td>Centronics Line Printer</td>
</tr>
<tr>
<td>TR:</td>
<td>TR</td>
<td>Teletype tape reader</td>
</tr>
<tr>
<td>CR:</td>
<td>CR</td>
<td>Card Reader</td>
</tr>
<tr>
<td>PR:</td>
<td>PP</td>
<td>Paper tape reader</td>
</tr>
<tr>
<td>PP:</td>
<td>PR</td>
<td>paper tape punch</td>
</tr>
<tr>
<td>TI:</td>
<td>STI</td>
<td>Silent 700 Cassette Tape Input</td>
</tr>
<tr>
<td>TO:</td>
<td>STO</td>
<td>Silent 700 Cassette Tape Output</td>
</tr>
</tbody>
</table>

15-8. SYSTEM DEVICE TABLE. The system device table is in the
operating system module SYSGEN.OBJ. It contains a mnemonic and a GLOBAL reference for each I/O device. The devices listed in paragraph 15-7 represent the standard System Device Table supplied for FLP-80DOS.

15-9. After system reset or power up the Monitor creates a RAM mnemonic table in scratchpad RAM starting at OFF2FH (See Appendix C). The RAM mnemonic table contains the mnemonics for the I/O devices which are supported by the operating system at execution time. Devices which are not in the RAM mnemonic table will generate the error message UNSUPPORTED DEVICE if an I/O transaction is attempted. In order for a device to be placed in the RAM mnemonic table during the Monitor initialization sequence the following conditions must be met.

1. The mnemonic for the device is in the System Device Table in the program module SYSGEN.OBJ which is linked into the operating system in OS.BIN 255.
2. The I/O driver itself is linked into the operating system (See STEP 2 in SYSGEN procedure paragraph 15-4).

15-10. ADDING NEW I/O DRIVERS

15-11. A new or modified I/O driver having a mnemonic which is in the system device table (e.g. LP:) may be linked directly into the operating system as outlined in STEP 2 of the SYSGEN procedure. However, if the mnemonic of the new driver is not in the System Device Table (See paragraph 15-7) the table can be modified by the user. Changes to the table are made by editing and assembling the file SYSGEN.SRC. After the System Device Table is modified the user should then link the new I/O driver module into the operating system (See STEP 2 of SYSGEN procedure).
15-12. CHANGING THE DEFAULT LOGICAL UNITS

15-13. The default dataset definitions for logical units 2-5 may be changed by the user with the SYSGEN procedure. Changes to the default vectors are made by editing and assembling the file SYSGEN.SRC and then linking the new SYSGEN module into the operating system (see STEP 2 of SYSGEN procedure, paragraph 15-4).

15-14. CHANGING THE NUMBER OF DISK UNITS IN THE SYSTEM

15-15. The variable NMUNIT in the files DKUNIT and DKTAB specifies the number of disk units in the system. NMUNIT is set to 2 for the standard Mostek system. If the user wishes to add additional disk drives (up to 4), NMUNIT should be modified in DKUNIT.SRC and DKTAB.SRC and these modules should be reassembled prior to performing the SYSGEN procedure.

15-16. SYSTEM GENERATION OF A 64K OPERATING SYSTEM

15-17. The hardware modifications to produce a system with 60K total RAM are discussed in the system hardware operations manual. For this configuration, the FLP-80DOS may be split to place most of the operating system below the PROM's (which start at 56K) and part of the operating system above the PROM's (which end at 60K). Here is the procedure to follow.

15-18. Create a module called 'SPACE' with the following instructions on it. Assemble the module.

```
PSECT ABS
ORG OEFFFH
NOP
END
```
$LINK MONITO,IOCS,SYSGEN,CSI,TASK,TK,TT,LPC,DKUNIT,DKTAB,DK,
SYSLNK TO TEST
OPTIONS? C U A
ENTER STARTING LINK ADDRESS > 5B35
DKO:MONITO.OBJ[1]
DKO:IOCS .OBJ[1]
DKO:SYSGEN.OBJ[1]
DKO:CSI .OBJ[1]
DKO:TASK .OBJ[1]
DKO:TK .OBJ[1]
DKO:TT .OBJ[1]
DKO:LPC .OBJ[1]
DKO:DKUNIT.OBJ[1]
DKO:DKTAB .OBJ[1]
DKO:DK .OBJ[1]
DKO:SYSLNK.OBJ[1]
CR LP PP PP TI
TO TR
UNDEFINED SYMBOLS 07
PASS 2
DKO:MONITO.OBJ[1] REL BEG ADDR 5A8A END ADDR 6129
DKO:IOCS .OBJ[1] REL BEG ADDR 612A END ADDR 6AC0
DKO:SYSGEN.OBJ[1] REL BEG ADDR 6AC1 END ADDR 6B62
DKO:CSI .OBJ[1] REL BEG ADDR 6B63 END ADDR 6DB4
DKO:TASK .OBJ[1] REL BEG ADDR 6DB5 END ADDR 6E7B
DKO:TK .OBJ[1] REL BEG ADDR 6E7C END ADDR 6EFE
DKO:TT .OBJ[1] REL BEG ADDR 6EFF END ADDR 6FB4
DKO:LPC .OBJ[1] REL BEG ADDR 6FB5 END ADDR 7070
DKO:DKUNIT.OBJ[1] ABS BEG ADDR 7071 END ADDR 7072
DKO:DKTAB .OBJ[1] REL BEG ADDR 7073 END ADDR 76BD
DKO:DK .OBJ[1] ABS BEG ADDR 76BE END ADDR 7FFF
DKO:SYSLNK.OBJ[1] ABS BEG ADDR 8000 END ADDR 8000

NOTE: The above example is the Linker Load Map resulting from the SYSGEN procedure on a system having 32K of RAM
(TOR=7FFF). Since the module SYSLNK is only used by the Linker to resolve global addresses the end address of the module DK is allowed to be the top location of addressable RAM. The end address of DK must not exceed that of the top of RAM. Since additional I/O drivers may be added and module sizes might change in the future, the starting link address should be adjusted during each system generation to correctly position the end address of DK.
15-19. Link the modules of the operating system together with 'SPACE.OBJ'. The constraints are as follows: 1) the lower part of the OS must have an end address below DFE0H; 2) the upper part of the OS will start above 'SPACE' (start address = OF000H), and it must have an end address below FFOOH.

15-20. Figure 15-2 shows an example of how to link a 64K OS. Note that both the centronics and Data Products line printer handlers (LPC and LPD) were linked in this example. The lower part of OS ends with module 'DKTAB' whose end address is below DFE0H. The upper part of OS starts with 'CSI' and ends with 'SYSLNK' whose end address is below FFOOH.
FIGURE 15-2 LINKING A 64K OPERATING SYSTEM

LINK
MONITOR, IOCS, SYSGEN, TASK, DUNIT, DTABLE, SPACE, CSI, TK, TT, LPC, LPD, DK,
SYSLNK TO TEST

CUIA
C7F7
Y
Y

LOAD MAP

<table>
<thead>
<tr>
<th>DKO:MONITOR.OBJ[1]</th>
<th>REL</th>
<th>BEG ADDR C7F7</th>
<th>END ADDR CE96</th>
</tr>
</thead>
<tbody>
<tr>
<td>DKO:IOCS.OBJ[1]</td>
<td>REL</td>
<td>BEG ADDR CE97</td>
<td>END ADDR D82D</td>
</tr>
<tr>
<td>DKO:SYSGEN.OBJ[1]</td>
<td>REL</td>
<td>BEG ADDR D82E</td>
<td>END ADDR D8CF1</td>
</tr>
<tr>
<td>DKO:TASK.OBJ[1]</td>
<td>REL</td>
<td>BEG ADDR D8DO</td>
<td>END ADDR D991</td>
</tr>
<tr>
<td>DKO:DUNIT.OBJ[1]</td>
<td>ABS</td>
<td>BEG ADDR D992</td>
<td>END ADDR D992</td>
</tr>
<tr>
<td>DKO:DTABLE.OBJ[1]</td>
<td>REL</td>
<td>BEG ADDR DFDE</td>
<td>END ADDR DFDE</td>
</tr>
<tr>
<td>DKO:SPACE.OBJ[1]</td>
<td>ABS</td>
<td>BEG ADDR EFFF</td>
<td>END ADDR EFFF</td>
</tr>
<tr>
<td>DKO:CSI.OBJ[1]</td>
<td>REL</td>
<td>BEG ADDR F000</td>
<td>END ADDR F251</td>
</tr>
<tr>
<td>DKO:TK.OBJ[1]</td>
<td>REL</td>
<td>BEG ADDR F252</td>
<td>END ADDR F2D4</td>
</tr>
<tr>
<td>DKO:TT.OBJ[1]</td>
<td>REL</td>
<td>BEG ADDR F2D5</td>
<td>END ADDR F377</td>
</tr>
<tr>
<td>DKO:LPC.OBJ[1]</td>
<td>REL</td>
<td>BEG ADDR F378</td>
<td>END ADDR F433</td>
</tr>
<tr>
<td>DKO:LPD.OBJ[1]</td>
<td>REL</td>
<td>BEG ADDR F434</td>
<td>END ADDR F4E7</td>
</tr>
<tr>
<td>DKO:DK.OBJ[1]</td>
<td>REL</td>
<td>BEG ADDR F4E8</td>
<td>END ADDR FDA3</td>
</tr>
<tr>
<td>DKO:SYSLNK.OBJ[1]</td>
<td>ABS</td>
<td>BEG ADDR FDA4</td>
<td>END ADDR FDA4</td>
</tr>
</tbody>
</table>
IPCODE Z80 OPCODE LISTING

LISTING

MOSTEK FLP-80 ASSEMBLER V2.0 PAGE 0001

ADDR OBJECT ST # SOURCE STATEMENT

DATASET = DKO:OPCODE.

0002 ; PSEUDO OPS
0003 ;
0004 NAME OPCODES

>0000
0005 ORG 0
0006 PSECT REL
0007 ;
0000 AA
0008 DEFB OAAH

>0001
0009 L2 DEFL S

>55AA
0010 L2 DEFL 55AAH

0001 41424344
0011 DEFM 'ABCD'

>0005
0012 NN DEFS 2

0007 BBAA
0013 DEFW 0AABBBH

>AAAB
0014 L1 EQU 0AABBBH

>0005
0015 IND EQU 5

>0020
0016 N EQU 20H

>0030
0017 DIS EQU 30H

0018 GLOBAL NN
0019 IF 0
0020 ; SHOULD NOT BE ASSEMBLED
0021 LD A,B
0022 ENDFI
0023 IF 1
0024 ; SHOULD BE ASSEMBLED

'0009 78
0025 LD A,B
0026 ENDFI
0027 ; TURN LISTING OFF
0032 ; LISTING SHOULD BE ON
0033 ;
0034 ;
0035 ;
0036 ; Z80 OPCODES
0037 ;

'000B 8E
0038 ADD A,(HL)

'000C DD8E05
0039 ADD A,(IX+IND)

'000F FD8E05
0040 ADD A,(IY+IND)

'0012 8F
0041 ADD A,A

'0013 88
0042 ADD A,B

'0014 89
0043 ADD A,C

'0015 8A
0044 ADD A,D

'0016 8B
0045 ADD A,E

'0017 8C
0046 ADD A,H

'0018 8D
0047 ADD A,L

'0019 CE20
0048 ADD A,N

'001B ED4A
0049 ADD HL,BC

'001D ED5A
0050 ADD HL,DE

'001F ED6A
0051 ADD HL,HL

'0021 ED7A
0052 ADD HL,SP

0053 ;

'0023 86
0054 ADD A,(HL)

'0024 DD8605
0055 ADD A,(IX+IND)

'0027 FD8605
0056 ADD A,(IY+IND)

'002A 87
0057 ADD A,A

'002B 80
0058 ADD A,B

'002C 81
0059 ADD A,C

'002D 82
0060 ADD A,D

'002E 83
0061 ADD A,E

'002F 84
0062 ADD A,H

'0030 85
0063 ADD A,L
<table>
<thead>
<tr>
<th>ADDR</th>
<th>OBJECT</th>
<th>ST #</th>
<th>SOURCE</th>
<th>STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>'0031</td>
<td>C620</td>
<td>0064</td>
<td>ADD</td>
<td>A,N</td>
</tr>
<tr>
<td>'0033</td>
<td>09</td>
<td>0065</td>
<td>ADD</td>
<td>HL,BC</td>
</tr>
<tr>
<td>'0034</td>
<td>19</td>
<td>0066</td>
<td>ADD</td>
<td>HL,DE</td>
</tr>
<tr>
<td>'0035</td>
<td>29</td>
<td>0067</td>
<td>ADD</td>
<td>HL,HL</td>
</tr>
<tr>
<td>'0036</td>
<td>39</td>
<td>0068</td>
<td>ADD</td>
<td>HL,SP</td>
</tr>
<tr>
<td>'0037</td>
<td>DD09</td>
<td>0069</td>
<td>ADD</td>
<td>IX,BC</td>
</tr>
<tr>
<td>'0039</td>
<td>DD19</td>
<td>0070</td>
<td>ADD</td>
<td>IX,DE</td>
</tr>
<tr>
<td>'003B</td>
<td>DD29</td>
<td>0071</td>
<td>ADD</td>
<td>IX,IX</td>
</tr>
<tr>
<td>'003D</td>
<td>DD39</td>
<td>0072</td>
<td>ADD</td>
<td>IX,SP</td>
</tr>
<tr>
<td>'003F</td>
<td>FD09</td>
<td>0073</td>
<td>ADD</td>
<td>IY,BC</td>
</tr>
<tr>
<td>'0041</td>
<td>FD19</td>
<td>0074</td>
<td>ADD</td>
<td>IY,DE</td>
</tr>
<tr>
<td>'0043</td>
<td>FD29</td>
<td>0075</td>
<td>ADD</td>
<td>IY,IY</td>
</tr>
<tr>
<td>'0045</td>
<td>FD39</td>
<td>0076</td>
<td>ADD</td>
<td>IY,SP</td>
</tr>
<tr>
<td>'0047</td>
<td>A6</td>
<td>0078</td>
<td>AND</td>
<td>(HL)</td>
</tr>
<tr>
<td>'0048</td>
<td>DDA605</td>
<td>0079</td>
<td>AND</td>
<td>(IX+IND)</td>
</tr>
<tr>
<td>'004B</td>
<td>FDA605</td>
<td>0080</td>
<td>AND</td>
<td>(IY+IND)</td>
</tr>
<tr>
<td>'004E</td>
<td>A7</td>
<td>0081</td>
<td>AND</td>
<td>A</td>
</tr>
<tr>
<td>'004F</td>
<td>A0</td>
<td>0082</td>
<td>AND</td>
<td>B</td>
</tr>
<tr>
<td>'0050</td>
<td>A1</td>
<td>0083</td>
<td>AND</td>
<td>C</td>
</tr>
<tr>
<td>'0051</td>
<td>A2</td>
<td>0084</td>
<td>AND</td>
<td>D</td>
</tr>
<tr>
<td>'0052</td>
<td>A3</td>
<td>0085</td>
<td>AND</td>
<td>E</td>
</tr>
<tr>
<td>'0053</td>
<td>A4</td>
<td>0086</td>
<td>AND</td>
<td>H</td>
</tr>
<tr>
<td>'0054</td>
<td>A5</td>
<td>0087</td>
<td>AND</td>
<td>L</td>
</tr>
<tr>
<td>'0055</td>
<td>E620</td>
<td>0088</td>
<td>AND</td>
<td>N</td>
</tr>
<tr>
<td>'0057</td>
<td>CB46</td>
<td>0090</td>
<td>BIT</td>
<td>0,(HL)</td>
</tr>
<tr>
<td>'0059</td>
<td>DDCB0546</td>
<td>0091</td>
<td>BIT</td>
<td>0,(IX+IND)</td>
</tr>
<tr>
<td>'005D</td>
<td>FDCB0546</td>
<td>0092</td>
<td>BIT</td>
<td>0,(IY+IND)</td>
</tr>
<tr>
<td>'0061</td>
<td>CB47</td>
<td>0093</td>
<td>BIT</td>
<td>0,A</td>
</tr>
<tr>
<td>'0063</td>
<td>CB40</td>
<td>0094</td>
<td>BIT</td>
<td>0,B</td>
</tr>
<tr>
<td>'0065</td>
<td>CB41</td>
<td>0095</td>
<td>BIT</td>
<td>0,C</td>
</tr>
<tr>
<td>'0067</td>
<td>CB42</td>
<td>0096</td>
<td>BIT</td>
<td>0,D</td>
</tr>
<tr>
<td>'0069</td>
<td>CB43</td>
<td>0097</td>
<td>BIT</td>
<td>0,E</td>
</tr>
<tr>
<td>'006B</td>
<td>CB44</td>
<td>0098</td>
<td>BIT</td>
<td>0,H</td>
</tr>
<tr>
<td>'006D</td>
<td>CB45</td>
<td>0099</td>
<td>BIT</td>
<td>0,L</td>
</tr>
<tr>
<td>'006F</td>
<td>CB46</td>
<td>0101</td>
<td>BIT</td>
<td>1,(HL)</td>
</tr>
<tr>
<td>'0071</td>
<td>DDCB054E</td>
<td>0102</td>
<td>BIT</td>
<td>1,(IX+IND)</td>
</tr>
<tr>
<td>'0075</td>
<td>FDCB054E</td>
<td>0103</td>
<td>BIT</td>
<td>1,(IY+IND)</td>
</tr>
<tr>
<td>'0079</td>
<td>CB4F</td>
<td>0104</td>
<td>BIT</td>
<td>1,A</td>
</tr>
<tr>
<td>'007B</td>
<td>CB48</td>
<td>0105</td>
<td>BIT</td>
<td>1,B</td>
</tr>
<tr>
<td>'007D</td>
<td>CB49</td>
<td>0106</td>
<td>BIT</td>
<td>1,C</td>
</tr>
<tr>
<td>'007F</td>
<td>CB4A</td>
<td>0107</td>
<td>BIT</td>
<td>1,D</td>
</tr>
<tr>
<td>'0081</td>
<td>CB4B</td>
<td>0108</td>
<td>BIT</td>
<td>1,E</td>
</tr>
<tr>
<td>'0083</td>
<td>CB4C</td>
<td>0109</td>
<td>BIT</td>
<td>1,H</td>
</tr>
<tr>
<td>'0085</td>
<td>CB4D</td>
<td>0110</td>
<td>BIT</td>
<td>1,L</td>
</tr>
<tr>
<td>'0087</td>
<td>CB56</td>
<td>0112</td>
<td>BIT</td>
<td>2,(HL)</td>
</tr>
<tr>
<td>'0089</td>
<td>DDCB0556</td>
<td>0113</td>
<td>BIT</td>
<td>2,(IX+IND)</td>
</tr>
<tr>
<td>'008D</td>
<td>FDCB0556</td>
<td>0114</td>
<td>BIT</td>
<td>2,(IY+IND)</td>
</tr>
<tr>
<td>'0091</td>
<td>CB57</td>
<td>0115</td>
<td>BIT</td>
<td>2,A</td>
</tr>
<tr>
<td>'0093</td>
<td>CB50</td>
<td>0116</td>
<td>BIT</td>
<td>2,B</td>
</tr>
<tr>
<td>'0095</td>
<td>CB51</td>
<td>0117</td>
<td>BIT</td>
<td>2,C</td>
</tr>
<tr>
<td>'0097</td>
<td>CB52</td>
<td>0118</td>
<td>BIT</td>
<td>2,D</td>
</tr>
<tr>
<td>'0099</td>
<td>CB53</td>
<td>0119</td>
<td>BIT</td>
<td>2,E</td>
</tr>
<tr>
<td>'009B</td>
<td>CB54</td>
<td>0120</td>
<td>BIT</td>
<td>2,H</td>
</tr>
<tr>
<td>'009D</td>
<td>CB55</td>
<td>0121</td>
<td>BIT</td>
<td>2,L</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>09F</td>
<td>CB5E</td>
<td>0123</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0A1</td>
<td>DDCB055E</td>
<td>0124</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0A5</td>
<td>FDCB055E</td>
<td>0125</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0A9</td>
<td>CB5F</td>
<td>0126</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0AB</td>
<td>CB58</td>
<td>0127</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0AD</td>
<td>CB59</td>
<td>0128</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0AF</td>
<td>CB5A</td>
<td>0129</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0B1</td>
<td>CB5B</td>
<td>0130</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0B3</td>
<td>CB5C</td>
<td>0131</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0B5</td>
<td>CB5D</td>
<td>0132</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0B7</td>
<td>CB66</td>
<td>0134</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0B9</td>
<td>DDCB0566</td>
<td>0135</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0BD</td>
<td>FDCB0566</td>
<td>0136</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0C1</td>
<td>CB67</td>
<td>0137</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0C3</td>
<td>CB60</td>
<td>0138</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0C5</td>
<td>CB61</td>
<td>0139</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0C7</td>
<td>CB62</td>
<td>0140</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0C9</td>
<td>CB63</td>
<td>0141</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0CB</td>
<td>CB64</td>
<td>0142</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0CD</td>
<td>CB65</td>
<td>0143</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0CF</td>
<td>CB6E</td>
<td>0145</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0D1</td>
<td>DDCB056E</td>
<td>0146</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0D5</td>
<td>FDCB056E</td>
<td>0147</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0D9</td>
<td>CB6F</td>
<td>0148</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0DB</td>
<td>CB68</td>
<td>0149</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0DD</td>
<td>CB69</td>
<td>0150</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0DF</td>
<td>CB6A</td>
<td>0151</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0E1</td>
<td>CB6B</td>
<td>0152</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0E3</td>
<td>CB6C</td>
<td>0153</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0E5</td>
<td>CB6D</td>
<td>0154</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0E7</td>
<td>CB76</td>
<td>0156</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0E9</td>
<td>DDCB0576</td>
<td>0157</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0ED</td>
<td>FDCB0576</td>
<td>0158</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0F1</td>
<td>CB77</td>
<td>0159</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0F3</td>
<td>CB70</td>
<td>0160</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0F5</td>
<td>CB71</td>
<td>0161</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0F7</td>
<td>CB72</td>
<td>0162</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0F9</td>
<td>CB73</td>
<td>0163</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0FB</td>
<td>CB74</td>
<td>0164</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0FD</td>
<td>CB75</td>
<td>0165</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>0FF</td>
<td>CB7E</td>
<td>0167</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>101</td>
<td>DDCB057E</td>
<td>0168</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>105</td>
<td>FDCB057E</td>
<td>0169</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>109</td>
<td>CB7F</td>
<td>0170</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>10B</td>
<td>CB78</td>
<td>0171</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>10D</td>
<td>CB79</td>
<td>0172</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>10F</td>
<td>CB7A</td>
<td>0173</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>111</td>
<td>CB7B</td>
<td>0174</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>113</td>
<td>CB7C</td>
<td>0175</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>115</td>
<td>CB7D</td>
<td>0176</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>117</td>
<td>DC0500'</td>
<td>0178</td>
</tr>
<tr>
<td>0122</td>
<td>;</td>
<td>11A</td>
<td>FC0500'</td>
<td>0179</td>
</tr>
<tr>
<td>ADDR</td>
<td>OBJECT</td>
<td>ST #</td>
<td>SOURCE STATEMENT</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>--------------</td>
<td>------</td>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td>'011D</td>
<td>D40500'</td>
<td>0180</td>
<td>CALL NC,NN</td>
<td></td>
</tr>
<tr>
<td>'0120</td>
<td>CD0500</td>
<td>0181</td>
<td>CALL NN</td>
<td></td>
</tr>
<tr>
<td>'0123</td>
<td>C40500</td>
<td>0182</td>
<td>CALL NZ,NN</td>
<td></td>
</tr>
<tr>
<td>'0126</td>
<td>F40500</td>
<td>0183</td>
<td>CALL P,NN</td>
<td></td>
</tr>
<tr>
<td>'0129</td>
<td>EC0500</td>
<td>0184</td>
<td>CALL PE,NN</td>
<td></td>
</tr>
<tr>
<td>'012C</td>
<td>E40500</td>
<td>0185</td>
<td>CALL PO,NN</td>
<td></td>
</tr>
<tr>
<td>'012F</td>
<td>CC0500</td>
<td>0186</td>
<td>CALL Z,NN</td>
<td></td>
</tr>
<tr>
<td>'0132</td>
<td>3F</td>
<td>0187</td>
<td>CCF</td>
<td></td>
</tr>
<tr>
<td>'0133</td>
<td>BE</td>
<td>0190</td>
<td>CP (HL)</td>
<td></td>
</tr>
<tr>
<td>'0134</td>
<td>DDBE05</td>
<td>0191</td>
<td>CP (IX+IND)</td>
<td></td>
</tr>
<tr>
<td>'0137</td>
<td>FDBE05</td>
<td>0192</td>
<td>CP (IY+IND)</td>
<td></td>
</tr>
<tr>
<td>'013A</td>
<td>BF</td>
<td>0193</td>
<td>CP A</td>
<td></td>
</tr>
<tr>
<td>'013B</td>
<td>B8</td>
<td>0194</td>
<td>CP B</td>
<td></td>
</tr>
<tr>
<td>'013C</td>
<td>B9</td>
<td>0195</td>
<td>CP C</td>
<td></td>
</tr>
<tr>
<td>'013D</td>
<td>BA</td>
<td>0196</td>
<td>CP D</td>
<td></td>
</tr>
<tr>
<td>'013E</td>
<td>BB</td>
<td>0197</td>
<td>CP E</td>
<td></td>
</tr>
<tr>
<td>'013F</td>
<td>BC</td>
<td>0198</td>
<td>CP H</td>
<td></td>
</tr>
<tr>
<td>'0140</td>
<td>BD</td>
<td>0199</td>
<td>CP L</td>
<td></td>
</tr>
<tr>
<td>'0141</td>
<td>FE20</td>
<td>0200</td>
<td>CP N</td>
<td></td>
</tr>
<tr>
<td>'0143</td>
<td>EDA9</td>
<td>0202</td>
<td>CPD</td>
<td></td>
</tr>
<tr>
<td>'0145</td>
<td>EDB9</td>
<td>0203</td>
<td>CPDR</td>
<td></td>
</tr>
<tr>
<td>'0147</td>
<td>EDA1</td>
<td>0204</td>
<td>CPI</td>
<td></td>
</tr>
<tr>
<td>'0149</td>
<td>EDB1</td>
<td>0205</td>
<td>CPIR</td>
<td></td>
</tr>
<tr>
<td>'014B</td>
<td>2F</td>
<td>0207</td>
<td>CPL</td>
<td></td>
</tr>
<tr>
<td>'014C</td>
<td>27</td>
<td>0208</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'014D</td>
<td>35</td>
<td>0209</td>
<td>DAA</td>
<td></td>
</tr>
<tr>
<td>'014E</td>
<td>DD3505</td>
<td>0210</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0151</td>
<td>FD3505</td>
<td>0211</td>
<td>DEC (HL)</td>
<td></td>
</tr>
<tr>
<td>'0154</td>
<td>3D</td>
<td>0212</td>
<td>DEC (IX+IND)</td>
<td></td>
</tr>
<tr>
<td>'0155</td>
<td>05</td>
<td>0213</td>
<td>DEC (IY+IND)</td>
<td></td>
</tr>
<tr>
<td>'0156</td>
<td>0B</td>
<td>0214</td>
<td>DEC A</td>
<td></td>
</tr>
<tr>
<td>'0157</td>
<td>0D</td>
<td>0215</td>
<td>DEC B</td>
<td></td>
</tr>
<tr>
<td>'0158</td>
<td>15</td>
<td>0216</td>
<td>DEC BC</td>
<td></td>
</tr>
<tr>
<td>'0159</td>
<td>1B</td>
<td>0217</td>
<td>DEC C</td>
<td></td>
</tr>
<tr>
<td>'015A</td>
<td>1D</td>
<td>0218</td>
<td>DEC D</td>
<td></td>
</tr>
<tr>
<td>'015B</td>
<td>25</td>
<td>0219</td>
<td>DEC DE</td>
<td></td>
</tr>
<tr>
<td>'015C</td>
<td>2B</td>
<td>0220</td>
<td>DEC E</td>
<td></td>
</tr>
<tr>
<td>'015D</td>
<td>DD2B</td>
<td>0221</td>
<td>DEC H</td>
<td></td>
</tr>
<tr>
<td>'015F</td>
<td>FD2B</td>
<td>0222</td>
<td>DEC HL</td>
<td></td>
</tr>
<tr>
<td>'0161</td>
<td>2D</td>
<td>0223</td>
<td>DEC IX</td>
<td></td>
</tr>
<tr>
<td>'0162</td>
<td>3B</td>
<td>0224</td>
<td>DEC IY</td>
<td></td>
</tr>
<tr>
<td>'0163</td>
<td>F3</td>
<td>0225</td>
<td>DEC L</td>
<td></td>
</tr>
<tr>
<td>'0164</td>
<td>102E</td>
<td>0226</td>
<td>DEC SP</td>
<td></td>
</tr>
<tr>
<td>'0166</td>
<td>FB</td>
<td>0227</td>
<td>DI</td>
<td></td>
</tr>
<tr>
<td>'0167</td>
<td>E3</td>
<td>0228</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0168</td>
<td>DDE3</td>
<td>0229</td>
<td>DJNZ DIS</td>
<td></td>
</tr>
<tr>
<td>'016A</td>
<td>FDE3</td>
<td>0230</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'016C</td>
<td>08</td>
<td>0231</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'016F</td>
<td>FB</td>
<td>0232</td>
<td>EI</td>
<td></td>
</tr>
<tr>
<td>'0167</td>
<td>E3</td>
<td>0233</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'0168</td>
<td>DDE3</td>
<td>0234</td>
<td>EX (SP),HL</td>
<td></td>
</tr>
<tr>
<td>'016A</td>
<td>FDE3</td>
<td>0235</td>
<td>EX (SP),IX</td>
<td></td>
</tr>
<tr>
<td>'016C</td>
<td>08</td>
<td>0236</td>
<td>EX (SP),IY</td>
<td></td>
</tr>
<tr>
<td>'016F</td>
<td>FB</td>
<td>0237</td>
<td>EX AF,AF'</td>
<td></td>
</tr>
<tr>
<td>ADDR</td>
<td>OBJECT</td>
<td>ST</td>
<td>#</td>
<td>SOURCE STATEMENT</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>----</td>
<td>---</td>
<td>------------------</td>
</tr>
<tr>
<td>'016D</td>
<td>EB</td>
<td>0238</td>
<td>EX</td>
<td>DE,HL</td>
</tr>
<tr>
<td>'016E</td>
<td>D9</td>
<td>0239</td>
<td>EXX</td>
<td></td>
</tr>
<tr>
<td>'016F</td>
<td>76</td>
<td>0240</td>
<td>HALT</td>
<td></td>
</tr>
<tr>
<td>'0170</td>
<td>ED46</td>
<td>0243</td>
<td>IM</td>
<td>0</td>
</tr>
<tr>
<td>'0172</td>
<td>ED56</td>
<td>0244</td>
<td>IM</td>
<td>1</td>
</tr>
<tr>
<td>'0174</td>
<td>ED5E</td>
<td>0245</td>
<td>IM</td>
<td>2</td>
</tr>
<tr>
<td>'0176</td>
<td>ED78</td>
<td>0246</td>
<td>IN</td>
<td>A,(C)</td>
</tr>
<tr>
<td>'0178</td>
<td>DB20</td>
<td>0248</td>
<td>IN</td>
<td>A,(N)</td>
</tr>
<tr>
<td>'017A</td>
<td>ED40</td>
<td>0249</td>
<td>IN</td>
<td>B,(C)</td>
</tr>
<tr>
<td>'017C</td>
<td>ED48</td>
<td>0250</td>
<td>IN</td>
<td>C,(C)</td>
</tr>
<tr>
<td>'017E</td>
<td>ED50</td>
<td>0251</td>
<td>IN</td>
<td>D,(C)</td>
</tr>
<tr>
<td>'0180</td>
<td>ED58</td>
<td>0252</td>
<td>IN</td>
<td>E,(C)</td>
</tr>
<tr>
<td>'0182</td>
<td>ED70</td>
<td>0253</td>
<td>IN</td>
<td>F,(C)</td>
</tr>
<tr>
<td>'0184</td>
<td>ED60</td>
<td>0254</td>
<td>IN</td>
<td>H,(C)</td>
</tr>
<tr>
<td>'0186</td>
<td>ED68</td>
<td>0255</td>
<td>IN</td>
<td>L,(C)</td>
</tr>
<tr>
<td>'0188</td>
<td>34</td>
<td>0256</td>
<td>INC</td>
<td>(HL)</td>
</tr>
<tr>
<td>'0189</td>
<td>FD3405</td>
<td>0257</td>
<td>INC</td>
<td>(IY+IND)</td>
</tr>
<tr>
<td>'018C</td>
<td>DD3405</td>
<td>0258</td>
<td>INC</td>
<td>(IX+IND)</td>
</tr>
<tr>
<td>'018F</td>
<td>3C</td>
<td>0259</td>
<td>INC</td>
<td>A</td>
</tr>
<tr>
<td>'0190</td>
<td>04</td>
<td>0260</td>
<td>INC</td>
<td>B</td>
</tr>
<tr>
<td>'0191</td>
<td>03</td>
<td>0261</td>
<td>INC</td>
<td>BC</td>
</tr>
<tr>
<td>'0192</td>
<td>0C</td>
<td>0262</td>
<td>INC</td>
<td>C</td>
</tr>
<tr>
<td>'0193</td>
<td>14</td>
<td>0263</td>
<td>INC</td>
<td>D</td>
</tr>
<tr>
<td>'0194</td>
<td>13</td>
<td>0264</td>
<td>INC</td>
<td>DE</td>
</tr>
<tr>
<td>'0195</td>
<td>1C</td>
<td>0265</td>
<td>INC</td>
<td>E</td>
</tr>
<tr>
<td>'0196</td>
<td>24</td>
<td>0266</td>
<td>INC</td>
<td>H</td>
</tr>
<tr>
<td>'0197</td>
<td>23</td>
<td>0267</td>
<td>INC</td>
<td>HL</td>
</tr>
<tr>
<td>'0198</td>
<td>DD23</td>
<td>0268</td>
<td>INC</td>
<td>IX</td>
</tr>
<tr>
<td>'019A</td>
<td>FD23</td>
<td>0269</td>
<td>INC</td>
<td>IY</td>
</tr>
<tr>
<td>'019C</td>
<td>2C</td>
<td>0270</td>
<td>INC</td>
<td>L</td>
</tr>
<tr>
<td>'019D</td>
<td>33</td>
<td>0271</td>
<td>INC</td>
<td>SP</td>
</tr>
<tr>
<td>'019E</td>
<td>EDAA</td>
<td>0272</td>
<td>IND</td>
<td></td>
</tr>
<tr>
<td>'01A0</td>
<td>EDBA</td>
<td>0273</td>
<td>INDR</td>
<td></td>
</tr>
<tr>
<td>'01A2</td>
<td>EDA2</td>
<td>0274</td>
<td>INI</td>
<td></td>
</tr>
<tr>
<td>'01A4</td>
<td>EDB2</td>
<td>0275</td>
<td>INIR</td>
<td></td>
</tr>
<tr>
<td>'01A6</td>
<td>E9</td>
<td>0276</td>
<td>JP</td>
<td>(HL)</td>
</tr>
<tr>
<td>'01A7</td>
<td>DDE9</td>
<td>0277</td>
<td>JP</td>
<td>(IX)</td>
</tr>
<tr>
<td>'01A9</td>
<td>FDE9</td>
<td>0278</td>
<td>JP</td>
<td>(IY)</td>
</tr>
<tr>
<td>'01AB</td>
<td>DA0500'</td>
<td>0279</td>
<td>JP</td>
<td>C,NN</td>
</tr>
<tr>
<td>'01AE</td>
<td>FA0500'</td>
<td>0280</td>
<td>JP</td>
<td>M,NN</td>
</tr>
<tr>
<td>'01B1</td>
<td>D20500'</td>
<td>0281</td>
<td>JP</td>
<td>NC,NN</td>
</tr>
<tr>
<td>'01B4</td>
<td>C30500'</td>
<td>0282</td>
<td>JP</td>
<td>NN</td>
</tr>
<tr>
<td>'01B7</td>
<td>C20500'</td>
<td>0283</td>
<td>JP</td>
<td>NZ,NN</td>
</tr>
<tr>
<td>'01BA</td>
<td>F20500'</td>
<td>0284</td>
<td>JP</td>
<td>P,NN</td>
</tr>
<tr>
<td>'01BD</td>
<td>EA0500'</td>
<td>0285</td>
<td>JP</td>
<td>PE,NN</td>
</tr>
<tr>
<td>'01C0</td>
<td>E20500'</td>
<td>0286</td>
<td>JP</td>
<td>PO,NN</td>
</tr>
<tr>
<td>'01C3</td>
<td>CA0500'</td>
<td>0287</td>
<td>JP</td>
<td>Z,NN</td>
</tr>
<tr>
<td>'01C6</td>
<td>382E</td>
<td>0288</td>
<td>JR</td>
<td>C,DIS</td>
</tr>
<tr>
<td>'01C8</td>
<td>182E</td>
<td>0289</td>
<td>JR</td>
<td>DIS</td>
</tr>
<tr>
<td>'01CA</td>
<td>302E</td>
<td>0290</td>
<td>JR</td>
<td>NC,DIS</td>
</tr>
<tr>
<td>'01CC</td>
<td>202E</td>
<td>0291</td>
<td>JR</td>
<td>NZ,DIS</td>
</tr>
<tr>
<td>ADDR</td>
<td>OBJECT</td>
<td>ST #</td>
<td>SOURCE</td>
<td>STATEMENT</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>------</td>
<td>--------</td>
<td>-----------</td>
</tr>
<tr>
<td>'01CE</td>
<td>282E</td>
<td>0296</td>
<td>JR</td>
<td>Z,DIS</td>
</tr>
<tr>
<td>'01D0</td>
<td>02</td>
<td>0298</td>
<td>LD</td>
<td>(BC),A</td>
</tr>
<tr>
<td>'01D1</td>
<td>12</td>
<td>0299</td>
<td>LD</td>
<td>(DE),A</td>
</tr>
<tr>
<td>'01D2</td>
<td>77</td>
<td>0300</td>
<td>LD</td>
<td>(HL),A</td>
</tr>
<tr>
<td>'01D3</td>
<td>70</td>
<td>0301</td>
<td>LD</td>
<td>(HL),B</td>
</tr>
<tr>
<td>'01D4</td>
<td>71</td>
<td>0302</td>
<td>LD</td>
<td>(HL),C</td>
</tr>
<tr>
<td>'01D5</td>
<td>72</td>
<td>0303</td>
<td>LD</td>
<td>(HL),D</td>
</tr>
<tr>
<td>'01D6</td>
<td>73</td>
<td>0304</td>
<td>LD</td>
<td>(HL),E</td>
</tr>
<tr>
<td>'01D7</td>
<td>74</td>
<td>0305</td>
<td>LD</td>
<td>(HL),H</td>
</tr>
<tr>
<td>'01D8</td>
<td>75</td>
<td>0306</td>
<td>LD</td>
<td>(HL),L</td>
</tr>
<tr>
<td>'01D9</td>
<td>3620</td>
<td>0307</td>
<td>LD</td>
<td>(HL),N</td>
</tr>
<tr>
<td>'01DB</td>
<td>DD7005</td>
<td>0309</td>
<td>LD</td>
<td>(IX+IND),A</td>
</tr>
<tr>
<td>'01DE</td>
<td>DD7005</td>
<td>0310</td>
<td>LD</td>
<td>(IX+IND),B</td>
</tr>
<tr>
<td>'01E1</td>
<td>DD7105</td>
<td>0311</td>
<td>LD</td>
<td>(IX+IND),C</td>
</tr>
<tr>
<td>'01E4</td>
<td>DD7205</td>
<td>0312</td>
<td>LD</td>
<td>(IX+IND),D</td>
</tr>
<tr>
<td>'01E7</td>
<td>DD7305</td>
<td>0313</td>
<td>LD</td>
<td>(IX+IND),E</td>
</tr>
<tr>
<td>'01EA</td>
<td>DD7405</td>
<td>0314</td>
<td>LD</td>
<td>(IX+IND),H</td>
</tr>
<tr>
<td>'01ED</td>
<td>DD7505</td>
<td>0315</td>
<td>LD</td>
<td>(IX+IND),L</td>
</tr>
<tr>
<td>'01F0</td>
<td>DD360520</td>
<td>0316</td>
<td>LD</td>
<td>(IX+IND),N</td>
</tr>
<tr>
<td>'01F4</td>
<td>FD7005</td>
<td>0318</td>
<td>LD</td>
<td>(IX+IND),A</td>
</tr>
<tr>
<td>'01F7</td>
<td>FD7005</td>
<td>0319</td>
<td>LD</td>
<td>(IX+IND),B</td>
</tr>
<tr>
<td>'01FA</td>
<td>FD7105</td>
<td>0320</td>
<td>LD</td>
<td>(IX+IND),C</td>
</tr>
<tr>
<td>'01FD</td>
<td>FD7205</td>
<td>0321</td>
<td>LD</td>
<td>(IX+IND),D</td>
</tr>
<tr>
<td>'0200</td>
<td>FD7305</td>
<td>0322</td>
<td>LD</td>
<td>(IX+IND),E</td>
</tr>
<tr>
<td>'0203</td>
<td>FD7405</td>
<td>0323</td>
<td>LD</td>
<td>(IX+IND),H</td>
</tr>
<tr>
<td>'0206</td>
<td>FD7505</td>
<td>0324</td>
<td>LD</td>
<td>(IX+IND),L</td>
</tr>
<tr>
<td>'0209</td>
<td>FD360520</td>
<td>0325</td>
<td>LD</td>
<td>(IX+IND),N</td>
</tr>
<tr>
<td>'02D0</td>
<td>320500'</td>
<td>0327</td>
<td>LD</td>
<td>(NN),A</td>
</tr>
<tr>
<td>'0210</td>
<td>ED430500</td>
<td>0328</td>
<td>LD</td>
<td>(NN),BC</td>
</tr>
<tr>
<td>'0214</td>
<td>ED530500</td>
<td>0329</td>
<td>LD</td>
<td>(NN),DE</td>
</tr>
<tr>
<td>'0218</td>
<td>220500'</td>
<td>0330</td>
<td>LD</td>
<td>(NN),HL</td>
</tr>
<tr>
<td>'021B</td>
<td>DD220500</td>
<td>0331</td>
<td>LD</td>
<td>(NN),IX</td>
</tr>
<tr>
<td>'021F</td>
<td>FD220500</td>
<td>0332</td>
<td>LD</td>
<td>(NN),IY</td>
</tr>
<tr>
<td>'0223</td>
<td>ED730500</td>
<td>0333</td>
<td>LD</td>
<td>(NN),SP</td>
</tr>
<tr>
<td>'0227</td>
<td>0A</td>
<td>0335</td>
<td>LD</td>
<td>A,(BC)</td>
</tr>
<tr>
<td>'0228</td>
<td>1A</td>
<td>0336</td>
<td>LD</td>
<td>A,(DE)</td>
</tr>
<tr>
<td>'0229</td>
<td>7E</td>
<td>0337</td>
<td>LD</td>
<td>A,(HL)</td>
</tr>
<tr>
<td>'022A</td>
<td>DD7E05</td>
<td>0338</td>
<td>LD</td>
<td>A,(IX+IND)</td>
</tr>
<tr>
<td>'022D</td>
<td>FD7E05</td>
<td>0339</td>
<td>LD</td>
<td>A,(IY+IND)</td>
</tr>
<tr>
<td>'0230</td>
<td>3A0500'</td>
<td>0340</td>
<td>LD</td>
<td>A,(NN)</td>
</tr>
<tr>
<td>'0233</td>
<td>7F</td>
<td>0341</td>
<td>LD</td>
<td>A,A</td>
</tr>
<tr>
<td>'0234</td>
<td>78</td>
<td>0342</td>
<td>LD</td>
<td>A,B</td>
</tr>
<tr>
<td>'0235</td>
<td>79</td>
<td>0343</td>
<td>LD</td>
<td>A,C</td>
</tr>
<tr>
<td>'0236</td>
<td>7A</td>
<td>0344</td>
<td>LD</td>
<td>A,D</td>
</tr>
<tr>
<td>'0237</td>
<td>7B</td>
<td>0345</td>
<td>LD</td>
<td>A,E</td>
</tr>
<tr>
<td>'0238</td>
<td>7C</td>
<td>0346</td>
<td>LD</td>
<td>A,H</td>
</tr>
<tr>
<td>'0239</td>
<td>ED57</td>
<td>0347</td>
<td>LD</td>
<td>A,I</td>
</tr>
<tr>
<td>'023B</td>
<td>7D</td>
<td>0348</td>
<td>LD</td>
<td>A,L</td>
</tr>
<tr>
<td>'023C</td>
<td>3E20</td>
<td>0349</td>
<td>LD</td>
<td>A,N</td>
</tr>
<tr>
<td>'023E</td>
<td>ED5F</td>
<td>0350</td>
<td>LD</td>
<td>A,R</td>
</tr>
<tr>
<td>'0240</td>
<td>46</td>
<td>0351</td>
<td>LD</td>
<td>B,(HL)</td>
</tr>
<tr>
<td>'0241</td>
<td>DD4605</td>
<td>0352</td>
<td>LD</td>
<td>B,(IX+IND)</td>
</tr>
<tr>
<td>ADDR</td>
<td>OBJECT</td>
<td>ST #</td>
<td>SOURCE</td>
<td>STATEMENT DATASET = DKO:OPCODE.</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>------</td>
<td>--------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>0244</td>
<td>FD4605</td>
<td>0354</td>
<td>LD</td>
<td>B,(IY+IND)</td>
</tr>
<tr>
<td>0247</td>
<td>47</td>
<td>0355</td>
<td>LD</td>
<td>B,A</td>
</tr>
<tr>
<td>0248</td>
<td>40</td>
<td>0356</td>
<td>LD</td>
<td>B,B</td>
</tr>
<tr>
<td>0249</td>
<td>41</td>
<td>0357</td>
<td>LD</td>
<td>B,C</td>
</tr>
<tr>
<td>024A</td>
<td>42</td>
<td>0358</td>
<td>LD</td>
<td>B,D</td>
</tr>
<tr>
<td>024B</td>
<td>43</td>
<td>0359</td>
<td>LD</td>
<td>B,E</td>
</tr>
<tr>
<td>024C</td>
<td>44</td>
<td>0360</td>
<td>LD</td>
<td>B,H</td>
</tr>
<tr>
<td>024D</td>
<td>45</td>
<td>0361</td>
<td>LD</td>
<td>B,L</td>
</tr>
<tr>
<td>024E</td>
<td>0620</td>
<td>0362</td>
<td>LD</td>
<td>B,N</td>
</tr>
<tr>
<td>0250</td>
<td>ED4B0500'</td>
<td>0364</td>
<td>LD</td>
<td>BC,(NN)</td>
</tr>
<tr>
<td>0254</td>
<td>010500'</td>
<td>0365</td>
<td>LD</td>
<td>BC,NN</td>
</tr>
<tr>
<td>0257</td>
<td>4E</td>
<td>0366</td>
<td>LD</td>
<td>C,(HL)</td>
</tr>
<tr>
<td>0258</td>
<td>DD4E05</td>
<td>0367</td>
<td>LD</td>
<td>C,(IX+IND)</td>
</tr>
<tr>
<td>0259</td>
<td>FD4E05</td>
<td>0368</td>
<td>LD</td>
<td>C,(IX+IND)</td>
</tr>
<tr>
<td>025E</td>
<td>4F</td>
<td>0369</td>
<td>LD</td>
<td>C,A</td>
</tr>
<tr>
<td>025F</td>
<td>48</td>
<td>0370</td>
<td>LD</td>
<td>C,B</td>
</tr>
<tr>
<td>0260</td>
<td>49</td>
<td>0371</td>
<td>LD</td>
<td>C,C</td>
</tr>
<tr>
<td>0261</td>
<td>4A</td>
<td>0372</td>
<td>LD</td>
<td>C,D</td>
</tr>
<tr>
<td>0262</td>
<td>4B</td>
<td>0373</td>
<td>LD</td>
<td>C,E</td>
</tr>
<tr>
<td>0263</td>
<td>4C</td>
<td>0374</td>
<td>LD</td>
<td>C,H</td>
</tr>
<tr>
<td>0264</td>
<td>4D</td>
<td>0375</td>
<td>LD</td>
<td>C,L</td>
</tr>
<tr>
<td>0265</td>
<td>0E20</td>
<td>0376</td>
<td>LD</td>
<td>C,N</td>
</tr>
<tr>
<td>0267</td>
<td>56</td>
<td>0377</td>
<td>LD</td>
<td>D,(HL)</td>
</tr>
<tr>
<td>0268</td>
<td>DD5605</td>
<td>0378</td>
<td>LD</td>
<td>D,(IX+IND)</td>
</tr>
<tr>
<td>026B</td>
<td>FD5605</td>
<td>0379</td>
<td>LD</td>
<td>D,(IX+IND)</td>
</tr>
<tr>
<td>026E</td>
<td>57</td>
<td>0380</td>
<td>LD</td>
<td>D,A</td>
</tr>
<tr>
<td>026F</td>
<td>50</td>
<td>0381</td>
<td>LD</td>
<td>D,B</td>
</tr>
<tr>
<td>0270</td>
<td>51</td>
<td>0382</td>
<td>LD</td>
<td>D,C</td>
</tr>
<tr>
<td>0271</td>
<td>52</td>
<td>0383</td>
<td>LD</td>
<td>D,D</td>
</tr>
<tr>
<td>0272</td>
<td>53</td>
<td>0384</td>
<td>LD</td>
<td>D,E</td>
</tr>
<tr>
<td>0273</td>
<td>54</td>
<td>0385</td>
<td>LD</td>
<td>D,H</td>
</tr>
<tr>
<td>0274</td>
<td>55</td>
<td>0386</td>
<td>LD</td>
<td>D,L</td>
</tr>
<tr>
<td>0275</td>
<td>1620</td>
<td>0387</td>
<td>LD</td>
<td>D,N</td>
</tr>
<tr>
<td>0277</td>
<td>ED5B0500'</td>
<td>0388</td>
<td>LD</td>
<td>DE,(NN)</td>
</tr>
<tr>
<td>027B</td>
<td>110500'</td>
<td>0389</td>
<td>LD</td>
<td>DE,NN</td>
</tr>
<tr>
<td>027E</td>
<td>5E</td>
<td>0390</td>
<td>LD</td>
<td>E,(HL)</td>
</tr>
<tr>
<td>027F</td>
<td>DD5E05</td>
<td>0391</td>
<td>LD</td>
<td>E,(IX+IND)</td>
</tr>
<tr>
<td>0282</td>
<td>FD5E05</td>
<td>0392</td>
<td>LD</td>
<td>E,(IX+IND)</td>
</tr>
<tr>
<td>0285</td>
<td>5F</td>
<td>0393</td>
<td>LD</td>
<td>E,A</td>
</tr>
<tr>
<td>0286</td>
<td>58</td>
<td>0394</td>
<td>LD</td>
<td>E,B</td>
</tr>
<tr>
<td>0287</td>
<td>59</td>
<td>0395</td>
<td>LD</td>
<td>E,C</td>
</tr>
<tr>
<td>0288</td>
<td>5A</td>
<td>0396</td>
<td>LD</td>
<td>E,D</td>
</tr>
<tr>
<td>0289</td>
<td>5B</td>
<td>0397</td>
<td>LD</td>
<td>E,E</td>
</tr>
<tr>
<td>028A</td>
<td>5C</td>
<td>0398</td>
<td>LD</td>
<td>E,H</td>
</tr>
<tr>
<td>028B</td>
<td>5D</td>
<td>0399</td>
<td>LD</td>
<td>E,L</td>
</tr>
<tr>
<td>028C</td>
<td>1E20</td>
<td>0400</td>
<td>LD</td>
<td>E,N</td>
</tr>
<tr>
<td>028E</td>
<td>66</td>
<td>0401</td>
<td>LD</td>
<td>H,(HL)</td>
</tr>
<tr>
<td>028F</td>
<td>DD6605</td>
<td>0402</td>
<td>LD</td>
<td>H,(IX+IND)</td>
</tr>
<tr>
<td>0292</td>
<td>FD6605</td>
<td>0403</td>
<td>LD</td>
<td>H,(IX+IND)</td>
</tr>
<tr>
<td>0295</td>
<td>67</td>
<td>0404</td>
<td>LD</td>
<td>H,A</td>
</tr>
<tr>
<td>0296</td>
<td>60</td>
<td>0405</td>
<td>LD</td>
<td>H,B</td>
</tr>
<tr>
<td>0297</td>
<td>61</td>
<td>0406</td>
<td>LD</td>
<td>H,C</td>
</tr>
</tbody>
</table>
OPCODE Z80 OPCODE LISTING

ADDR | OBJECT | ST # | SOURCE | STATEMENT
--- | --- | --- | --- | ---
'0298 62 | 0412 | LD | H,D | DATASET = DKO:OPCODE.
'0299 63 | 0413 | LD | H,E |
'029A 64 | 0414 | LD | H,H |
'029B 65 | 0415 | LD | H,L |
'029C 2620 | 0416 | LD | H,N |
'029E 2A0500' | 0417 | LD | HL,(NN) |
'02A1 210500' | 0418 | LD | HL,NN |
'02A4 ED47 | 0420 | LD | I,A |
'02A6 DD2A0500' | 0421 | LD | IX,(NN) |
'02AA DD210500' | 0422 | LD | IX,NN |
'02AE FD2A0500' | 0423 | LD | IY,(NN) |
'02B2 FD210500' | 0424 | LD | IY,NN |
'02B6 6E | 0425 | LD | L,(HL) |
'02B7 DD6E05 | 0426 | LD | L,(IX+IND) |
'02BA FD6E05 | 0427 | LD | L,(IY+IND) |
'02BD 6F | 0428 | LD | L,A |
'02BE 68 | 0429 | LD | L,B |
'02BF 69 | 0430 | LD | L,C |
'02C0 6A | 0431 | LD | L,D |
'02C1 6B | 0432 | LD | L,E |
'02C2 6C | 0433 | LD | L,H |
'02C3 6D | 0434 | LD | L,L |
'02C4 2E20 | 0435 | LD | L,N |
'02C6 ED4F | 0436 | LD | R,A |
'02C8 ED7B0500' | 0437 | LD | SP,(NN) |
'02CC F9 | 0438 | LD | SP,HL |
'02CD DDF9 | 0439 | LD | SP,IX |
'02CF FDF9 | 0440 | LD | SP,IY |
'02D1 310500' | 0441 | LD | SP,NN |
'02D4 EDA8 | 0442 | LDD |
'02D6 EDB8 | 0443 | LDDR |
'02D8 EDAO | 0444 | LDI |
'02DA EDB0 | 0445 | LDIR |
'02DC ED44 | 0446 | NEG |
'02DE 00 | 0447 | NOP |
'02DF B6 | 0448 | OR | (HL) |
'02E0 DDB605 | 0449 | OR | (IX+IND) |
'02E2 FDB605 | 0450 | OR | (IY+IND) |
'02E3 DDB605 | 0451 | OR | A |
'02E4 B0 | 0452 | OR | B |
'02E5 B1 | 0453 | OR | C |
'02E6 B2 | 0454 | OR | D |
'02E7 B3 | 0455 | OR | E |
'02E8 B4 | 0456 | OR | H |
'02E9 B5 | 0457 | OR | L |
'02EA B6 | 0458 | OR | N |
'02EB F620 | 0459 | OR | (IX+IND) |
'02EC 620 | 0460 | OR | (IY+IND) |
<table>
<thead>
<tr>
<th>ADDR</th>
<th>OBJECT</th>
<th>ST #</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>02EF</td>
<td>EDBB</td>
<td>0470</td>
<td>OTDR</td>
</tr>
<tr>
<td>02F1</td>
<td>EDB3</td>
<td>0471</td>
<td>OTIR</td>
</tr>
<tr>
<td>02F3</td>
<td>ED79</td>
<td>0472</td>
<td>;</td>
</tr>
<tr>
<td>02F5</td>
<td>ED41</td>
<td>0473</td>
<td>OUT (C),A</td>
</tr>
<tr>
<td>02F7</td>
<td>ED49</td>
<td>0474</td>
<td>OUT (C),B</td>
</tr>
<tr>
<td>02F9</td>
<td>ED51</td>
<td>0475</td>
<td>OUT (C),C</td>
</tr>
<tr>
<td>02FB</td>
<td>ED59</td>
<td>0476</td>
<td>OUT (C),D</td>
</tr>
<tr>
<td>02FD</td>
<td>ED61</td>
<td>0477</td>
<td>OUT (C),E</td>
</tr>
<tr>
<td>02FF</td>
<td>ED69</td>
<td>0478</td>
<td>OUT (C),H</td>
</tr>
<tr>
<td>0301</td>
<td>D320</td>
<td>0479</td>
<td>OUT (C),L</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0480</td>
<td>OUT (N),A</td>
</tr>
<tr>
<td>0303</td>
<td>EDAB</td>
<td>0481</td>
<td>;</td>
</tr>
<tr>
<td>0305</td>
<td>EDA3</td>
<td>0482</td>
<td>OUTD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0483</td>
<td>OUTI</td>
</tr>
<tr>
<td>0307</td>
<td>F1</td>
<td>0484</td>
<td>;</td>
</tr>
<tr>
<td>0308</td>
<td>C1</td>
<td>0485</td>
<td>POP AF</td>
</tr>
<tr>
<td>0309</td>
<td>D1</td>
<td>0486</td>
<td>POP BC</td>
</tr>
<tr>
<td>030A</td>
<td>E1</td>
<td>0487</td>
<td>POP DE</td>
</tr>
<tr>
<td>030B</td>
<td>DDE1</td>
<td>0488</td>
<td>POP IX</td>
</tr>
<tr>
<td>030D</td>
<td>DDE1</td>
<td>0489</td>
<td>POP IY</td>
</tr>
<tr>
<td>030F</td>
<td>F5</td>
<td>0490</td>
<td>POP AF</td>
</tr>
<tr>
<td>0310</td>
<td>C5</td>
<td>0491</td>
<td>PUSH BC</td>
</tr>
<tr>
<td>0311</td>
<td>D5</td>
<td>0492</td>
<td>PUSH DE</td>
</tr>
<tr>
<td>0312</td>
<td>E5</td>
<td>0493</td>
<td>PUSH HL</td>
</tr>
<tr>
<td>0313</td>
<td>DDE5</td>
<td>0494</td>
<td>PUSH IX</td>
</tr>
<tr>
<td>0315</td>
<td>DDE5</td>
<td>0495</td>
<td>PUSH IY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0496</td>
<td>;</td>
</tr>
<tr>
<td>0317</td>
<td>CB86</td>
<td>0497</td>
<td>RES 0,(HL)</td>
</tr>
<tr>
<td>0319</td>
<td>DDCB0586</td>
<td>0498</td>
<td>RES 0,(IX+IND)</td>
</tr>
<tr>
<td>031D</td>
<td>FDCB0586</td>
<td>0499</td>
<td>RES 0,(IY+IND)</td>
</tr>
<tr>
<td>0321</td>
<td>CB87</td>
<td>0500</td>
<td>RES 0,A</td>
</tr>
<tr>
<td>0323</td>
<td>CB80</td>
<td>0501</td>
<td>RES 0,B</td>
</tr>
<tr>
<td>0325</td>
<td>CB81</td>
<td>0502</td>
<td>RES 0,C</td>
</tr>
<tr>
<td>0327</td>
<td>CB82</td>
<td>0503</td>
<td>RES 0,D</td>
</tr>
<tr>
<td>0329</td>
<td>CB83</td>
<td>0504</td>
<td>RES 0,E</td>
</tr>
<tr>
<td>032B</td>
<td>CB84</td>
<td>0505</td>
<td>RES 0,H</td>
</tr>
<tr>
<td>032D</td>
<td>CB85</td>
<td>0506</td>
<td>RES 0,L</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0507</td>
<td>;</td>
</tr>
<tr>
<td>032F</td>
<td>CB8E</td>
<td>0508</td>
<td>RES 1,(HL)</td>
</tr>
<tr>
<td>0331</td>
<td>DDCB0586</td>
<td>0509</td>
<td>RES 1,(IX+IND)</td>
</tr>
<tr>
<td>0335</td>
<td>FDCB0586</td>
<td>0510</td>
<td>RES 1,(IY+IND)</td>
</tr>
<tr>
<td>0339</td>
<td>CB8F</td>
<td>0511</td>
<td>RES 1,A</td>
</tr>
<tr>
<td>033B</td>
<td>CB88</td>
<td>0512</td>
<td>RES 1,B</td>
</tr>
<tr>
<td>033D</td>
<td>CB89</td>
<td>0513</td>
<td>RES 1,C</td>
</tr>
<tr>
<td>033F</td>
<td>CB8A</td>
<td>0514</td>
<td>RES 1,D</td>
</tr>
<tr>
<td>0341</td>
<td>CB8B</td>
<td>0515</td>
<td>RES 1,E</td>
</tr>
<tr>
<td>0343</td>
<td>CB8C</td>
<td>0516</td>
<td>RES 1,H</td>
</tr>
<tr>
<td>0345</td>
<td>CB8D</td>
<td>0517</td>
<td>RES 1,L</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0518</td>
<td>;</td>
</tr>
<tr>
<td>0347</td>
<td>CB96</td>
<td>0519</td>
<td>RES 2,(HL)</td>
</tr>
<tr>
<td>0349</td>
<td>DDCB0596</td>
<td>0520</td>
<td>RES 2,(IX+IND)</td>
</tr>
<tr>
<td>034D</td>
<td>FDCB0596</td>
<td>0521</td>
<td>RES 2,(IY+IND)</td>
</tr>
<tr>
<td>0351</td>
<td>CB97</td>
<td>0522</td>
<td>RES 2,A</td>
</tr>
<tr>
<td>0353</td>
<td>CB90</td>
<td>0523</td>
<td>RES 2,B</td>
</tr>
<tr>
<td>0355</td>
<td>CB91</td>
<td>0524</td>
<td>RES 2,C</td>
</tr>
<tr>
<td>0357</td>
<td>CB92</td>
<td>0525</td>
<td>RES 2,D</td>
</tr>
<tr>
<td>0359</td>
<td>CB93</td>
<td>0526</td>
<td>RES 2,E</td>
</tr>
<tr>
<td>ADDR</td>
<td>OBJECT</td>
<td>ST</td>
<td>#</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>----</td>
<td>---</td>
</tr>
<tr>
<td>'035B</td>
<td>CB94</td>
<td>0528</td>
<td>RES</td>
</tr>
<tr>
<td>'035D</td>
<td>CB95</td>
<td>0529</td>
<td>RES</td>
</tr>
<tr>
<td>'035F</td>
<td>CB9E</td>
<td>0530 ;</td>
<td>RES</td>
</tr>
<tr>
<td>'0361</td>
<td>DDCB059E</td>
<td>0531</td>
<td>RES</td>
</tr>
<tr>
<td>'0365</td>
<td>FDCB059E</td>
<td>0532</td>
<td>RES</td>
</tr>
<tr>
<td>'0369</td>
<td>CB9F</td>
<td>0533</td>
<td>RES</td>
</tr>
<tr>
<td>'036B</td>
<td>CB98</td>
<td>0534</td>
<td>RES</td>
</tr>
<tr>
<td>'036D</td>
<td>CB99</td>
<td>0535</td>
<td>RES</td>
</tr>
<tr>
<td>'036F</td>
<td>CB9A</td>
<td>0536</td>
<td>RES</td>
</tr>
<tr>
<td>'0371</td>
<td>CB9B</td>
<td>0537</td>
<td>RES</td>
</tr>
<tr>
<td>'0373</td>
<td>CB9C</td>
<td>0538</td>
<td>RES</td>
</tr>
<tr>
<td>'0375</td>
<td>CB9D</td>
<td>0539</td>
<td>RES</td>
</tr>
<tr>
<td>'0377</td>
<td>CBA6</td>
<td>0540</td>
<td>RES</td>
</tr>
<tr>
<td>'0379</td>
<td>DDCB05A6</td>
<td>0541</td>
<td>RES</td>
</tr>
<tr>
<td>'037D</td>
<td>FDCB05A6</td>
<td>0542</td>
<td>RES</td>
</tr>
<tr>
<td>'0381</td>
<td>CBA7</td>
<td>0543</td>
<td>RES</td>
</tr>
<tr>
<td>'0383</td>
<td>CBA0</td>
<td>0544</td>
<td>RES</td>
</tr>
<tr>
<td>'0385</td>
<td>CBA1</td>
<td>0545</td>
<td>RES</td>
</tr>
<tr>
<td>'0387</td>
<td>CBA2</td>
<td>0546</td>
<td>RES</td>
</tr>
<tr>
<td>'0389</td>
<td>CBA3</td>
<td>0547</td>
<td>RES</td>
</tr>
<tr>
<td>'038B</td>
<td>CBA4</td>
<td>0548</td>
<td>RES</td>
</tr>
<tr>
<td>'038D</td>
<td>CBA5</td>
<td>0549</td>
<td>RES</td>
</tr>
<tr>
<td>'038F</td>
<td>CBAE</td>
<td>0550</td>
<td>RES</td>
</tr>
<tr>
<td>'0391</td>
<td>DDCB05AE</td>
<td>0551</td>
<td>RES</td>
</tr>
<tr>
<td>'0395</td>
<td>FDCB05AE</td>
<td>0552</td>
<td>RES</td>
</tr>
<tr>
<td>'0399</td>
<td>CBAF</td>
<td>0553</td>
<td>RES</td>
</tr>
<tr>
<td>'039B</td>
<td>CBA8</td>
<td>0554</td>
<td>RES</td>
</tr>
<tr>
<td>'039D</td>
<td>CBA9</td>
<td>0555</td>
<td>RES</td>
</tr>
<tr>
<td>'039F</td>
<td>CBA0</td>
<td>0556</td>
<td>RES</td>
</tr>
<tr>
<td>'03A1</td>
<td>CBA1</td>
<td>0557</td>
<td>RES</td>
</tr>
<tr>
<td>'03A3</td>
<td>CBA2</td>
<td>0558</td>
<td>RES</td>
</tr>
<tr>
<td>'03A5</td>
<td>CBA3</td>
<td>0559</td>
<td>RES</td>
</tr>
<tr>
<td>'03A7</td>
<td>CBB6</td>
<td>0560</td>
<td>RES</td>
</tr>
<tr>
<td>'03A9</td>
<td>DDCB05B6</td>
<td>0561</td>
<td>RES</td>
</tr>
<tr>
<td>'03AD</td>
<td>FDCB05B6</td>
<td>0562</td>
<td>RES</td>
</tr>
<tr>
<td>'03B1</td>
<td>CBB7</td>
<td>0563</td>
<td>RES</td>
</tr>
<tr>
<td>'03B3</td>
<td>CBB0</td>
<td>0564</td>
<td>RES</td>
</tr>
<tr>
<td>'03B5</td>
<td>CBB1</td>
<td>0565</td>
<td>RES</td>
</tr>
<tr>
<td>'03B7</td>
<td>CBB2</td>
<td>0566</td>
<td>RES</td>
</tr>
<tr>
<td>'03B9</td>
<td>CBB3</td>
<td>0567</td>
<td>RES</td>
</tr>
<tr>
<td>'03BB</td>
<td>CBB4</td>
<td>0568</td>
<td>RES</td>
</tr>
<tr>
<td>'03BD</td>
<td>CBB5</td>
<td>0569</td>
<td>RES</td>
</tr>
<tr>
<td>'03BF</td>
<td>CBBE</td>
<td>0570</td>
<td>RES</td>
</tr>
<tr>
<td>'03C1</td>
<td>DDCB05BE</td>
<td>0571</td>
<td>RES</td>
</tr>
<tr>
<td>'03C5</td>
<td>FDCB05BE</td>
<td>0572</td>
<td>RES</td>
</tr>
<tr>
<td>'03C9</td>
<td>CBBF</td>
<td>0573</td>
<td>RES</td>
</tr>
<tr>
<td>'03CB</td>
<td>CB8</td>
<td>0574</td>
<td>RES</td>
</tr>
<tr>
<td>'03CD</td>
<td>CB9</td>
<td>0575</td>
<td>RES</td>
</tr>
<tr>
<td>'03CF</td>
<td>CB1</td>
<td>0576</td>
<td>RES</td>
</tr>
<tr>
<td>'03D1</td>
<td>CB2</td>
<td>0577</td>
<td>RES</td>
</tr>
<tr>
<td>'03D3</td>
<td>CB3</td>
<td>0578</td>
<td>RES</td>
</tr>
<tr>
<td>'03D5</td>
<td>CB4</td>
<td>0579</td>
<td>RES</td>
</tr>
<tr>
<td>'03D9</td>
<td>CB5</td>
<td>0580</td>
<td>RES</td>
</tr>
<tr>
<td>'03D7</td>
<td>CB6</td>
<td>0581</td>
<td>RES</td>
</tr>
<tr>
<td>'03D5</td>
<td>CB7</td>
<td>0582</td>
<td>RES</td>
</tr>
<tr>
<td>'03D9</td>
<td>CB8</td>
<td>0583</td>
<td>RES</td>
</tr>
<tr>
<td>'03DB</td>
<td>CB9</td>
<td>0584</td>
<td>RES</td>
</tr>
<tr>
<td>'03DF</td>
<td>CB1</td>
<td>0585 ;</td>
<td>RES</td>
</tr>
<tr>
<td>'03D3</td>
<td>CB2</td>
<td>RES</td>
<td>7, D</td>
</tr>
<tr>
<td>'03D5</td>
<td>CB3</td>
<td>RES</td>
<td>7, E</td>
</tr>
<tr>
<td>'03D7</td>
<td>CB4</td>
<td>RES</td>
<td>7, F</td>
</tr>
<tr>
<td>'03DB</td>
<td>CB5</td>
<td>RES</td>
<td>7, L</td>
</tr>
<tr>
<td>ADDR</td>
<td>OBJECT</td>
<td>OBJ</td>
<td>ST #</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>-----</td>
<td>------</td>
</tr>
<tr>
<td>'03D7</td>
<td>C9</td>
<td>0586</td>
<td>RET</td>
</tr>
<tr>
<td>'03D8</td>
<td>D8</td>
<td>0587</td>
<td>RET</td>
</tr>
<tr>
<td>'03D9</td>
<td>F8</td>
<td>0588</td>
<td>RET</td>
</tr>
<tr>
<td>'03DA</td>
<td>D0</td>
<td>0589</td>
<td>RET</td>
</tr>
<tr>
<td>'03DB</td>
<td>C0</td>
<td>0590</td>
<td>RET</td>
</tr>
<tr>
<td>'03DC</td>
<td>F0</td>
<td>0591</td>
<td>RET</td>
</tr>
<tr>
<td>'03DD</td>
<td>E8</td>
<td>0592</td>
<td>RET</td>
</tr>
<tr>
<td>'03DE</td>
<td>E0</td>
<td>0593</td>
<td>RET</td>
</tr>
<tr>
<td>'03DF</td>
<td>C8</td>
<td>0594</td>
<td>RET</td>
</tr>
<tr>
<td>'03E0</td>
<td>ED4D</td>
<td>0596</td>
<td>RETI</td>
</tr>
<tr>
<td>'03E2</td>
<td>ED45</td>
<td>0597</td>
<td>RETN</td>
</tr>
<tr>
<td>'03E4</td>
<td>CB16</td>
<td>0599</td>
<td>RL</td>
</tr>
<tr>
<td>'03E6</td>
<td>DDCB0516</td>
<td>0600</td>
<td>RL</td>
</tr>
<tr>
<td>'03EA</td>
<td>FDCB0516</td>
<td>0601</td>
<td>RL</td>
</tr>
<tr>
<td>'03EE</td>
<td>CB17</td>
<td>0602</td>
<td>RL</td>
</tr>
<tr>
<td>'03F0</td>
<td>CB10</td>
<td>0603</td>
<td>RL</td>
</tr>
<tr>
<td>'03F2</td>
<td>CB11</td>
<td>0604</td>
<td>RL</td>
</tr>
<tr>
<td>'03F4</td>
<td>CB12</td>
<td>0605</td>
<td>RL</td>
</tr>
<tr>
<td>'03F6</td>
<td>CB13</td>
<td>0606</td>
<td>RL</td>
</tr>
<tr>
<td>'03F8</td>
<td>CB14</td>
<td>0607</td>
<td>RL</td>
</tr>
<tr>
<td>'03FA</td>
<td>CB15</td>
<td>0608</td>
<td>RL</td>
</tr>
<tr>
<td>'03FC</td>
<td>17</td>
<td>0610</td>
<td>RLA</td>
</tr>
<tr>
<td>'03FD</td>
<td>CB06</td>
<td>0612</td>
<td>RLC</td>
</tr>
<tr>
<td>'03FF</td>
<td>DDCB0506</td>
<td>0613</td>
<td>RLC</td>
</tr>
<tr>
<td>'0403</td>
<td>FDCB0506</td>
<td>0614</td>
<td>RLC</td>
</tr>
<tr>
<td>'0407</td>
<td>CB07</td>
<td>0615</td>
<td>RLC</td>
</tr>
<tr>
<td>'0409</td>
<td>CB00</td>
<td>0616</td>
<td>RLC</td>
</tr>
<tr>
<td>'040B</td>
<td>CB01</td>
<td>0617</td>
<td>RLC</td>
</tr>
<tr>
<td>'040D</td>
<td>CB02</td>
<td>0618</td>
<td>RLC</td>
</tr>
<tr>
<td>'040F</td>
<td>CB03</td>
<td>0619</td>
<td>RLC</td>
</tr>
<tr>
<td>'0411</td>
<td>CB04</td>
<td>0620</td>
<td>RLC</td>
</tr>
<tr>
<td>'0413</td>
<td>CB05</td>
<td>0621</td>
<td>RLC</td>
</tr>
<tr>
<td>'0415</td>
<td>07</td>
<td>0623</td>
<td>RLCA</td>
</tr>
<tr>
<td>'0416</td>
<td>ED6F</td>
<td>0625</td>
<td>RLD</td>
</tr>
<tr>
<td>'0418</td>
<td>CB1E</td>
<td>0627</td>
<td>RR</td>
</tr>
<tr>
<td>'041A</td>
<td>DDCB051E</td>
<td>0628</td>
<td>RR</td>
</tr>
<tr>
<td>'041E</td>
<td>FDCB051E</td>
<td>0629</td>
<td>RR</td>
</tr>
<tr>
<td>'0422</td>
<td>CB1F</td>
<td>0630</td>
<td>RR</td>
</tr>
<tr>
<td>'0424</td>
<td>CB18</td>
<td>0631</td>
<td>RR</td>
</tr>
<tr>
<td>'0426</td>
<td>CB19</td>
<td>0632</td>
<td>RR</td>
</tr>
<tr>
<td>'0428</td>
<td>CB1A</td>
<td>0633</td>
<td>RR</td>
</tr>
<tr>
<td>'042A</td>
<td>CB1B</td>
<td>0634</td>
<td>RR</td>
</tr>
<tr>
<td>'042C</td>
<td>CB1C</td>
<td>0635</td>
<td>RR</td>
</tr>
<tr>
<td>'042E</td>
<td>CB1D</td>
<td>0636</td>
<td>RR</td>
</tr>
<tr>
<td>'0430</td>
<td>1F</td>
<td>0638</td>
<td>RRA</td>
</tr>
<tr>
<td>'0431</td>
<td>CB0E</td>
<td>0640</td>
<td>RRC</td>
</tr>
<tr>
<td>'0433</td>
<td>DDCB050E</td>
<td>0641</td>
<td>RRC</td>
</tr>
<tr>
<td>'0437</td>
<td>FDCB050E</td>
<td>0642</td>
<td>RRC</td>
</tr>
<tr>
<td>'043B</td>
<td>CB0F</td>
<td>0643</td>
<td>RRC</td>
</tr>
<tr>
<td>ADDR</td>
<td>OBJECT</td>
<td>ST #</td>
<td>SOURCE STATEMENT</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>'043D</td>
<td>CB08</td>
<td>0644</td>
<td>RRC B</td>
</tr>
<tr>
<td>'043F</td>
<td>CB09</td>
<td>0645</td>
<td>RRC C</td>
</tr>
<tr>
<td>'0441</td>
<td>CB0A</td>
<td>0646</td>
<td>RRC D</td>
</tr>
<tr>
<td>'0443</td>
<td>CB0B</td>
<td>0647</td>
<td>RRC E</td>
</tr>
<tr>
<td>'0445</td>
<td>CB0C</td>
<td>0648</td>
<td>RRC H</td>
</tr>
<tr>
<td>'0447</td>
<td>CB0D</td>
<td>0649</td>
<td>RRC L</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0650 ;</td>
<td></td>
</tr>
<tr>
<td>'0449</td>
<td>0F</td>
<td>0651</td>
<td>RRCA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0652 ;</td>
<td></td>
</tr>
<tr>
<td>'044A</td>
<td>ED67</td>
<td>0653</td>
<td>RRD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0654 ;</td>
<td></td>
</tr>
<tr>
<td>'044C</td>
<td>C7</td>
<td>0655</td>
<td>RST 0</td>
</tr>
<tr>
<td>'044D</td>
<td>CF</td>
<td>0656</td>
<td>RST 08H</td>
</tr>
<tr>
<td>'044E</td>
<td>D7</td>
<td>0657</td>
<td>RST 10H</td>
</tr>
<tr>
<td>'044F</td>
<td>DF</td>
<td>0658</td>
<td>RST 18H</td>
</tr>
<tr>
<td>'0450</td>
<td>E7</td>
<td>0659</td>
<td>RST 20H</td>
</tr>
<tr>
<td>'0451</td>
<td>EF</td>
<td>0660</td>
<td>RST 28H</td>
</tr>
<tr>
<td>'0452</td>
<td>F7</td>
<td>0661</td>
<td>RST 30H</td>
</tr>
<tr>
<td>'0453</td>
<td>FF</td>
<td>0662</td>
<td>RST 38H</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0663 ;</td>
<td></td>
</tr>
<tr>
<td>'0454</td>
<td>9E</td>
<td>0664</td>
<td>SBC A,(HL)</td>
</tr>
<tr>
<td>'0455</td>
<td>DD9E05</td>
<td>0665</td>
<td>SBC A,(IX+IND)</td>
</tr>
<tr>
<td>'0458</td>
<td>FD9E05</td>
<td>0666</td>
<td>SBC A,(IY+IND)</td>
</tr>
<tr>
<td>'045B</td>
<td>9F</td>
<td>0667</td>
<td>SBC A,A</td>
</tr>
<tr>
<td>'045C</td>
<td>98</td>
<td>0668</td>
<td>SBC A,B</td>
</tr>
<tr>
<td>'045D</td>
<td>99</td>
<td>0669</td>
<td>SBC A,C</td>
</tr>
<tr>
<td>'045E</td>
<td>9A</td>
<td>0670</td>
<td>SBC A,D</td>
</tr>
<tr>
<td>'045F</td>
<td>9B</td>
<td>0671</td>
<td>SBC A,E</td>
</tr>
<tr>
<td>'0460</td>
<td>9C</td>
<td>0672</td>
<td>SBC A,H</td>
</tr>
<tr>
<td>'0461</td>
<td>9D</td>
<td>0673</td>
<td>SBC A,L</td>
</tr>
<tr>
<td>'0462</td>
<td>DE20</td>
<td>0674</td>
<td>SBC A,N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0675 ;</td>
<td></td>
</tr>
<tr>
<td>'0464</td>
<td>ED42</td>
<td>0676</td>
<td>SBC HL,BC</td>
</tr>
<tr>
<td>'0466</td>
<td>ED52</td>
<td>0677</td>
<td>SBC HL,DE</td>
</tr>
<tr>
<td>'0468</td>
<td>ED62</td>
<td>0678</td>
<td>SBC HL,HL</td>
</tr>
<tr>
<td>'046A</td>
<td>ED72</td>
<td>0679</td>
<td>SBC HL,SP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0680 ;</td>
<td></td>
</tr>
<tr>
<td>'046C</td>
<td>37</td>
<td>0681</td>
<td>SCF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0682 ;</td>
<td></td>
</tr>
<tr>
<td>'046D</td>
<td>CBC6</td>
<td>0683</td>
<td>SET 0,(HL)</td>
</tr>
<tr>
<td>'046F</td>
<td>DDCB05C6</td>
<td>0684</td>
<td>SET 0,(IX+IND)</td>
</tr>
<tr>
<td>'0473</td>
<td>FDCB05C6</td>
<td>0685</td>
<td>SET 0,(IY+IND)</td>
</tr>
<tr>
<td>'0477</td>
<td>CBC7</td>
<td>0686</td>
<td>SET 0,A</td>
</tr>
<tr>
<td>'0479</td>
<td>CBC0</td>
<td>0687</td>
<td>SET 0,B</td>
</tr>
<tr>
<td>'047B</td>
<td>CBC1</td>
<td>0688</td>
<td>SET 0,C</td>
</tr>
<tr>
<td>'047D</td>
<td>CBC2</td>
<td>0689</td>
<td>SET 0,D</td>
</tr>
<tr>
<td>'047F</td>
<td>CBC3</td>
<td>0690</td>
<td>SET 0,E</td>
</tr>
<tr>
<td>'0481</td>
<td>CBC4</td>
<td>0691</td>
<td>SET 0,H</td>
</tr>
<tr>
<td>'0483</td>
<td>CBC5</td>
<td>0692</td>
<td>SET 0,L</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0693 ;</td>
<td></td>
</tr>
<tr>
<td>'0485</td>
<td>CBCE</td>
<td>0694</td>
<td>SET 1,(HL)</td>
</tr>
<tr>
<td>'0487</td>
<td>DDCB05CE</td>
<td>0695</td>
<td>SET 1,(IX+IND)</td>
</tr>
<tr>
<td>'048B</td>
<td>FDCB05CE</td>
<td>0696</td>
<td>SET 1,(IY+IND)</td>
</tr>
<tr>
<td>'048F</td>
<td>CBCF</td>
<td>0697</td>
<td>SET 1,A</td>
</tr>
<tr>
<td>'0491</td>
<td>CBC8</td>
<td>0698</td>
<td>SET 1,B</td>
</tr>
<tr>
<td>'0493</td>
<td>CBC9</td>
<td>0699</td>
<td>SET 1,C</td>
</tr>
<tr>
<td>'0495</td>
<td>CBCA</td>
<td>0700</td>
<td>SET 1,D</td>
</tr>
<tr>
<td>'0497</td>
<td>CBCB</td>
<td>0701</td>
<td>SET 1,E</td>
</tr>
</tbody>
</table>
7499 CBCC 0702 SET 1, H
749B CBCD 0703 SET 1, L
749D CBDE 0704 SET 2, (HL)
749F DDCB05D6 0705 SET 2, (IX+IND)
74A3 FDCB05D6 0706 SET 2, (IX+IND)
74A7 CBD7 0707 SET 2, A
74A9 CBD0 0708 SET 2, B
74AB CBD1 0709 SET 2, C
74AD CBD2 0710 SET 2, D
74AF CBD3 0711 SET 2, E
74B1 CBD4 0712 SET 2, H
74B3 CBD5 0713 SET 2, L
74B5 CBDE 0714 SET 3, (HL)
74B7 DDCB05DE 0715 SET 3, (IX+IND)
74BB FDCB05DE 0716 SET 3, (IX+IND)
74BF CBDF 0717 SET 3, A
74C1 CBD8 0718 SET 3, B
74C3 CBD9 0719 SET 3, C
74C5 CBDA 0720 SET 3, D
74C7 CBDB 0721 SET 3, E
74C9 CBDC 0722 SET 3, H
74CB CBDD 0723 SET 3, L
74CD CBE6 0724 SET 4, (HL)
74CF DDCB05E6 0725 SET 4, (IX+IND)
74D3 FDCB05E6 0726 SET 4, (IX+IND)
74D7 CBE7 0727 SET 4, A
74D9 CBE0 0728 SET 4, B
74DB CBE1 0729 SET 4, C
74DD CBE2 0730 SET 4, D
74DF CBE3 0731 SET 4, E
74E1 CBE4 0732 SET 4, H
74E3 CBE5 0733 SET 4, L
74E5 CBE6 0734 SET 5, (HL)
74E7 DDCB05EE 0735 SET 5, (IX+IND)
74EB FDCB05EE 0736 SET 5, (IX+IND)
74EF CBEF 0737 SET 5, A
74F1 CBE8 0738 SET 5, B
74F3 CBE9 0739 SET 5, C
74F5 CBEA 0740 SET 5, D
74F7 CBE8 0741 SET 5, E
74F9 CBE7 0742 SET 5, H
74FB CBE6 0743 SET 5, L
74FD CBF6 0744 SET 6, (HL)
74FF DDCB05F6 0745 SET 6, (IX+IND)
0503 FDCB05F6 0746 SET 6, (IX+IND)
0507 CBF7 0747 SET 6, A
0509 CBF0 0748 SET 6, B
050B CBF1 0749 SET 6, C
050D CBF2 0750 SET 6, D
050F CBF3 0751 SET 6, E
0511 CBF4 0752 SET 6, H
0513 CBF5 0753 SET 6, L
0759 ;
<table>
<thead>
<tr>
<th>ADDR</th>
<th>OBJECT</th>
<th>ST #</th>
<th>SOURCE</th>
<th>STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>'0515</td>
<td>CBFE 0760</td>
<td>SET</td>
<td>7,(HL)</td>
<td></td>
</tr>
<tr>
<td>'0517</td>
<td>DDCB05FE 0761</td>
<td>SET</td>
<td>7,(IX+IND)</td>
<td></td>
</tr>
<tr>
<td>'051B</td>
<td>FDCB05FE 0762</td>
<td>SET</td>
<td>7,(IY+IND)</td>
<td></td>
</tr>
<tr>
<td>'051F</td>
<td>CBFF 0763</td>
<td>SET</td>
<td>7,A</td>
<td></td>
</tr>
<tr>
<td>'0521</td>
<td>CBFB 0764</td>
<td>SET</td>
<td>7,B</td>
<td></td>
</tr>
<tr>
<td>'0523</td>
<td>CBFA 0765</td>
<td>SET</td>
<td>7,C</td>
<td></td>
</tr>
<tr>
<td>'0525</td>
<td>CBFB 0766</td>
<td>SET</td>
<td>7,D</td>
<td></td>
</tr>
<tr>
<td>'0527</td>
<td>CBFC 0767</td>
<td>SET</td>
<td>7,~</td>
<td></td>
</tr>
<tr>
<td>'0529</td>
<td>CBFD 0768</td>
<td>SET</td>
<td>7,H</td>
<td></td>
</tr>
<tr>
<td>'052B</td>
<td>CB26 0770</td>
<td>SLA</td>
<td>(HL)</td>
<td></td>
</tr>
<tr>
<td>'052D</td>
<td>CB26 0771</td>
<td>SLA</td>
<td>(IX+IND)</td>
<td></td>
</tr>
<tr>
<td>'052F</td>
<td>DDCB0526 0772</td>
<td>SLA</td>
<td>(IY+IND)</td>
<td></td>
</tr>
<tr>
<td>'0537</td>
<td>CB27 0774</td>
<td>SLA</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>'0539</td>
<td>CB20 0775</td>
<td>SLA</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>'053B</td>
<td>CB21 0776</td>
<td>SLA</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>'053D</td>
<td>CB22 0777</td>
<td>SLA</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>'053F</td>
<td>CB23 0778</td>
<td>SLA</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>'0541</td>
<td>CB24 0779</td>
<td>SLA</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>'0543</td>
<td>CB25 0780</td>
<td>SLA</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>'0545</td>
<td>CB2E 0782</td>
<td>SRA</td>
<td>(HL)</td>
<td></td>
</tr>
<tr>
<td>'0547</td>
<td>DDCB052E 0783</td>
<td>SRA</td>
<td>(IX+IND)</td>
<td></td>
</tr>
<tr>
<td>'054B</td>
<td>FDCB052E 0784</td>
<td>SRA</td>
<td>(IY+IND)</td>
<td></td>
</tr>
<tr>
<td>'054F</td>
<td>CB2F 0785</td>
<td>SRA</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>'0551</td>
<td>CB28 0786</td>
<td>SRA</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>'0553</td>
<td>CB29 0787</td>
<td>SRA</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>'0555</td>
<td>CB2A 0788</td>
<td>SRA</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>'0557</td>
<td>CB2B 0789</td>
<td>SRA</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>'0559</td>
<td>CB2C 0790</td>
<td>SRA</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>'055B</td>
<td>CB2D 0791</td>
<td>SRA</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>'055D</td>
<td>CB3E 0793</td>
<td>SRL</td>
<td>(HL)</td>
<td></td>
</tr>
<tr>
<td>'055F</td>
<td>DDCB053E 0794</td>
<td>SRL</td>
<td>(IX+IND)</td>
<td></td>
</tr>
<tr>
<td>'0563</td>
<td>FDCB053E 0795</td>
<td>SRL</td>
<td>(IY+IND)</td>
<td></td>
</tr>
<tr>
<td>'0567</td>
<td>CB3F 0796</td>
<td>SRL</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>'0569</td>
<td>CB38 0797</td>
<td>SRL</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>'056B</td>
<td>CB39 0798</td>
<td>SRL</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>'056D</td>
<td>CB3A 0799</td>
<td>SRL</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>'056F</td>
<td>CB3B 0800</td>
<td>SRL</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>'0571</td>
<td>CB3C 0801</td>
<td>SRL</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>'0573</td>
<td>CB3D 0802</td>
<td>SRL</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>'0575</td>
<td>96 0804</td>
<td>SUB</td>
<td>(HL)</td>
<td></td>
</tr>
<tr>
<td>'0576</td>
<td>DD9605 0805</td>
<td>SUB</td>
<td>(IX+IND)</td>
<td></td>
</tr>
<tr>
<td>'0579</td>
<td>FD9605 0806</td>
<td>SUB</td>
<td>(IY+IND)</td>
<td></td>
</tr>
<tr>
<td>'057C</td>
<td>97 0807</td>
<td>SUB</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>'057D</td>
<td>90 0808</td>
<td>SUB</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>'057E</td>
<td>91 0809</td>
<td>SUB</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>'057F</td>
<td>92 0810</td>
<td>SUB</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>'0580</td>
<td>93 0811</td>
<td>SUB</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>'0581</td>
<td>94 0812</td>
<td>SUB</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>'0582</td>
<td>95 0813</td>
<td>SUB</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>'0583</td>
<td>D620 0814</td>
<td>SUB</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>'0585</td>
<td>AE 0816</td>
<td>XOR</td>
<td>(HL)</td>
<td></td>
</tr>
<tr>
<td>'0586</td>
<td>DDAE05 0817</td>
<td>XOR</td>
<td>(IX+IND)</td>
<td></td>
</tr>
</tbody>
</table>
CODE Z80 OP CODE LISTING

SOURCE STATEMENT

END

\texttt{XOR (IY+IND)}

\texttt{XOR A}

\texttt{XOR B}

\texttt{XOR C}

\texttt{XOR D}

\texttt{XOR E}

\texttt{XOR H}

\texttt{XOR L}

\texttt{XOR N}

\texttt{END}

\texttt{RORS=0000}
APPENDIX B

MOSTEK OBJECT OUTPUT DEFINITION
APPENDIX B

MOSTEK OBJECT OUTPUT DEFINITION

B-1. INTRODUCTION

B-2. Each record of an object module begins with a delimiter (colon or dollar sign) and ends with carriage return and line feed. A colon (:) is used for data records and end of file record. A dollar sign ($) is used for records containing relocation information and linking information. An Intel loader will ignore such information and allow loading of non-relocatable, non-linkable programs. All information is in ASCII. Each record is identified by a "type". The type appears in the 8th and 9th bytes of the record and can take the following values:

00 - data record
01 - end-of-file
02 - internal symbol
03 - external symbol
04 - relocation information
05 - module definition

B-3. DATA RECORD FORMAT (TYPE 00)

Byte 1 Colon (:) delimiter.
2-3 Number of binary bytes of data in this record.
The maximum is 32 binary bytes (64 ASCII bytes).
4-5 Most significant byte of the start address of data.
6-7 Least significant byte of start address of data.
8-9 ASCII zeros. This is the "record type" for data.
10- Data bytes.
Last two bytes - Checksum of all bytes except the delimiter, carriage return, and line feed. The
checksum is the negative of the binary sum of all
bytes in the record.

CRLF Carriage return, line feed.

B-4. END-OF-FILE RECORD (TYPE 01)
Byte 1 Colon (:) delimiter.
2-3 ASCII zeros.
4-5 Most significant byte of the transfer address of
the program. This transfer address appears as an
argument in the 'END' Pseudo-op of a program. It
represents the starting execution address of the
program.
6-7 Least significant byte of the transfer address.
8-9 Record type 01.
10-11 Checksum.
CRLF Carriage return, line feed.

B-5. INTERNAL SYMBOL RECORD (TYPE 02)

Byte 1 Dollar sign ($) delimiter.
2-7 Up to 6 ASCII characters of the internal symbol
name. The name is left justified, blank filled.
8-9 Record type 02.
10-13 Address of the internal symbol, most significant
byte first.
14-15 Binary checksum. Note that the ASCII letters of
the symbol are converted to binary before the
checksum is calculated. Binary conversion is done
without regard to errors.
CRLF Carriage return, line feed.

B-6. EXTERNAL SYMBOL RECORD (TYPE 03)

Byte 1 Dollar Sign ($) Delimiter.
CODE Z80Opcode Listing

499  CBCC 0702  SET 1,H
49B  CBBD 0703  SET 1,L

49D  CBDD 0704 ;
49F  DDCB05D6 0705  SET 2,(HL)
4A3  FDCB05D6 0706  SET 2,(IX+IND)
4A7  CBDD 0707  SET 2,(IY+IND)
4A9  CBDD 0708  SET 2,A
4AB  CBDD 0709  SET 2,B
4AD  CBDD 0710  SET 2,C
4AF  CBDD 0711  SET 2,D
4B1  CBDD 0712  SET 2,E
4B3  CBDD 0713  SET 2,H

4B5  CBDD 0714  SET 2,L
4B7  DDCB05DE 0715 ;
4BB  FDCB05DE 0716  SET 3,(HL)
4BF  CBDD 0717  SET 3,(IX+IND)
4C1  CBDD 0718  SET 3,(IY+IND)
4C3  CBDD 0719  SET 3,A
4C5  CBDD 0720  SET 3,B
4C7  CBDD 0721  SET 3,C
4C9  CBDD 0722  SET 3,D
4CB  CBDD 0723  SET 3,E
4CD  CBDD 0724  SET 3,H
4CB  CBDD 0725  SET 3,L

4CF  DDCB05DE 0726 ;
4D3  FDCB05DE 0727  SET 4,(HL)
4D7  CBDD 0728  SET 4,(IX+IND)
4D9  CBDD 0729  SET 4,(IY+IND)
4DB  CBDD 0730  SET 4,A
4DD  CBDD 0731  SET 4,B
4DF  CBDD 0732  SET 4,C
4E1  CBDD 0733  SET 4,D
4E3  CBDD 0734  SET 4,E
4E5  CBDD 0735  SET 4,H
4E7  DDCB05EE 0736  SET 4,L
4EB  FDCB05EE 0737 ;
4EF  CBDD 0738  SET 5,(HL)
4E7  DDCB05EE 0739  SET 5,(IX+IND)
4EB  FDCB05EE 0740  SET 5,(IY+IND)
4EF  CBDD 0741  SET 5,A
4F1  CBDD 0742  SET 5,B
4F3  CBDD 0743  SET 5,C
4F5  CBDD 0744  SET 5,D
4F7  CBDD 0745  SET 5,E
4F9  CBDD 0746  SET 5,F
4FB  CBDD 0747  SET 5,G

4FD  CBDD 0748 ;
4FF  DDCB05F6 0749  SET 6,(HL)
503  FDCB05F6 0750  SET 6,(IX+IND)
507  CBFD 0751  SET 6,(IY+IND)
509  CBFD 0752  SET 6,A
50B  CBFD 0753  SET 6,B
50D  CBFD 0754  SET 6,C
50F  CBFD 0755  SET 6,D
511  CBFD 0756  SET 6,E
513  CBFD 0757  SET 6,F
515  CBFD 0758  SET 6,G
517  CBFD 0759 ;
OPCODE Z80 OPCODE LISTING

<table>
<thead>
<tr>
<th>ADDR</th>
<th>OBJECT</th>
<th>ST #</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
</table>

1. '0515 CBFE 0760 SET 7,(HL)
2. '0517 DDC05FE 0761 SET 7,(IX+IND)
3. '051B FDC05FE 0762 SET 7,(IY+IND)
4. '051F CBFF 0763 SET 7,A
5. '0521 CB8 0764 SET 7,B
6. '0523 CB9 0765 SET 7,C
7. '0525 CBFA 0766 SET 7,D
8. '0527 CBFB 0767 SET 7,E
9. '0529 CBFC 0768 SET 7,H
10. '052B CBFD 0769 SET 7,L
11. '052D CB26 0771 SLA (HL)
12. '052F DDC0526 0772 SLA (IX+IND)
13. '0533 FDC0526 0773 SLA (IY+IND)
14. '0537 CB27 0774 SLA A
15. '0539 CB20 0775 SLA B
16. '053B CB21 0776 SLA C
17. '053D CB22 0777 SLA D
18. '053F CB23 0778 SLA E
19. '0541 CB24 0779 SLA H
20. '0543 CB25 0780 SLA L
21. '0545 CB2E 0782 SRA (HL)
22. '0547 DDC052E 0783 SRA (IX+IND)
23. '054B FDC052E 0784 SRA (IY+IND)
24. '054F CB22F 0785 SRA A
25. '0551 CB28 0786 SRA B
26. '0553 CB29 0787 SRA C
27. '0555 CB2A 0788 SRA D
28. '0557 CB2B 0789 SRA E
29. '0559 CB2C 0790 SRA H
30. '055B CB2D 0791 SRA L
31. '055D CB3E 0793 SRL (HL)
32. '055F DDC053E 0794 SRL (IX+IND)
33. '0563 FDC053E 0795 SRL (IY+IND)
34. '0567 CB3F 0796 SRL A
35. '0569 CB38 0797 SRL B
36. '056B CB39 0798 SRL C
37. '056D CB3A 0799 SRL D
38. '056F CB3B 0800 SRL E
39. '0571 CB3C 0801 SRL H
40. '0573 CB3D 0802 SRL L
41. '0575 96 0803 SUB (HL)
42. '0576 DD9605 0804 SUB (IX+IND)
43. '0579 FD9605 0805 SUB (IY+IND)
44. '057C 97 0806 SUB A
45. '057D 90 0807 SUB B
46. '057E 91 0808 SUB C
47. '057F 92 0809 SUB D
48. '0580 93 0810 SUB E
49. '0581 94 0811 SUB F
50. '0582 95 0812 SUB H
51. '0583 D620 0813 SUB L
52. '0584 96 0814 SUB N
53. '0585 AE 0815 XOR (HL)
54. '0586 DDAE05 0816 XOR (IX+IND)
2-7 Up to 6 ASCII characters of the external symbol name. The name is left justified, blank filled.

8-9 Record type 03.

10-13 Last address which uses the external symbol. This is the start of a link list in the object data records which is described below. The most significant byte is first.

14-15 Binary checksum.

CRLF Carriage return, line feed.

The Assembler outputs the external symbol name and the last address in the program where the symbol is used. The data records which follow contain a link list pointing to all occurrences of that symbol in the object code. This is illustrated in Figure B-1.

1. The external symbol record shows the symbol ('LAB') and the last location in the program which uses the symbol (212AH).

2. The object code at 212AH has a pointer which shows where the previous reference to the external symbol occurred (200FH).

3. This backward reference list continues until a terminator ends the list. This terminator is FFFFH. This method is easy to generate and decode. It has the advantage of reducing the number of bytes of object code needed to define all external references in a program.

B-7. RELOCATING INFORMATION RECORD (TYPE 04)

The addresses in the program which must be relocated are explicitly defined in these records. Up to 16 addresses (64 ASCII characters) may be defined in each record.

Byte 1 Dollar sign ($) delimiter.

2-3 Number of sets of 2 ASCII characters, where 2 sets define an address.
4-7 ASCII zeros.
8-9 Record type 04.
10- Addresses which must be relocated, most significant byte first.
   Last two bytes - Binary checksum.
CRLF Carriage return, line feed.

B-8. MODULE DEFINITION RECORD (TYPE 05)

This record has the name of the module (defined by the 'NAME'
pseudo-op) and a loading flag byte. The flag byte is determined
by the 'PSECT' pseudo-op.

Byte 1  Dollar sign ($) delimiter.
2-7  Name of the module, left justified, blank filled.
8-9  Record type 05.
10-11  Flag byte. When converted to binary, the flag
   byte is defined as follows:
   Bit 0 = 0 For absolute
        = 1 For relocatable
   Bit 1 = 0 For Z80 Data Format
          (LSB First)
        = 1 For 3870 Data Format
          (MSB First)
12-13  Binary checksum.
CRLF Carriage return, line feed.
FIGURE B-1. EXTERNAL SYMBOL LINK LIST

$ \text{LAB}03212A$ defines last reference to external symbol 'LAB'

: 00
: 00
: 00
: 00
: 00 200F

Object Data
APPENDIX C

SCRATCHPAD MEMORY MAP
APPENDIX C

SCRATCHPAD MEMORY MAP

C-1. INTRODUCTION

C-2. The FLP-80DOS operating system makes extensive use of the 256 x 8 scratchpad memory from OFF00H to OFFFFH for system variables. This area is reserved for the operating system and should not be modified by user programs.

C-3. DESCRIPTION OF PARAMETERS

<table>
<thead>
<tr>
<th>MEMORY LOCATION</th>
<th>MNEMONIC</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF00-01</td>
<td>TOR</td>
<td>Top of contiguous RAM Memory (calculated by Monitor)</td>
</tr>
<tr>
<td>FF02-03</td>
<td>BALR</td>
<td>Bottom of Allocated RAM</td>
</tr>
<tr>
<td>FF04-05</td>
<td>CDATE</td>
<td>Current Date</td>
</tr>
<tr>
<td>FF06</td>
<td>Count</td>
<td>DDT variables</td>
</tr>
<tr>
<td>FF07</td>
<td>CMDSV</td>
<td>Disk Controller Command Save location</td>
</tr>
<tr>
<td>FF08</td>
<td>TRK</td>
<td>Disk Controller Variable</td>
</tr>
<tr>
<td>FF09</td>
<td>ERSTAT</td>
<td>Disk Controller Error Status Flag</td>
</tr>
<tr>
<td>FF0A</td>
<td>SCTR</td>
<td>Disk Controller Variable</td>
</tr>
<tr>
<td>FF0B</td>
<td>SCTRSIZE</td>
<td>Disk Controller Variable</td>
</tr>
<tr>
<td>FF0C-FF11</td>
<td></td>
<td>Software Break Point Control</td>
</tr>
<tr>
<td>FF12</td>
<td>FLAG</td>
<td>Debug Flag</td>
</tr>
<tr>
<td>FF13</td>
<td>LONG</td>
<td>Register Map long/short flag</td>
</tr>
<tr>
<td>MEMORY LOCATION</td>
<td>MNEMONIC</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>FF14-FF19</td>
<td>OPR1, OPR2, OPR3</td>
<td>DDT OPERANDS, 124 Byte Buffer for Reset Boot sequence</td>
</tr>
<tr>
<td>FF1A</td>
<td>OPFLG</td>
<td>DDT Operand Flag</td>
</tr>
<tr>
<td>FF1B</td>
<td>NXCHR</td>
<td>DDT Variable</td>
</tr>
<tr>
<td>FF1C</td>
<td>CMD</td>
<td>DDT Variable</td>
</tr>
<tr>
<td>FF1D-1E</td>
<td>MAP</td>
<td>DDT Variable</td>
</tr>
<tr>
<td>FF1F-20</td>
<td>OFFSET</td>
<td>DDT Offset Address</td>
</tr>
<tr>
<td>FF21</td>
<td>EXCTL</td>
<td>DDT Variable</td>
</tr>
<tr>
<td>FF22</td>
<td>FSAVE</td>
<td>DDT Variable</td>
</tr>
<tr>
<td>FF23</td>
<td>BUSY FLG</td>
<td>IOCS busy flag</td>
</tr>
<tr>
<td>FF24</td>
<td>MINFLG</td>
<td>DDT Variable</td>
</tr>
<tr>
<td>FF25</td>
<td>TKST</td>
<td>DDT Variable</td>
</tr>
<tr>
<td>FF26-28</td>
<td>JTASK</td>
<td>Jump to the routine TASK</td>
</tr>
<tr>
<td>FF29-2B</td>
<td>JIOCS</td>
<td>Jump to IOCS</td>
</tr>
<tr>
<td>FF2C-2E</td>
<td></td>
<td>Not used. Reserved for future use</td>
</tr>
<tr>
<td>FF2F-5F</td>
<td></td>
<td>Ram Mnemonic Table*</td>
</tr>
<tr>
<td>FF60-FF8F</td>
<td></td>
<td>Monitor I/O Vector, Reset Boot Vector</td>
</tr>
<tr>
<td>FF90-FF98</td>
<td></td>
<td>AIM-80 Flags</td>
</tr>
<tr>
<td>FFA9</td>
<td></td>
<td>User Stack Origin</td>
</tr>
<tr>
<td>FFAA-FFDF</td>
<td></td>
<td>Monitor and DDT Stack and Breakpoint Area</td>
</tr>
<tr>
<td>FFE0</td>
<td>BRATE</td>
<td>Baud Rate Flag</td>
</tr>
<tr>
<td>FFE1</td>
<td></td>
<td>Not used. Reserved for future use</td>
</tr>
<tr>
<td>FFE2-E3</td>
<td>SPSV</td>
<td>FLP-80 Disk Controller Stack Pointer Save</td>
</tr>
<tr>
<td>FFE4-FFFF</td>
<td></td>
<td>DDT User Register Save Area</td>
</tr>
</tbody>
</table>

*The RAM mnemonic table is initialized by the Monitor. It contains device mnemonics for I/O drivers which are linked into the operating system during the System Generation procedure (See Section 15).*
APPENDIX D

TESTING/DIAGNOSTICS
APPENDIX D

TESTING/DIAGNOSTICS

D-1. INTRODUCTION

D-2. This Appendix contains a description of Software/Firmware troubleshooting techniques and instructions for using the Disk Diagnostic Utility. For problems in areas other than those listed above, consult the appropriate hardware or software manual.

D-3. SOFTWARE/FIRMWARE TROUBLESHOOTING

D-4. Double check the hardware and associated interfaces. Assure that the FLP-80DOS PROMs are in the correct sockets and that the strapping options are correct. Double check connections from the terminal to the serial port. If you suspect a hardware problem perform the diagnostic tests listed in the hardware manuals.

D-5. POWER UP SEQUENCE WITHOUT DISKETTE

1. Assure that no diskettes are in the drives.
2. Power up the system.
3. Depress "carriage return" on your terminal. The system should print the following:
   DSK ERR
   The dot is the DDT-80 prompt.
4. If the above message was not printed and all hardware appears correct, the problem is probably bad PROM's which should be replaced.

D-6. POWER UP SEQUENCE WITH DISKETTE
1. Assure that no diskettes are in the drives.
2. Power up the system.
3. Place a system diskette in the right hand drive (DK0:).
4. Depress "carriage return" on your terminal.
5. The disk should be accessed.
6. If the disk was not accessed, then a controller or disk controller Firmware problem is indicated. Double check the strapping options on the disk drive board. Then proceed to paragraph D-7, DISK CONTROLLER Firmware TEST.
7. If the sign-on message was printed on the terminal but a disk error was indicated (**ERROR OA DISK I/O ERROR), then the diskette is bad and should be replaced.
8. If the following message is displayed on the terminal:
   OS.BIN 255 NOT FOUND
   the operating system binary file is not on the disk in DK0: . The period is the DDT-80 prompt.
9. If the sign-on message and Monitor prompt ('$') appeared on the terminal, proceed to paragraph D-8, MONITOR CHECKOUT.

D-7. DISK CONTROLLER Firmware TEST (only for FLP-80 card. See Hardware manual for other cards).

1. Perform the following sequence.
   .F 0,7F,AA(CR)
   .E EC06(CR)
   SAVE ADR, #SCTRS: 0,1(CR)
   UNIT,TRK,SCTR: 0,A,1(CR)
If a disk error is indicated, then a disk controller problem is indicated for WRITE.
   .F 0,7F,0(CR)
   .E EC09(CR)
   LOAD ADR: 0(CR)
   UNIT,TRK,SCTR: 0,A,1(CR)
.M 0,7F(CR)
Check locations 0-7FH for the pattern AAH. If any discrepancies are found, then failure in the disk controller or disk unit is indicated for READ.
Consult the FLP-80 Operations Manual; MK78560.

D-8. MONITOR CHECKOUT

D-9. A major portion of the system software and hardware can be checked out by performing the following procedure:

$DDT
.F 0,FF,AA(CR)
.Q(CR)
$SAVE 0,FF,TEST(CR)
$DDT(CR)
.F 0,FF,O(CR)
.Q(CR)
$GET TEST(CR)
$DDT(CR)
.M 0,FF(CR)
...

All of the displayed locations should have AA in them. If not, then the Disk Diagnostic should be executed.

D-10. DISK DIAGNOSTIC UTILITY

D-11. PURPOSE

D-12. The Disk Diagnostic Utility allows the user to perform a battery of tests on the disk controller and individual disk drives.

D-13. USER INTERFACE

D-14. The Disk Diagnostic Utility is executed by the user
by entering the following while in the Monitor environment.

$DSKDI A(CR)

D-15. At this point, the program will print a list of available tests and how to call for them. A brief description of the available tests follows.

D-16. DESCRIPTION OF TESTS

1. TEST 20 -- Write and read every sector. This test causes random data to be written to and read from each sector of the diskette in the unit specified. The data is verified as it is read in.

2. TEST 21 -- read every sector. Every sector of the diskette in the unit specified is read. No check of the input data is performed, however format information is checked.

3. TEST 22 -- read ID. This test allows the user to specify a random track and sector address, which the program will then attempt to access.

4. TEST 23 -- random write and read (single drive). Random track and sector addresses are generated and random data is written to the sector at that address. The data is then read and verified.

5. TEST 24 -- random write and read (both drives). This test is the same as the 23 except that both drives are used.

6. TEST 27 -- format diskette. The diskette in the unit specified is formatted in IBM compatible format (Note, this is not to be confused with the PIP format command).

7. TEST 30 -- Memory test. This tests all memory locations from the end of the program to location 7FFFH (32K system).
8. TEST 31 -- fifo test. This test causes writing to and reading from the fifo on the disk controller board.

NOTE -- the removal of disks containing data to be saved from their respective drives is highly recommended immediately after the Disk Diagnostic Utility is loaded. This will prevent accidental overwriting of data during tests 20, 23, 24, and 27.
APPENDIX E

FLP-80DOS ERROR DICTIONARY
APPENDIX E
ERROR MESSAGE/DESCRIPTION

1 INVALID RQST
A request word was specified which is not a valid DOS request.

2 DUPLICATE FILE
An attempt was made to create a directory entry for a file that already exists. Can occur only on create or rename. In the case of OPENW, the file is opened but this error is reported only as a flag.

3 FILE TABLE FULL
An attempt was made to insert another entry in the active file table when it is full. Can occur only on open or create. Up to 7 files can be open at one time.

4 FILE NOT FOUND
The requested file was not found in the directory. Can occur only on open or rename.

5 DIR FULL
There is no more space to insert another directory entry. The directory can have up to 192 entries in it.

6 DISK WRITE PROTECT
Diskette is write protected and an attempt has been made to write on it. Write protection is documented in the Shugart SA800/801 OEM Manual, paragraphs 8.2 and 8.3.

7 I/O TIME OUT
The maximum time allowed for an I/O device to go ready has been exceeded. This is a non-terminating error printed on
the console device by an I/O device handler. In MOSTEK I/O handlers, the message is output every 20 seconds until the I/O device is made ready by the user. The user may terminate the wait loop via RESET or Console Escape (CNTL-C or CNTL-X from the keyboard).

8 FILE NOT OPEN
An attempt was made to close or perform some record operation on a file which had not been opened. Can occur on any operation except initialize, open, or create.

9 READ PAST EOF
An attempt was made to advance the pointer beyond the last record in the file. The error can occur on read next, skip forward, or delete. In the case of delete it points to a null record, with the previous record being the last one.

OA DISK I/O ERR
A disk I/O error occurred during the operation. Data may have been lost. Can occur on any operation except rewind.

OB DISK FULL
Diskette is full and will not allow the allocation of another record. Can occur only on insert.

OC DISK PTR ERR
The pointers read do not agree with the next or previous record. Can occur on any record operation except rewind. Pointer errors occur because a sector is not readable or because an application program has written on a non-initialized disk.

OD DIR MAP ERR
A read or write error occurred during operations involving
the disk directory or sector and track maps. If operation occurred during a close or erase, directory or maps could be destroyed.

OE  FILE ALREADY OPEN
An attempt was made to open or create a file which is currently active.

OF  DISK NOT READY
Can occur on any operation when a diskette is not fully inserted and ready.

10 INITIALIZE
A file is being closed on a disk whose ID is different from the one currently in memory. This can occur if disks are changed during operations without initializing. Can occur only on close and erase. Recovery is by initializing disks before operations begin (INIT command).

11 BAD UNIT
A unit has been specified other than 0/-3 for any command.

12 INVALID RQST
An invalid request code was passed to IOCS in the IOCS vector. The programmer should assure that each request code is one which is described in Section 9 of this manual and that the code is allowed for the selected device.

13 UNIT ALREADY OPEN
An attempt was made to open the same device more than once. This applies to non-file-structured devices and file structured devices. The user should open a device only once. The device must be closed via a CLOSE request before it can be opened again.
14 UNIT NOT OPEN
An I/O operation was attempted on a device which had not been opened. This applies to non-file-structured devices and file structured devices. The user should assure that any device to be accessed is opened for read or write via an OPENR or OPENW request.

15 UNSUPPORTED DEVICE
An operation was attempted on a device whose two character device name was not recognized by the system. The user should assure that an allowable device name is being used. Alternatively, new device names may be added to the system (See Section 7-29). This error occurs at the IOCS level. Allowed device names are shown in Section 9-12.

16 INVALID FMAT
The format specification (FMAT) in the IOCS vector is invalid. The programmer should assure that a valid format specification is used (See Section 9).

17 ALLOC ERR
This error occurs if the user attempts to open more than 16 files or devices requiring physical buffers at the same time.

18 DE-ALLOC ERR
This error occurs during a CLOSE request if the physical buffer number (PBFFR) in the IOCS vector contained an erroneous number, or if the physical buffer had previously been de-allocated.

19 BAD FILE NAME
An invalid file name was specified. A file name may have
up to 6 alphanumeric characters and must start with an alphabetic character.

1A An attempt was made to read from or write into the directory area of the diskette. These operations are not allowed via the FDH, but they are allowed via the Disk Controller Firmware (DCF). Occurrence of this error during normal operation of the software indicates that the diskette has not been initialized or that track and sector pointers on the diskette have been corrupted. The diskette should be reformatted via PIP's FORMAT command.

1B BAD UNIT, TRK, OR SCTR Controller has received invalid drive number, or sector and track out of normal range.

1C SEEK ERR Controller not able to locate track during seek, read, or write operation.

1D SCTR NOT FOUND Sector address marks not readable.

1E CRC ERR Incorrect data has been flagged by CRC check during reading.

1F DATA LOST Hardware problem causing data overrun in reading or writing.

20 INVALID DEVICE SPEC An I/O device was specified in a command which is not al-
allowed in the system. The user should assure that an allowable device mnemonic is being used. See Section 9-12. Alternatively, new mnemonics may be added to the system (See Section 15-6). This error occurs at the system program level and is used in PIP. The Append command, for example, is only supported on the disk device DK.

21 INCOMPATIBLE EXTENSIONS
An attempt was made to perform some PIP command on files whose extensions are not compatible. Specifically, binary files (extension 'BIN') cannot be intermixed with non-binary files. The user should assure that binary file operations are associated only with binary files. The PIP commands Rename and Copy will generate this error if the extensions are incompatible.

22 BINARY EXTENSION NOT ALLOWED
Binary files (extension 'BIN') cannot be appended. This error is generated by the PIP Append command.

23 RESERVED FOR FUTURE USE

24 I/O FILES EQUAL
An input and output file in a PIP copy command were the same file. The user should assure that any file is not used for both input and output in PIP.

25-2B Reserved for future use.

MONITOR ERROR MESSAGES

2C INVALID LUN
The Logical Unit Number (LUN) specified in a Monitor com-
mand was not allowed. LUN's may be 0/-FEH. LUN FFH is reserved for applications in which the LUN is not to be redirected.

2D  SAVE TOO LARGE
The amount of memory to be saved as a binary file via the Monitor SAVE command exceeded the maximum allowable, which is 256x124=31744 bytes. The user should assure that the maximum size of the area to be saved does not exceed 31744 bytes.

2E  INVALID EXTENSION
A valid extension consists of one to three alphanumeric digits.

2F  ASSIGN TABLE FULL
Too many redirects were attempted via the Monitor ASSIGN command. The maximum number of allowed redirects is 6. The user should eliminate some of the redirects via the Monitor CLEAR command.

30  MEMORY FAULT LOC
A memory location was found to be faulty. The address is printed out.

31  CHECKSUM
A checksum error was encountered by the LINKER within an object module. The user should regenerate the object module and then try linking it.

32  GLOBAL DOUBLE DEF
The LINKER generates this error when a global symbol is multiply defined in two different modules.
33-34 RESERVED FOR FUTURE USE

35 MODULE SEQUENCE ERROR
During use of the LINKER, specification of modules to be linked did not match during both passes.

36 NOT ENOUGH MEMORY AVAILABLE
During use of the LINKER, the largest object module to be linked exceeded the available memory.

37-3E Reserved for future use.

ASSEMBLER ERROR MESSAGES

3F BAD RELOCATABLE USAGE
A relocatable value was used in an 8-bit operand. The user should assure that relocatable quantities are used only for 16-bit operand values (addresses), or the PSECT ABS pseudo-op should be used.

40 BAD LABEL
An invalid label was specified. A label may consist of any printable ASCII characters except '( ) * + , - = . / : ; or space. In addition, the first character cannot be a number. A label may start in any column if followed by a colon. It does not require a colon if started in column one.

41 BAD OPCODE
An invalid Z80 opcode or pseudo-op was specified. This error will also occur for a label which starts beyond column 1 and is not followed by a colon.
42 BAD OPERAND
An invalid operand or combination of operands was specified for a given opcode.

43 BAD SYNTAX
The specification of an operand was invalid.

44 UNDEF SYMBOL
A symbol was used in an operand which was not defined in the program, either locally or as an external symbol.

45 MULTIPLE DEF
A symbol was defined more than once in the same program.

46 MULTIPLE PSECT USAGE
A PSECT pseudo-op was used more than once or was defined after the first code producing opcode. The PSECT pseudo-op should be used only once at the beginning of a program.

47 SYMBOL TABLE FULL
The symbol table of the Assembler is full and will accept no more symbols. The user should reduce the number of symbols in his program or break the program up into one or more linkable modules.

48 BAD EXTERNAL USAGE
An external symbol was used in an expression or as the operand of an 'EQU' or 'DEFL' pseudo-op. The user should assure that an external symbol is not used in these situations.

49 MACROS NOT ALLOWED WITH THIS VERSION
The current version of the Assembler does not support macros.
4A  UNBALANCED QUOTES
An uneven number of quote characters (') occurred in an operand or operands.

4B  LABEL REQUIRED
A label was not used on an 'EQU' or 'DEFL' pseudo-op. Each 'EQU' or 'DEFL' pseudo-op must have a label associated with it.

4C  OVERFLOW IN EXPRESSION
In evaluating an expression, the value of the expression exceeded 65536 (0/FFFFH). The user should check the expression for validity. Alternatively, the .RES. operation may be used to ignore the overflow condition and only the least significant 16 bits of the expression will be used.

4D  OPERAND OUT OF RANGE
The final value of an operand was found to be out of the range allowed for the given opcode. For example, the valid range of the JR operand is -126 through +129.

4E  BAD DIGIT
An invalid digit was found in a number.

4F  BAD OPERATOR
An invalid operator was found in an expression.

50  BAD SYMBOL TABLE LIMITS
The available RAM is not sufficient for the Assembler symbol table. The user should assure that 'BALR' (Bottom of Allocated RAM) is correct for his configuration. 'BALR' is defined in locations FF02H and FF03H. All system routines exist above BALR and must not be overwritten. See SYSGEN,
51 INPUT TRUNCATED
The input statement exceeded 80 characters in length. This is the system input limit for all FLP-80DOS Software.

52 MULTIPLE NAME
The 'NAME' pseudo-op was used more than once in the same program. The user should use the NAME pseudo-op only once per source module.

53 The 'INCLUDE' pseudo-op was nested. The user should assure that the 'INCLUDE' pseudo-op is not used in the body of an included module.

54 The expression evaluator stack reached its limit. The user should reduce the complexity of the expression in the statement with caused the error.

55 The cross reference table became too large. This is a warning message indicating that not all cross references will be output in the cross reference listing.
APPENDIX F

SYSTEM LINKAGES
(SYSLNK)
F-1. INTRODUCTION

F-2. FLP-80DOS system routines are documented in Section 13 of this manual. The linkage addresses for these routines are documented here, and they are set up in a file on the system diskette called SYSLNK. SYSLNK contains linkages for all system routines resident in PROM (E000-EFFF). It also contains the variable JTASK which is the linkage to the RAM resident system routines in the operating system (See Section 13), and the linkage to JIOCS for calls to IOCS.

F-3. Any program using a system routine should declare that routine name as an external global symbol.

EXAMPLE
GLOBAL RDCHR
GLOBAL WRCHR
GLOBAL JTASK

F-4. When the user program is loaded or linked, the SYSLNK.OBJ file should be linked in with it to resolve these external references.

EXAMPLE
$LINK MYFILE,SYSLNK(CR)

F-5. The source and object files SYSLNK.SRC and SYSLNK.OBJ are both included on FLP-80DOS system diskettes.
0002 NAME    SYSLNK
0003 PSECT    ABS
0004 ;************************************************
0005 ;* SYSTEM LINKAGES FOR FLP-80DOS V2.0  *
0006 ;* ID: SYSLNK VERSION 2.0  5/22/78  *
0007 ;* PROGRAMMER: JOHN BATES  *
0008 ;* DESCRIPTION:  *
0009 ;* THIS IS AN ABSOLUTE LINK BLOCK FOR  *
0010 ;* FLP-80DOS SYSTEM ROUTINES. MOST OF  *
0011 ;* THESE ROUTINES ARE RESIDENT IN THE  *
0012 ;* SYSTEM FIRMWARE AREA (E000-EFFF).  *
0013 ;* ADDITIONAL RAM RESIDENT SYSTEM ROUTINES  *
0014 ;* IN OS.BIN[255] MAY BE ACCESSED THROUGH  *
0015 ;* LINKAGES IN SCRATCH PAD RAM (E.G.TASK).  *
0016 ;* EACH SYSTEM SUBROUTINE IS IDENTIFIED BY  *
0017 ;* ITS ASSIGNED NAME AND ITS ASSOCIATED  *
0018 ;* STARTING ADDRESS. THIS SOURCE MODULE  *
0019 ;* SHOULD BE ASSEMBLED SO ITS OBJECT MODULE  *
0020 ;* MAY BE LINKED WITH USER PROGRAMS OR  *
0021 ;* SYSTEM PROGRAMS (E.G. PIP).  *
0022 ;************************************************
0023 ;* SYSTEM SUBROUTINES IN FIRMWARE SPACE (E000-EFFF)  *
0024 ;************************************************
0025 >E56A GLOBAL AORN
0026 0032 AORN EQU OE56AH
0027 0033 GLOBAL ASBIN
0028 >E583 GLOBAL EQU
0029 0034 ASBIN EQU OE583H
0030 0035 GLOBAL CRLF
0031 >E59C GLOBAL EQU
0032 CRLF EQU OE59CH
0033 0035 GLOBAL ECHO
0034 >E597 GLOBAL EQU
0035 ECHO EQU OE597H
0036 0039 GLOBAL EH
0037 >E003 GLOBAL EQU
0038 EH EQU OE003H ;ERROR HANDLER
0039 0040 GLOBAL FATAL
0041 >EC23 GLOBAL EQU
0042 FATAL EQU OE23H ;FATAL ERROR EXIT
0043 0044 GLOBAL FLOPPY
0044 >EC00 GLOBAL EQU
0045 FLOPPY EQU OE00H ;FLOPPY CONTROLLER
0045 0046 GLOBAL LOADER
0046 >EC03 GLOBAL EQU
0047 LOADER EQU OE03H ;LINKED FILE LOADER
0047 0047 GLOBAL MINDIS
0048 >E3B3 GLOBAL EQU
0048 MINDIS EQU OE3B3H ;DISABLE MINIMAL LISTNER
0049 0049 GLOBAL MINEN
0050 >E534 GLOBAL EQU
0050 MINEN EQU OE534H ;ENABLE MINIMAL LISTNER
0051 0051 GLOBAL PACC
0052 >E58B GLOBAL EQU
0052 PACC EQU OE58BH
0053 0053 GLOBAL PADDO
0054 >E61C GLOBAL EQU
0054 PADDO EQU OE61CH
0055 0055 GLOBAL PASP
0056 >E5AA GLOBAL EQU
0056 PASP EQU OE5AAH ;PRINT ACC AND SPACE
0057 0057 GLOBAL PTXT
0058 >E3C7 GLOBAL EQU
0058 PTXT EQU OE3C7H
0059 0059 GLOBAL RDCHR
>E522 0060 RDCHR EQU OE522H
   0061 GLOBAL RENTRY
>E11D 0062 RENTRY EQU OE11DH ;DDT-80 RENTRY POINT
   0063 GLOBAL ENTRY
>E066 0064 ENTRY EQU OE066H ;BREAK PT REENTRY
   0065 GLOBAL RUN
>EFE1 0066 RUN EQU OEFE1H ;EXIT FOR IMPLIED RUN CMD
   0067 GLOBAL SCAN
>E414 0068 SCAN EQU OE414H
   0069 GLOBAL SPACE
>E5A5 0070 SPACE EQU OE5A5H
   0071 GLOBAL SRCHU
>E547 0072 SRCHU EQU OE547H
   0073 GLOBAL WRCHR
>E527 0074 WRCHR EQU OE527H
   0075 GLOBAL REBOOT
>E006 0076 REBOOT EQU OE006H
   0077 ; SCRATCH PAD VARIABLES
   0078 ;
   0080 GLOBAL ERSTAT
>FF09 0081 ERSTAT EQU OFF09H ;ERROR STATUS
   0082 GLOBAL JTASK
>FF26 0083 JTASK EQU OFF26H ;JUMP TO TASK
   0084 GLOBAL JIOCS
>FF29 0085 JIOCS EQU OFF29H ;JUMP TO IOCS
   0086 ;
   0087 END

ERRORS=0000
APPENDIX G

DISK RECOVERY UTILITY
APPENDIX G

DISK RECOVERY UTILITY

G-1. INTRODUCTION

G-2. The Disk Recovery Utility may be used to recover ASCII text files that are inaccessible to other programs due to some form of error within the file. Typically, the Disk Recovery Utility would be used to recover files that have experienced a pointer error.

G-3. USER INTERFACE

G-4. The file to be recovered must be on the diskette currently in unit DK1:. As its contents are recovered, they are copied to a file on unit DKO: (the file is automatically created by the Disk Recovery Utility).

G-5. The Disk Recovery Utility is invoked by entering the following from the console while in the monitor environment:

   $DSKREC DK1:sfilename TO DKO:dfilename (CR)

G-6. The parameter 'sfilename' is the name of the input (source) file that is to be recovered. The parameter 'dfilename' is the name of the output (destination) file that is to receive the recovered data. This is optional and defaults to the name of the source file.

G-7. After the above is entered by the user, the program attempts to recover the source file. One or more of the following messages may then be printed.
G-8. MESSAGES

G-9. Error messages that may be printed by the Disk Recovery Utility are listed in Appendix E (FLP-80DOS ERROR MESSAGES/DESCRIPTION).

G-10. The following messages indicates normal termination of the Disk Recovery Utility:

DSKREC> FILE VERIFIED--NO ERRORS

This indicates that the file was recovered and that no errors of any sort were detected.

DSKREC> FILE RECOVERED--POSSIBLE ERRORS

The source file has been partially recovered. An error was detected in the file and therefore some data may have been lost.

G-11. When some form of error is detected in a file being recovered, the Disk Recovery Utility inserts a message into the recovered copy of the file at the point were the error occurred. This message is highly visible and enables the user to quickly locate the area in the recovered file at which data may be garbled and/or lost. This message should be deleted from the recovered copy of the file when the user has verified the data in the area of the message. The message will appear as follows:

*************************************
* I/O OR POINTER ERROR OCCURRED HERE*
*************************************

G-12. METHOD OF OPERATION

G-13. The procedure used by the Disk Recovery Utility to recover disk files is described below.

G-14. The directory entry for the source (input) file is
obtained from the disk file directory. Within this entry the addresses of the first and last sectors in the source file are found. These are copied and saved. At this point the destination file is created on unit DKO:

G-15. The source file is then read and copied to the destination file sector by sector until either an end of file or error condition is detected. If an end of file condition is detected, the output file is closed and a message is printed on the console indicating that no errors were detected. The program returns control to the Monitor. If an error condition is detected, the program retries the operation 50 times. If the error is still present, the program then writes a message to the destination file that will aid the user in locating the area in the file where data is suspect.

G-16. The program then begins reading sectors backward starting at the last sector in the file (the address was saved previously). No sectors are written to the destination file during this pass. Reading continues until an error condition is detected and 50 retries are performed.

G-17. Sectors are then read forward, beginning at the last sector correctly read (in G-16, above). These sectors are written to the destination file. Reading and copying continues until the end of the source file is detected, at which time a message is printed on the console indicating that errors have been detected. The program then returns control to the Monitor.