NOTICE

The information in this document is provided for reference only. Pro-Log does not assume any liability arising out of the application or use of the information or products described herein.

This document may contain or reference information and products protected by copyrights or patents and does not convey any license under the patent rights of Pro-Log, nor the rights of others.

Printed in U.S.A. Copyright © 1981 by Pro-Log Corporation, Monterey, CA 93940. All rights reserved. However, any part of this document may be reproduced with Pro-Log Corporation cited as the source.
# 7803 USER'S MANUAL

## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>SECTION</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRODUCT OVERVIEW</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>THE STD BUS</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>STD BUS Summary</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7803 Pin Utilization</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Control Bus Signal Table</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Processor Status Signals</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dynamic RAM Control</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7803 SPECIFICATIONS</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Power Requirements</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Drive Capability and Loading</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Clock Generator</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Timing Specifications and Waveforms</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mechanical</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Environmental</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Z80 ARCHITECTURE AND INSTRUCTION SET</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Z80 Programming Model</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Program Compatibility with 8080, 8085</td>
<td></td>
</tr>
<tr>
<td></td>
<td>STD Instruction Mnemonics</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Z80 Instruction Set</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Interrupts</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>PROGRAM INSTRUCTION TIMING</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>Introduction</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Machine Cycles</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WAIT States</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DMA Mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Instruction Timing Table</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Instruction Timing Example</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>MEMORY AND I/O MAPPING AND CONTROL</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>Memory Addressing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12K-Byte Onboard Memory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Input/Output Port Addressing</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>PROGRAM AND HARDWARE DEBUGGING</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>Microprocessor Logic State Analysis</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Instruction Diagnostic Tables</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M824 System Analyzer</td>
<td></td>
</tr>
<tr>
<td>APPENDIX A</td>
<td>7803 STRAPPING OPTIONS</td>
<td>52</td>
</tr>
<tr>
<td>APPENDIX B</td>
<td>SCHEMATIC AND ASSEMBLY DIAGRAMS</td>
<td>55</td>
</tr>
</tbody>
</table>
7803
Z-80 PROCESSOR CARD

This card combines a buffered and fully expandable Z-80 microprocessor with onboard RAM and PROM sockets.

The 7803 includes 1K byte of RAM with sockets for up to 4K bytes and sockets for up to 8K bytes of ROM or EPROM. An STD BUS system using the 7803 card can be expanded to the full Z-80 memory and I/O capability. The 7803 STD BUS interface may be disabled for DMA applications.

FEATURES
- Z-80 Processor
- 4096 bytes RAM capacity (2114)
- 1024 bytes RAM included
- 8192 bytes ROM capacity onboard (2716 EPROM)
- 3 State Address, Data, Control Bus
- Crystal controlled 400 ns clock
- Power-on reset or pushbutton reset input
- Dynamic RAM refresh control
- All IC's socketed
- Single +5V operation
- Use Pro-Log D1004 1Kx8 memories (two 2114L's)

FIGURE 1: 7803 BLOCK DIAGRAM
SECTION TWO - THE STD BUS

The STD BUS standardizes the physical and electrical aspects of modular 8-bit microprocessor card systems, providing a dedicated, orderly interconnect scheme. The STD BUS is dedicated to internal communication and power distribution between cards, with all external communication made via I/O connectors which are suitable to the application. The standardized pinout and 56-pin connector lends itself to a bussed motherboard that allows any card to work in any slot.

As the system processor and primary system control card, the 7803 is responsible for maintaining the signal functionality defined by the STD BUS standard.

A complete copy of the STD BUS standard is contained in the SERIES 7000 STD BUS TECHNICAL MANUAL, available from Pro Log Corporation, 2411 Garden Road, Monterey, California 93940.

STD BUS Summary

The 56-pin STD BUS is organized into five functional groups of backplane signals:

1. Logic Power Bus pins 1-6
2. Data Bus pins 7-14
3. Address Bus pins 15-30
4. Control Bus pins 31-52
5. Auxiliary Power pins 53-56

Figure 2 shows the organization and pinout of the STD BUS with mnemonic function and signal flow relative to the 7803 Processor card:

<table>
<thead>
<tr>
<th>COMPONENT SIDE</th>
<th>CIRCUIT SIDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN</td>
<td>MNEMONIC</td>
</tr>
<tr>
<td>LOGIC POWER BUS</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>+5V</td>
</tr>
<tr>
<td>3</td>
<td>GND</td>
</tr>
<tr>
<td>5</td>
<td>-5V</td>
</tr>
<tr>
<td>DATA BUS</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>D3</td>
</tr>
<tr>
<td>9</td>
<td>D2</td>
</tr>
<tr>
<td>11</td>
<td>D1</td>
</tr>
<tr>
<td>13</td>
<td>D0</td>
</tr>
<tr>
<td>ADDRESS BUS</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>A07</td>
</tr>
<tr>
<td>17</td>
<td>A6</td>
</tr>
<tr>
<td>19</td>
<td>A5</td>
</tr>
<tr>
<td>21</td>
<td>A4</td>
</tr>
<tr>
<td>23</td>
<td>A3</td>
</tr>
<tr>
<td>25</td>
<td>A2</td>
</tr>
<tr>
<td>27</td>
<td>A1</td>
</tr>
<tr>
<td>29</td>
<td>A0</td>
</tr>
<tr>
<td>CONTROL BUS</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>WR*</td>
</tr>
<tr>
<td>33</td>
<td>IORD*</td>
</tr>
<tr>
<td>35</td>
<td>IOEXP*</td>
</tr>
<tr>
<td>37</td>
<td>REFRESH*</td>
</tr>
<tr>
<td>39</td>
<td>STATUS 1*</td>
</tr>
<tr>
<td>41</td>
<td>BUSAK*</td>
</tr>
<tr>
<td>43</td>
<td>INTRACK*</td>
</tr>
<tr>
<td>45</td>
<td>WAITRO*</td>
</tr>
<tr>
<td>47</td>
<td>SYRESET*</td>
</tr>
<tr>
<td>49</td>
<td>CLOCK*</td>
</tr>
<tr>
<td>51</td>
<td>PCO</td>
</tr>
<tr>
<td>POWER BUS</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>AUX GND</td>
</tr>
<tr>
<td>55</td>
<td>AUX +V</td>
</tr>
</tbody>
</table>

*Low Level Active Indicator
STD BUS Pin Utilization by 7803

Since the STD BUS standard does not specify timing or require that all available pins be used, the timing and signal allocation assumes many of the characteristics of the microprocessor type used. The timing characteristics of the 7803 are those of its Z80 microprocessor, with LSTTL buffering added to enhance the card's drive capability.

The allocation of STD BUS lines for the 7803 is given below.

1. **Logic Power Bus:** +5V (pins 1, 2) and Logic Ground (Pins 3, 4) supply operating power to the 7803. Pins 5 and 6 are open.

2. **Data Bus:** Pins 7 through 14 form an 8-bit bidirectional 3-state data bus as shown in Figure 2. High level active data flows between the 7803 and its peripheral cards over this bus. When the 7803 fetches data from its onboard memory sockets, this data also appears on the STD Data Bus. Except during Direct Memory Access (DMA) operations, the 7803 controls the direction of data flow with its MEMRQ*, IORQ*, RD*, WR*, and INTAK* control signal outputs. Peripheral cards are required to release the data bus to the high impedance state except when addressed and directed to drive the data bus by the 7803.

The 7803 releases the Data Bus when BUSAK* is active in response to BUSRQ*, as in DMA operations.

3. **Address Bus:** Pins 15 through 30 form a 16-bit 3-state address bus as shown in Figure 3. The 7803 drives high level active 16-bit memory addresses over these lines, and 8-bit I/O port addresses over the eight low-order address lines (A0 through A7 on pins 15, 17, 19, 21, 23, 25, 27 and 29).

The 7803 releases the Address Bus when BUSAK* is active in response to BUSRQ*, as in DMA operations.

4. **Control Bus:** Pins 31 through 52 provide control signals for memory, I/O, interrupt, and fundamental system operations. Figure 3 summarizes these signals and shows how they are derived from Z80 signals.

The 7803 releases the Control Bus during BUSAK* in response to BUSRQ*, except for the following output signals: MEMEX*, I0EXP*, BUSAK*, PC0, CLOCK*.

5. **Auxiliary Power Bus:** Pins 53 through 56 are not used by the 7803 and are electrically open.

The 7803 meets all of the signal requirements of the STD BUS standard. Detailed timing information and specifications are in Section 5.
<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>PIN</th>
<th>IN/OUT</th>
<th>FUNCTION</th>
<th>HOW DERIVED—Z80 NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>WR*</td>
<td>31</td>
<td>Out#</td>
<td>Write to memory or I/O</td>
<td>[WR*]</td>
</tr>
<tr>
<td>RD*</td>
<td>32</td>
<td>Out#</td>
<td>Read from memory or I/O</td>
<td>[RD*]</td>
</tr>
<tr>
<td>IORQ*</td>
<td>33</td>
<td>Out#</td>
<td>AQ-A7 hold valid I/O address</td>
<td>[IORQ*]</td>
</tr>
<tr>
<td>MEMRQ*</td>
<td>34</td>
<td>Out#</td>
<td>AQ-A15 hold valid memory address</td>
<td>[MEMRQ*]</td>
</tr>
<tr>
<td>IOEXP*</td>
<td>35</td>
<td>Out</td>
<td>1/O expansion control</td>
<td>User-removable ground</td>
</tr>
<tr>
<td>MEMEX*</td>
<td>36</td>
<td>Out</td>
<td>Memory expansion control</td>
<td>User-removable ground</td>
</tr>
<tr>
<td>REFRESH*</td>
<td>37</td>
<td>Out#</td>
<td>Dynamic RAM refresh control</td>
<td>[RFSH*]</td>
</tr>
<tr>
<td>MCSYNC*</td>
<td>38</td>
<td>Out#</td>
<td>One pulse per machine cycle</td>
<td>[RD*]+[WR*]+<a href="M1*">IORQ*</a></td>
</tr>
<tr>
<td>STATUS 1*</td>
<td>39</td>
<td>Out#</td>
<td>Active during opcode fetch</td>
<td>[M1*]</td>
</tr>
<tr>
<td>STATUS 0*</td>
<td>40</td>
<td>-</td>
<td>(Not used)</td>
<td>Electrically open</td>
</tr>
<tr>
<td>BUSAK*</td>
<td>41</td>
<td>Out</td>
<td>Acknowledges BUSRQ*</td>
<td>[BUSAK*]</td>
</tr>
<tr>
<td>BUSRQ*</td>
<td>42</td>
<td>In</td>
<td>Bus request for DMA; synchronous processor halt and 3-state driver disable</td>
<td>[BUSRQ*]</td>
</tr>
<tr>
<td>INTAK*</td>
<td>43</td>
<td>Out#</td>
<td>Acknowledges INTRQ* and replaces [RD*]MEMRQ* to read interrupt vector</td>
<td><a href="M1*">IORQ*</a></td>
</tr>
<tr>
<td>INTRQ*</td>
<td>44</td>
<td>In</td>
<td>Maskable interrupt request</td>
<td>[INT*]</td>
</tr>
<tr>
<td>WAIT#</td>
<td>45</td>
<td>In</td>
<td>Synchronous processor halt</td>
<td>[WAIT*]</td>
</tr>
<tr>
<td>NMIRQ*</td>
<td>46</td>
<td>In</td>
<td>Nonmaskable interrupt request</td>
<td>[NMI*]</td>
</tr>
<tr>
<td>SYSRESET*</td>
<td>47</td>
<td>Out#</td>
<td>System power-on and pushbutton reset output</td>
<td>Onboard one-shot</td>
</tr>
<tr>
<td>PBRESET*</td>
<td>48</td>
<td>In</td>
<td>Pushbutton reset input</td>
<td></td>
</tr>
<tr>
<td>CLOCK*</td>
<td>49</td>
<td>Out</td>
<td>Time state clock (1/2 crystal frequency)</td>
<td>Onboard oscillator</td>
</tr>
<tr>
<td>CNTRL*</td>
<td>50</td>
<td>In</td>
<td>External clock input (2 times desired time state frequency)</td>
<td></td>
</tr>
<tr>
<td>PC1/PCO</td>
<td>52/51</td>
<td>In/Out</td>
<td>Priority chain</td>
<td>PC1 shorted to PC0; no other 7803 connection</td>
</tr>
</tbody>
</table>

* Low level active
# Output buffer disabled when BUSAK* active
[] Denotes equivalent Z80 signal name

FIGURE 3 : 7803 CONTROL BUS SIGNALS
MCSYNC* and STATUS 1* signals provide status information which is peculiar to the Z80 microprocessor. These signals are useful for displaying processor status in logic signal analyzers, and can be used to drive Z80 peripheral chips and systems designed to work with the Z80 specifically. The use of these signals is not recommended in systems where microprocessor device-type independence is a design goal.

MCSYNC* is obtained by ORing the read, write, and interrupt acknowledge signals. Thus MCSYNC* occurs once in each machine cycle (Section 3), and can be used to allow a logic signal analyzer to select a specific cycle within a multi-cycle instruction for analysis. The timing of MCSYNC* varies according to machine cycle type.

STATUS 1* is equivalent to the Z80's M1 signal, which denotes the opcode fetch or interrupt acknowledge cycle (M1 is ANDed with IORQ* internally to produce INTAK, and externally with MEMRQ* to denote opcode fetch). Note that the Z80 has both 1-byte and 2-byte opcodes (2-byte opcodes are identified by a first byte equal to CB, DD, ED, or FD hexadecimal). Accordingly, the processor asserts STATUS 1* in each opcode byte, or twice per instruction cycle for these instructions.

Dynamic RAM Control: REFRESH*

The Z80 microprocessor chip is specifically designed for refreshing standard 16-pin dynamic RAM chips with multiplexed address lines and 4K x 1 or 16K x 1 internal organization. These devices can be refreshed transparently during the opcode fetch memory cycle without complex processor synchronization circuitry and without delaying processor instruction execution time.

The REFRESH* output signal occurs during T3 and T4 of the opcode fetch cycle (fig. 8) and is used to indicate that a memory refresh address is present on the Address Bus. The address is composed of a presettable, autounting 7-bit address (A0-A6) which is the lower seven bits of the Z80's R (Refresh) Register, and an eighth bit (A7) which is the R Register's most significant bit and is program-settable in the high or low state.

For more information on dynamic RAM refreshing, refer to the following publications:

Interfacing 16 Pin Dynamic RAMs to the Z80A Microprocessor
available from Zilog, 10460 Bubb Road, Cupertino, CA 95014

Z80 Dynamic RAM Interfacing Techniques
available from Mostek, 1215 W. Crosby Rd., Carrollton, TX 75006
SECTION 3 - 7803 SPECIFICATIONS

Power Requirements

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
<th>MIN</th>
<th>MAX</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vcc (Note 1)</td>
<td>4.75</td>
<td>5.00</td>
<td>5.25</td>
<td>0</td>
<td>5.50</td>
<td>Volts</td>
</tr>
<tr>
<td>Icc (Note 2)</td>
<td>1.15</td>
<td>1.65</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Ampere</td>
</tr>
</tbody>
</table>

FIGURE 4: 7803 POWER SUPPLY SPECIFICATION

NOTES: 1. In order to guarantee correct operation, the following power supply considerations apply:
   a. Vcc rise must be monotonic, rising from +0.50 Volt to +4.75 Volts in 10 ms or less.
   b. If Vcc drops below +4.75 Volts at any time it must be reduced to less than +0.50 Volt before restoration to the specified operating range.

2. Icc specification assumes that all EPROM and RAM sockets on the 7803 are loaded. Subtract 75 mA per 2716 EPROM and 50 mA per 2114L RAM for each device not used.

The 2114L devices require 10 milliseconds minimum after initial power-on for stabilization of internal bias oscillators. The 7803's power-on reset one-shot provides adequate stabilization delay only if Vcc risetime is less than 10 milliseconds.

Drive Capability and Loading

The 7803's STD BUS Edge Connector Pin List (Figure 5) gives input loading and output drive capability in LSTTL loads as defined by the SERIES 7000 TECHNICAL MANUAL.

In general, input lines and disabled 3-state outputs present 5 LSTTL input loads maximum (one LSTTL or MOS input plus 4.7K pullup resistor). Output lines can drive a minimum of 50 LSTTL loads. Pins which are unspecified in Figure 5 are electrically open.

Exceptions to the general loading rules are:
   a. PBRESET* input, which is 15 LSTTL loads.
   b. CLOCK* output, which can drive 10 LSTTL loads
   c. PCI and PCO, which are connected together but to nothing else on the 7803.
### STD/7803 EDGE CONNECTOR PIN LIST

<table>
<thead>
<tr>
<th>PIN NUMBER</th>
<th>OUTPUT (LSTTL DRIVE)</th>
<th>INPUT (LSTTL LOADS)</th>
<th>MNEMONIC</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IN</td>
<td>+5 VOLTS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>IN</td>
<td>GROUND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>IN</td>
<td>-5V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>IN</td>
<td>D7</td>
<td>5</td>
<td>60</td>
</tr>
<tr>
<td>5</td>
<td>IN</td>
<td>D6</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>6</td>
<td>IN</td>
<td>D5</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>7</td>
<td>IN</td>
<td>D4</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>8</td>
<td>IN</td>
<td>A15</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>9</td>
<td>IN</td>
<td>A14</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>10</td>
<td>IN</td>
<td>A13</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>11</td>
<td>IN</td>
<td>A12</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>12</td>
<td>IN</td>
<td>A11</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>13</td>
<td>IN</td>
<td>A10</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>14</td>
<td>IN</td>
<td>A9</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>15</td>
<td>IN</td>
<td>A8</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>16</td>
<td>IN</td>
<td>RD*</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>17</td>
<td>IN</td>
<td>MEMREQ*</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>18</td>
<td>IN</td>
<td>MEMEX* (GROUND)</td>
<td>OUT</td>
<td>36</td>
</tr>
<tr>
<td>19</td>
<td>IN</td>
<td>MCSYNC*</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>20</td>
<td>IN</td>
<td>STATUS 0*</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>21</td>
<td>IN</td>
<td>BUSRQ*</td>
<td>5</td>
<td>42</td>
</tr>
<tr>
<td>22</td>
<td>IN</td>
<td>INTRQ*</td>
<td>5</td>
<td>44</td>
</tr>
<tr>
<td>23</td>
<td>IN</td>
<td>NMIROQ*</td>
<td>5</td>
<td>46</td>
</tr>
<tr>
<td>24</td>
<td>IN</td>
<td>PBRESET*</td>
<td>5</td>
<td>48</td>
</tr>
<tr>
<td>25</td>
<td>IN</td>
<td>CNTRL* - EXT CLK IN</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>26</td>
<td>IN</td>
<td>PCI</td>
<td>IN</td>
<td>52</td>
</tr>
<tr>
<td>27</td>
<td>IN</td>
<td>AUX GND</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>IN</td>
<td>AUX -V</td>
<td>56</td>
<td></td>
</tr>
</tbody>
</table>

*Designates Active Low Level Logic
Clock Generator

The 7803's clock oscillator serves as the primary timing element in a 7803-based system. The oscillator's output is divided by two to drive the Z80 microprocessor, producing the time state clock. The time state clock's period is the shortest program-related period of interest in the system. Instruction execution times are computed as whole multiples of the time state clock period (Section 5).

The 7803 is shipped with a crystal installed which sets the system's time state period. If desired, the user can substitute a slower crystal or replace the crystal with a TTL-compatible clock signal generated elsewhere. Details of this option are given in Appendix A.

<table>
<thead>
<tr>
<th>CRYSTAL OR EXTERNAL CLOCK FREQUENCY</th>
<th>RESULTING TIME STATE PERIOD</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 MHz</td>
<td>400 ns</td>
<td>7803 time state; fastest allowable rate for Z80 device</td>
</tr>
<tr>
<td>1 MHz</td>
<td>2000 ns</td>
<td>Slowest recommended rate for Z80 device</td>
</tr>
</tbody>
</table>

FIGURE 6: CLOCK OSCILLATOR FREQUENCY RANGE
Bus Timing Specifications

An understanding of the 7803's signal timing characteristics is necessary for the selection of speed-compatible memory devices, I/O functions, other peripheral STD BUS cards, and for real-time logic analysis of 7803-based STD BUS card systems.

The 7803's timing characteristics are established by its Z80 microprocessor, with additional delays added by LSTTL buffers. The basic operations performed by the 7803 and the signals controlling these operations are shown in Figure 7

<table>
<thead>
<tr>
<th>SIGNALS</th>
<th>OPERATION</th>
<th>WAVEFORM</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEMRQ*, RD*</td>
<td>Read from memory</td>
<td>Figures 8 and 9</td>
</tr>
<tr>
<td>AO-A15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEMRQ*, WR*</td>
<td>Write to memory</td>
<td>Figure 9</td>
</tr>
<tr>
<td>AO-A15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IORQ*, RD*</td>
<td>Read from an input port</td>
<td>Figure 10</td>
</tr>
<tr>
<td>AO-A7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IORQ*, WR*</td>
<td>Write to an output port</td>
<td>Figure 10</td>
</tr>
<tr>
<td>AO-A7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTAK*</td>
<td>Read an interrupt instruction vector</td>
<td>Figure 11</td>
</tr>
<tr>
<td></td>
<td>(in response to INTRQ* only)</td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 7 : BASIC 7803 OPERATIONS**

The waveforms on the following pages show timing measurements as a 5-letter code as follows:

- First letter is always T for Timing measurement.
- Second letter is the abbreviation of the signal which starts the measurement. (C=Clock)
- Third letter is the condition of the start signal. (H=High)
- Fourth letter is the abbreviation of the signal which ends the measurement. (A=Address bus)
- Fifth letter is the condition of the end signal. (V=Valid)

For example, TCHAV stands for Time from Clock High until Address Valid. Specific abbreviations are given in the Legend on each page of the specification.

In the case of the Clock, it is necessary to note which time state is of interest; refer to figures 8 through 13.
MACHINE CYCLE

T1 | T2 | T3 | T4 | T1

CLOCK*

ADDRESS BUS
AO-A15

INSTRUCTION ADDRESS

REFRESH ADDRESS

MEMREQ*

TMLMH1

TAVML

TMLMH2

RD*

TRLRH

DATA BUS
D0-D7

OPCODE IN

DATA BUS
D0-D7

TAVRL

TRLDZ

TDVCL

TCLDX

TRLRH

TDVCL

TCLDX

REFRESH*

TSLAV

TSLSH

STATUS 1*

TCLFL

TFLFH

LEGEND
A | AO-A15
D | D0-D7
M | MEMREQ*
R | RD*
F | REFRESH*
S | STATUS 1*
E | CLOCK*
L | Low state
H | High state
V | Valid
Z | High impedance
X | Don't care
* | Low active

SYMBOL | PARAMETER | MIN | TYP | MAX
TAVDV | Address valid before data valid (access time) | 550 | 580
TAVME | Address valid before MEMREQ* active | 75
TMLMH | MEMREQ* pulse width (1 Opcode Fetch) | 600
TAVRL | Address valid before RD* active | 165
TRLRH | RD* pulse width | 370
TRLDZ | Data Bus in high impedance read mode after RD* active | 50 100
TDVCL | Data Bus setup time before clock transition ends T2 | 85
TCLDX | Data Bus hold time after T2 | 0
TCLFL | REFRESH* active after start of T3 | 200
TFLFH | REFRESH* pulse width | 770
TSLAV | STATUS 1* active after address valid | 0
TSLSH | STATUS 1* pulse width | 800
TCLAV | Address valid after start of T1 in any memory or 1/O machine cycle (Figures through ) | 160

FIGURE 8 : OPCODE FETCH AND MEMORY REFRESH MACHINE CYCLE
FIGURE 9: MEMORY READ (EXCEPT OPCODE) AND MEMORY WRITE
MACHINE CYCLES
Note: In onboard memory read operations (Section 6), the Data Bus does not enter the high impedance read mode. Instead the 7803 drives data fetched from the onboard memory sockets onto the STD Data Bus to facilitate logic state analysis at the motherboard. The access time for onboard memory devices may not exceed the values shown for TAVDV in Figure 8. The state of the Data Bus prior to TDVCL is unspecified for an onboard read operation.
**MACHINE CYCLE**

- **T1**
- **T2**
- **TW** (Note)
- **T3**
- **T1**

**CLOCK**

**ADDRESS BUS AO-A7**

- **PORT ADDRESS**
  - **TAVIL**
  - **TILIH**

**I:QRQ**

- **TAVRL**
- **TRLRH**

**RD**

- **TRLDZ**
- **TDVCH**
- **TCHDX**

**DATA BUS DO-D7**

- **TAVWL**
- **TAVDV1**
- **TAVWH**

**WR**

- **TAVDV2**
- **TDVWH**
- **TWHDX**

**DATA BUS DO-D7**

- **DATA OUT**

**LEGEND**

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>PARAMETER</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>AO-A7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>DO-D7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>I:QRQ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>RD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>WR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>CLOCK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>Low active</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>Low state</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>High state</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>Valid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>High impedance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Don't care</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TAADVQ** Address valid to data valid (input cycle)
**TAVWH** Address valid to WR active (output cycle)
**TAVIL** Address valid to I:QRQ active
**TILI** H:QRQ pulse width
**TAVRL** Address valid to RD active
**TRLDZ** Data Bus in high impedance read mode after RD low
**TDVCH** Input data setup time before clock high in T3
**TCHDX** Input data hold time after clock high in T3
**TAVWL** Address valid to WR active
**TWLWH** WR pulse width
**TAADV2** Output data valid after address valid
**TDVWH** Output data setup time before WR rising edge
**TWHDX** Output data hold time after WR rising edge

**NANOSECONDS**

- 1100
- 1530
- 300
- 345
- 1000
- 300
- 355
- 1000
- 50
- 100
- 95
- 0
- 300
- 335
- 1000
- 300
- 1070
- 0

Note: TW (WAIT state) inserted automatically by Z80 in I/O cycles.
### LEGEND

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>PARAMETER</th>
<th>UNIT</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>CLOCK*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td>INTRQ*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>INTAK*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>DATA BUS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>Low active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>Low level</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>High level</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>High impedance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Don’t care</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### SYMBOLS

- **TQLCL**: INTRQ* setup time prior to last time state in instruction cycle prior to interrupt
- **TCLQL**: INTRQ* hold time after clock low
- **TCHKL**: INTAK* asserted in first TW after clock high
- **TKLKH**: INTAK* pulse width
- **TKLDZ**: Data Bus in high impedance read mode after INTAK* low
- **TDVCL**: Data Bus setup time prior to clock low
- **TCLDX**: Data Bus hold time after clock low

(TWA and TWB)

**Notes:**

1. Two WAIT states are automatically inserted by the Z80 to allow for priority chain propagation time.
2. In interrupt mode 1, INTAK* is asserted but the data bus is ignored.
3. The above time state sequence assumes that the ENI (enable interrupt) instruction is in effect.
4. INTAK* = [(MI*)*(IORQ*)] plus buffer delays.

**FIGURE II**: INTERRUPT REQUEST/ACKNOWLEDGE CYCLE
WAIT REQUEST

The WAITRQ* input allows the 7803 to enter the WAIT state in any memory, I/O or interrupt acknowledge cycle while a slow memory device responds, or until a control function such as an analog-to-digital converter finishes. WAITRQ* can also be used to single-step the 7803. Figure 12 shows the required timing for the WAITRQ* input.

![MACHINE CYCLE](image)

**FIGURE 12**: WAIT STATE INSERTION IN Opcode FETCH, Memory READ, AND MEMORY WRITE MACHINE CYCLES

WAIT state insertion in all memory cycles is similar to the Opcode Fetch, Memory Read, and Memory Write cycles shown in Figures 8-9. While WAITRQ* is sampled halfway through time state T2 in these cycles, however, it is sampled at different times in I/O and interrupt acknowledge cycles.

I/O machine cycles sample WAITRQ* at the rising edge of CLOCK* during TW, the single wait state inserted automatically by the Z80 in I/O cycles. User-inserted wait states occur after TW and prior to T3.

Interrupt machine cycles sample WAITRQ* at the rising edge of CLOCK* during TWB, the second wait state inserted automatically by the Z80 during interrupt acknowledge cycles. User-inserted wait states occur after TWB and prior to T3.
BUS REQUEST

The BUSRQ* input and BUSAK* output allow Direct Memory Access (DMA) operations, giving another system controller card access to the 7803's peripheral cards. Figure 13 shows the timing for these signals.

**LEGEND**
- **CLOCK***: Low state
- **BUSSES**: Low state
- **BUSERQ***: Low state
- **BUSAK***: Low state
- **LOW impedance**: Low state
- **Drivers on**: Low state

**SYMBOL** | **PARAMETER** | **NANoseconds**
--- | --- | ---
TQLCL | BUSRQ* setup time prior to last time state in last instruction, last cycle preceding DMA | 130
TCLKL | BUSAK* active after start of first DMA cycle | 185
TJKBZ | Busses float after BUSAK* active | 35 65
TCHKH | BUSAK* inactive after clock rising edge in last DMA cycle | 160
TKHBD | Busses driven after BUSAK* inactive | 35 65

**NOTE:** Busses refers to the Address Bus AO-A15; the Data Bus DO-D7; and the Control Bus lines MEMRQ*, IORQ*, RD*, WR*, INTAK*, REFRESH*, MCSYNC*, STATUS 1*, and SYSRESET*. Other Control Bus lines are not floated.

**FIGURE 13: BUSRQ*/BUSAK* (DMA) MACHINE CYCLES**
Mechanical

The 7803 meets all STD BUS mechanical specifications. Refer to the Series 7000 Technical Manual for outline dimensions.

Environmental

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free Air Ambient Operating Temperature</td>
<td>0</td>
<td>25</td>
<td>55</td>
<td>°Celsius</td>
</tr>
<tr>
<td>Absolute Nonoperating Free Air Ambient Temperature</td>
<td>-40</td>
<td>75</td>
<td></td>
<td>°Celsius</td>
</tr>
<tr>
<td>Relative Humidity, Noncondensing</td>
<td>5</td>
<td>95</td>
<td></td>
<td>%</td>
</tr>
<tr>
<td>Absolute Nonoperating Relative Humidity, Noncondensing</td>
<td>0</td>
<td>100</td>
<td></td>
<td>%</td>
</tr>
</tbody>
</table>

FIGURE 14: ENVIRONMENTAL SPECIFICATIONS
SECTION 4 : Z80 ARCHITECTURE AND INSTRUCTION SET

FIGURE 15 : Z80 PROGRAMMING MODEL
Z80 Architecture

The Z80 architecture (Figure 15) consists of a 16-bit Instruction Register, a 16-bit Program Address Counter, a 16-bit Stack Pointer, two 16-bit Index Registers, an 8-bit Interrupt Page Register, and two bank-selectable sets of General Purpose Registers plus two bank-selectable Arithmetic/Logical Units (ALUs). A 6-bit Flag Register (in each ALU bank) holds processor condition code information. An 8-bit autounting Refresh Register supports dynamic RAMs.

Instruction Register: The 16-bit Instruction Register provides storage and decoding for instruction opcodes as they are received from program memory.

The Z80 executes all of the 8080 instructions as a subset of instructions with 8-bit (one byte) opcodes, and adds a large number of additional instructions of which most have 16-bit (two byte) opcodes. The processor receives the first opcode byte from memory and decodes it to determine if a second opcode byte follows. The instructions with 2-byte opcodes are identified by a first byte equal to hexadecimal CB, DD, ED, or FD.

The complete instruction word may consist of address or data information in addition to a 1-byte or 2-byte opcode. The full instruction may be up to four bytes (32 bits) long. Additional words of multi-byte instructions bypass the instruction register. These words may be immediate data for registers, a memory or I/O port address for direct addressing, or an offset address for indexed relative addressing.

Program Address Counter (PC): The 16-bit Program Address Counter keeps track of the location of the next instruction to be executed from the program memory. The PC increments automatically for each instruction word unless the instruction is a jump or subroutine return which modifies the count by loading a new address.

Stack Pointer (SP): A 16-bit auto-counting Stack Pointer provides the address of the subroutine return address stack location in RAM memory. The SP is used for controlling subroutines and interrupts, and can also be used to "push" and "pull" data in memory at high speed.

Subroutine return addresses are automatically stored on the stack when a jump-to-subroutine instruction is executed, and are retrieved when a return-from-subroutine instruction is executed. Z80 mode 1 and 2 interrupts are treated as subroutine jumps, taking advantage of the SP's return address storage and retrieval ability.

All of the General Purpose Register Pairs and the ALU registers can be stored and retrieved from memory using the SP as an indirect address register. The resulting 16-bit data movement and automatic increment/decrement of the SP offer fast memory data manipulation.

The current memory address in the SP can be brought into the HL Register Pair for arithmetic manipulation, then restored to the SP by the program.

General Purpose Registers: Two identical banks of General Purpose Registers are provided in the Z80. Each consists of six 8-bit registers (B,C,D,E,H,L) which can also be treated as three 16-bit Register Pairs (BC, DE, HL). The banks can be switched by a single instruction, providing fast interrupt response by saving the time required to store the register content in memory. Or they can be used as general fast access data storage in non-interrupt applications.
The instruction set allows individual 8-bit registers to be loaded from any other register, loaded and stored in memory indirectly, or loaded immediately from the second byte of the instruction. All registers can be incremented and decremented, added to or subtracted from the Accumulator, perform logic with the Accumulator, shifted or rotated arithmetically or logically. Each bit in each register can be addressed separately for testing, setting, and clearing. Register C can be used for indirect I/O port addressing.

The three 16-bit register pairs can be loaded immediately from the second and third bytes of the instruction, incremented and decremented, stored directly in memory, added or subtracted to the HL pair and the Index Registers, and used as indirect address registers for operations with other 8-bit registers, memory, and the Accumulator. In arithmetic operations, carries and borrows are propagated from the low-order 8-bit register into the high-order register automatically.

Arithmetic/Logical Unit (ALU): The ALU consists of an 8-bit Accumulator Register (Register A) and a 8-bit condition code or Flag Register (Register F), plus arithmetic, logical, shift, rotate, and control circuitry needed to execute the program instructions. The A and F register are treated as the AF Register Pair for push and pull operations involving the Stack Pointer Register. The ALU is duplicated in two banks, with bank switching accomplished by a single instruction similar to the General Purpose Registers. The enabled ALU provides add and subtract with or without carry; AND, OR, Exclusive OR, compare, shift, rotate, and byte complement operations. ALU operations are performed on the Accumulator from other registers or memory, with direct, indirect, indexed, or immediate addressing. The Accumulator is the primary register for I/O communication. The Accumulator can be decimally adjusted and allows 4-bit nibble swap operations with memory for Binary Coded Decimal (BCD) arithmetic.

Register F contains the following flags:

- **C** - Carry/Borrow from Accumulator bit 7 (arithmetic or rotate)
- **D** - Carry for BCD arithmetic from Accumulator bit 3
- **N** - Specifies whether last operation was subtract, allowing different algorithm for BCD operations.
- **Z** - Zero resulted from the last Accumulator operation.
- **S** - Sign for signed binary arithmetic (same as Accumulator bit 7)
- **PV** - Parity/Overflow (signed binary arithmetic); dual function flag, function depending on last instruction

PV shows whether (NTR0#) is enabled when tested after the LDAI instruction (Figure 21).

The C, Z, S, and PV flags can be tested by the conditional jump and subroutine return instructions. Special instructions allow the C flag to be set, cleared, and complemented. When pushed/pulled on the Stack via the Stack Pointer as part of the AF register pair, the F register occupies 8 bits with the state of bits 3 and 5 unspecified. The states of the six flags can be preset by pulling program-prepared bits into Register F; the states of the untestable flags (D,N) can be determined by pushing Register F onto the Stack, then pulling the Stack data into a General Purpose register.

Index Registers (IX and IY): The 16-bit Index Registers are used as indirect memory address registers. The address supplied by IX and IY is modified by a relative offset which is one byte of the multibyte indexed instructions. The Index Registers address memory to allow memory bytes to take part in the arithmetic
and logical operations described above for the single 8-bit General Purpose Registers. When modifying the address content of the Index Registers, IX and IY are treated as Register Pairs. The arithmetic, logical, and load/store operations that can be performed on the General Purpose Register Pairs can generally be performed on the Index Registers, although fewer instructions apply to IX and IY than to the BC, DE, and HL pairs.

Input and Output Ports (I/O): I/O is mapped independently of memory with separate control signals and instructions. The OPA instruction writes data from the Accumulator to output ports, and the IPA instruction reads data from input ports to the Accumulator. A specific port is specified by the second byte of the instruction, allowing up to 256 each 8-bit input and output ports. In the 7803, all I/O ports are provided on separate cards.

Communication with the ports can be direct from memory using HL as an address pointer when the Z80's Compound Instructions (below) are used.

Compound Instructions: The Z80's instruction set contains several compound instructions which perform multiple functions, with or without automatic looping. These are implied sequence instructions which execute a fixed sequence of other instructions in the instruction set. They perform block (multiple byte) moves within memory, search memory, and input or output to or from memory from the I/O ports. Register C as a port address pointer and HL as a memory address pointer. One compound instruction performs automatic count and jump functions for loop control alone.

The compound instructions require approximately as much time as the instruction sequences they replace, but offer program memory savings by eliminating instruction storage for common program functions.

Interrupt: The Z80 offers three interrupt modes:

0 - identical to the 8080 interrupt system
1 - implied vector interrupt (Restart at 0038 always)
2 - supplied vector interrupt, with a single byte supplied by the interrupting device.

In interrupt mode 2, the content of the Interrupt (I) Register is the vector page address, and the byte supplied by the interrupting device is the vector line address. Together they form a 16-bit memory address which is the indirect address of the interrupt service routine for that device.

More interrupt information is given at the end of this section.

Refresh Register (R): The Z80 contains an 8-bit Refresh Register which is used to address dynamic RAM devices external to the 7803 card. In conjunction with the REFRESH* control signal, the processor can automatically refresh dynamic RAMs during the opcode fetch portion of the instruction cycle. This function is applicable primarily to certain dynamic RAM devices available from the manufacturers of the Z80 chip.
Both the Z80 and the 8085 include all of the 8080 instructions as a subset, and these instructions are all machine-language compatible. Programs written exclusively in 8080 opcodes will execute on the Z80 with the following considerations:

1. Execution times of 8080-identical instructions vary due to the number of time states required for execution (some more, some less) in the Z80 and 8085, even if the processors are all operated at the same clock rate. Consequently, programmed timing (such as count-and-test time delays) will generally require modification.

2. Flag Register bit 2 is the PV (Parity/Overflow) flag in the Z80, and parity only in the 8080/8085. The added overflow function is for signed binary arithmetic. Since the parity and overflow functions are unrelated, occurring at different times in most programs, incompatibility does not usually result. However the flag's activity is different overall and the program should be examined for sensitivity to the PV flag.

Except for the differences noted, the Z80 resets to 8080 compatibility and its additional features must be deliberately invoked by the program.
STD INSTRUCTION MNEMONICS

The STD Instruction Mnemonics are a standard set of processor instruction abbreviations suitable for use as an assembly language for writing programs. These mnemonics are standard in that they do not change but keep the same meaning regardless of the processor they are applied to. They are also standard in that they are derived from a set of easily understood rules.

The instruction mnemonic is an abbreviated action statement containing an operator, a locator and a qualifier plus a supplemental and separate modifier.

1. The operator is a unique two letter abbreviation that suggests the action.
2. The locator follows the operator and designates the operand or data to be operated on. Instructions without operands ignore the locator.
3. The qualifier states the addressing mode or provides further qualifying information for compound instructions.
4. The modifier carries detailed support information: labels, conditions, addressing and data.

The operator, locator and qualifier letters are strung together to form the instruction mnemonic. The modifier, when needed, stands alone either in its own separate column or separated by spaces or additional lines in written text.

<table>
<thead>
<tr>
<th>OPERATOR</th>
<th>LOCATOR</th>
<th>QUALIFIER</th>
<th>MODIFIER</th>
<th>INSTRUCTION DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTS</td>
<td>RT</td>
<td>S</td>
<td></td>
<td>Return from Subroutine</td>
</tr>
<tr>
<td>CLA</td>
<td>CL</td>
<td>A</td>
<td></td>
<td>Clear A</td>
</tr>
<tr>
<td>LDAD</td>
<td>LD</td>
<td>A</td>
<td>D</td>
<td>Load A Direct</td>
</tr>
<tr>
<td>LDA</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>Load A with B</td>
</tr>
<tr>
<td>LDAN (BC)</td>
<td>LD</td>
<td>A</td>
<td>N (BC)</td>
<td>Load A indirect using BC as an Address Pointer</td>
</tr>
<tr>
<td>JS (LABEL)</td>
<td>JS</td>
<td>(LABEL)</td>
<td></td>
<td>Jump to Subroutine Located at (LABEL)</td>
</tr>
</tbody>
</table>

Figure 14: Examples of Instruction Mnemonic Structure
The following table lists the STD mnemonic operations, locators, modifiers, qualifiers, and other notation used in the instruction tables for the Z80 in this section.

### STANDARD MNEMONICS DEFINITIONS

<table>
<thead>
<tr>
<th>OPERATIONS</th>
<th>LOCATORS, MODIFIERS</th>
<th>QUALIFIERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD (AC) ADD (WITH CARRY)</td>
<td>A,ACC</td>
<td>C</td>
</tr>
<tr>
<td>AJ ADJUST DECIMAL</td>
<td>B,CDI</td>
<td>J</td>
</tr>
<tr>
<td>AN AND</td>
<td>E,HL</td>
<td>D</td>
</tr>
<tr>
<td>BS BANK SELECT</td>
<td>F</td>
<td>DIRECT ADDRESS, OR DECIMAL</td>
</tr>
<tr>
<td>CL CLEAR</td>
<td>G</td>
<td>IMMEDIATE DATA</td>
</tr>
<tr>
<td>CM COMPLEMENT</td>
<td>H</td>
<td>INDIRECT ADDRESS</td>
</tr>
<tr>
<td>CP COMPARE</td>
<td>I</td>
<td>INDEXED ADDRESS</td>
</tr>
<tr>
<td>DC DECREMENT</td>
<td>J</td>
<td>RELATIVE ADDRESS</td>
</tr>
<tr>
<td>DS DISABLE</td>
<td>K</td>
<td>TOP OF STACK</td>
</tr>
<tr>
<td>EN ENABLE</td>
<td>L</td>
<td>WORD FORWARD</td>
</tr>
<tr>
<td>HLT HALT</td>
<td>M</td>
<td>BLOCK FORWARD</td>
</tr>
<tr>
<td>IC INCREMENT</td>
<td>N</td>
<td>WORD BACKWARD</td>
</tr>
<tr>
<td>IF INPUT FROM PORT</td>
<td>O</td>
<td>BLOCK BACKWARD</td>
</tr>
<tr>
<td>JS JUMP TO INTERRUPT</td>
<td>P</td>
<td>WORD ADDRESS</td>
</tr>
<tr>
<td>JP JUMP</td>
<td>Q</td>
<td>16-BIT DATA, POINTERS</td>
</tr>
<tr>
<td>JS JUMP TO SUBROUTINE</td>
<td>R</td>
<td>UNCONDITIONAL</td>
</tr>
<tr>
<td>LD LOAD</td>
<td>S</td>
<td>NONMASKABLE INTERRUPT</td>
</tr>
<tr>
<td>MV MOVE MEMORY</td>
<td>T</td>
<td>MOST SIGNIFICANT BIT</td>
</tr>
<tr>
<td>NOP NO OPERATION</td>
<td>U</td>
<td>LEAST SIGNIFICANT BIT</td>
</tr>
<tr>
<td>OP OUTPUT TO PORT</td>
<td>V</td>
<td>JUMP CONDITION</td>
</tr>
<tr>
<td>OR INCLUSIVE</td>
<td>W</td>
<td>I/O PORT ADDRESS</td>
</tr>
<tr>
<td>PULP PUSH, PULL VIA stack</td>
<td>X</td>
<td>MEMORY LINE ADDRESS</td>
</tr>
<tr>
<td>RLRAR ROTATE LEFT, RIGHT</td>
<td>Y</td>
<td>16-BIT ADDRESS</td>
</tr>
<tr>
<td>RES SET</td>
<td>Z</td>
<td>8-BIT OFFSET</td>
</tr>
<tr>
<td>RT RETURN VIA STACK</td>
<td>P,F</td>
<td>8-BIT DATA</td>
</tr>
<tr>
<td>SE SET</td>
<td>G,P</td>
<td></td>
</tr>
<tr>
<td>ST STORE</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>SU (SC) SUBTRACT (WITH CARRY)</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>TS TEST</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>XC EXCHANGE</td>
<td>M</td>
<td></td>
</tr>
</tbody>
</table>

### MISCELLANEOUS

| UN UNCONDITIONAL |
| NMI NONMASKABLE INTERRUPT |
| MSB MOST SIGNIFICANT BIT |
| LSB LEAST SIGNIFICANT BIT |
| CSJ JUMP CONDITION |
| PS I/O PORT ADDRESS |
| ML MEMORY LINE ADDRESS |
| HMP 16-BIT ADDRESS |
| RF RELATIVE OFFSET |
| BD 8-BIT DATA |

**FIGURE 17: STD Mnemonics**
The Z80 Instruction Set

Figures 18, 19 and 20 show the full Z80 instruction set with STD mnemonics and hexadecimal operation codes. The tables are grouped by 8-bit register operations, 16-bit register pair operations, ALU (Accumulator and Carry), program address control, I/O, machine control, and compound instructions.

Figure 21 shows the bit organization of the 8-bit and 16-bit registers, the effect of the shift and rotate instructions, the allocation of memory by certain instructions, and the action of the flags (instructions not listed in the Flag Summary have no effect on the flags). Figure 22 shows relative addressing constants.

### 8-BIT LOAD, STORE

<table>
<thead>
<tr>
<th>INST</th>
<th>MOD</th>
<th>MOD1</th>
<th>MOD2</th>
<th>MOD3</th>
<th>MOD4</th>
<th>MOD5</th>
<th>IMMEDIATE</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>7E</td>
<td>7E</td>
<td>7E</td>
<td>7E</td>
<td>7E</td>
<td>7E</td>
<td>DD7Eh</td>
<td>F07Er</td>
</tr>
<tr>
<td>LDB</td>
<td>47</td>
<td>42</td>
<td>41</td>
<td>40</td>
<td>44</td>
<td>46</td>
<td>DD46h</td>
<td>F046h</td>
</tr>
<tr>
<td>LDC</td>
<td>4F</td>
<td>48</td>
<td>49</td>
<td>4A</td>
<td>48</td>
<td>40</td>
<td>DD40h</td>
<td>F040h</td>
</tr>
<tr>
<td>LDD</td>
<td>57</td>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>50</td>
<td>DD50h</td>
<td>F050h</td>
</tr>
<tr>
<td>LDE</td>
<td>5F</td>
<td>58</td>
<td>59</td>
<td>5A</td>
<td>5B</td>
<td>5C</td>
<td>DD5Ch</td>
<td>F05Ch</td>
</tr>
<tr>
<td>LDI</td>
<td>67</td>
<td>60</td>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td>DD64h</td>
<td>F064h</td>
</tr>
<tr>
<td>LDH</td>
<td>6F</td>
<td>68</td>
<td>69</td>
<td>6A</td>
<td>6B</td>
<td>6C</td>
<td>DD6Ch</td>
<td>F06Ch</td>
</tr>
<tr>
<td>STX</td>
<td>77</td>
<td>70</td>
<td>71</td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>DD74h</td>
<td>F074h</td>
</tr>
<tr>
<td>STY</td>
<td>7D</td>
<td>70</td>
<td>71</td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>DD74h</td>
<td>F074h</td>
</tr>
</tbody>
</table>

### 8-BIT ARITHMETIC

<table>
<thead>
<tr>
<th>INST</th>
<th>MOD</th>
<th>MOD1</th>
<th>MOD2</th>
<th>MOD3</th>
<th>MOD4</th>
<th>MOD5</th>
<th>IMMEDIATE</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICX</td>
<td>3C</td>
<td>04</td>
<td>0C</td>
<td>14</td>
<td>1C</td>
<td>24</td>
<td>DD24h</td>
<td>F024h</td>
</tr>
<tr>
<td>DEX</td>
<td>3D</td>
<td>05</td>
<td>00</td>
<td>15</td>
<td>10</td>
<td>25</td>
<td>DD25h</td>
<td>F025h</td>
</tr>
<tr>
<td>ADD</td>
<td>57</td>
<td>80</td>
<td>81</td>
<td>82</td>
<td>83</td>
<td>84</td>
<td>DD84h</td>
<td>F084h</td>
</tr>
<tr>
<td>ACA</td>
<td>5F</td>
<td>88</td>
<td>89</td>
<td>8A</td>
<td>8B</td>
<td>8C</td>
<td>DD8Ch</td>
<td>F08Ch</td>
</tr>
<tr>
<td>SAA</td>
<td>97</td>
<td>90</td>
<td>91</td>
<td>92</td>
<td>93</td>
<td>94</td>
<td>DD94h</td>
<td>F094h</td>
</tr>
<tr>
<td>SCA</td>
<td>9F</td>
<td>98</td>
<td>99</td>
<td>9A</td>
<td>9B</td>
<td>9C</td>
<td>DD9Ch</td>
<td>F09Ch</td>
</tr>
</tbody>
</table>

### 8-BIT LOGIC

<table>
<thead>
<tr>
<th>INST</th>
<th>MOD</th>
<th>MOD1</th>
<th>MOD2</th>
<th>MOD3</th>
<th>MOD4</th>
<th>MOD5</th>
<th>IMMEDIATE</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANA</td>
<td>A7</td>
<td>A8</td>
<td>A9</td>
<td>A1</td>
<td>A2</td>
<td>A3</td>
<td>DD3Ah</td>
<td>F03Ah</td>
</tr>
<tr>
<td>EGA</td>
<td>AF</td>
<td>A8</td>
<td>A9</td>
<td>A9</td>
<td>AC</td>
<td>AD</td>
<td>DD2Ah</td>
<td>F02Ah</td>
</tr>
<tr>
<td>ORA</td>
<td>87</td>
<td>88</td>
<td>89</td>
<td>8A</td>
<td>8B</td>
<td>8C</td>
<td>DD2Ch</td>
<td>F02Ch</td>
</tr>
<tr>
<td>CPA</td>
<td>8F</td>
<td>88</td>
<td>89</td>
<td>8A</td>
<td>8B</td>
<td>8C</td>
<td>DD2Ch</td>
<td>F02Ch</td>
</tr>
</tbody>
</table>

### ROTATE, SHIFT

<table>
<thead>
<tr>
<th>INST</th>
<th>MOD</th>
<th>MOD1</th>
<th>MOD2</th>
<th>MOD3</th>
<th>MOD4</th>
<th>MOD5</th>
<th>IMMEDIATE</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>RLX</td>
<td>CB07</td>
<td>CB08</td>
<td>CB09</td>
<td>CB0A</td>
<td>CB0B</td>
<td>CB0C</td>
<td>DD3Bh</td>
<td>F03Bh</td>
</tr>
<tr>
<td>RAXA</td>
<td>CB0F</td>
<td>CB06</td>
<td>CB07</td>
<td>CB08</td>
<td>CB09</td>
<td>CB0A</td>
<td>DD3Bh</td>
<td>F03Bh</td>
</tr>
<tr>
<td>RLX</td>
<td>CB17</td>
<td>CB18</td>
<td>CB19</td>
<td>CB1A</td>
<td>CB1B</td>
<td>CB1C</td>
<td>DD3Bh</td>
<td>F03Bh</td>
</tr>
<tr>
<td>RAX</td>
<td>CB1F</td>
<td>CB16</td>
<td>CB17</td>
<td>CB18</td>
<td>CB19</td>
<td>CB1A</td>
<td>DD3Bh</td>
<td>F03Bh</td>
</tr>
</tbody>
</table>

For all instructions, MOD will be either (HL), (IX), or (IY), (I), (W).
### Accumulator, Carry Control

<table>
<thead>
<tr>
<th>INSTR</th>
<th>MOD</th>
<th>CODE</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLAC</td>
<td>AF</td>
<td>CLEAR ACC, CARRY</td>
<td></td>
</tr>
<tr>
<td>CLC</td>
<td>B7</td>
<td>CLEAR CARRY FLAG</td>
<td></td>
</tr>
<tr>
<td>SEC</td>
<td>37</td>
<td>SET CARRY FLAG</td>
<td></td>
</tr>
<tr>
<td>CMC</td>
<td>3F</td>
<td>COMPLEMENT CARRY</td>
<td></td>
</tr>
<tr>
<td>CMAL</td>
<td>2F</td>
<td>COMPLEMENT ACC LOGICAL</td>
<td></td>
</tr>
<tr>
<td>CMAA</td>
<td>ED44</td>
<td>COMPLEMENT ACC ARITHMETIC</td>
<td></td>
</tr>
<tr>
<td>AJAD</td>
<td>27</td>
<td>ADJUST ACC DEGITALLY</td>
<td></td>
</tr>
<tr>
<td>RLA</td>
<td>07</td>
<td>ROTATE ACC</td>
<td></td>
</tr>
<tr>
<td>RRA</td>
<td>0F</td>
<td>(8 BIT)</td>
<td></td>
</tr>
<tr>
<td>RLC</td>
<td>17</td>
<td>ROTATE CARRY AND CARRY</td>
<td></td>
</tr>
<tr>
<td>RRC</td>
<td>1F</td>
<td>(9 BIT)</td>
<td></td>
</tr>
<tr>
<td>RLAM</td>
<td>(HL)</td>
<td>ED6F</td>
<td>ROTATE ACC MULTIPLE WITH MEMORY</td>
</tr>
<tr>
<td>RRAM</td>
<td>(HL)</td>
<td>ED67</td>
<td></td>
</tr>
<tr>
<td>LDA</td>
<td>R</td>
<td>ED5F</td>
<td>LOAD ACC FROM REFRESH</td>
</tr>
<tr>
<td>LDR</td>
<td>A</td>
<td>ED4F</td>
<td>STORE ACC IN REFRESH</td>
</tr>
<tr>
<td>LOAD</td>
<td>3ANLMP</td>
<td>LOAD ACC DIRECT</td>
<td></td>
</tr>
<tr>
<td>STAD</td>
<td>3LNMP</td>
<td>STORE ACC DIRECT</td>
<td></td>
</tr>
</tbody>
</table>

### Machine Control Instructions

<table>
<thead>
<tr>
<th>INSTR</th>
<th>MOD</th>
<th>CODE</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENI</td>
<td>F8</td>
<td>ENABLE INTERRUPT</td>
<td></td>
</tr>
<tr>
<td>DS1</td>
<td>F3</td>
<td>DISABLE INTERRUPT</td>
<td></td>
</tr>
<tr>
<td>SEIM</td>
<td>0</td>
<td>ED46</td>
<td>INTERRUPT MODE 0</td>
</tr>
<tr>
<td>SEIM</td>
<td>1</td>
<td>ED56</td>
<td>INTERRUPT MODE 1</td>
</tr>
<tr>
<td>SEIM</td>
<td>2</td>
<td>ED5E</td>
<td>INTERRUPT MODE 2</td>
</tr>
<tr>
<td>LDA</td>
<td>1</td>
<td>ED57</td>
<td>LOAD ACC FROM INTERRUPT REG</td>
</tr>
<tr>
<td>LDI</td>
<td>A</td>
<td>ED45</td>
<td>LOAD INTERRUPT REG FROM ACC</td>
</tr>
<tr>
<td>RTI</td>
<td>ED40</td>
<td>RETURN FROM INTRO</td>
<td></td>
</tr>
<tr>
<td>RTN</td>
<td>ED45</td>
<td>RETURN FROM NMIRO</td>
<td></td>
</tr>
<tr>
<td>NOP</td>
<td>00</td>
<td>NO OPERATION</td>
<td></td>
</tr>
<tr>
<td>HLT</td>
<td>76</td>
<td>HALT</td>
<td></td>
</tr>
<tr>
<td>LDA</td>
<td>R</td>
<td>ED5F</td>
<td>LOAD ACC FROM REFRESH REG</td>
</tr>
<tr>
<td>LDR</td>
<td>A</td>
<td>ED4F</td>
<td>LOAD REFRESH REG FROM ACC</td>
</tr>
</tbody>
</table>

### Input-Output Instructions

<table>
<thead>
<tr>
<th>INSTR</th>
<th>MOD</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>H</th>
<th>L</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPa</td>
<td>Ps</td>
<td>0B90</td>
<td>ED70</td>
<td>ED40</td>
<td>ED50</td>
<td>ED40</td>
<td>ED56</td>
<td>ED40</td>
<td>ED48</td>
<td>INPUT DIRECT FROM PORT Ps TO ACC</td>
</tr>
<tr>
<td>IPIN</td>
<td>(C)</td>
<td>ED70</td>
<td>ED40</td>
<td>ED50</td>
<td>ED40</td>
<td>ED56</td>
<td>ED40</td>
<td>ED48</td>
<td>INPUT INDIRECT FROM PORT DEFINED BY (C) TO REGISTER NAMED</td>
<td></td>
</tr>
<tr>
<td>OPA</td>
<td>Ps</td>
<td>0D90</td>
<td>ED70</td>
<td>ED41</td>
<td>ED50</td>
<td>ED51</td>
<td>ED40</td>
<td>ED48</td>
<td>OUTPUT DIRECT FROM ACC TO PORT Ps</td>
<td></td>
</tr>
<tr>
<td>OPIN</td>
<td>(C)</td>
<td>ED70</td>
<td>ED41</td>
<td>ED50</td>
<td>ED51</td>
<td>ED40</td>
<td>ED48</td>
<td>INPUT INDIRECT FROM REGISTER NAMED TO PORT DEFINED BY (C)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Compound Instructions

<table>
<thead>
<tr>
<th>INSTR</th>
<th>MOD</th>
<th>WORD</th>
<th>BLOCK</th>
<th>WORD</th>
<th>BLOCK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>FORWARD</td>
<td>FORWARD</td>
<td>BACKWARD</td>
<td>BACKWARD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16*16F</td>
<td>16*8F</td>
<td>16*8B</td>
<td>32*8B</td>
</tr>
<tr>
<td>MVX</td>
<td>EDA0</td>
<td>ED80</td>
<td>ED80</td>
<td>ED80</td>
<td>ED80</td>
</tr>
<tr>
<td>CPX</td>
<td>EDA1</td>
<td>ED81</td>
<td>ED80</td>
<td>ED80</td>
<td>ED80</td>
</tr>
<tr>
<td>IPX</td>
<td>EDA2</td>
<td>ED92</td>
<td>EDAA</td>
<td>ED8A</td>
<td>ED8A</td>
</tr>
<tr>
<td>OPX</td>
<td>EDA3</td>
<td>ED83</td>
<td>ED80</td>
<td>ED88</td>
<td>ED88</td>
</tr>
</tbody>
</table>

### Operate

<table>
<thead>
<tr>
<th>INSTRO</th>
<th>MOD</th>
<th>WID</th>
<th>BLOCK</th>
<th>BLOCK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>16W</td>
<td>16W</td>
<td>16W</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOVE MEMORY WORD FROM (HL) TO (DE):</td>
</tr>
<tr>
<td>INCREMENT FORWARD OR DECREMENT BACKWARD DE AND HL:</td>
</tr>
<tr>
<td>DECREMENT (COUNT) BC; IF BLOCK, REPEAT UNTIL (BC) = 0.</td>
</tr>
<tr>
<td>COMPAR TO ACC WITH (HL), RESULT TO F:</td>
</tr>
<tr>
<td>INCREMENT FORWARD OR DECREMENT BACKWARD HL;</td>
</tr>
<tr>
<td>DECREMENT (COUNT) BC; IF BLOCK, REPEAT UNTIL ACC = 0 OR BC &lt; 0.</td>
</tr>
<tr>
<td>INPUT FROM PORT DEFINED BY (C), STORE IN (HL):</td>
</tr>
<tr>
<td>INCREMENT FORWARD OR DECREMENT BACKWARD HL;</td>
</tr>
<tr>
<td>DECREMENT COUNT R; IF BLOCK REPEAT UNTIL R = 0.</td>
</tr>
<tr>
<td>OUTPUT DATA FROM (HL) TO PORT DEFINED BY (C):</td>
</tr>
<tr>
<td>INCREMENT FORWARD OR DECREMENT BACKWARD HL;</td>
</tr>
<tr>
<td>DECREMENT R; IF BLOCK, REPEAT UNTIL R = 0.</td>
</tr>
</tbody>
</table>

| DCRQ  | 10W | DECREMENT B; IF 0, 1; JUMP TO (PC+1). |

**Figure 20: 280 Instruction Set**
### Automatic Memory Operations

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>DATA FLOW</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>JL JS, ANY INTERRUPT</td>
<td>mP—(SP-1)</td>
<td>SP-2 AFTER</td>
</tr>
<tr>
<td></td>
<td>mP—(SP-2)</td>
<td></td>
</tr>
<tr>
<td>ANY RET</td>
<td>(SP)=mL</td>
<td>SP-2 AFTER</td>
</tr>
<tr>
<td></td>
<td>mL—(SP-1)</td>
<td></td>
</tr>
<tr>
<td>LOAD/STORE, PLP/PPS ANY REGISTER PAIR IN MEMORY</td>
<td>dL—mlnP</td>
<td>dL—REGISTERS F, C, E, L</td>
</tr>
<tr>
<td></td>
<td>dH—mlnP+1</td>
<td>dH—REGISTERS A, B, D, H</td>
</tr>
</tbody>
</table>

**NOTE:** This table shows how the processor allocates memory automatically when certain instructions are executed. For example, the PLP instruction pulls a line address or low-order register (C,E,F,L); increments the SP, then pulls a page address or high-order register (A,B,D,H); and increments the SP again, leaving the SP two counts higher than its initial value.

### Flag Summary

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>S</th>
<th>Z</th>
<th>D</th>
<th>O</th>
<th>P</th>
<th>V</th>
<th>N</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADA, ACA</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>SUA, SCA, CPA, CMAA</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>ANA</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>ORA, XRA, CLC</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>CLAC</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>TSs</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>ICx</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>DCs</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>LDA L, LDA R</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>AJA</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>CMAL</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>SEC</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>RLA, RRA, RLAC, RRAC</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>SLA, SRA, SRAC, RLAC</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>IP, IPRA</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>ADP</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>ACP</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>SCP</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>IPWF, IPWB, OPWF, OPWB</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>IPBF, IPBB, OPBF, OPBB</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>MVWF, MVWB</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>MVBF, MVBB</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
<tr>
<td>CPWF, CPWB, OPWF, OPWB</td>
<td>S</td>
<td>Z</td>
<td>D</td>
<td>O</td>
<td>P</td>
<td>V</td>
<td>N</td>
<td>C</td>
</tr>
</tbody>
</table>

### Shift, Rotate Summary

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>SHIFT DIRECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>RLA.RRA</td>
<td>R</td>
</tr>
<tr>
<td>RLA.RRA</td>
<td>R</td>
</tr>
<tr>
<td>RLA.RRA</td>
<td>R</td>
</tr>
<tr>
<td>RLA.RRA</td>
<td>R</td>
</tr>
<tr>
<td>SLA</td>
<td>L</td>
</tr>
<tr>
<td>SLA</td>
<td>L</td>
</tr>
<tr>
<td>SLA</td>
<td>L</td>
</tr>
<tr>
<td>SLA</td>
<td>L</td>
</tr>
<tr>
<td>RR</td>
<td>R</td>
</tr>
<tr>
<td>RR</td>
<td>R</td>
</tr>
<tr>
<td>RR</td>
<td>R</td>
</tr>
<tr>
<td>RR</td>
<td>R</td>
</tr>
<tr>
<td>RLAM</td>
<td>R</td>
</tr>
<tr>
<td>RLAM</td>
<td>R</td>
</tr>
<tr>
<td>RLAM</td>
<td>R</td>
</tr>
<tr>
<td>RLAM</td>
<td>R</td>
</tr>
</tbody>
</table>

**NOTE:** This table shows how the processor allocates memory automatically when certain instructions are executed. For example, the PLP instruction pulls a line address or low-order register (C,E,F,L); increments the SP, then pulls a page address or high-order register (A,B,D,H); and increments the SP again, leaving the SP two counts higher than its initial value.

### Register Organization

![Register Organization Diagram](image)

**Figure 2.1: Supplementary Instruction Information**
FIGURE 22: DECIMAL/HEXADECIMAL RELATIVE OFFSET TABLES
Interrupts

The 7803 has two interrupt request inputs which are accessible at the STD BUS backplane: NMIRQ* (pin 46) and INTRQ* (pin 44). The characteristics of these interrupts are:

NMIRQ* - Nonmaskable interrupt request cannot be disabled by the program.

The processor stores the address of the next instruction in its program on the Stack using the SP as a memory pointer, then jumps to memory address location 0066 hexadecimal. Any return-from-subroutine instruction may be used to resume the interrupted program, but a special RTN (Return from nonmaskable interrupt) instruction is included to inform any Z80 peripheral chips in the system that the interrupt is over.

INTRQ* - Maskable interrupt request can be disabled and enabled by the program, and can operate in one of three modes:

1. Mode 0 is identical to the 8080 interrupt system.

   The processor issues INTAK* (interrupt acknowledge), which is used as an enable signal by the interrupting device. During INTAK* the interrupting device places an instruction opcode on the 7803 Data Bus, which the processor will execute. Either a 1-byte or 2-byte opcode may be used. If the opcode is part of a multi-byte instruction, one or two additional bytes must be placed on the Bus following the opcode (for example, a jump instruction consists of a 1-byte opcode and two additional bytes of jump-address information).

   Note: The Z80 will execute one interrupt acknowledge cycle and issue one INTAK* pulse for a one-byte opcode, or two cycles with two INTAK* pulses for a 2-byte opcode. However, it will not generate INTAK* during any subsequent cycles that may be required by the specific instruction being executed.

2. Mode 1 is the implied vector mode, with the implied vector address equal to 0038 hexadecimal. In Mode 1, any INTRQ* results in a subroutine jump to 0038.

3. Mode 2 is the supplied vector mode. The user preloads Register 1 with the page address of an interrupt vector lookup table which is part of the program. When the interrupt is acknowledged by the processor, a single INTAK* pulse is issued which causes the interrupting device to place the correct memory line number of the interrupt vector lookup table onto the STD Data Bus. The processor will then go to the lookup table at the address supplied by the peripheral; read a 16-bit memory address from two sequential entries in the table (line address followed by page address), and jump to that location in memory.
INTRQ* is enabled by the ENI instruction, and disabled by any of the following:

- Power-on or reset
- The DS1 instruction
- Previous response to INTRQ*
- Previous response to NMIRQ*

Z80 Peripheral Chip Considerations: When used with Z80 peripheral chips, such as the P10 or S10, these considerations and others may apply:

- In Mode 2, the 1-byte vector supplied by the interrupting device must be an even number with bit 0 = 0. This is a requirement of the peripheral chips, not the 7803 which will accept odd or even vectors.

- In Mode 0 with either JS or JI instructions inserted, and in Modes 1 and 2, the interrupt routine should be terminated with RTI (for INTRQ*) or RTN (for NMIRQ*) instructions. These execute like RTS in the 7803, but the special opcodes inform the peripheral chips that the interrupt routine is over. The peripheral chips then respond by restoring the state of the serial Priority Chain.

It is recommended that the user thoroughly acquaint himself with all the characteristics of any peripheral chips before attempting the program design.
SECTION 5 - PROGRAM INSTRUCTION TIMING

Introduction

The execution of a program instruction is a sequential process. The time state clock is used to step the Z80 through a specific sequence for each instruction type. The execution time for each instruction is the total of the time states needed by the instruction, with the time state period set by the processor's clock oscillator.

An understanding of the Z80's instruction execution timing is important in real time programming, where the program's execution rate is precisely matched to the speed requirements of the application. When using a signal or logic analyzer, a knowledge of the time state sequence makes it possible to predict the data and control states present on the STD BUS backplane and at the Z80 chip pins at any given instant in the execution of a program (Figure 31).

Machine Cycles

Each transaction between the Z80 and its memory and I/O ports requires a distinct time period called a machine cycle. Machine cycles are composed in turn of time states, with specific activity occurring in each time state. Although the number of time states and machine cycles vary among different types of instructions, they are precisely predictable for any given instruction.

Figure 23 is a timing diagram for the STAD (STore Accumulator Direct) instruction. This instruction requires four machine cycles (M1 through M4) with a total of 13 time states. Four machine cycles are necessary because the instruction accesses memory four times.

![Figure 23: Processor Timing for STAD Instruction](image-url)
The first machine cycle in the instruction (M1 in Figure 23) is used to read and decode the operation code (opcode) from program memory. M1 is called the opcode fetch cycle, and can be identified by an active pulse on the STATUS 1* output from the 7803 (also at the M1* pin on the Z80 chip).

Many Z80 instructions use 2-byte opcodes. If the first byte has a hexadecimal value of CB, DD, ED, or FD, a second byte is required. The Program Counter is incremented and the M1 cycle is repeated to read the rest of the opcode. STATUS 1* is asserted a second time.

Each M1 cycle requires a minimum of four time states (T1 through T4 in Figure 25), but this may be stretched to up to 11 time states in some instructions, allowing time for the instruction to fully execute if no additional machine cycles are needed. The shortest instructions use one machine cycle with four time states; the longest require six machine cycles (two M1 opcode fetches plus M2 through M5 for additional memory accesses) with a total of 23 time states.

When the Z80 interprets the first opcode byte during M1, it will add additional machine cycles to the instruction if it finds that:

a. The instruction has a 2-byte opcode; and/or
b. The instruction has 1 or 2 additional bytes of data, memory address, port address, or relative offset appended to the opcode; and/or
c. The instruction requires the processor to access memory or an I/O port as part of the function performed by the instruction.

For example, the STAD instruction in Figure 23 is a 3-byte instruction (1-byte opcode plus 16-bit memory address in the two bytes appended to the opcode), and STAD is an instruction whose function is to store data in memory. Therefore STAD requires 4 machine cycles with M1 used to read the opcode, M2 and M3 used to read the specified memory address, and M4 used to perform the operation of storing data in memory.

**WAIT States**

Although the minimum number of time states in any given machine cycle is fixed, the user can insert one or more WAIT states in the cycle. WAIT states are added by driving the 7803's WAITRQ* line active during the T2 time state in the machine cycle where the WAIT state is desired (Section 3 for timing). The WAIT state is a do-nothing time period that can be used to interface slow memories to the 7803, or to cause the processor to pause while a slow system function (such as an analog-to-digital converter or arithmetic processor) completes its task. The effect of holding WAITRQ* active indefinitely is to halt the processor; when WAITRQ* is released, the processor resumes operation with no change in its internal data or control states.

Note that the Z80 adds one WAIT state to all I/O access machine cycles automatically. Additional WAIT states can be added by the user if desired.

Naturally the addition of WAIT states must be included in the computation of program execution time in real-time control applications. Each WAIT state requires one full time state clock period.
DMA Mode

Direct Memory Access (DMA) operations are controlled by driving the 7803's BUSRQ* line active when sampled at the end of any time state. The processor will complete the current instruction, then float its Data Bus, Address Bus, and many Control Bus lines (Figure 3). BUSAK* then goes active.

The BUSAK* output signifies that the 7803's 3-state bus drivers are in the OFF condition, allowing an alternate system controller card to operate the 7803's memory, I/O, and other peripheral cards. Internally, the Z80 is halted in a manner similar to the WAIT state, with internal data and control states unaffected by the DMA operation. BUSRQ* can be held active indefinitely, but the dynamic RAM refresh operation is halted during DMA operations.

Note: the 7803's onboard memory sockets are not accessible in DMA mode, and the processor can't be interrupted by INTRQ* or NMIRQ*.

Instruction Timing Table

The table in Figure 2.4 shows the actual number of memory bytes, machine cycles and time states required for all of the Z80 instructions. Two time state periods are included for convenience with the full execution time of the instructions shown for each.
<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>DESCRIPTION</th>
<th>BYTES</th>
<th>CYCLES</th>
<th>STATES</th>
<th>0.25 μs</th>
<th>0.50 μs</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOAD STORE</td>
<td>REGISTER TO REGISTER</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1.9</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>IMMEDIATE TO REGISTER</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>2.8</td>
<td>1.75</td>
</tr>
<tr>
<td></td>
<td>ACCUMULATOR TO OR FROM MEMORY DIRECT</td>
<td>3</td>
<td>4</td>
<td>13</td>
<td>5.2</td>
<td>3.25</td>
</tr>
<tr>
<td></td>
<td>ACCUMULATOR TO OR FROM MEMORY INDIRECT (BC) (DE) (HL)</td>
<td>1</td>
<td>2</td>
<td>7</td>
<td>2.8</td>
<td>1.75</td>
</tr>
<tr>
<td></td>
<td>REGISTER TO OR FROM MEMORY INDIRECT (HL)</td>
<td>4</td>
<td>2</td>
<td>7</td>
<td>2.8</td>
<td>1.75</td>
</tr>
<tr>
<td></td>
<td>IMMEDIATE TO MEMORY INDIRECT (HL)</td>
<td>2</td>
<td>3</td>
<td>10</td>
<td>4.0</td>
<td>2.50</td>
</tr>
<tr>
<td></td>
<td>REGISTER TO OR FROM MEMORY INDEXED (IX) OR (IY)</td>
<td>3</td>
<td>5</td>
<td>19</td>
<td>7.6</td>
<td>4.75</td>
</tr>
<tr>
<td></td>
<td>IMMEDIATE TO MEMORY INDEXED (IX) OR (IY)</td>
<td>4</td>
<td>5</td>
<td>19</td>
<td>7.6</td>
<td>4.75</td>
</tr>
<tr>
<td></td>
<td>ACCUMULATOR TO OR FROM INTERRUPT OR REFRESH</td>
<td>2</td>
<td>2</td>
<td>9</td>
<td>3.6</td>
<td>2.25</td>
</tr>
<tr>
<td>ACCUMULATOR, CARRY</td>
<td>AJAD, CMAL, CLAC, CLC, SEC</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1.9</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>CMIAA</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>3.2</td>
<td>2.00</td>
</tr>
<tr>
<td>ADD, SUBTRACT, LOGICAL</td>
<td>REGISTER</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1.9</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>IMMEDIATE</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>2.8</td>
<td>1.75</td>
</tr>
<tr>
<td></td>
<td>MEMORY INDIRECT (HL)</td>
<td>1</td>
<td>2</td>
<td>7</td>
<td>2.8</td>
<td>1.75</td>
</tr>
<tr>
<td></td>
<td>MEMORY INDEXED (IX) OR (IY)</td>
<td>3</td>
<td>5</td>
<td>19</td>
<td>7.6</td>
<td>4.75</td>
</tr>
<tr>
<td>INCREMENT DECREMENT</td>
<td>REGISTER</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1.9</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>MEMORY INDIRECT (HL)</td>
<td>1</td>
<td>3</td>
<td>11</td>
<td>4.4</td>
<td>2.75</td>
</tr>
<tr>
<td></td>
<td>MEMORY INDEXED (IX) OR (IY)</td>
<td>3</td>
<td>6</td>
<td>23</td>
<td>9.2</td>
<td>5.75</td>
</tr>
<tr>
<td>TEST BIT</td>
<td>REGISTER</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>3.2</td>
<td>2.00</td>
</tr>
<tr>
<td></td>
<td>MEMORY INDIRECT (HL)</td>
<td>2</td>
<td>3</td>
<td>12</td>
<td>4.8</td>
<td>3.00</td>
</tr>
<tr>
<td></td>
<td>MEMORY INDEXED (IX) OR (IY)</td>
<td>4</td>
<td>5</td>
<td>20</td>
<td>8.0</td>
<td>5.00</td>
</tr>
<tr>
<td>SET, CLEAR BIT</td>
<td>REGISTER</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>3.2</td>
<td>2.00</td>
</tr>
<tr>
<td></td>
<td>MEMORY INDIRECT (HL)</td>
<td>2</td>
<td>4</td>
<td>15</td>
<td>6.0</td>
<td>3.75</td>
</tr>
<tr>
<td></td>
<td>MEMORY INDEXED (IX) OR (IY)</td>
<td>4</td>
<td>6</td>
<td>23</td>
<td>9.2</td>
<td>5.75</td>
</tr>
<tr>
<td>SHIFT, ROTATE</td>
<td>ACCUMULATOR</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1.9</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>REGISTER</td>
<td>1</td>
<td>2</td>
<td>8</td>
<td>3.2</td>
<td>2.00</td>
</tr>
<tr>
<td></td>
<td>ACCUMULATOR MULTIPLE WITH MEMORY (HL)</td>
<td>2</td>
<td>5</td>
<td>18</td>
<td>7.2</td>
<td>4.50</td>
</tr>
<tr>
<td></td>
<td>MEMORY INDIRECT (HL)</td>
<td>2</td>
<td>4</td>
<td>15</td>
<td>6.0</td>
<td>3.75</td>
</tr>
<tr>
<td></td>
<td>MEMORY INDEXED (IX) OR (IY)</td>
<td>4</td>
<td>6</td>
<td>23</td>
<td>9.2</td>
<td>5.75</td>
</tr>
<tr>
<td>ADD SUBTRACT</td>
<td>ADD TO HL</td>
<td>1</td>
<td>3</td>
<td>11</td>
<td>4.4</td>
<td>2.75</td>
</tr>
<tr>
<td></td>
<td>ADD, SUBTRACT WITH CARRY TO HL</td>
<td>2</td>
<td>4</td>
<td>15</td>
<td>6.0</td>
<td>3.75</td>
</tr>
<tr>
<td></td>
<td>ADD TO IX OR IV</td>
<td>2</td>
<td>4</td>
<td>15</td>
<td>6.0</td>
<td>3.75</td>
</tr>
<tr>
<td>INCREMENT DECREMENT</td>
<td>INCREMENT, DECREMENT PAIR EXCEPT IX OR IY</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>2.4</td>
<td>1.50</td>
</tr>
<tr>
<td></td>
<td>INCREMENT, DECREMENT IX OR IY</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>4.0</td>
<td>2.50</td>
</tr>
<tr>
<td>LOAD</td>
<td>LOAD IMMEDIATE TO BC, DE, HL, SP</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>4.0</td>
<td>2.50</td>
</tr>
<tr>
<td></td>
<td>LOAD IMMEDIATE TO IX OR IX</td>
<td>4</td>
<td>4</td>
<td>14</td>
<td>5.6</td>
<td>3.50</td>
</tr>
<tr>
<td></td>
<td>LOAD HL TO OR FROM MEMORY DIRECT</td>
<td>3</td>
<td>5</td>
<td>16</td>
<td>6.4</td>
<td>4.00</td>
</tr>
<tr>
<td></td>
<td>LOAD BC, DE, SP, IX OR IY TO OR FROM MEMORY DIRECT</td>
<td>4</td>
<td>6</td>
<td>20</td>
<td>8.0</td>
<td>5.00</td>
</tr>
<tr>
<td></td>
<td>LOAD SP WITH HL</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>4.0</td>
<td>2.50</td>
</tr>
<tr>
<td></td>
<td>LOAD SP WITH IX OR IY</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>4.0</td>
<td>2.50</td>
</tr>
<tr>
<td>PUSH</td>
<td>PUSH AF, BC, DE, HL</td>
<td>1</td>
<td>3</td>
<td>11</td>
<td>4.4</td>
<td>2.75</td>
</tr>
<tr>
<td></td>
<td>PUSH IX OR IX</td>
<td>2</td>
<td>4</td>
<td>15</td>
<td>6.0</td>
<td>3.75</td>
</tr>
<tr>
<td>PULL</td>
<td>PULL AF, BC, DE, HL</td>
<td>1</td>
<td>3</td>
<td>10</td>
<td>4.0</td>
<td>2.50</td>
</tr>
<tr>
<td></td>
<td>PULL IX OR IX</td>
<td>2</td>
<td>4</td>
<td>14</td>
<td>5.6</td>
<td>3.50</td>
</tr>
<tr>
<td>BANK SELECT</td>
<td>REGISTER BANK AF OR BC/DE/HL</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1.6</td>
<td>1.00</td>
</tr>
<tr>
<td>EXCHANGE</td>
<td>DE WITH HL</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1.6</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>TOP OF STACK WITH HL</td>
<td>1</td>
<td>5</td>
<td>19</td>
<td>7.6</td>
<td>4.75</td>
</tr>
<tr>
<td></td>
<td>TOP OF STACK WITH IX OR IY</td>
<td>2</td>
<td>6</td>
<td>23</td>
<td>9.2</td>
<td>5.75</td>
</tr>
<tr>
<td>I/O</td>
<td>INPUT OR OUTPUT DIRECT</td>
<td>2</td>
<td>3</td>
<td>11</td>
<td>4.4</td>
<td>2.75</td>
</tr>
<tr>
<td></td>
<td>INPUT OR OUTPUT INDIRECT (C)</td>
<td>2</td>
<td>3</td>
<td>12</td>
<td>4.8</td>
<td>3.00</td>
</tr>
</tbody>
</table>

**Figure 24A: Z80 Instruction Timing Summary**
<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>INSTRUCTION</th>
<th>DESCRIPTION</th>
<th>BYTES</th>
<th>CYCLES</th>
<th>STATES</th>
<th>0.4 µs</th>
<th>0.25 µs</th>
</tr>
</thead>
<tbody>
<tr>
<td>JUMP</td>
<td>INDIRECT: LOAD PC WITH HL</td>
<td>1 1 4</td>
<td>1.6</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LOAD PC WITH IX OR IY</td>
<td>2 2 6</td>
<td>3.2</td>
<td>2.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JUMP TO INTERRUPT</td>
<td>1 3 9</td>
<td>1.4</td>
<td>2.75</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIRECT, ANY CONDITION</td>
<td>3 3 10</td>
<td>4.0</td>
<td>2.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RELATIVE</td>
<td>CONDITION: UN</td>
<td>2 3 12</td>
<td>4.8</td>
<td>3.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOT MET</td>
<td>2 3 7</td>
<td>2.8</td>
<td>1.75</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>UN</td>
<td>3 5 17</td>
<td>6.8</td>
<td>4.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE</td>
<td>CONDITION: MET</td>
<td>3 5 17</td>
<td>6.8</td>
<td>4.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOT MET</td>
<td>3 3 10</td>
<td>4.0</td>
<td>2.50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RETURN</td>
<td>CONDITION: UN</td>
<td>1 3 11</td>
<td>6.8</td>
<td>4.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOT MET</td>
<td>1 3 11</td>
<td>6.8</td>
<td>4.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOOP</td>
<td>D.C.BT</td>
<td>IF $0</td>
<td>2 2 8</td>
<td>3.2</td>
<td>2.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IF $0/0</td>
<td>2 3 13</td>
<td>5.2</td>
<td>3.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOCK LOAD</td>
<td>MOVE WORD</td>
<td>2 4 18</td>
<td>6.4</td>
<td>4.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLOCK</td>
<td>IF BC 0</td>
<td>2 4 16</td>
<td>6.4</td>
<td>4.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IF BC / 0</td>
<td>2 5 21</td>
<td>6.4</td>
<td>4.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOCK SEARCH</td>
<td>COMPARE WORD</td>
<td>2 4 18</td>
<td>6.4</td>
<td>4.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLOCK</td>
<td>IF BC 0</td>
<td>2 4 16</td>
<td>6.4</td>
<td>4.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IF BC / 0</td>
<td>2 5 21</td>
<td>6.4</td>
<td>4.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOCK INPUT-OUTPUT</td>
<td>INPUT OR OUTPUT WORD</td>
<td>2 4 18</td>
<td>6.4</td>
<td>4.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLOCK</td>
<td>IF $0</td>
<td>2 4 18</td>
<td>6.4</td>
<td>4.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IF $0/0</td>
<td>2 5 21</td>
<td>6.4</td>
<td>4.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTERRUPT</td>
<td>ENABLE/DISABLE INTERRUPT</td>
<td>1 1 4</td>
<td>1.6</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SET INTERRUPT MODE</td>
<td>2 2 6</td>
<td>3.2</td>
<td>2.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RETURN FROM INTERRUPTS</td>
<td>2 4 10</td>
<td>5.6</td>
<td>3.50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LOAD INTERRUPT REGISTER TO OR FROM ACCUMULATOR</td>
<td>2 2 9</td>
<td>3.6</td>
<td>2.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MISC</td>
<td>NOP</td>
<td>1 1 4</td>
<td>1.6</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HALT</td>
<td>1 1 4</td>
<td>1.6</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LOAD REFRESH REGISTER TO OR FROM ACCUMULATOR</td>
<td>2 2 9</td>
<td>3.6</td>
<td>2.25</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 24 B: Z80 Instruction Timing Summary**
Instruction Timing Example

The execution time for any routine or program segment is found by totalling all of the time states in all of the instructions executed. The factors affecting the execution time of a program segment are:

a. The clock frequency, which determines the time state period (Section 3).

b. The specific instructions used, which determine the number of time states in the segment (Figure 2.4).

c. The instantaneous Flag (Register F) bit states which summarize processor conditions when the conditional instructions (jump, jump-to-subroutine, return-from-subroutine) are executed (Figure 2.1).

d. The number of instruction loops within the instruction sequence, and the number of times each loop is executed (loop iterations).

e. If the program segment has more than one entrance or exit, every combination of routes through the segment that are used by the program should be considered.

The following example shows how to compute execution times in a program segment. The Z80 is programmed to generate a series of five short pulses at an output port bit line. Determine the overall execution time of the program segment and the period of the pulses generated (the output port bit lines are low when the segment is entered; only the bit 7 line is of interest).

---

Figure 2.5: Instruction Segment Timing Example
In the example in Figure 2.10, six of the program segment's nine instructions are within the loop and are executed five times each. Three of the instructions (LDBI, CLAC, OPA) are outside the loop and executed only once.

<table>
<thead>
<tr>
<th>FLOW DIAGRAM FUNCTION</th>
<th>TIMES PERFORMED</th>
<th>INSTRUCTIONS</th>
<th>TIME STATES</th>
<th>EXECUTION TIME IN 400 NS 7803 SYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set loop count = 5</td>
<td>Once</td>
<td>LDBI 05</td>
<td>7</td>
<td>2.8 us</td>
</tr>
<tr>
<td>Pulse output line once</td>
<td>Five times</td>
<td>CLAC</td>
<td>4</td>
<td>1.6 us</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OPA PORT 00</td>
<td>11</td>
<td>4.4 us</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LDAI 80</td>
<td>7</td>
<td>2.8 us</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OPA PORT 00</td>
<td>11</td>
<td>4.4 us</td>
</tr>
<tr>
<td>Test for end</td>
<td>Five times</td>
<td>DCB</td>
<td>4</td>
<td>1.6 us</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP ZO LOOP</td>
<td>10</td>
<td>4.0 us</td>
</tr>
<tr>
<td>Leave output line low</td>
<td>Once</td>
<td>CLAC</td>
<td>7</td>
<td>1.6 us</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OPA PORT 00</td>
<td>11</td>
<td>4.4 us</td>
</tr>
</tbody>
</table>

**FIGURE 2.10: SAMPLE TIMING CALCULATION**

The total execution time for the instructions performed once, outside the loop, is

$$2.8 + 1.6 + 4.4 = 8.8 \text{ us.}$$

One pass through the loop requires

$$1.6 + 4.4 + 2.8 + 4.4 + 1.6 + 4.0 = 18.8 \text{ us.}$$

The loop is repeated five times, so the total execution time for the program segment is

$$8.8 + [(5)(18.8)] = 102.8 \text{ us.}$$

The period of the pulses is found by adding the time the pulse is low to the time the pulse is high. The pulse is low from the end of the first OPA instruction to the end of the second:

$$2.8 + 4.4 = 7.2 \text{ us.}$$

The pulse is high from the end of the second OPA instruction until the end of the first (around the loop) or until the end of the third OPA (the fifth time through the loop):

$$1.6 + 4.0 + 1.6 + 4.4 = 11.6 \text{ us.}$$

The total period of each pulse is

$$7.2 + 11.6 = 18.8 \text{ us.}$$
SECTION 6 - MEMORY AND I/O MAPPING AND CONTROL

Memory Addressing

The 7803's 16-bit Address Bus can directly address a 65,536-byte (64K) memory. A specific memory location is addressed when these conditions are met:

a. The Address Bus contains the specific address of the memory location (0000 through FFFF hexadecimal);

b. MEMRQ* (memory request) and RD* (read) or WR* (write) control signals are active;

c. MEMEX* (memory expansion) is active.

Other factors affecting the 7803's control of its memory are:

a. In the Interrupt Acknowledge Cycle, the 7803 issues INTAK* in place of the memory enable signals, when responding to INTRQ*. This causes the interrupting device to provide an instruction or vector to the 7803 over the STD Data Bus.

b. The 7803 can pause to wait for a slow memory-mapped device, or be single-stepped, by inserting WAIT states in memory access machine cycles. See WAITRQ*, Section

c. The 7803 can disconnect from the STD BUS and enter the WAIT state while Direct Memory Access (DMA) operations are conducted by an alternate system controller card. DMA is controlled by the BUSRQ*/BUSAK* (Bus Request/Bus Acknowledge) signals.

A typical memory implementation is shown in Figure 2.8

12K-Byte Onboard Memory

The 7803 card has a combined EPROM/ROM and RAM memory on the card which is large enough to store the program and variable data required in many applications, without the need for additional external memory cards. The card is shipped with 1K of RAM and sockets which allow the user to add up to 8K of EPROM or masked ROM devices and to expand the RAM to 4K. The onboard memory sockets have addressing restrictions (Figure 2.7) and are not accessible in DMA operations.

The onboard memory is organized as follows:

a. EPROM/ROM sockets: provide capacity for four 2716 or equivalent single +5V supply EPROM devices which can be mixed in any combination with 2316E or equivalent masked ROMs. Each device is a 2048-byte (2K) read-only memory for a total capacity of 8192 (8K) bytes. All of these devices are supplied by the user.

b. RAM and RAM sockets: provides two 2114L or equivalent RAM devices organized as a 1024-byte (1K) memory, and sockets for six additional user-supplied 2114 RAMs. The 2114 is a 1024x4 device and two chips are required for each 1K of RAM added to the card. The total RAM capacity of the 7803 with all sockets loaded is 4096 (4K) bytes.

Figure 2.7 summarizes the addressing options for each of the memory chip sockets.
## Table: 7803 Onboard Memory Sockets Address Mapping

<table>
<thead>
<tr>
<th>MEMORY DEVICE DESIGNATION</th>
<th>FULL HEXADECIMAL ADDRESS FIELD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AS SHIPPED</td>
</tr>
<tr>
<td>ROM 0</td>
<td>0000 - 07FF</td>
</tr>
<tr>
<td>ROM 1</td>
<td>0800 - 0FFF</td>
</tr>
<tr>
<td>ROM 2</td>
<td>1000 - 17FF</td>
</tr>
<tr>
<td>ROM 3</td>
<td>1800 - 1FFFF</td>
</tr>
<tr>
<td>RAM U20,U24</td>
<td>2000 - 23FF (Note 2)</td>
</tr>
<tr>
<td>RAM U19,U23</td>
<td>2400 - 27FF</td>
</tr>
<tr>
<td>RAM U18,U22</td>
<td>2800 - 28FF</td>
</tr>
<tr>
<td>RAM U17,U21</td>
<td>2C00 - 2FFF</td>
</tr>
<tr>
<td>UNUSABLE (Note 3)</td>
<td>3000 - 3FFFF</td>
</tr>
</tbody>
</table>

### Notes:
1. Refer to Appendix A for remapping option.
2. 1K of RAM (two 2114L devices) mapped in addresses 2000-23FF are supplied with the 7803.
3. Maximum 7803 addressing range is 60K (12K onboard memory plus 48K on external memory cards) when the 7803 onboard memory is used. If the onboard memory is disabled (Appendix A), maximum system memory size without bank selection is 64K and no mapping restrictions are imposed by the 7803.
**Input/Output (I/O) Port Addressing**

The 7803 can address up to 256 each input ports and output ports. The port address appears on the low-order half of the Address Bus (A0-A7) and is repeated on the high-order half of the Address Bus (A8-A15). A specific I/O port is addressed when the following conditions are met:

a. The Address Bus (A0-A7) contains the specific address of the I/O port (00 through FF hexadecimal);

b. IORQ* (I/O Request) is active

c. IOEXP* (I/O Expansion) is active

d. RD* (read) is active to select an input port, or WR* (write) is active to select an output port.

The 8-bit input ports provide a means for reading data or status lines into the processor to take part in programmed operations. The 8-bit output ports provide a means for outputting program-generated data or control states. Typical input and output port circuits are shown in Figure 29.
This figure illustrates the Bus interface and I/O port address decoding circuitry and device types typically used to implement I/O ports. Pro Log's 7500, 7600, and 7900 Series I/O modules are similar to this example.
SECTION 7 - PROGRAM AND HARDWARE DEBUGGING

Microprocessor Logic State Analysis

An attempt at monitoring the execution of a microprocessor program in real time using a conventional multitrace oscilloscope will be found to be impossible for practical purposes. The capacity of the scope and the operator will be quickly exhausted by the following characteristics:

a. Parallel data and addresses. Data is transferred as byte-parallel information (the address bus is 2 bytes wide). Individual bits on these buses have little meaning in program debugging. It is necessary to see the full content of both busses at once, and a hexadecimal display of numeric values is much more meaningful than binary waveforms.

b. Display Trigger Qualification. As many as 20 signals (combined address and control signals) may be used simultaneously to qualify the enabling of a peripheral memory card, for example. In order to capture this event, the test instrumentation must also be trigger-qualified by the same group of signals. Conventional oscilloscopes lack the number of trigger channels and operating modes needed to interface with a processor system such as the 7803.

c. Data Bus Voltage Levels and Timing. The 7803 and all of its peripheral cards in a given system will drive the Data Bus at different times, and will do so with a variety of logic high and logic low levels, all of which are different but within specification.

This presents two problems:

1. The operator will find it difficult to identify the source of any given waveform on the scope display.
2. In order to see a specific data segment on the Data Bus, the operator will find it necessary to synchronize the display with the processor's software program rather than with the voltage output of any one element of system hardware.

The logic state analyzer solves these problems by displaying formatted high/low or numeric logic states rather than analog waveforms, and by offering enough trigger channels and coincidence logic to allow literal program/display synchronization.

A logic state analyzer is considered an essential troubleshooting aid for both program development and system maintenance in any 7803-based system where the needs of the Manufacturing Test and Field Service organizations are important considerations.

The logic state analyzer performs these basic functions:

a. Tracks the actual instruction sequence as the program executes, facilitating program debugging.

b. Monitors control states and data passing between the processor and the system it controls, allowing the system external to the processor card to be observed at the same time as the program flow, using the same display.

c. Provides a multi-qualified trigger to a conventional oscilloscope when analog measurements are unavoidable (e.g. propagation delay through a suspected memory device).
Instruction Diagnostic Tables

The Instruction Diagnostic Tables on the following pages are used with a logic signal analyzer. They show the type of data on the Data Bus for time states T1, T2, and T3 within each machine cycle, and the machine cycles within any given instruction. This information is useful when debugging a program or troubleshooting the 7803 or any hardware under the 7803's control.

In addition to expected data and processor status for T1, T2, and T3, the TIME STATES column in each machine cycle shows the total number of time states for that cycle. If there are one or more time states after T3, the processor is performing an internal operation; the signals at the Z80 chip pins are either unchanged from T3 or undefined, with no new information available until the next T1.

Because of the size of the Z80 instruction set, the Instruction Diagnostic Tables are separated into the following sheets by instruction type:

<table>
<thead>
<tr>
<th>INSTRUCTION CATEGORY</th>
<th>INSTRUCTION TYPE</th>
<th>FIGURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-Bit Register &amp; Memory Data</td>
<td>Load/Store</td>
<td>31A</td>
</tr>
<tr>
<td></td>
<td>Accumulator &amp; Carry</td>
<td>31A</td>
</tr>
<tr>
<td></td>
<td>Arithmetic &amp; Logical</td>
<td>31A</td>
</tr>
<tr>
<td></td>
<td>Increment &amp; Decrement</td>
<td>31B</td>
</tr>
<tr>
<td></td>
<td>Bit Test/Set/Clear</td>
<td>31B</td>
</tr>
<tr>
<td></td>
<td>Shift &amp; Rotate</td>
<td>31B</td>
</tr>
<tr>
<td>16-Bit Register &amp; Memory Data</td>
<td>Add &amp; Subtract</td>
<td>31C</td>
</tr>
<tr>
<td></td>
<td>Increment &amp; Decrement</td>
<td>31C</td>
</tr>
<tr>
<td></td>
<td>Load &amp; Store</td>
<td>31C</td>
</tr>
<tr>
<td></td>
<td>Push &amp; Pull</td>
<td>31C</td>
</tr>
<tr>
<td></td>
<td>Bank Select</td>
<td>31C</td>
</tr>
<tr>
<td></td>
<td>Exchange</td>
<td>31C</td>
</tr>
<tr>
<td>I/O</td>
<td>Input &amp; Output</td>
<td>31D</td>
</tr>
<tr>
<td>Address</td>
<td>Jump &amp; Return</td>
<td>31D</td>
</tr>
<tr>
<td>Compound</td>
<td>Loop</td>
<td>31D</td>
</tr>
<tr>
<td></td>
<td>Block Memory</td>
<td>31D</td>
</tr>
<tr>
<td></td>
<td>Move &amp; Search</td>
<td>31D</td>
</tr>
<tr>
<td></td>
<td>Block I/O</td>
<td>31D,E</td>
</tr>
<tr>
<td>Machine Control</td>
<td>Interrupt</td>
<td>31E</td>
</tr>
<tr>
<td></td>
<td>Halt, NOP</td>
<td>31E</td>
</tr>
</tbody>
</table>

FIGURE 30: INSTRUCTION DIAGNOSTIC TABLES INDEX
### Instruction Diagnostic Table

**Figure 31A**

<table>
<thead>
<tr>
<th>Address</th>
<th>Status</th>
<th>Data</th>
<th>Time</th>
<th>Address</th>
<th>Status</th>
<th>Data</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>READ</td>
<td>OPERA TION CODE</td>
<td>0</td>
<td>PC + 1</td>
<td>READ</td>
<td>OPERA TION CODE</td>
<td>0</td>
</tr>
<tr>
<td>REG</td>
<td>READ</td>
<td>OPERA TION CODE</td>
<td>0</td>
<td>REG</td>
<td>READ</td>
<td>OPERA TION CODE</td>
<td>0</td>
</tr>
<tr>
<td>REG</td>
<td>READ</td>
<td>OPERA TION CODE</td>
<td>0</td>
<td>REG</td>
<td>READ</td>
<td>OPERA TION CODE</td>
<td>0</td>
</tr>
<tr>
<td>REG</td>
<td>READ</td>
<td>OPERA TION CODE</td>
<td>0</td>
<td>REG</td>
<td>READ</td>
<td>OPERA TION CODE</td>
<td>0</td>
</tr>
<tr>
<td>REG</td>
<td>READ</td>
<td>OPERA TION CODE</td>
<td>0</td>
<td>REG</td>
<td>READ</td>
<td>OPERA TION CODE</td>
<td>0</td>
</tr>
<tr>
<td>REG</td>
<td>READ</td>
<td>OPERA TION CODE</td>
<td>0</td>
<td>REG</td>
<td>READ</td>
<td>OPERA TION CODE</td>
<td>0</td>
</tr>
<tr>
<td>REG</td>
<td>READ</td>
<td>OPERA TION CODE</td>
<td>0</td>
<td>REG</td>
<td>READ</td>
<td>OPERA TION CODE</td>
<td>0</td>
</tr>
<tr>
<td>REG</td>
<td>READ</td>
<td>OPERA TION CODE</td>
<td>0</td>
<td>REG</td>
<td>READ</td>
<td>OPERA TION CODE</td>
<td>0</td>
</tr>
</tbody>
</table>

---

**Legend:**
- **PC:** Program Counter
- **REG:** Register
- **READ:** Read
- **WRITE:** Write
- **OPERATION CODE:** Code for the operation
- **DATA:** Data being read or written
- **TIME:** Time taken for the operation

---

**Notes:**
- The table above illustrates the diagnostic table for machine cycles.
- Each row represents a specific operation, detailing the address, status, data, and time for each operation.
- The table is used to diagnose and analyze the behavior of machine cycles in a computer system.
<table>
<thead>
<tr>
<th>NUMBER OF INSTRUCTION WORDS</th>
<th>MACHINE CYCLES</th>
<th>ADDRESS</th>
<th>STATUS</th>
<th>DATA</th>
<th>1</th>
<th>DATA</th>
<th>EXECUTION TIME (MIC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PC READ MEMORY</td>
<td>0</td>
<td>MEMORY ADDRESS</td>
<td>(2)</td>
<td>4</td>
<td>0</td>
<td>1.0 1.00</td>
</tr>
<tr>
<td>2</td>
<td>PC READ MEMORY</td>
<td>0</td>
<td>MEMORY ADDRESS</td>
<td>(2)</td>
<td>4</td>
<td>0</td>
<td>4.0 2.50</td>
</tr>
<tr>
<td>3</td>
<td>PC READ MEMORY</td>
<td>0</td>
<td>MEMORY ADDRESS</td>
<td>(2)</td>
<td>4</td>
<td>0</td>
<td>5.0 5.00</td>
</tr>
</tbody>
</table>

FIGURE 31B: INSTRUCTION DIAGNOSTIC TABLE
<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>MACHINE CYCLES</th>
<th>MACHINE CYCLES</th>
<th>EXECUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>INSTRUCTION</td>
<td>STATUS</td>
<td>TIME</td>
</tr>
<tr>
<td>0</td>
<td>INPUT TO BUS</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>INPUT DATA FROM</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>ACCUMULATOR</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>JUMP ON CONDITION</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>JUMP UNCONDITIONAL</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>RETURN</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>RETURN ON CONDITION</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>RETURN ON CONDITION</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>JUMP TO INTERRUPT</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>RETURN ON CONDITION</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>JUMP TO ZEROPAGE</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>JUMP TO ZEROPAGE</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>JUMP TO ZEROPAGE</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>JUMP TO ZEROPAGE</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>JUMP TO ZEROPAGE</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>JUMP TO ZEROPAGE</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>JUMP TO ZEROPAGE</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>JUMP TO ZEROPAGE</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>JUMP TO ZEROPAGE</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>JUMP TO ZEROPAGE</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>19</td>
<td>JUMP TO ZEROPAGE</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>20</td>
<td>JUMP TO ZEROPAGE</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>JUMP TO ZEROPAGE</td>
<td>0</td>
<td>21</td>
</tr>
<tr>
<td>22</td>
<td>JUMP TO ZEROPAGE</td>
<td>0</td>
<td>22</td>
</tr>
<tr>
<td>23</td>
<td>JUMP TO ZEROPAGE</td>
<td>0</td>
<td>23</td>
</tr>
<tr>
<td>24</td>
<td>JUMP TO ZEROPAGE</td>
<td>0</td>
<td>24</td>
</tr>
<tr>
<td>25</td>
<td>JUMP TO ZEROPAGE</td>
<td>0</td>
<td>25</td>
</tr>
</tbody>
</table>

**FIGURE 310: INSTRUCTION DIAGNOSTIC TABLE**
The M824 is a logic signal analyzer designed specifically for program debugging and hardware troubleshooting in Z80-based systems such as the 7803 Processor.

Figure 32 below summarizes the ability of the M824 to capture, format, and display the information available from all the time states within a Z80 machine cycle at any instruction step in the program. The M824 operates in dynamic, single-step, and breakpoint modes; tracks interrupts and DMA operations; can pick instructions out of nested loops for display; and can trigger other test equipment with program instruction synchronization.

The M824 is portable and clips onto the Z80 on the 7803, eliminating the need for test probes and a long setup procedure.
APPENDIX A - 7803 USER STRAPPING OPTIONS

In new 7803 applications, system characteristics such as memory mapping are often arbitrary. The as-shipped configuration of the 7803 is recommended to minimize system assembly costs as well as field service and repair documentation efforts. Most other Pro-Log Series 7000 cards can be used with the 7803 without any jumper changes.

Jumper-wire strapping options are provided on the 7803 to allow processor upgrading in existing applications, firmware, and compatibility with similar cards from other manufacturers.

The strapping options for the 7803 are identified by the letters A through F on the Schematic (Pro Log document #103218), Assembly Diagram (#103219) and by silkscreened letters on the 7803 circuit card. The options include:

a. Clock (jumpers A and F): output clock to STD BUS, or input external clock signal in place of the 7803's crystal.

b. Mapping and Bank Control (jumpers B-E): remap or disable the onboard RAM and EPROM memory sockets, and allow external control of I/O and memory bank selection (IOEXP* and MEMEX* lines).

Clock (Figure 33)

Output: Some devices and instruments require access to the system clock. Jumper A (Figure ) places the system clock on STD BUS pin 43. Note that the clock output driver is not floated during DMA operations.

Input: an external clock can be used to drive the 7803's clock oscillator. This should be a TTL-compatible signal in the range of 1 to 5 MHz with a 25% to 75% duty cycle. The 7803's clock circuit will divide this signal's frequency in half, producing time states in the range 2000 ns to 400 ns.

The external clock input signal is assigned STD BUS pin 50 (CNTRL*). Remove the following components from the 7803: Crystal Y1; 2.2K resistors R3 and R4; 1000 pF capacitor C5. Replace C5 with a wire jumper. Add wire jumper F.

Figure shows the clock circuit before and after the external clock input modification.

Mapping (Figure 34)

The 7803's onboard memory sockets can occupy the lower quadrant of memory (0000-3FFF hexadecimal, as shipped) or the upper quadrant (C000-FFFFF), or be disabled.

Figure summarizes these selections and shows the jumpers required to obtain them.

Bank Selection (Figure 33)

Jumpers D and E hold MEMEX* and IOEXP*, respectively, active by connecting the bus traces to ground on the 7803 card. At least one additional 64K memory bank and one 256-1/0 port bank could be enabled on the same motherboard by employing memory and I/O cards which regard MEMEX* and IOEXP* as high level active signals.
Schematic coordinates B6,7,8

Internal clock with clock output on STD BUS

External clock drive with clock output removed from STD BUS

Jumper pad locations
D,E JUMPERS ON CARD WIRING SIDE

FIGURE 33:
Jumper options for external clock drive and clock output

Note: Some 7803 versions prior to May 1980 use E1,E2,E3...jumper notation instead of A,B,C....
MEMORY ADDRESS ASSIGNMENT | JUMPER WIRES
--- | ---
EPROM | RAM | UNUSABLE | B | C
--- | --- | --- | --- | ---
0000-1FFF | 2000-2FFF | 3000-3FFF | OPEN | JUMPER
C000-0FFF | E000-EFFF | F000-FFFF | JUMPER | OPEN
DISABLED | DISABLED | NONE | OPEN | OPEN

Schematic coordinates B4,5

Jumper shown in C (as shipped) position.

FIGURE 3A: ONBOARD MEMORY MAPPING OPTIONS
APPENDIX B: DOCUMENTATION