P&T-488 INTERFACE
INSTRUCTION MANUAL
Custom Software Package

PICKLES & TROUT
P. O. BOX 1206, GOLETA, CA 93116, (805) 685-4641
P&T-488 INTERFACE
INSTRUCTION MANUAL

copyright 1981 by

Pickles & Trout
P.O. Box 1206
Goleta, CA 93116
All Rights Reserved

WARRANTY

This Pickles & Trout product is warranted against defects in materials and workmanship for 90 days from the date of shipment. Pickles & Trout will, at its option, repair or replace products which prove to be defective within the warranty period provided they are returned to Pickles & Trout. Repairs necessitated by modification, alteration or misuse of this product are not covered by this warranty.

NO OTHER WARRANTIES ARE EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. PICKLES & TROUT IS NOT LIABLE FOR CONSEQUENTIAL DAMAGES.
This manual contains the information necessary to understand and use the P&T-488 interface as well as provide instruction in the basic concepts of the IEEE-488 bus.

Those who are already familiar with the IEEE-488 bus (also known as the HP-IB, GPIB and ASCII bus) and the concepts of a Talker, Listener and Controller may skip to the chapter "Installation of the P&T-488". It is recommended that those who are not acquainted with Talkers, Listeners and Controllers read the chapter "The IEEE-488 Bus" first.

The P&T-488 interface consists of two major components: the P&T-488 interface board and the P&T-488 custom system interface software package. The software package consists of a single program named PNT488. Also included is a program named 488TEST which performs a complete functional test of the P&T-488 interface board. Additional programs are provided as examples of how one can use the P&T-488 interface to communicate with 488 devices.
# Table of Contents

<table>
<thead>
<tr>
<th>Page</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IEEE-488 BUS</td>
<td>An introduction to the three primary occupants of the IEEE-488 bus: the Talker, Listener and Controller.</td>
</tr>
<tr>
<td>7</td>
<td>Hardware Description</td>
<td>A brief description of the P&amp;T-488 interface board. Instructions are included for changing the I/O port addresses used. The significance of each port is also explained.</td>
</tr>
<tr>
<td>12</td>
<td>Functional Test</td>
<td>Instruction on the use of the Functional test routine (488TEST). This routine performs a complete check of the operation of the P&amp;T-488 interface board and its 488 cable.</td>
</tr>
<tr>
<td>CS-1</td>
<td>Installation of the P&amp;T-488</td>
<td>A step by step account of how to install the P&amp;T-488 interface board and test it.</td>
</tr>
<tr>
<td>CS-2</td>
<td>Custom System Routines</td>
<td>A description of each of the routines in the Custom system package, and how each is to be used.</td>
</tr>
<tr>
<td>CS-2</td>
<td>Design Philosophy</td>
<td>An overview of the goals set for the design and use of the Custom Software Package.</td>
</tr>
<tr>
<td>CS-3</td>
<td>Jump Table</td>
<td>The organization and relative addresses of the various routines in the Custom Software Package.</td>
</tr>
<tr>
<td>CS-4</td>
<td>Single &amp; Double Byte Addresses</td>
<td>A description of the meaning of single and double byte 488 addresses, and a summary of how the P&amp;T-488 uses them.</td>
</tr>
<tr>
<td>CS-5</td>
<td>Serial Poll &amp; Service Request</td>
<td>An explanation of the interaction of the Service Request and Serial Poll functions.</td>
</tr>
<tr>
<td>CS-5</td>
<td>Summary of Functions</td>
<td>A summary of the IEEE-488 functions provided by the Custom Software Package.</td>
</tr>
<tr>
<td>CS-7</td>
<td>CNTRL</td>
<td>The routine which makes the P&amp;T-488 the Active Controller.</td>
</tr>
<tr>
<td>CS-8</td>
<td>GIM</td>
<td>A routine which gives the user direct access to the IFC, SRQ, REN and EOI lines.</td>
</tr>
<tr>
<td>CS-8</td>
<td>INIT</td>
<td>A routine which initializes the P&amp;T-488 interface and can optionally send an IFC on the 488 bus.</td>
</tr>
<tr>
<td>CS-8</td>
<td>LISTN</td>
<td>The routine which causes the P&amp;T-488 to become a Listener.</td>
</tr>
<tr>
<td>CS-9</td>
<td>PPIDL</td>
<td>A routine which puts the P&amp;T-488 Parallel Poll function into the Idle (PPIS) state.</td>
</tr>
<tr>
<td>CS-9</td>
<td>PPQRY</td>
<td>A routine which causes the P&amp;T-488 to perform a Parallel Poll.</td>
</tr>
<tr>
<td>CS-9</td>
<td>PISTT</td>
<td>Sets the &quot;ist&quot; (individual status) message of the P&amp;T-488 true.</td>
</tr>
</tbody>
</table>
## Table of Contents

**CS-10 PISTF**
Sets the "ist" message of the P&T-488 false.

**CS-10 Additional Comments**
A description of how the Parallel Poll response function of the P&T-488 may be used in an IEEE-488 system.

**CS-11 SPIDL**
A routine which puts the P&T-488 Service Request function into the idle (NPRS) state.

**CS-11 SPQRY**
A routine which conducts a 488 Serial Poll.

**CS-12 SPSRQ**
A routine which places a Service Request on the 488 bus and then responds to a Serial Poll.

**CS-12 STADR**
A routine which sets the Listen and Talk addresses of the P&T-488. It also sets the Parallel Poll Response byte and the End-of-String (EOS) character.

**CS-13 STATE**
The routine which summarizes the state of the P&T-488 interface.

**CS-13 State Table**
A table showing how the states of the various 488 interface functions are stored in memory.

**CS-15 TALK**
The routine which makes the P&T-488 a Talker.

**CS-15 XCTRL**
The routine which updates the state of the P&T-488 in response to commands sent by an external Controller.

**CS-17 User-Supplied Routines**
Brief descriptions of each of the routines the user must supply to complete the integration of the P&T-488 into his system.

**CS-17 Jump Table**
The required organization of the jump table for the user-supplied routines.

**CS-17 BREAK**
The routine which allows the user to defer or interrupt 488 communication after each byte.

**CS-18 BUFUL**
The routine which is called when the Listen buffer is full and cannot accept more bytes.

**CS-18 DVCLR**
A routine which is called whenever a Device Clear message is detected.

**CS-18 IFCLR**
A routine which is called when an IFC (sent by an external Controller) is detected.

**CS-18 NOLSN**
A routine which is called upon the discovery that there are no Acceptors on the bus when the P&T-488 is trying to be a Source (that is a Controller or Talker).

**CS-18 POC**
A routine which is called whenever an S-100 RESET or Power-On Clear has been detected.
Table of Contents

CS-18 SVCRQ A routine which is called upon detection of a 488 device requesting service.

CS-19 TRIGR A routine which is called whenever the Group Execute Trigger (GET) command is received from the Controller.

CS-19 XATN A routine which is called whenever an external Controller is attempting to issue commands.

CS-20 488 Bus Monitor A sample program which shows how to use the P&T-488 to monitor communication on the 488 bus. The source listing (in Intel mnemonics) is included.

CS-25 Sample Program A sample program which shows how to set up the P&T-488 as a Controller to send out bus commands and then become a Listener. The source listing (in Intel mnemonics) is included.

CS-30 DINK A sample program which allows the user to exercise most of the functions provided by the Custom Software package.

CS-35 Special Considerations A review of how the P&T-488 works and the consequences in terms of communications on the IEEE-488 bus and the operation of the S-100 computer.

*** Appendices ***

A-1 Unofficial Phrasebook A dictionary which expands the IEEE-488 Standard mnemonics into English. There are also some definitions, and many of the mnemonics are cross-referenced to the pages in the IEEE-488 1975 Standard document which define their meaning and use.

B-1 Functional Test Program Comments on how to modify the Functional Test program so that it can be used in any 8080 or Z-80 system. The source listing is included.

C-1 Bitwiggler™ Instructions on how to use the Audio Cassette input port of the P&T-488 to read tapes recorded in Kansas City format. The source listing of the Bitwiggler™ program is included. (Provided only when the software is supplied on cassette.)

D-1 Version 1.4 Listing The source listing for the Custom Software Package Version 1.4.

E-1 Code Assignments for Command Mode of Operation A table giving the binary, Hex and ASCII codes for the commands sent by the 488 Controller.
The 488 bus is populated by three major types of devices. One is the Controller, which sends commands over the bus to other devices. Another is the Talker, which sends data over the bus to one or more devices of the third kind: the Listeners. The Listeners and Talker communicate with a handshake on each data transfer, and the communication proceeds at the maximum rate allowed by the Talker and the slowest Listener. This communication is completely asynchronous and may be interrupted at specific points in the handshake cycle without causing any loss of data.

It can be useful to liken the bus to a meeting which has a chairman (Controller), a recognized speaker (Talker) and an audience (Listeners). As is true of most meetings, some of the audience is paying no attention whatever to the proceedings (some of the devices on the bus may be Idle), while some of those that are listening want to interrupt the Talker. Sometimes a member of the audience is audacious enough to indicate that it should be the chairman. The 488 bus specification allows the Controller to designate another device as his successor.

It is the Controller's responsibility to make sure that communication takes place in an orderly manner: it is he that says who can talk and who should listen at any given time. It is also the Controller that takes care of such matters as telling everyone to shut up (Universal Untalk UNT command), everyone to go back to their desks (Interface Clear IFC), or listen to someone trying to gain the floor (Service Request SRQ). Even though the Controller has (in theory) complete command over everyone else, problems can arise. One possible problem is that the Controller has made the unwise choice of telling more than one device that it can be a Talker, which results in sheer bedlam. Another way for the Controller to lose control of the situation is if a Talk Only (ton) device is placed on the bus. Some Talk Only devices are notoriously deaf and don't pay any attention to anybody, even the Controller!

A Talker, on the other hand, leads a simple life. It does not concern itself with disputes over who has the right to be heard, and when. It only puts data on the bus, waits until the slowest listener indicates it is ready for data, says the data is valid, waits until the slowest Listener says it has accepted the data, then says that it is removing the data and follows up on its threat. About the only thing that bothers a Talker is to find that no one is listening to him. Most get really upset and let the Controller know about this impolite state of affairs. Talkers that don't complain have a tendency to sit there with their mouths open, caught in mid-word. Either way, no communication is taking place and this is not considered a desirable state of affairs.

Listeners can be a little more complicated. They let the Talker know when they are ready for another word and when they have received it. Some also let the Controller know that they want some special attention. The Controller waits until the Talker can be interrupted so that no Listener is deprived of the latest bit of wisdom imparted by the Talker. Then the Controller tries to find out which device wants the attention. Two ways to do this are Serial Poll, in which each device is allowed to speak (one at a time) and Parallel Poll, which allows several devices to simultaneously inform the Controller of their need by a bit pattern each puts onto the eight data lines.
The 488 bus is made up of 16 signal lines: eight are used for data, three are needed for the interlocking handshake used to communicate the data, and the remaining five are used for bus management. Since there are eight data lines, a full eight bit byte can be communicated in each handshake cycle. This is what is meant by the phrase "bit parallel - byte serial" transmission. It is an alternative to the slower RS 232C standard, in which only one data line is used (and which is referred to as being a "bit serial" interface standard).

There are three basic concepts which are important to an understanding of how the hardware of the 488 bus works. The first is that only one of two voltages is allowed on each line, and the lower allowed voltage is ground. The second is that the 488 bus uses negative true logic, which means that the lower of the two voltage levels has the value TRUE, while the higher voltage has the value FALSE. The third is that the bus uses open-collector drivers. An open-collector driver can be thought of as a switch with one terminal connected to the line and the other to ground. When the driver is ON, it is as if the switch is closed, and so connects the line to ground. If the driver is OFF, it is as if the switch is open, so no connection is made between the line and ground. There is a resistor connecting the line to a voltage supply, so the voltage on the line rises to the higher of the two allowed levels if the line is not grounded. Since the 488 uses negative true logic, a line is given the value TRUE by turning the open-collector driver ON, or the value FALSE by turning the driver OFF. The phrases "active true" and "passive false" are used to describe this system; active true because the line must be actively connected to ground to impress a value of true on it, passive false because no action is needed (no connection is made) to make the value of the line false.
Each 488 device has one open-collector driver for each 488 line that it uses. More than one open-collector driver (that is, more than one 488 device) can be connected to each line. If all drivers are off the voltage on the line will be high, which means it has the value false. However, if one or more open-collector drivers are on, the line's voltage will be low, and it will have the value true. This is called a "wire-or" system because the logical value of the line is the logical OR of the logical values impressed on it by the several open-collector drivers connected to it. Thus each 488 device sends a true to the line by turning on its driver, or a false by turning the driver off. Note that if any device asserts a particular line true, that line will have the value true. However, if a device asserts a false (high) signal, it will be overridden by any device which asserts a true.

The eight data lines are named DIO1 through DIO8 (DIO stands for Data Input/Output). The least significant bit appears on DIO1, the most significant on DIO8. One point of possible confusion is that the data bits in an S-199 system are numbered 0 through 7, while the 488 data lines are numbered 1 through 8. Another is that S-199 systems assume positive true logic (high means TRUE, low means FALSE). Just remember that S-199 data bit 7 appears on DIO8, etc. and a 488 byte is the one's complement of an S-199 byte and everything should be all right.

The proper IEEE title for the three handshake lines is "Data Byte Transfer Control" lines. They are individually known as follows:
- DAV (Data Valid) - when true the data on the eight data lines is valid.
- NRFD (Not Ready For Data) - when true the 488 devices are not ready to accept data.
- NDAC (Not Data Accepted) - when true the devices have not yet accepted the data.

The remaining five lines are known as the "General Interface Management" lines. They are as follows:
- IFC (Interface Clear) - place all 488 devices in their default state.
- ATN (Attention) - used to distinguish between a Controller and a Talker.
- SRQ (Service Request) - indicates that a device needs attention.
- REN (Remote Enable) - allows 488 devices to be programmed either by their local controls (front panel switches, etc.), or by information sent over the 488 bus.
- EOI (End or Identify) - indicates the end of a string if ATN is false, otherwise it indicates a Parallel Poll is in progress.

- BYTE COMMUNICATION -

Byte communication requires that there be a device which is generating the byte to be communicated (the "source") and one or more devices which receive the byte (the "acceptors"). The Source and Acceptors communicate by use of an interlocking handshake using the three Data Byte Transfer Control lines (DAV, NRFD and NDAC). The byte itself is sent on the eight data lines (DIO1 through DIO8). The handshake is schematized in the following flow chart.
**SOURCE (SH)**

Set DAV high (false)

Are NRFD and NDAC both high (false)?

YES - error: no Acceptors on bus

NO - place the byte on D101-D108

Is NRFD false (high)?

NO - goto B

YES - continue

Has it been at least 2 microseconds since the byte was placed on the data bus?

NO - goto C

YES - assert DAV true (data available)

Is NDAC false (high)?

NO - goto D

YES - data has been accepted, so prepare to send next byte.

More data to send?

YES - goto A

NO - continue

Warn that data will change

Assert DAV false (high)

Remove data

Assert D101 through D108 false (high)

**ACCEPTORS (AH)**

Initialize handshake

Set NRFD, NDAC low (true)

Each Acceptor passively asserts NRFD false (high) as it becomes ready for data. The NRFD line goes high (false) when all are ready.

Is DAV true (low)?

NO - goto U

YES - as each Acceptor finishes getting the byte it passively asserts NDAC false and actively asserts NRFD true (low). When all have accepted the byte, NDAC finally goes false (high).

Is DAV false (high)?

NO - goto V

YES - actively assert NDAC true (low), because the new byte which has not yet been sent is not accepted yet

goto T
- A More Detailed Look at the 488 Inhabitants -

A TALKER is a device which sends data over the 488 interface to other devices. There are two major types and various subtypes. One major type is the Talk Only (ton), which may be used in a 488 system which has no Controller. This device always talks, and so it must be the only device which can talk. The other major type must be told when to talk ("addressed to talk"). A Controller is needed because it is the only kind of 488 device that is allowed to address Talkers and Listeners. All Talkers use the Source Handshake (SH) function to send a message over the 488 bus.

A LISTENER is a device which receives data over the 488 interface. As with the Talker, there are two major types: Listen Only (lon) and addressed Listener. A Listen Only device always listens to the 488 bus, while an addressed Listener listens only when the Controller tells it to. The Listen Only device can operate in a 488 system which does not have a Controller since it does not need to be told what to do and when to do it. All Listeners use the Acceptor Handshake (AH) function to receive messages on the 488 bus.

A CONTROLLER is a device which issues commands on the 488 bus. These include commands which are used to reset all devices on the bus Interface Clear (IFC), indicate which device is to Talk (when the Controller relinquishes the bus) and which devices are to Listen (i.e. it sends the Talk and Listen addresses of those devices over the bus), perform a Poll of 488 devices (Serial Poll and Parallel Poll), and a myriad of other special functions. The commands fall into two general classifications: Uniline and Multiline. Each uniline command uses only one line out of the five General Interface Management lines. Examples of uniline messages are Remote Enable (REN), Interface Clear (IFC) and Parallel Poll. Multiline messages use the eight data (DIO1-DIO8) lines to issue the command. Examples of multiline messages include performing a Serial Poll and commanding 488 devices to Talk or Listen. Multiline messages are sent using the Source Handshake (SH) function, just like a Talker. The way that a device determines whether it is hearing a Talker or the Controller is that the ATN (Attention) line is true (low) when the Controller is issuing a message, but false (high) when a Talker is saying something. The Controller is the device which controls the ATN line. Whenever ATN is true, all addressed Talkers shut up so that the Controller can say its piece. However, some Talk Only devices don't, and so they garble commands issued by the Controller. Generally speaking, a Talk Only device should be used only in a 488 system which has no Controller. Whenever the Controller passively asserts ATN false (lets it go high), the Talker (if any) begins to send its message.

- MULTILINE COMMANDS -

Telling a 488 device to Listen is one example of a multiline command. The Controller places the Listen address of the selected device on the data lines (DIO1 through DIO8) and then performs the Source Handshake (SH) function. In other words, it "speaks" the address while ATN is true (low). Whenever the Controller is active (that is, whenever ATN is true), all devices on the 488 bus interpret whatever is said (via the data lines and the Source Handshake function) as a command rather than data. ALL devices hear what is said by the Controller. They ALL execute the Acceptor Handshake function, without regard to whether they are normally a Talker, Listener or whatever.
Another example of a multiline command is the **Serial Poll**. The order of events is that the Controller sends out the Serial Poll Enable (SPE) command, which tells each device that when it is addressed as a Talker that it is to say either **SBN** (Status Byte – service Not requested) or **SBA** (Status Byte – service request Acknowledged). Those are the only two messages that are allowed. Then the Controller addresses each device as a Talker in turn and listens to the response of each. To conclude a Serial Poll, the Controller sends the Serial Poll Disable (SPD) command so that any device later addressed as a Talker can speak data (instead of SBN or SBA). Finally, the Controller performs whatever service is needed, which is device dependent.

---

**UNILINE COMMANDS**

An example of a uniline command is **Parallel Poll**. Parallel Poll is both simpler and more complicated than Serial Poll. It is simpler because only one command is given (Identify IDY: the logical AND of ATN and EOI) and all devices respond at once. It is possibly more complicated in that it may be more difficult to sort out which device wants service. Whenever a 488 device receives the IDY message, it immediately places its Parallel Poll Response byte on the eight data lines. For systems of eight devices or less, it is common for each device to be assigned a unique bit which it asserts true when it needs service. For example, one device would have a Parallel Poll response byte in which bit 1 is true if it needs service, otherwise bit 1 is false, and bits 2 through 8 are always false. Another device would use bit 2 to indicate its need for service and all other bits would always be false in its response byte. A third device would use bit 3. When a Parallel Poll is performed, the response sensed by the Controller will be the logical OR of all the Parallel Poll Response bytes (due to the fact that the 488 bus is a wire-or system). If the response has bits 1 and 3 true, and all other bits false, it means that the first and third devices need service, while the second does not.

If the 488 system uses more than eight devices, some alternate scheme must be used. One would be to have only eight devices respond to a Parallel Poll, and use Serial Poll on the remaining devices. Another scheme would be to have several devices share the same Parallel Poll Response byte. If the response to a Parallel Poll shows that at least one of the devices that shares a common response needs service, a Serial Poll can be used to find which ones they are.
- OVERVIEW -

The P&T-488 has four read/write registers which appear as four input/output (I/O) ports to the S-100 host machine. The ports are addressed as four consecutive I/O ports with the first port address an integral multiple of 4 (0, 4, 8, 12, ..., N*4, ..., FC). For ease of description these registers will be referred to as registers 0 through 3, even though what is called register 0 may be Port 0, 4, 8, ..., N*4, ..., FC.

The addresses used by the P&T-488 are set by means of a DIP switch on the upper left corner of the interface board. All boards are set at the factory for I/O ports 7C through 7F Hex, and all software supplied by Pickles & Trout assumes these addresses. The address used by both the board and the software can be changed by the user. The addresses used by the software and the board must be the same. To change the addresses assumed by the software, refer to the instructions given with the program.

To change the addresses used by the board, first note that the labels "A7" through "A2n" appear to the left of the switch. Switches A2 through A7 are set according to the following table:

<table>
<thead>
<tr>
<th>Address (Hex)</th>
<th>A7</th>
<th>A6</th>
<th>A5</th>
<th>A4</th>
<th>A3</th>
<th>A2</th>
</tr>
</thead>
<tbody>
<tr>
<td>00-03</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>04-07</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>08-0B</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>0C-0F</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>10-13</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>14-17</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>18-1B</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>1C-1F</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>20-23</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>24-27</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>28-2B</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>2C-2F</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>30-33</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>34-37</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>38-3B</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>3C-3F</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>40-43</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>44-47</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>48-4B</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>4C-4F</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>50-53</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>54-57</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>58-5B</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>5C-5F</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>60-63</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>64-67</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>68-6B</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
</tr>
</tbody>
</table>

-7-
<table>
<thead>
<tr>
<th>Address (Hex)</th>
<th>A7</th>
<th>A6</th>
<th>A5</th>
<th>A4</th>
<th>A3</th>
<th>A2</th>
</tr>
</thead>
<tbody>
<tr>
<td>6C-6F</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>70-73</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>74-77</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>78-7B</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>7C-7F</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>80-83</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>84-87</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>88-8B</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>8C-8F</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>90-93</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>94-97</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>98-9B</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>9C-9F</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>A0-A3</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>A4-A7</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>A8-AF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>AC-AF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>B0-B3</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>B4-B7</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>B8-BB</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>BC-BF</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>C0-C3</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>C4-C7</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>C8-CB</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>CC-CF</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>D0-D3</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>D4-D7</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>D8-DB</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>DC-DF</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>E0-E3</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>E4-E7</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>E8-EF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>EC-EF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>F0-F3</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>F4-F7</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>F8-FB</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>FC-FF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
</tbody>
</table>

For example, to address the P&T-488 interface board to use I/O ports 7C through 7F Hex, A7 must be ON and A2 through A6 OFF.

The P&T-488 allows direct access to the 8 signal lines of the IEEE 488-1978 (hereafter called 488) data bus (Register 2) and the 8 lines of the 488 Data Byte Transfer Control Bus and General Interface Management Bus (Register 1). In addition, a register is provided to allow a software settable response to a Parallel Poll (Register 3). Finally, a register is provided which indicates transitions occurring on the various 488 Control Bus and Management Bus lines (Register 9). Additional features of the P&T-488 include software disable of interrupts from the P&T-488 (without having to disable all interrupts of the S-100 system) and immediate response of the interface to Attention (ATN), Interface Clear (IFC) and Parallel Poll without intervention of the S-100 system's CPU.
The data transfer rate is highly dependent on the software, CPU and system memory of the S-100 system, but with the supplied software, an 8080 running at 2.0 MHz and no memory wait states, the transfer rate is about 3 KBytes/sec. For applications requiring higher rates, the same S-100 system can get data rates of over 9 KBytes/sec in the Talk Only mode.

REGISTER FUNCTIONS

<table>
<thead>
<tr>
<th>No.</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Interrupt Status (read only)</td>
</tr>
<tr>
<td>0</td>
<td>Interrupt Reset (write only)</td>
</tr>
<tr>
<td>1</td>
<td>Command Line Register (read and write)</td>
</tr>
<tr>
<td>2</td>
<td>Data Line Register (read and write)</td>
</tr>
<tr>
<td>3</td>
<td>Parallel Poll Response (write only)</td>
</tr>
</tbody>
</table>

REGISTER BIT MAP

<table>
<thead>
<tr>
<th>No.</th>
<th>I/O</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>IN</td>
<td>DAV</td>
<td>NRFD</td>
<td>NDAC</td>
<td>XIFC</td>
<td>XATN</td>
<td>SRQ</td>
<td>REN</td>
<td>POC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>OUT</td>
<td>DAV</td>
<td>NRFD</td>
<td>NDAC</td>
<td>XIFC</td>
<td>XATN</td>
<td>SRQ</td>
<td>TALK/</td>
<td>LISTN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>I/O</td>
<td>DAV</td>
<td>NRFD</td>
<td>NDAC</td>
<td>IFC</td>
<td>ATN</td>
<td>SRQ</td>
<td>REN</td>
<td>E01</td>
</tr>
<tr>
<td>2</td>
<td>I/O</td>
<td>D108</td>
<td>D107</td>
<td>D106</td>
<td>D105</td>
<td>D104</td>
<td>D103</td>
<td>D102</td>
<td>D101</td>
</tr>
<tr>
<td>3</td>
<td>OUT</td>
<td>D108</td>
<td>D107</td>
<td>D106</td>
<td>D105</td>
<td>D104</td>
<td>D103</td>
<td>D102</td>
<td>D101</td>
</tr>
</tbody>
</table>

NOTES:

+ means the bit goes low on a LOW to HIGH transition
- means the bit goes low on a HIGH to LOW transition

D1 means 488 interface interrupts are disabled
E1 means 488 interface interrupts are enabled

The 488 data lines are numbered from 1 to 8, while the data lines on the S-100 system are numbered 0 to 7

X as in XATN, XIFC signifies that some device other than the P&T-488 has made the level on the line (ATN or IFC) active true (low).
- REGISTER 3 -

This register holds the Parallel Poll Response byte. Whatever has been output to Register 3 will appear on the 488 data lines in response to a Parallel Poll (ATN and EOI).

- REGISTER 2 -

This register is connected to the 488 data lines through bus transceivers. The state of the data lines can be sensed by reading Register 2, and the P&T-488 will assert on the data lines whatever was last written into Register 2. However, if either the XATN flag or XIFC flag in Register 0 is set, the output buffers to the 488 bus are disabled which precludes assertion of what was last written into Register 2. Remember that the 488 bus uses negative logic so that any bit that is low is asserted (or logically true). Also the 488 bus is a wire-or system, so if any piece of equipment is asserting a particular line true, that line will be a logical true. But if a device asserts a false (high) signal, it is overridden by any device that asserts a true. Hence the terminology of active true and passive false. Thus if the P&T-488 is being used as a Listener all bits of Register 2 should be written high (logic false) so that the data asserted by the Talker can be properly read.

- REGISTER 1 -

This register allows direct setting and sensing of the 488 Control and Management bus lines. If the XIFC flag is set in Register 0, the interface will not assert any of the lines, regardless of what was last written into Register 1. Similarly, if XATN flag is set in Register 0, the interface will not assert any line except Not Ready For Data (NRFD) and Service Request (SRQ). SRQ will be asserted active true (low) only if the SRQ bit (bit D2) of Register 1 was written low. NRFD will always be asserted active true (low). The reason that NRFD is asserted true is so that the System Controller will not send any commands until the S-100 CPU is ready to accept them. Note that XATN has precedence over XIFC, so an externally applied IFC followed by an externally applied ATN will cause NRFD to be active true, SRQ to be true if the SRQ bit in Register 1 was written low, and all other 488 lines will be passive false.

- REGISTER 0 -

This is the Interrupt Status/Reset Register. Since the P&T-488 uses only one interrupt vector, one needs to be able to determine which condition caused the interrupt. Each bit of this register is associated with an interrupt-causing condition. By writing a low in the corresponding bits, one can individually reset the status bits associated with Data Valid (DAV), Not Ready For Data (NRFD), Not Data Accepted (NDAC), External Interface Clear (XIFC), External Attention (XATN) and Service Request (SRQ). If Bit 1 is set low status bit 7 will ignore any activity on the DAV line. This is useful when the interface is used as a Talker or Controller. If Bit 1 is set high, Bits 5 and 6 will ignore any activity on the NDAC and NRFD lines, which is useful when the interface is used as a Listener. If Bit 0 is set low, status Bits 0 (POC/RESET) and 1 (REN) will be cleared and the P&T-488 will be prevented from interrupting the S-100 system (but the interrupt status bits will continue to respond to 488 Control and Management line activity). If Bit 0
is set high the interface can interrupt the S-100 system.

If Bit 4 (IFC) of Register 1 is asserted there is no way of determining if an external Controller is also asserting IFC, so interrupt status bit 4 (XIFC) will ignore any activity due to an external Controller. A similar argument is true for ATN and XATN (Bit 3 of Registers 1 and 0). This is not a problem because the IEEE standard allows only the System Controller to assert IFC, and only the Controller-in-Charge may assert ATN. The standard further specifies that there may be no more than one System Controller and no more than one Controller-in-Charge.
The program 488TST81 performs seven different kinds of tests on the P&T-488 interface board and its 488 cable. The first group of four are done with no 488 device or test plug connected to the P&T-488. The last three are made with the special test plug connected to the P&T-488.

The program starts by printing a message to the operator to disconnect all 488 devices from the P&T-488. The operator signifies this has been done by pressing any key on the keyboard. After a key has been pressed the program begins its tests.

NOTE: Any time a Control C is pressed, the program is aborted and control is returned to the monitor (operating system).

The first test checks the data register (Register 2) by outputting a byte to the 488 data lines then reading the data lines to see if their state corresponds to the byte output to them. Each of the 256 possible bytes is tried in turn. If any errors occur, a message "DATA ERROR - bits in error are ..." with the bit names is printed. If there are no errors, no message is printed.

In a similar manner, the second test checks the command line register (Register 1). If there are any errors, the message "COMMAND LINE ERROR - bits in error are ..." is printed. Again, if there is no error, no message is printed.

The third test checks the Parallel Poll Response register (Register 3) by first making ATN and EOI true. Thus anything output to the Parallel Poll Response Register should appear on the 488 data lines. If the Command Line test failed with bits 0 and/or 3 in error, the results of this third test are meaningless. As with the first two tests, each of the 256 possible byte values is tried and any errors are reported: this time the error message is "PARALLEL POLL ERROR - bits in error are ...".

The fourth test checks the Interrupt Service Register (Register 0). If the second test failed, this one will probably fail also. Errors are reported with the message "INTERRUPT SERVICE REGISTER ERROR - bits in error are ...".

After these four tests have been made, (they take less than a tenth of a second), the operator is told to attach the special test plug and then press any key on the keyboard to continue the tests. The plug connects the eight data lines to the eight 488 command lines, so that the 488 cable can be tested for continuity, shorts or incorrect wiring. It also allows testing the response of the P&T-488 board to ATN and IFC asserted true by an external Controller.

The fifth test checks the 488 cable and reports any bits in error. If either the first (data line) or second (command line) tests failed, the results of this test will be meaningless. If the first four tests were passed without error, but this one shows errors, it means either the cable and/or test plug is open, shorted, miswired or improperly plugged. If all bits are in error, the 488 cable is either not connected to the P&T-488 interface board or the special test plug is not plugged into the cable.
The sixth test checks the response of the P&T-488 to an IFC (Interface Clear) presented by an external Controller. What is really done, of course, is to use the data port to assert a true on the IFC line through the special shorting plug, but the P&T-488 can't tell the difference between this and an external Controller making IFC true. The results are meaningful only if the first five tests passed with no errors.

The seventh test checks the response of the P&T-488 to an ATN (Attention) presented by an external Controller. The technique is the same as used in the sixth test. Again, the results are meaningful only if the first five tests were passed without any errors.

After the seventh test has been completed, the message NO ERRORS is printed if all tests were passed without error. Then the message "P&T 488 functional test complete" is printed and the program jumps back to the monitor.

WHAT TO DO IN CASE OF ERROR –

If any of the first four tests fail, check the following:

1. The P&T-488 interface board must be addressed to the same ports that the test routine tests. The base address (lowest address of the four) used by the P&T-488 must be in location 103 Hex for CP/M systems, 3003 Hex for North Star. The program is supplied with the base address set to 7C Hex.

2. All 488 devices must be disconnected from the P&T-488.

3. Make sure you are using the correct test routine. 488TST81 is to be used on ONLY Revision 81A boards (serial number 5000 and up). 488TEST is to be used on ONLY boards with serial numbers under 5000.

If any of the first four tests fail, try disconnecting the 488 cable from the P&T-488 interface board. If they STILL fail, the P&T-488 is faulty and should be returned to Pickles & Trout for repair or replacement. Be sure to include a printout of the test results. If the first four tests are passed without error after the cable has been disconnected, the cable is defective (a short between lines or a short to ground).

If no error message is printed before the "Attach test plug..." message to the operator, the first four tests were passed without error. If the error message "EXTERNAL ATN ERROR – bits in error are 2" is displayed, it is likely that you are using the wrong test routine. 488TEST is to be used on only boards with serial numbers under 5000; 488TST81 is to be used only on boards with serial numbers over 4999. USE THE CORRECT TEST. If the error message "EXTERNAL INTERFACE CLEAR ERROR – ..." is printed with no error message preceding it, the P&T-488 is faulty. If the error message "EXTERNAL ATN ERROR – ..." is printed, and either there is no other error message or only the EXTERNAL INTERFACE CLEAR ERROR message, the P&T-488 is faulty and should be returned for repair or replacement.

RETURN POLICY –

The P&T-488 interface board, its 488 connecting cable and the special test plug are warranted to be free of defects in materials and workmanship for 90 days from the date of sale. If they should be found faulty within the warranty period, Pickles & Trout will
(at its option) repair or replace them upon receipt of the defective pieces. Repairs necessitated by alteration, modification or misuse of these products are not covered by this warranty. Out of warranty interface boards which have not been modified or otherwise tampered with will be repaired or replaced for a flat fee. As of January, 1981, the fee is $45.00.

NOTICE - A handling fee of $45.00 will be charged for any board that is returned for repair because the wrong test routine was used. THIS INCLUDES BOARDS STILL IN WARRANTY.

When returning equipment to Pickles & Trout, be sure to include the following information:

1 NAME and ADDRESS of the owner.

2 NAME and PHONE NUMBER of the person who is using the P&T-488.

3 Description of the failure and how it was found. PRINTOUT OF THE TEST RESULTS IS REQUIRED.

4 Description of the S-100 machine and operating system. Include manufacturer and model name of the CPU board, system clock rate, and the name of the organization that authored the operating system, as well as any information on systemic modifications made to it.

For example: IMSAI 8080 with Ithaca Audio Z-80 CPU board with a system clock of 4 MHz, North Star single density 5.25" floppy disk drive and controller, Digital Research CP/M as modified by Lifeboat Associates for North Star disks.

5 If the equipment is still in warranty, enclose a copy of the bill of sale. Otherwise enclose a check for the repair and shipping and handling fees. The shipping and handling fee is $5.00 for addresses within the contiguous US, $7.50 for Alaska and Hawaii. There is no shipping fee for foreign addresses because the equipment will be returned freight collect.

The repairs/replacements will be made within five business days and the equipment returned postage paid to US addresses, freight collect to foreign addresses.
INSTALLATION of the P&T-488

The P&T-488 interface card uses four contiguous I/O ports, and is supplied configured to use ports 7C through 7F Hex. Be sure there is no port address conflict with other I/O boards in your S-100 system BEFORE installing the P&T-488. If it is necessary to change the I/O ports that the P&T-488 uses, refer to the chapter entitled "Hardware Description" for instructions.

When you are satisfied that there is no I/O port address conflict between the P&T-488 interface and other devices in your S-100 system, turn off the power to the S-100 system and wait at least twenty seconds (to allow sufficient time for the S-100 power supply to discharge) before installing the P&T-488 card. Attach the cable to the back panel of the S-100 system using the metric hardware supplied with the cable (this hardware mates with the standard lock screws used on 488 cables supplied by Hewlett-Packard, Beldon and others), and plug the cable onto the top edge connector of the P&T-488 interface card. Note that the plug and edge connector are keyed.

If the I/O port addresses of the board have been changed from 7C through 7F Hex, it will be necessary to modify 488TEST and PNT488. The fourth byte in the program 488TEST is supposed to contain the lowest address of the four that is used by the P&T-488 interface card. If, for example, the card has been addressed to use ports 60 through 63 Hex, you should change the value in the location BASPRT (103Hex of 488TEST) to 60 Hex.

The programs 488TEST and PNT488 should now be loaded so they can be modified (if necessary) and run. Programs supplied on cassette tape are recorded in Kansas City format and may be read by the BITWIGGLER™ (see Appendix C for the source listing) or any other cassette interface which understands the Kansas City format.

Next the P&T-488 should be tested for proper operation. Make any necessary modifications to 488TEST (see Appendix B for details) and then run the modified program. Refer to the chapter "Functional Test" for information about the meanings of the various messages.

After the test has been completed with no errors, you are ready to use the 488 interface. You will have to write a set of short routines to complete the integration of the P&T-488 with your particular system. The chapters "Custom Package Routines" and "User-Supplied Routines" define the purpose of each of the routines, and the chapters "488 Bus Monitor" and "Sample Program" each give examples of how the routines can be written and used.
<table>
<thead>
<tr>
<th>PAGE</th>
<th>ROUTINE</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS-7</td>
<td>CNTRL</td>
<td>Performs the Controller function (sends commands)</td>
</tr>
<tr>
<td>CS-8</td>
<td>GIM</td>
<td>Allows direct control of the General Interface Management lines</td>
</tr>
<tr>
<td>CS-8</td>
<td>INIT</td>
<td>Clears the interface, leaves all lines passive false</td>
</tr>
<tr>
<td>CS-8</td>
<td>LISTN</td>
<td>Performs the Listen-Only function</td>
</tr>
<tr>
<td>CS-9</td>
<td>PPIDL</td>
<td>Puts the Parallel Poll function in the Idle state</td>
</tr>
<tr>
<td>CS-9</td>
<td>PPQRY</td>
<td>Performs a Parallel poll</td>
</tr>
<tr>
<td>CS-9</td>
<td>PISTT</td>
<td>Sets the &quot;ist&quot; (individual status) message true</td>
</tr>
<tr>
<td>CS-10</td>
<td>PISTF</td>
<td>Sets the &quot;ist&quot; message false</td>
</tr>
<tr>
<td>CS-11</td>
<td>SPIDL</td>
<td>Puts the Service Request function in the Idle state</td>
</tr>
<tr>
<td>CS-11</td>
<td>SPQRY</td>
<td>Serial Poll query routine (performs a Serial Poll)</td>
</tr>
<tr>
<td>CS-12</td>
<td>SPSRQ</td>
<td>Service Request routine</td>
</tr>
<tr>
<td>CS-12</td>
<td>STADR</td>
<td>Sets the talker, listener addresses</td>
</tr>
<tr>
<td>CS-13</td>
<td>STATE</td>
<td>Passes information on the state of the interface to the user</td>
</tr>
<tr>
<td>CS-15</td>
<td>TALK</td>
<td>Performs the Talk-Only function</td>
</tr>
<tr>
<td>CS-15</td>
<td>XCTRL</td>
<td>Respond to an External Controller</td>
</tr>
</tbody>
</table>

**DESIGN PHILOSOPHY**

This software package was written with several objectives in mind. The first is that the routines should relieve the user of as much of the burden of dealing with the 488 bus protocol as possible. In place of having to test and respond to the signals on the bus the user need only set up a buffer (when appropriate) for the commands or data to be sent or received and then call a routine. The second is that ALL commands actually appear on the bus: there is nothing more frustrating than trying to debug a system in which a "smart" controller sees that it is going to address itself as a Talker, and then does so without putting the Talk address on the bus. The third consideration is that the design be closely identifiable with the state-space representation of bus functions. The memory locations TSTAT, LSTAT, etc. hold the present state of the interface functions. The fourth consideration is that the code for the interface routines be "pure" so that it can be put into ROM (Read Only Memory).
The routines supplied with the P&T-488 interface board allow it to act as a Controller, Talker or Listener, and provide the additional ability of conveniently handling commonly encountered situations. These include requesting service, either by means of the SRQ (Service Request) function or the PP (Parallel Poll) function, ceasing to request service, performing a Parallel Poll, performing a Serial Poll and responding to an external Controller (i.e., a Controller that is not the P&T-488 itself).

The P&T-488 interface depends heavily on the support software in order to communicate on the 488 bus. For this reason it is necessary for the S-100 system to execute P&T-488 routines in order to perform 488 bus functions. This includes not only the "assertive" functions, such as Talk and Control, but also the "responsive" functions, which include responding to a Serial Poll, being addressed as a Talker or a Listener by the Controller, etc. The only 488 bus function which the P&T-488 interface board can complete without any software intervention is respond to a Parallel Poll.

Communication between the S-100 system and the P&T-488 takes place by means of jump tables, state tables and string buffers. The user accesses routines within the P&T-488 software package by means of a jump table that resides within it. The user supplies several routines which are used by the P&T-488: these routines are accessed by means of a jump table which the user also supplies. The jump table within the P&T-488 interface software package is near the beginning and starts at memory location ENTBL. The user is expected to use it and it only as the means of calling the various P&T-488 routines. The reason the jump table should be used instead of going directly to the P&T-488 routine is that later versions of the interface software may change the location of the routine, while the placement of the jump to that routine in the jump table WILL NOT change. Thus if the user uses only the jump table, he can use subsequent versions of the interface software without changing his software in any way.

### P&T-488 Ver. 1.4 Jump Table Organization

<table>
<thead>
<tr>
<th>Routine</th>
<th>Entry Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>ENTBL</td>
</tr>
<tr>
<td>TALK</td>
<td>ENTBL+3</td>
</tr>
<tr>
<td>LISTN</td>
<td>ENTBL+6</td>
</tr>
<tr>
<td>STADR</td>
<td>ENTBL+9</td>
</tr>
<tr>
<td>CNTRL</td>
<td>ENTBL+12</td>
</tr>
<tr>
<td>GIM</td>
<td>ENTBL+15</td>
</tr>
<tr>
<td>STATE</td>
<td>ENTBL+18</td>
</tr>
<tr>
<td>XCTRL</td>
<td>ENTBL+21</td>
</tr>
<tr>
<td>SPQRY</td>
<td>ENTBL+24</td>
</tr>
<tr>
<td>SPSRQ</td>
<td>ENTBL+27</td>
</tr>
<tr>
<td>SPIDL</td>
<td>ENTBL+30</td>
</tr>
<tr>
<td>PPQRY</td>
<td>ENTBL+33</td>
</tr>
<tr>
<td>PISTT</td>
<td>ENTBL+36</td>
</tr>
<tr>
<td>PISTF</td>
<td>ENTBL+39</td>
</tr>
<tr>
<td>PPIDL</td>
<td>ENTBL+42</td>
</tr>
</tbody>
</table>

The P&T-488 interface software needs several user supplied routines in order to complete the integration into his system. It is expected that the user will provide a jump table which points to these routines. The details of the jump table and the operation of the routines appears in the section User-Supplied Routines.
Many of the P&T-488 interface routines cause the 488 interface functions to change state. The routine STATE allows the user to quickly determine the state of the more commonly desired interface functions. If the user needs additional detailed information about the states of the various interface functions he may look at the state table which is stored in memory starting at location TSTAT.

The P&T-488 routines which allow the S-100 system to be the 488 bus Controller or a Talker require strings which are stored in output string buffers. The user informs the P&T-488 routines of the location of the buffer by setting the register pair HL to the beginning address of the string and DE to the address of the end of the string before calling the P&T-488 routine. This technique allows the user flexibility in the definition of the strings and their length. For those strings which are needed on a recurring basis, the user may just point to that string rather than copying it into an intermediate buffer before calling the P&T-488 routine.

One other P&T-488 interface function may require a string buffer. That function is the 488 Listen routine. The conditions under which it needs a buffer are detailed in the description of the routine LISTN. If a buffer is needed, the location of that buffer is passed to the routine by the HL and DE register pairs, just as was done for the Talk and Control functions.

Single and Double Byte Addresses
And How the P&T-488 Uses Them

The IEEE-488 standard defines two general ways of addressing Talkers and Listeners. One way is by a single byte, and is called "single byte address" or "non-extended address". In terms of function mnemonics, the Talker function is known as the T Interface function, and the Listener as the L Interface function. The other method of addressing is known as "extended address" or "two byte address". The corresponding function mnemonics are TE and LE for Extended Talker and Extended Listener Interface functions, respectively. The P&T-488 and this software package are set up so that the P&T-488 may be addressed either way. If the Controller sends the primary Listen address of the P&T-488 and follows it with a secondary address, the secondary address is stored in the memory location LSTNS. If the primary address was not followed by a secondary address, a dummy secondary address of 7F Hex (which is an illegal secondary address) is stored in that location. The memory location TALKS is used in a similar manner to record the secondary address (or lack thereof) sent by the Controller after the P&T-488's primary Talk address. The user can make use of the optional secondary address for many different purposes. One example of a use of multiple secondary addresses is the following: Assume that the S-100 system is monitoring activity of the 488 bus and printing the results on its printer. Assume also that there are several different print formats possible and that the user wants the 488 Controller to be able to specify which format is to be used. One way of accomplishing this goal is to assign two different Listen addresses to the P&T-488: one for passing formatting information and the other for passing characters to be printed. The two addresses must have the same primary address and so differ only in the secondary address. Assume that the P&T-488 has been assigned the primary Listen address of 1 (21 Hex), and the secondary address for formatting information is b (62 Hex), while that for data to be printed is a (61 Hex). Whenever the S-100 system calls the Listen function it first looks at the memory location LSTNS to see what the secondary Listen address is. If it finds the character b, it interprets the string that is heard as formatting information. If it finds the character a, it prints the string, for it is data. And if it finds any other character it means that neither of these functions has been called for.

CS-4
This brings up a point that should be made about good practice concerning configuration of the IEEE-488 bus. It is generally a good rule to assign a given primary Listen or Talk address to only one 488 device. This way if an address gets garbled (the wrong secondary address sent with the proper primary address), it becomes obvious that there is an error.

Serial Poll and Service Request

Overview

The two functions Serial Poll and Service Request are closely intertwined. Basically, the Service Request function is used by a 488 device to tell the Controller that it needs some special attention. The Serial Poll function is used by the Controller to determine which one of the devices attached to the 488 bus is calling for help.

All 488 devices which have the Service Request function share the single 488 line known as SRQ. Any one which needs special attention asserts an active true (connects the line to ground). It can be seen that the SRQ line is false (high) only when all the devices do not need service. Since several devices share the one line, the Controller must find which device(s) need attention before it can service it (them). This is done by performing a Serial Poll, which consists of first informing all devices that a Serial Poll is going to begin (the Controller sends the Serial Poll Enable message), addressing each device as a Talker one at a time, and listening to its response. The response byte has a true (low) value on line DIO7 if that device is requesting service, and that device also asserts a passive false (high) on the SRQ line as it sends the response byte. If the device is not requesting service, line DIO7 is false (high).

When the Controller has finished the Serial Poll, it informs all devices that the function is finished by sending the SPD (Serial Poll Disable) message. This is done so that any device which is subsequently addressed as a Talker will speak normal data instead of the Serial Poll response byte.

Summary of Functions

IEEE-488 Functions Implemented

The IEEE-488 standard assigns mnemonics to the allowed subsets of each interface function, so a 488 device can be tersely but fully described by just a few words. The following table indicates what interface functions are implemented by the P&T-488 and Ver 1.4 software, and includes a brief description of the meaning of the mnemonics used.

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AH1</td>
<td>Complete Acceptor Handshake capability</td>
</tr>
<tr>
<td>SH1</td>
<td>Complete Source Handshake capability</td>
</tr>
<tr>
<td>T5</td>
<td>The device can operate as a Basic Talker, respond to a Serial Poll, be placed into a Talk Only mode of operation, and will unaddress itself as a Talker if the Controller sends its Listen Address. This last operation means that the device will cease being an addressed Talker when the Controller commands it to be a Listener.</td>
</tr>
</tbody>
</table>

CS-5
TE5
The device can operate as a Basic Extended Talker, respond to a Serial Poll, be
placed into a Talk Only mode of operation, and will unaddress itself as a Talker if
the Controller sends its Listen Address. This last operation means that the device
will cease being an addressed Talker when the Controller commands it to be a
Listener.

L3
The device can operate as a Basic Listener, can be placed into a Listen Only mode of
operation, and will unaddress itself as a Listener if the Controller sends its Talk
Address. This last operation means that the device will cease being an addressed
Listener when the Controller commands it to be a Talker.

LE3
The device can operate as a Basic Extended Listener, can be placed into a Listen Only
mode of operation, and will unaddress itself as a Listener if the Controller sends
its Talk Address. This last operation means that the device will cease being an
addressed Listener when the Controller commands it to be a Talker.

SR1
The device has complete Service Request capability.

RLO
The device has no Remote-Local function capability.

PP1
The device has complete Parallel Poll response capability. This means that the
Parallel Poll function can be configured by the Controller (which in turn means that
the Controller can assign a specific Parallel Poll response message to the device).

PP2
The device is not capable of being configured (assigned a Parallel Poll response
message) by the Controller. The response is assigned by the local message Ipe,
which in this case is done by the S-100 system.

The user should note that the PP1 and PP2 functions are mutually exclusive. The
P&T-488 and its associated software package have been constructed so that the user
could pick whichever function is most suited to his needs. But for proper operation
of the 488 bus, it is imperative that he use only one of the two functions in any
particular bus configuration.

DC1
The device has complete Device Clear capability.

DT1
The device has complete Device Trigger capability.

C1
The device can operate as the System Controller.

C2
The device can send IFC and take charge of the 488 bus.

C3
The device can send the REN (Remote Enable) message.
The device can respond to the SRQ (Service Request) message.

The device can send IF messages (e.g., Listen and Talk addresses, etc.), can perform a Parallel Poll and can Take Control Synchronously. However, the device cannot pass or receive control to or from another Controller.

The user should be aware of the fact that these are capabilities offered by the P&T-488 and that he does not have to use all of them. Indeed, some are mutually contradictory so he must not use both. The mutually exclusive capabilities offered are the T5/TE5 pair, the L3/LE3 pair and the PP1/PP2 pair. It is the user's obligation to pick at most only one function capability out of each of these pairs. It is allowable for the user to pick neither, but it is not allowable for the user to pick both.

**CNTRL**

*Become the Controller*

This routine is used to perform the various Controller functions, such as addressing Listeners, Talkers, sending Remote Enable, etc. It is important that this routine be called only when the user is sure that the DAV line is passive high, (i.e., take Control synchronously); otherwise there is the possibility of the current Talker being interrupted by the Controller while it is in the middle of transferring a byte of data. This could result in a spurious command being sent over the 488 bus and may destroy the data byte as well. In those cases where the P&T-488 is not participating in data transfer on the 488 bus but it is necessary for it to become the Controller from time to time, one can use the non-buffered Listener function provided by the routine LISTN to insure that the P&T-488 will take control synchronously. Note that the routines TALK and LISTN either return to the user's calling routine or call his routine BREAK at a point in the handshake cycle where a call to CNTRL will result in a synchronous assumption of the Controller function by the P&T-488.

The register pair HL must contain the address of the first character of the command string to be sent, DE contains the address of the last character of the string, and BC contains the address of the beginning of the user-supplied jump table. CNTRL calls the user routine BREAK after each character in the string has been sent (this allows the user to interrupt or defer further commands while other devices on the S-100 system are being serviced). If a Service Request (SRQ) is detected from some 488 device, a call is made to the user-supplied routine SVCRQ.

When CNTRL has finished sending the string of commands, it returns to the user's calling routine with the address of the last character sent in register pair HL, and the 488 lines ATN and DAV are left passive false. (Thus the P&T-488 has relinquished control of the bus.) If the P&T-488 has been selected as a Listener or is to perform Listen Handshake, the 488 line NRFD is left active true. This prevents the Talker from saying anything until the S-100 system has started execution of the routine LISTN. Finally, the Controller is left in STANDBY (CSBS in IEEE 488 notation). Thus the P&T-488 is assumed by the other programs to be the Controller-In-Charge until CSTAT (a memory location) is set to the Controller Idle State (CIDS) either directly by the user, or by the user executing the routine INIT.
GIM
General Interface Management

This routine allows the user to directly control several of the General Interface Management lines. A call to GIM is made with the appropriate bit pattern in the A register.

\[
\begin{array}{cccccccc}
D7 & D6 & D5 & D4 & D3 & D2 & D1 & D0 \\
X & X & X & IFC & X & SRQ & REN & EOI \\
\end{array}
\]

If a bit is high (positive logic 1), the corresponding line is made active true. Those bits marked by an X are disregarded. For example, if it is desired to make EOI active true, and IFC, SRQ and REN passive false, one would call GIM with 01 Hex in the A register. (Because of the disregarded bits, the A register could contain 09 Hex, 21 Hex, etc. without changing the result.) GIM returns to the calling routine with all registers restored except the accumulator and flags.

INIT
Initialize Interface

A call to INIT clears the P&T-488 by setting all data and control lines passive false, sets the Parallel Poll Response to all lines passive false, and sets all functions (Talker, Controller, Listener, etc) to their idle states. If the B register is zero when INIT is called, an IFC (Interface Clear) pulse is also sent on the 488 bus to initialize all devices to a known state. Note that only the Controller is allowed to send the IFC message, so the user should set register B non-zero if the P&T-488 is not the Controller.

LISTN
Listen-Only

This routine performs the Listen function, which allows another device on the 488 bus to send information to the S-100 computer. The information can be in any byte-oriented form: it may be ASCII characters with or without parity, it may be BCD values, binary values, etc.

The accumulator (A register) determines which of four modes is selected: if Bit 0 of A is 0 no buffer is used and the user must get the byte of data by looking at the A register each time BREAK is called. If the Bit 0 is 1 when LISTN is called, the data is put into a buffer as well as appearing in the A register each time BREAK is called. Bit 1 of the A register determines whether the Listen function will terminate on a End Of String (EOS) byte. If Bit 1 is 1, then an EOS will cause LISTN to return to the calling program. The routine BREAK is called as each byte of data is received, which allows the user to interrupt or defer further 488 transactions while he performs some other operation, or allows him to check each byte for special information.

The register pair HL must contain the address of the beginning of the listen buffer, and DE contain the address of the end of the buffer. Note that HL and DE need to be defined only if a buffer is used. The register pair BC contains the address of the beginning of the user-supplied jump table.

A jump is made to the user-supplied routine BUFUL when the buffer is filled, so the user can then transfer or otherwise manipulate the data and clear the buffer.
When the buffer is emptied, a call to LISTN will continue the transfer of data. LISTN returns to the calling routine when it senses EOI (End Or Identify) true.

The SRQ (Service Request) line is tested before each byte is received, and if it is active true, the routine determines whether the P&T-488 is the Controller-In-Charge. If it is, then a call is made to the user-supplied routine SVCRQ. After the user has serviced the Service Request, he need only execute a RETURN to continue listening from where LISTN left off.

This routine implements the Listen Only (Ion) function described in the IEEE-488 standard. Thus execution of this routine sets the Listen State byte to Listener Addressed. Execution of this routine also resets the Talk State byte to the Idle (TIDS) State.

If the user wishes instead to implement the Addressed Listen function described in the IEEE-488 standard (i.e., the transition from LIDS to LADS should occur only if the Controller has addressed the P&T-488 as a Listener), he should call the routine STATE and then call LISTN only if the Listen State byte shows that the P&T-488 is addressed to Listen.

The non-buffered Listen function can be used for those cases where the P&T-488 is not the Talker or Listener but is expected to assume Control from time to time. The technique is to use the LISTN routine but ignore the data. Each time BREAK is called is a time that the P&T-488 can assume Controller status without garbling a data byte. So each time BREAK is called the S-100 system determines whether it needs to become the 488 Controller: if so, it does so then, but if not it merely RETURNS to the calling routine. Note that the routine BREAK is called AFTER each data byte has been communicated; this technique will lock up the S-100 system until the Talker says something. If it turns out that there is no Talker or the Talker never speaks, there is no way for the S-100 system to regain control.

PPIDL
Parallel Poll Idle

This routine puts the Parallel Poll response function in the Idle state. Thus, whenever the Controller performs a Parallel Poll, the P&T-488 will give a non-affirmative response, regardless of the state of the "ist" (individual status) message and the Sense bit of the most recent PPE (Parallel Poll Enable) message received by the P&T-488.

PPQRY
Parallel Poll

This routine causes the P&T-488 to conduct a Parallel Poll. The response to the Parallel Poll is returned in the accumulator and also in the memory location LBYTE. Note that the IEEE-488 standard specifies that only the Controller is allowed to conduct a Parallel Poll; it is up to the user to refrain from using this routine unless the P&T-488 is the 488 Controller.

PISTT
Parallel Poll - ist True

This routine sets the "ist" (individual status) message in the P&T-488 true. If the sense bit of the most recent PPE (Parallel Poll Enable) message received by the P&T-488 is the same as the value of the "ist" message, (in this case, true), the
affirmative response byte is put into the Parallel Poll response register of the P&T-488. Otherwise, the non-affirmative response byte is put into the Parallel Poll response register. What all this means is that when the 488 Controller conducts a Parallel Poll, the P&T-488 will respond affirmatively if the sense bit of the PPE message was true, non-affirmatively if the sense bit of the PPE message was false. This routine also places the Parallel Poll function in the Standby (PPSS) state. Note that the Parallel Poll response will change if the routines PISTF or PPIDL are called or if the 488 Controller sends another PPE to the P&T-488.

PISTF
Parallel Poll – i.st False

This routine is the same as PISTT except that it sets the "ist" message false. Thus if the sense bit of the most recent PPE message received by the P&T-488 is FALSE, the AFFIRMATIVE response is put into the Parallel Poll Response register. Otherwise the NON-AFFIRMATIVE response is put there. Note that this is just the opposite of what happens when the routine PISTT is called. Execution of this routine places the Parallel Poll function in the Standby (PPSS) state.

Additional Comments
Parallel Poll – How to use it

There are several ways in which the Parallel Poll response function may be programmed using the P&T-488 and this interface software package. One way is for the 488 Controller (which may or may not be the P&T-488 itself) to address the P&T-488 as a Listener, send the PPC (Parallel Poll Configure) message, then send the PPE (Parallel Poll Enable) message. This will put the Parallel Poll function of the P&T-488 into the Standby (PPSS) state and also define which one of the eight 488 data lines will be used by the P&T-488 when the Controller performs a Parallel Poll. Another method is to put the PPE byte into the memory location reserved for the Parallel Poll response byte. This can be done by defining a five byte string consisting of the P&T-488’s Primary Talk address, Primary Listen address, Serial Poll Response byte, Parallel Poll response byte (the desired PPE message), and the EOS (End Of String) byte, then calling the routine STADR. This method defines the response byte, but the Parallel Poll response function of the P&T-488 still needs to be enabled (put into the Standby state). Do do this, a call can be made to the routine PISTT or PISTF. PISTT will make the "ist" message true, while PISTF will make it false. Since an affirmative Parallel Poll response is given only if the "ist" and sense bit of the PPE have the same logical value, one would call PISTT if he wanted the P&T-488 to respond affirmatively to a Parallel Poll and the PPE message was the character h, i, j, k, l, n or o.

By the use of the routines PISTT and PISTF one can readily cause the P&T-488 to give either a non-affirmative or an affirmative Parallel Poll response. One use of this ability would be to define an affirmative response as meaning that the S-100 system wants the Controller to perform some special function (which could be something as simple as to alert the operator that the printer is out of paper), and a non-affirmative response means that the Controller is to continue with normal operation. For the sake of a concrete example, assume that the P&T-488’s Listen address is the character l (21 Hex). Assume also that the Controller has sent the string ?<PPC>h? where the characters <PPC> mean that the PPC message (05 Hex) was sent, not that the five characters <, P, P, C and > were sent. Thus the sense bit of the PPE is true, and the P&T-488 is assigned to use data line DIO1 for its Parallel Poll response. Now assume that the S-100 system is listening to transactions on the 488 bus (via the Listen function of the P&T-488) and printing
each character on a printer as it is heard. Whenever the printer's status indicates
that it is out of paper, the routine PISTT should be called, for it will set the
"list" message true and cause the P&T-488 to respond affirmatively to a Parallel Poll.
When the printer has been serviced, the routine PISTF should be called so that the
P&T-488's response to a Parallel Poll will be non-affirmative.

One thing that the user should be aware of is that all Listeners which are in
the addressed state will be assigned the same Parallel Poll response byte when the
Controller sends the string <PPC><PPE>. This can give rise to utter confusion when a
Parallel Poll is actually executed, so it is wise to have the Controller explicitly
unaddress all Listeners (with the Unlisten command, which is the character ?),
address the Listener that is to have its Parallel Poll response byte configured, then
send the PPC and PPE message, followed by another Unlisten.

The P&T-488 along with this software package implements the full Parallel Poll
(PP1) function as defined by the IEEE-488 standard. As such, the function may be put
back into its Idle state (PPIS) by the Controller addressing the P&T-488 as a
Listener and sending the PPC character followed by the PPD character, or by the
Controller sending the PPU (Parallel Poll Unconfigure) message, or by calling the
routine PPIDL, which implements the "local poll not enabled" message defined in the
standard.

SPIDL
Service Request Idle

This routine resets the Service Request function to the Idle state. As a
consequence, it also insures that the P&T-488 is passively asserting SRQ false and
that the Serial Poll response byte is non-affirmative. Thus execution of this
routine is equivalent to the S-100 system making the local message rsv (request
service) false. This routine is the complement of the routine SPSRQ, which makes the
local message rsv true.

SPQRY
Serial Poll Query

This routine is called when the user wishes to determine (by means of Serial
Poll) which device is requesting service. The Talker addresses in the buffer are
sent out one by one and the response monitored to find which one is requesting
service. The routine returns when the appropriate device is found or when the buffer
with the Talker addresses is emptied.

The register pair HL must contain the address of the first byte of the Serial
Poll Query buffer, DE must contain the address of the end of the buffer, and BC the
first address of the user-supplied jump table. The Serial Poll Query buffer must
contain a character string made up of the Talk or Talk Extended addresses (in any
order) of the devices to be tested for Service Request.

This routine causes the Controller function of the P&T-488 to enter the Active
state, issue a UNL (Unlisten) message so that devices that had been addressed to
Listen will not hear the Serial Poll response bytes sent by each Talker, then issue a
SPE (Serial Poll Enable) message, and then send each Talk address in turn. As a
precaution against the possibility of a device not unaddressing itself as a Talker
whenever another Talk address is sent over the 488 bus, each Talk address is
preceeded by a UNT (Untalk) command. When a Talker responds affirmatively to the
Poll or when there are no more Talker addresses left in the buffer, this routine
issues a SPD (Serial Poll Disable) message and then returns to the calling program.

To allow for the possibility of addressing both normal (single address byte) and extended address (two address bytes) Talkers (otherwise known as T and TE Talkers), this routine sends the first address and then looks to see if a secondary address is to be sent also. If not, it listens for the Talker's response. If there is a secondary address to be sent, it sends it then listens to the Talker's response.

If a Talker responded affirmatively to the Serial Poll, the routine returns to the calling program with 00 Hex in the accumulator, the Serial Poll response byte in register B, and the register pair HL points to the buffer location that contains the Primary Address of that Talker. If no Talker responds affirmatively, the A register contains 40 Hex, register B contains the response of the last Talker, and HL points to the memory location holding the address of that last Talker.

Note that the IEEE-488 standard allows only the Controller to perform a Serial Poll. It is up to the user to insure that this routine is called by his programs only when the P&T-488 is the 488 Controller. Another point the user should be aware of is that this routine does not check for valid Talk addresses. It is the user's responsibility to put only valid Talk addresses in the buffer. Since the P&T-488 must wait for the addressed Talker to respond to the Serial Poll, if a non-existant Talk address is in the buffer, the P&T-488 will wait forever for the non-existant Talker to speak its Serial Poll response byte.

SPSRQ
Service Request

A call to this routine causes the P&T-488 to make the SRQ (Service Request) line active true and puts the Service Request function of the P&T-488 into the Service Request (SRQS) state. Thus execution of this routine is equivalent to the S-100 system making the local message rsv (request service) true. This routine then tests the Controller State of the interface. If it is Not Idle, a jump is made to the user-supplied routine SVCRQ. Otherwise the routine waits until the Talker address of the interface is sent out and responds properly to the Serial Poll performed by an external controller. After it has responded, the routine returns to the calling program. The register pair BC must contain the base address of the user-supplied jump table before this routine is called.

If the P&T-488 Controller state is Idle, the P&T-488 ignores all data communication on the 488 bus until it has been polled by the Controller. Thus if the P&T-488 had been a Listener, it will miss everything the Talker says between the time SPSRQ was called and a Serial Poll is conducted by the Controller.

STADR
Set Talker, Listener addresses

This routine copies the Talker and Listener addresses, Parallel Poll and Serial Poll Response bytes and the End Of String (EOS) byte from a table to the P&T-488 interface routines. The register pair HL must contain the address of the beginning of this table. Note that the Parallel Poll response byte is not copied into the interface Parallel Poll Response register. The Parallel Poll Response byte is interpreted in the same manner as the PPE/PPD (Parallel Poll Enable/Parallel Poll Disable) messages received from the Controller during a Parallel Poll Configure.
STATE
Show the state of the P&T-488

This routine passes abbreviated state information to the user in the A register and sets HL to the beginning of the State table. Thus the user can determine the states of the various interface functions if the abbreviated information returned in the A register is insufficient.

The states of various interface functions are mapped into the following bit positions of the A register:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Both Talk and Listen functions are idle</td>
</tr>
<tr>
<td>01</td>
<td>TIDS- (Not Talker Idle State)</td>
</tr>
<tr>
<td>10</td>
<td>LIDS- (Not Listener Idle State)</td>
</tr>
<tr>
<td>11</td>
<td>PPS- (Parallel Poll Idle State)</td>
</tr>
<tr>
<td>10</td>
<td>PPSS (Parallel Poll Standby State)</td>
</tr>
<tr>
<td>01</td>
<td>LOCS (Local State)</td>
</tr>
<tr>
<td>00</td>
<td>LWLS (Local With Lockout State)</td>
</tr>
<tr>
<td>11</td>
<td>REMS (Remote State)</td>
</tr>
<tr>
<td>10</td>
<td>RWLS (Remote With Lockout State)</td>
</tr>
<tr>
<td>0</td>
<td>CIDS (Controller Idle State)</td>
</tr>
<tr>
<td>1</td>
<td>CIDS- (Not Controller Idle State)</td>
</tr>
</tbody>
</table>

Example: If the Controller State is Not Idle, the Remote-Local State is LOCAL, Parallel poll is Idle and Talker Not Idle, the A register would contain 41 Hex.

The state table itself is comprised of six bytes, each one of which is associated with one 488 interface function. The actual state of the function is represented by the bit pattern of its associated state byte. Some states have the same bit pattern and are distinguished only by what routine is being executed. For example, if you look at the encoding for the Talk states you will find that TADS, TACS and SPAS are all represented by the same bit pattern. However, the P&T-488 interface software can distinguish among them by the fact that if it is not running either the Talk routine or the Serial Poll response routine, the state is TADS. If it is running the Talk routine, the state is TACS, and if it is running the Serial Poll response routine, the state is SPAS. The user does not need to concern himself with which one of the three states the Talk function is in because he only needs to know whether the Talk function has been addressed by a Controller, and he will make the inquiry at a time when neither the Talk nor the Serial Poll response routines are being executed.

State Table

<table>
<thead>
<tr>
<th>TSTAT</th>
<th>Talk Interface Function State byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>TIDS Talk Idle State</td>
</tr>
<tr>
<td>01</td>
<td>TADS Talk Addressed State</td>
</tr>
<tr>
<td>11</td>
<td>TACS Talk Active State</td>
</tr>
<tr>
<td>10</td>
<td>SPAS Serial Poll Active State</td>
</tr>
<tr>
<td>00</td>
<td>SPIS Serial Poll Idle State</td>
</tr>
<tr>
<td>11</td>
<td>SPMS Serial Poll Mode State</td>
</tr>
<tr>
<td>01</td>
<td>TPIS Talk Primary Idle State</td>
</tr>
<tr>
<td>10</td>
<td>TPAS Talk Primary Addressed State</td>
</tr>
</tbody>
</table>
### LISTEN Interface Function State byte

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Listen Idle State</td>
</tr>
<tr>
<td>...1</td>
<td>Listen Addressed State</td>
</tr>
<tr>
<td>...1</td>
<td>Listen Active State</td>
</tr>
<tr>
<td>.0..</td>
<td>Listen Primary Idle State</td>
</tr>
<tr>
<td>.1..</td>
<td>Listen Primary Addressed State</td>
</tr>
<tr>
<td>0...</td>
<td>non-buffered Listen function</td>
</tr>
<tr>
<td>1...</td>
<td>buffered Listen function</td>
</tr>
<tr>
<td>...0</td>
<td>do not return from Listen routine upon receipt of EOS message</td>
</tr>
<tr>
<td>...1</td>
<td>return from Listen routine upon receipt of EOS message</td>
</tr>
</tbody>
</table>

### SERVICE Request Interface Function State byte

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.00</td>
<td>Negative Poll Response State</td>
</tr>
<tr>
<td>.01</td>
<td>Service Request State</td>
</tr>
<tr>
<td>.10</td>
<td>Affirmative Poll Response State</td>
</tr>
</tbody>
</table>

### REMOTE-LOCAL Interface Function State byte

(Note: This function is not implemented, but these definitions will be used when it is.)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>...0</td>
<td>Local State</td>
</tr>
<tr>
<td>...1</td>
<td>Local With Lockout State</td>
</tr>
<tr>
<td>...1</td>
<td>Remote State</td>
</tr>
<tr>
<td>...1</td>
<td>Remote With Lockout State</td>
</tr>
</tbody>
</table>

### PARALLEL Poll Interface Function State byte

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>...0</td>
<td>Parallel Poll Idle State</td>
</tr>
<tr>
<td>...1</td>
<td>Parallel Poll Standby State</td>
</tr>
<tr>
<td>...1</td>
<td>Parallel Poll Active State</td>
</tr>
<tr>
<td>.0..</td>
<td>ist (individual status) message is false</td>
</tr>
<tr>
<td>.1..</td>
<td>ist message is true</td>
</tr>
<tr>
<td>.0..</td>
<td>Parallel Poll Unaddressed to Configure</td>
</tr>
<tr>
<td>.1..</td>
<td>Parallel Poll Addressed to Configure</td>
</tr>
</tbody>
</table>

### CONTROLLER Interface Function State byte

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>Controller Idle State</td>
</tr>
<tr>
<td>0001</td>
<td>Controller Addressed State</td>
</tr>
<tr>
<td>0010</td>
<td>Controller Transfer State (not yet implemented)</td>
</tr>
<tr>
<td>0011</td>
<td>Controller Active State</td>
</tr>
<tr>
<td>0011</td>
<td>Controller Parallel Poll Wait State</td>
</tr>
<tr>
<td>0011</td>
<td>Controller Parallel Poll State</td>
</tr>
<tr>
<td>0011</td>
<td>Controller Active Wait State</td>
</tr>
<tr>
<td>0110</td>
<td>Controller Standby State</td>
</tr>
<tr>
<td>1000</td>
<td>Controller Synchronous Wait State</td>
</tr>
<tr>
<td>...0</td>
<td>Controller Service Not Requested State</td>
</tr>
<tr>
<td>...1</td>
<td>Controller Service Requested State</td>
</tr>
<tr>
<td>.0.</td>
<td>System Control Not Active State</td>
</tr>
<tr>
<td>.1.</td>
<td>System Control Active State</td>
</tr>
</tbody>
</table>

CS-14
This routine allows the user to send data from the S-100 system to other devices on the 488 bus. The data may be in any byte oriented form: ASCII characters (with or without parity), BCD, binary, etc. The information is put into a buffer in memory before the routine is called.

The register pair HL must contain the address of the beginning of the buffer, DE must contain the address of the end, and BC the address of the beginning of the user-supplied jump table. If the accumulator (A register) contents are non-zero, the last byte in the buffer will be sent with EOI (End Or Identify) active true, otherwise the last byte will be sent with EOI passive false. All other bytes of the string are sent with EOI passive false.

A call is made to the user-supplied routine BREAK after each byte is sent, which allows the user to interrupt or defer further 488 bus transactions while he executes some other routine. To continue the Talk function, he need only execute a RETurn. All registers may be changed between the time BREAK was entered and the RETurn to the Talker routine was executed.

The SRQ (Service Request) line is checked after each byte is transmitted, and if it is active true, the routine determines whether the P&T-488 is the Controller-In-Charge. (Actually, CSTAT is tested to see if the Controller function is in the non-Idle state.) If it is the Controller-In-Charge, then a call is made to the user-supplied routine SVCRQ. After the user has serviced the Service Request, he need only execute a RETurn to continue talking from where the routine left off.

This routine implements the Talk Only (ton) function described in the IEEE-488 standard. Thus execution of this routine sets the Talk State byte to Talker Addressed. Execution of this routine also resets the Listen State byte to the Idle (LIDS) State.

If the user wishes instead to implement the Addressed Talker function described in the IEEE-488 standard, (i.e., the transition from TIDS to TADS should occur only if the Controller has addressed the P&T-488 as a Talker), he should call the routine STATE and then call TALK only if the Talk State byte shows that the P&T-488 is addressed to Talk.

XCTRL
Respond to External Controller

Each command presented by an external Controller (some device other than the P&T-488) is examined in turn and the states of the various interface functions are modified as necessary. A return is made to the calling program when the external Controller relinquishes the bus (asserts ATN passive false). An exception is made when the external Controller is conducting a Serial Poll: in this case the routine responds appropriately to the poll and returns to the calling program after the poll is concluded (a Serial Poll Disable command has been received followed by ATN going passive false).

This routine is to be called only upon ATN being made active true (low) by an external Controller. Load the register pair BC with the base address of the user-supplied jump table before calling XCTRL.
Since the states of the interface functions may have changed (due to commands from the external Controller), it may not be appropriate to return to the routine that was interrupted by the external Controller.
PAGE ROUTINE FUNCTION NAME

CS-17 BREAK Allows S-100 operations during buffered 488 communication

CS-18 BUFUL Fixup for Listen Buffer full

CS-18 DVCLR Application dependent. A Device Clear (DCL) was detected

CS-18 IFCLR Re-initialize due to 488 Interface Clear (IFC)

CS-18 NOLSN No listeners on 488 bus - ERROR

CS-18 POC Re-initialize due to S-100 Power-On Clear or Reset

CS-18 SVCRQ The 488 Service Request line is active true
Find the device and service it

CS-19 TRIGR Start whatever function that was waiting for Group Execute Trigger (GET)

CS-19 XATN Some other device made the 488 ATN line true

The P&T-488 interface software uses a jump table to access the user-supplied routines. It is the user's responsibility to provide the jump table, and it must have the form shown below. The user must set the register pair BC to the address of the first entry of the user jump table before calling routines supplied in the P&T-488 software package.

**User-Supplied Jump Table**

<table>
<thead>
<tr>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>JMP TRIGR</td>
</tr>
<tr>
<td>JMP DVCLR</td>
</tr>
<tr>
<td>JMP BUFUL</td>
</tr>
<tr>
<td>JMP IFCLR</td>
</tr>
<tr>
<td>JMP BREAK</td>
</tr>
<tr>
<td>JMP NOLSN</td>
</tr>
<tr>
<td>JMP SVCRQ</td>
</tr>
<tr>
<td>JMP POC</td>
</tr>
<tr>
<td>JMP XATN</td>
</tr>
</tbody>
</table>

**BREAK**

After each data byte or command is transferred on the 488 bus, a call is made to BREAK. The accumulator (A register) contains the byte last communicated, and the register pair HL points to the buffer location of the last byte sent or received. This routine allows the user to interrupt or defer until later any further 488 transactions, so that he may perform other operations. Examples include polling the keyboard for operator input, performing a background print routine, etc. It also
User-Supplied Routines

P&T-488

gives the user the opportunity to regain control of the S-100 system short of pushing RESET or turning off the power.

The BREAK routine is also useful for those cases in which the Talker does not make EOI true on the last byte; since the routine LISTN does not return to the user's calling routine until it sees an EOI (or optionally an EOS), one can see there is a fundamental problem. However, since a call is made to BREAK after each byte, the user can test each byte and determine if it is the end of transmission.

The only register that needs to be preserved is the Stack Pointer (SP). Transactions on the 488 bus may be resumed by executing a RETurn.

**BUFUL**

Listen Buffer Full

A jump is made to this routine when the Listen buffer is filled. The user should empty or redefine the buffer, then continue Listening by reinitializing all registers (A, BC, DE and HL) and calling LISTN.

**DVCLR**

Detected a Device Clear

A jump is executed to this routine whenever the Controller sends a Device Clear command. The user should perform whatever function Device Clear means in his system. (The proper response is device dependent.)

**IFCLR**

Detected an Interface Clear

A jump is made to this routine whenever an external Controller sends an Interface Clear (IFC) command. The P&T-488 must be re-initialized (for example, use INIT followed by STADR).

**NOLSN**

Nobody's Listening

A jump is made to this routine whenever the P&T-488 was to have said something as a Talker but found that no one was Listening. This is an error condition: correct it, reinitialize the registers and then call TALK again. (The only time that Not Ready For Data (NRFD) and Not Data Accepted (NDAC) can both be false at the same time is if there are no Listeners. It is this condition that causes a jump to NOLSN.)

**POC**

S-100 Power-On Clear or Reset

A jump is made to this routine whenever the P&T-488 interface senses an S-100 Reset or Power-On Clear. It will have to be re-initialized (use INIT followed by STADR).

**SVCRQ**

488 Service Request

This routine is CALLe d whenever the 488 Service Request (SRQ) line is true and the P&T-488 is the Controller-In-Charge. Find the device (by using SPQRY), service CS-18
it, then execute a RETurn to resume 488 transactions. The only register that needs to be preserved is the Stack Pointer.

TRIGR
488 Group Execute Trigger

This routine is CALLe d whenever the Group Execute Trigger (GET) command is received. Start whatever function was waiting for the trigger, then RETurn to resume 488 transactions. The only register that needs to be preserved is the Stack Pointer.

XATN
An External Controller wants Control

This routine is CALLe d whenever some other device on the 488 bus has made ATN active true (low). Call STATE to get the present Talker, Listener, etc. state information. Save this information, put the base address of the user-supplied jump table in register pair BC and call XCTRL. Then call STATE again to find out if the external Controller has changed the states of the Talk, Listen, etc. functions. If not, just execute a RETurn to resume 488 transactions from where they were interrupted by the external Controller. If the states are changed, perform whatever function the external Controller has commanded.
488 Bus Monitor

Description

This program shows all data and all commands sent over the IEEE-488 bus. Common non-printing characters (space, horizontal tab, carriage return and line feed) are shown as a message enclosed in angle brackets. As an example, "<HT>" is printed on the console printer each time a horizontal tab is detected.

The program begins by placing dummy Listen and Talk addresses in the interface. The parity bit is set (logic 1), so there is no way that the 488 interface can be addressed as either a Listener or a Talker by the Controller. (The parity bit of each address sent by the Controller is set to zero before comparing it to the interface Listen and Talk addresses.)

After the addresses are set up, the interface is cleared by a call to the routine INIT. Note that the B register is non-zero because we do not want to send an IFC (interface clear) signal over the bus. Only the System Controller is allowed to send IFC, and we are not he.

Then the interface is set to the Controller Standby state (at statement label RST2) which causes the 488 routines to assume that we are the Controller-in-Charge. We are not, but this is done so that the Listen routine will branch to the user-supplied routine SREQ each time a Service Request (SRQ) is detected. Otherwise there is no easy way of making this program print a special message each time a Service Request is pending.

Finally the Stack Pointer is reset, register pair BC is set pointing to the jump table of user-supplied routines, and the Listen routine is called. No buffer is used and the End-of-String (EOS) byte is ignored. The Listen routine will return each time it receives an END byte (a data byte with the EOI line active true). A special message is printed on the system console to show that an END byte was received and then the program is restarted.

The user-supplied routine BRK is called each time a byte of data or command appears on the 488 bus. All printing characters are sent to the console printer as is and a RETURN is made to the calling routine. The non-printing characters space (20 Hex), Horizontal Tab (9) and Line Feed (OA Hex) are replaced with the messages <SPACE>, <HT> and <LF> respectively. The non-printing character Carriage Return (OD Hex) causes the message <CR> to be printed followed by a carriage return and a line feed.

The user-supplied routine XTN prints a message to show that a Controller is active (ATN active true) and then calls the routine XCTRL to listen to the commands sent by the Controller. Each byte sent by the Controller is placed in location LBYTE and a branch is made to the routine BRK.

Special Cases:

Each time the Controller becomes active (asserts ATN active true), a carriage return-line feed is sent to the console device, followed by the string "COMMAND:", followed by another carriage return-line feed pair. Similarly, each time the Controller becomes inactive (ATN is false), a carriage return, line feed, the string "DATA:", carriage return and a line feed is sent to the console. All characters
printed after "COMMAND:" and before "DATA:" are sent by the Controller, and are instructions to the various 488 devices (for example, "?" means Unlisten, which means that no device should be a Listener when the Controller relinquishes the bus).

All characters which are printed after "DATA:" and before "COMMAND:" are data (otherwise known as device-dependent messages). They may be readings from a DVM which has been commanded to be a Talker, etc.

; 488 BUS MONITOR PROGRAM
;
; All activity on the 488 bus is shown by messages printed on the console printer.
;
; ORG 0B700H
;
CSTAT EQU 800AH ;controller state byte
ENTBL EQU 8026H ;memory addr of beginning of P&T-488 jump table
;
INIT EQU ENTBL
LISTN EQU ENTBL+6
XCTRL EQU ENTBL+15H
STADR EQU ENTBL+51H
;
MNITR EQU 0000H ;system monitor entry address
PRT EQU 0D106H ;console print routine entry address
;
; It is assumed that the routine PRT prints the character held in the A register, then returns to the calling routine. All registers (except the flags) are assumed to be unmodified by PRT.
;
START: LXI SP,STAK ;initialize the stack pointer
LXI H,DUMAD ;set up dummy listen, talk addresses
CALL STADR
RSTRT: MVI B,2 ;clear 488 interface, but do not send IFC
CALL INIT
RST2: MVI A,6 ;set CSTAT to standby (thus fooling the other routines into jumping to SVCRQ
STA CSTAT ;upon detection of a service request)
LXI SP,STAK ;initialize stack pointer
LXI B,JTBL ;set up pointers
MVI A,0 ;non-buffered listener, ignore EOS byte
CALL LISTN
LXI B,ENDMS ;show that an <END> message has been received
CALL MSG
JMP RST2
;
;
USER-SUPPLIED JUMP TABLE

CS-21
; JTBL:  JMP  TRGR
       JMP  DVCL
       JMP  BFL
       JMP  ICLR
       JMP  BRK
       JMP  NLS
       JMP  SREQ
       JMP  POCRST
       JMP  XTN

; TRGR:  LXI  B,TMS  ;print trigger message
        CALL  MSGCR
        RET

; DVCL:  LXI  B,DVMS  ;print device clear message
        CALL  MSGCR
        RET

; BFL:   LXI  B,BMS  ;we should never get this message
        CALL  MSGCR
        JMP  MNITR

; ICLR:  LXI  B,IFMS  ;print interface clear message
        CALL  MSGCR
        JMP  RSTRT  ;restart (initialize 488 interface)

; LOOK AT THE LAST COMMUNICATED CHARACTER

; BRK:   CPI  0DH  ;<CR>?
        JZ  CRMSG  ;...print <CR> message
        CPI  0AH  ;<LF>?
        JZ  LFMSG  ;...print <LF> message
        CPI  9  ;<HORIZONTAL TAB>?
        JZ  HTMSG  ;...print <HT> message
        CPI  20H  ;<SPACE>?
        JZ  SPMSG  ;...print <SPACE> message
        CALL  PRT  ;print char
        RET

; CRMSG:  PUSH  A  ;save character for later
         LXI  B,CRMS  ;print <CR> message
         CALL  MSG
         POP  A
         CALL  PRT  ;then do the carriage return
         MVI  A,0AH  ;finish with a line feed
         CALL  PRT
         RET

; LFMSG:  LXI  B,LFMS  ;print <LF> message
         CALL  MSG
         RET

; HTMSG:  LXI  B,HTMS  ;print <HT> message
         CALL  MSG

P&T-488

488 Bus Monitor Program

; RET
; SMSG: LXI B,SPMS ;print <SPACE> message
    CALL MSG
    RET
; NLS: LXI B,NLMS ;we should never reach this point
    CALL MSGCR ;but if we do, print message and
    JMP MNITR ; go to the monitor
; SREQ: LXI B,SREQS ;print service request message
    CALL MSGCR ;let the controller-in-charge take care
    JMP MNITR ; of the service request
; POCSRST: LXI B,POCSRST ;print S-100 reset message
    CALL MSGCR
    JMP RSTRT ;re-initialize the 488 interface
; XTN: LXI B,XTNMS ;print external ATN message
    CALL MSGCR
    LXI B,XTNMS
    CALL XCTRL ;listen to the commands and update
    JMP MSGCR ; state of interface
; MSG: LDAX B ;print message
    CALL PRT
    ANI B0H ;see if parity set
    INX B
    JZ MSG ;.. no, so print some more
    RET
; MSGCR: CALL MSG ;print the message, terminate with CRLF
    MVI A,0DH ; output a carriage return
    CALL PRT
    MVI A,0AH ; then a line feed
    CALL PRT
    RET

; DUMMY LISTEN, TALK ADDRESSES-
; The parity bit is set, preventing the 488 interface
; from ever recognizing a talk or listen address
; DUMAD: DB 0A2H ; dummy listen address
    DB 0C0H ; dummy talk address
    DB 0FFH ; parallel poll response byte (no response)
    DB 0FFH ; serial poll response byte (no response)
    DB 0AH ; EOS CHARACTER (IGNORED IN THIS PROGRAM)
TMS: DB 'DEVICE TRIGGER', 0D2H
DVMS: DB 'DEVICE CLEA', 0D2H
BMS: DB 'LISTEN BUFFER FUL', 0CCH
488 Bus Monitor Program

IFMS:    DB  $0DH, $0AH,'INTERFACE CLEA', $0D2H
NLMS:    DB  'NO LISTEN', $0D2H
SRQMS:   DB  $0DH,$0AH,'SRQ ACTIVE TRU', $0C5H
POCMS:   DB  $0DH,$0AH,'POC/RESET TRU', $0C5H
XTNMS:   DB  $0DH,$0AH,'COMMAND', $0BAH
ENDMS:   DB  '<END>',$0BEH
CRMS:    DB  '<CR>',$0BEH
LFMS:    DB  '<LF>',$0BEH
HTMS:    DB  '<HT>',$0BEH
SPMS:    DB  '<SPACE>',$0BEH
DATMS:   DB  $0DH,$0AH,'DATA', $0BAH

; DS       $64D  ; stack area
STAK:
; END
488 Sample Program

Description

This program demonstrates how to set up the P&T-488 as a Controller to send out bus commands (in this case, the Talk and Listen addresses of two devices), then become a Listener. It also illustrates how to allow for an abort command (by the use of the routine BRK).

The program begins by setting up the Stack Pointer and then sets the Listen and Talk addresses of the P&T-488 interface. The 488 bus and interface are cleared by a call to the routine INIT, which is followed by a call to CNTRL, which sends out the contents of the buffer CMDSTR. These commands first tell all active Listeners to stop Listening, then all active Talkers to stop Talking. Talker 5 is then told it is the designated Talker, and Listener 3 (which in this case is the P&T-488) is told it is the sole Listener.

The state of the interface is checked by a call to the routine STATE after the commands are sent. If the Listen state is in the IDLE mode, a jump is made to the routine NTLSN, which prints an error message on the printer and then jumps to the system monitor. (Since the Listen address of the P&T-488 was sent as a command this particular branch should never be executed.)

As preparation for the use of the routine LISTN, the mode switch (A register) is set so that a buffer will be used and the EOS byte will not cause LISTN to RETURN to the calling program. Each time the LISTN routine returns (due to an END byte; i.e., a data byte sent with EOI active true) or the buffer fills (i.e., a branch is made to BFL), the contents of the buffer are printed, the buffer pointers are reset and the LISTN routine is called again.

The user-supplied routine BRK is used to allow the user to suspend 488 transactions and jump back to the system monitor by pressing Control C on the keyboard. It is assumed that the keyboard status is available at Port 0, bit 2 is zero when a key has been depressed, and the keycode is available at Port 1.

The skeleton of the user-supplied routine SREQ is shown, in which a Serial Poll is made of 488 devices 1, 17, 7 and 3. The address of the first device to respond is placed in the A register but the rest of the routine is device dependent. For example, a printer may request service when it is out of paper, the ribbon jams, or some other error condition. A reasonable response to a paper out condition would be a message sent to the console (assuming it is not the printer needing service) informing the operator of the printer's problem.
488 SAMPLE PROGRAM

Assert control, send out the talk address of some other device, the listen address of the P&T-488, and then listen to the talker.

ORG 0B700H

CSTAT EQU 800AH ;addr of controller state byte
ENTBL EQU 8026H ;addr of beginning of P&T-488 jump table

INIT EQU ENTBL
LISTN EQU ENTBL+6
CNTRL EQU ENTBL+0CH
STATE EQU ENTBL+12H
XCTRL EQU ENTBL+15H
SPORCY EQU ENTBL+18H
STADR EQU ENTBL+51H

MNITR EQU 0000H ;system monitor entry address
PRT EQU 0D105H ;console print routine entry address

It is assumed that the routine PRT prints the character held in the A register, then returns to the calling routine. All registers (except the flags) are assumed to be unmodified by PRT.

START: LXI SP,STAK ;initialize the stack pointer
LXI H,ADRTBL ;set up P&T-488 listen, talk addresses
CALL STADR
MVI B,0 ;clear 488 interface and send IFC
CALL INIT
LXI H,CMDSTR ;load HL with beginning address of command string
LXI D,CMDEND ;load DE with end addr of command string
LXI B,JTBL ;load BC with beginning addr of jump table
CALL CNTRL ;send the commands
CALL STATE ;find out what P&T-488 state is
ANI 2 ;keep only listener bit
JZ NTLSN ;..P&T-488 in listener idle mode

LSNLUP: MVI A,1 ;use buffer, ignore EOS character
LXI H,LSNBL
LXI D,LSNEND ;addr of last byte of listen buffer
LXI B,JTBL ;set up pointers
CALL LISTN

LSNPRT: LXI D,LSNBL ;now print the contents of the listen buffer

DCX D

LSNPRL: INX D ;point to next byte in buffer
LDAX D
CALL PRT
P&T-488

Sample Program

MOV A,E ;have we done the last byte yet?
CMP L
JNZ LSNPR1
MOV A,D
CMP H
JNZ LSNPR1
JMP LSNLUP ;printed the last byte, so start
; listening again
;
; USER-SUPPLIED JUMP TABLE
;
JTBL:
JMP TRGR
JMP DVCL
JMP BFL
JMP ICLR
JMP BRK
JMP NLS
JMP SREQ
JMP POCRST
JMP XTN
;
; The following routines should not be entered in
; this program. If they are, a message is printed
; (to aid in debugging) and then a jump is made to
; the system monitor.
;
TRGR:
LXI B,TMS ;print trigger message
CALL MSGCR
JMP MNITR
;
DVCL:
LXI B,DVMS ;print device clear message
CALL MSGCR
JMP MNITR
;
ICLR:
LXI B,IFMS ;print interface clear message
CALL MSGCR
JMP MNITR
;
NLS:
LXI B,NLMS ;print no listener message
CALL MSGCR
JMP MNITR
;
POCRST:
LXI B,POCMS ;print S-100 reset/power-on clear
CALL MSGCR
JMP MNITR
;
XTN:
LXI B,XTNMS ;print external controller message
CALL MSGCR
JMP MNITR
;
NTLSN:
LXI B,NTLMS ;get P&T not listening message
CALL MSGCR ;print it
JMP MNITR ;then go to the system monitor
;
;*************** END OF ABNORMAL BRANCHES ***************

CS-27
; Sample Program

; BFL:       JMP    LSNPRT ; print the contents of the buffer
;           ; then continue listening
;
; BRK:       IN     0  ; get keyboard status
;            ANI    2  ; look at ready bit
;            RNZ    ; .. no key has been depressed
;            IN     1  ; get char from keyboard
;            ANI    7FH ; strip parity bit
;            CPI    3  ; <Control C>?
;            RNZ    ; .. no, continue with 488 transactions
;            JMP    MNITR ; user pressed Control C. ABORT!!!!!!
;
; SREQ:      LXI    H,SPSTR ; put beginning address of serial poll
;            LXI    D,SPEND ; string in HL
;            LXI    B,SPTRB ; and end address in DE
;            CALL    SQRB  ; find jump table address in BC
;            MOV    A,M    ; put device's address in A register
*            *            THE REST IS DEPENDENT ON THE DEVICE
*            *            RET
;
; MSG:       LDAX    B    ; print message
;            CALL    PRT  ; see if parity set
;            ANI    80H
;            INX    B
;            JZ     MSG    ; .. no, so print some more
;            RET
;
; MSGCR:     CALL    MSG  ; print the message, terminate with CRLF
;            MVI    A,9DH  ; output a carriage return
;            CALL    PRT  ; then a line feed
;            CALL    PRT  ; RET
;
; ADRTBL:    DB     '!' ; listen address 3
;            DB     'C'  ; talk address 3
;            DB     0F0FH ; parallel poll response byte (no response)
;            DB     0FFH  ; serial poll response byte (no response)
;            DB     0AH   ; EOS character (ignored in this program)
;            DB     '?'  ; universal unlisten
;            DB     ''   ; universal untalk
;            DB     'F'  ; primary talk address 5
;            DB     '#'  ; primary listen address 3 (P&T-488)
;
; SPSTR:     DB     'A'  ; primary talk address 1
;            DB     'Q'  ; primary talk address 7
;            DB     'G'  ; primary talk address 3
;
; TMS:       DB     'DEVICE TRIGGE', 0D2H
;            DB     'DEVICE CLEA', 0D2H

CS-28
Sample Program

'B\textit{LISTEN BUFFER FUL}', \texttt{OCCH}
\texttt{BDH, BAH,'INTERFACE CLEA', BD2H}
\texttt{BLMS: DB 'NO LISTENE', BD2H}
\texttt{POCMS: DB BDH, BAH,'POC/RESET TRU', BC5H}
\texttt{XTNMS: DB BDH, BAH,'EXTERNAL CONTROLLE', BD2H}
\texttt{NTLMS: DB BDH, BAH,'P&T NOT ADDRESSED AS A LISTENE', BD2H}

; \texttt{LSNTBL: DS 255 ;listen buffer}
\texttt{LSNEND: DS 1 ;last byte of listen buffer}

; \texttt{DS 44D ;stack area}
\texttt{STAK:}

\texttt{END}
The program DINK has been included for several reasons. The first is that it allows
the user to easily exercise the functions provided by the P&T-488 Custom Software
package and interface card. Another is that it allows the user to easily experiment with a
488 device so that he can thoroughly understand what messages it needs before he writes
the assembly language code. Finally, by looking at how DINK is written, the user can see
how the P&T-488 Custom Software package can be used. It should be noted, however,
that not all functions provided by the P&T-488 software package are used in DINK. As
an example, DINK uses only the non-buffered Listen function, so one cannot learn from
DINK how to use the buffered Listen function.

The routine DINK was written so that it is fairly easy to see what is going on. As
a consequence, the code is not optimal, either in execution speed or in the amount of
memory that it requires. One could shorten it considerably, but at the expense of
clarity.

In order to use DINK, the user must first add two routines: the first is for console
input (which is called KBIN) and the other is console output (called PRT). The routine
KBIN should get a character from the console keyboard and return with the character in
the accumulator. No other register (except the flags) may be changed. The routine PRT
should print the character held in the accumulator on the console output device, and then
return. Again, no register (except the flags) may be altered. Examples of these routines
are given at the end of the listing of DINK. The examples shown use the console input
and output routines which are available in the CP/M operating system (CP/M is a product
of Digital Research). The console output routine of CP/M needs the character in register
C, and the CP/M input routine returns with the character in register A.

Once these two routines have been added to DINK, the user should modify (if
necessary) the EQUate for ENTBL and the ORG and then assemble DINK. (The EQUate
for ENTBL must be modified if the ORG of PNT488 has been changed.) PNT488 should
also be assembled, then it and DINK should be loaded into memory. Finally the routine
DINK should be executed from the location START.

Now that DINK is running, what does one do with it? The first thing is to respond
to the message it sent out. Assuming that PRT was correctly written and DINK was
properly assembled, loaded and run, the user should see the message
DINK 1-2-80
Enter P&T-488 Listen and Talk addresses, Parallel Poll response
Serial Poll status and the End-of-String (EOS) bytes
If this message did not appear on the console, the subroutine PRT should be carefully
checked, and the steps of assembly, loading and executing DINK should be tried again.

Now that the message has appeared on the console answer it with the appropriate
characters. The computer will store the characters in a line buffer but will not act upon
them until the user indicates that he is finished with his response by pressing the <carriage
return> key. The line input routine incorporates several editing functions. Individual
characters may be "erased" from the line by pressing the <delete> (sometimes labelled
RUBOUT) key. The computer will "forget" the preceding character and the console output
device will print the character DELCHR in response. (This character can be changed by
the user to whatever code is appropriate for his console. The usual characters are Ø8
Hex (backspace) or 7F Hex (delete).) Multiple characters may be erased by pressing the
delete key once for each character to be erased. The whole line can be erased by typing a Control X (press and hold the CONTROL key, then press the X key, then release both keys). A # will be printed and the console will advance to the next line to show that the line is being restarted.

The line input routine has one more special function key: ESCAPE. The line input routine will not perform any special function associated with the first key depressed after the ESCAPE key. Instead, it will put the key code into the line buffer just as it does for any normal character. Thus the ESCAPE key allows any key code to be placed into the buffer, including the codes for <carriage return>, <ESCAPE>, <Control X> and <delete>. For instance, if one types ABC<Control X>EF<carriage return> the computer accepts this as the same as EF<carriage return> (remember that Control X> erases everything that was typed before it). However, if ABC<ESCAPE><Control X>EF<carriage return> were typed, the computer remembers this as the key codes ABC<Control X>EF because the ESCAPE caused the line input routine to place the key following the ESCAPE into the buffer instead of performing the special function.

Valid Listen addresses are any single character from <space> through >, inclusive. (See the table Code Assignments for Command Mode of Operation for further details.) Valid Talk addresses are any single character @ through †, inclusive. The Parallel Poll response byte should be one character selected from <accent grave> through 0, inclusive. This byte is really the same as a Parallel Poll Enable byte sent by the Controller, in that the three least significant bits of the byte indicate which data (DIO) line will be used by the P&T-488 to respond to a Parallel Poll, and the fourth least significant bit is the Sense bit which selects an affirmative poll response if it has the same logical value as the Ist (individual status) message. The Serial Poll status byte and the EOS byte may be set to any character. (Remember that <delete>, <Control X>, <ESCAPE> or <carriage return> must be preceded by <ESCAPE> to prevent the line input routine from deleting a character, deleting the line or terminating the collection of the string, respectively). These characters are used to set up the P&T-488's own Listen and Talk addresses as well as the bytes it will respond with when it responds to a Parallel or Serial Poll. The EOS byte may be used by the Listen function to detect the end of a string sent by the Talker. If it is desired to make the EOS character a carriage return, remember to press the <ESCAPE> key before the carriage return.

After the line has been entered DINK will print

Enter function code

on the console. The code is a single character, and the following table shows the codes and their corresponding functions.

<table>
<thead>
<tr>
<th>Code</th>
<th>Function Performed</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Get new Listen, Talk addresses, Poll response bytes</td>
</tr>
<tr>
<td>C</td>
<td>Become the 488 Controller</td>
</tr>
<tr>
<td>G</td>
<td>Use function GIM to control 488 General Interface Management lines manually</td>
</tr>
<tr>
<td>I</td>
<td>Initialize the P&amp;T-488 and optionally send IFC true</td>
</tr>
<tr>
<td>L</td>
<td>Listen to the Talker and print what he says</td>
</tr>
<tr>
<td>M</td>
<td>Put the P&amp;T-488 into the Parallel Poll idle (PPIS) state</td>
</tr>
<tr>
<td>N</td>
<td>Make the local &quot;ist&quot; (individual status) message false</td>
</tr>
<tr>
<td>O</td>
<td>Make the local &quot;ist&quot; (individual status) message true</td>
</tr>
<tr>
<td>P</td>
<td>Do a Parallel Poll and print the response</td>
</tr>
<tr>
<td>Q</td>
<td>Do a Serial Poll and print the response</td>
</tr>
</tbody>
</table>
P&T-488

R  Put the P&T-488 into the active Service Request (SRQS) state
S  Print a summary of the state of the P&T-488 and of the 488 Data and GIM lines (all numbers in Hex)
T  Talk on the 488 bus
V  Put the P&T-488 into the No Service Requested (NPRS) state

The following paragraphs are expansions of the descriptions of each of the functions.

**Function A** sets the Listen and Talk addresses, the Parallel and Serial Poll responses, and the End-of-String bytes, just as was done when DINK was first started. By use of this function one can change the addresses or the poll responses of the P&T-488. Note that the Parallel Poll response can also be changed by the Controller. It can send the Listen address of the P&T-488 followed by the PPC (Parallel Poll Configure) byte, then the PPE (Parallel Poll Enable) byte. The PPE sets the Parallel Poll response byte of the P&T-488.

**Function C** causes the P&T-488 to become the 488 Controller. Note that it asserts control immediately, so the user must take care that he is not interrupting the current Talker if it is desired to take control synchronously. Then the routine asks for a string. When the user has typed in the string and terminated it with a carriage return, the string is sent over the bus. Remember that the characters of the string have special meaning, as they are now commands. For example, the character _ (underscore) means UNTALK (all talkers are to revert to the Talker Idle (TIDS) state).

This function will lock up the S-100 system until all commands have been sent. If one of the devices on the 488 bus is performing the Acceptor Handshake but does not complete it, the S-100 system will remain locked up. If there are not any devices connected to the P&T-488, it will send the command string on the bus (even though no one is there to hear it) because another section of the P&T-488 is performing the Acceptor Handshake. It is doing this so that the state of the P&T-488 will be updated in response to what the Controller says.

**Function G** allows one to manually set or reset selected General Interface Management lines of the 488 bus. It is provided so that this software package is compatible with programs written for an older package (Version 1.3). In general, the user should be discouraged from using the function GIM because most of the functions can be better performed by calling other routines.

The lines that function G allows access to are IFC, SRQ, REN and EOI. IFC is better controlled from the routine INIT (function code I), SRQ from the routines SPIDL (function code V) and SPSRQ (function code R), and EOI from TALK (function code T) or PPQRY (function code P). The only line that is not accessible from a better routine is REN.

One can set/reset these lines by typing in an appropriate character followed by a carriage return. The character is placed in the A register and then the PNT488 routine GIM is called. By referring to the description of GIM, it is seen that the IFC, SRQ and EOI lines can be made false while REN can be made true by using any one of the characters <Control B>, #, B or b.

The user should be aware that the routines in PNT488 may not be aware of changes made to the lines by use of this function, and things can get quite confused. The ONLY
reason that this function should even be considered is to gain access to the REN line. If it is used, the user should note the state of the other three lines and preserve their state while changing REN.

Function I causes all P&T-488 states to revert back to their Idle state. The user is asked whether an IFC (Interface Clear) is to be sent over the 488 bus also. If he answers with a Y an IFC will be sent; this will cause all the other 488 devices to revert back to their initialized states. If the answer is N then an IFC will not be sent and the 488 devices other than the P&T-488 will not be affected by this function. If any other character is typed as the first character of the string a message is printed on the console informing the user that only these two responses are allowed.

Function L sets up the P&T-488 as a non-buffered Listener. Each character heard by the P&T-488 is printed on the console as it is heard. Control (non-printing) characters are printed as two-character strings, the first being uparrow (↑) and the second being the character with 40 Hex added to it to make it printable. For example, a null will be printed as ↑@, a <Control X> (otherwise known as CAN or CANCEL) as ↑X, etc. The user is asked Return upon receipt of EOS byte? If the response is Y or y the function will terminate when a character matching the EOS byte is received. Upon termination of this function the user is asked to select the next function. The function will always terminate upon receipt of an END message (the EOI line made true by the Talker while speaking a byte). In this case the message <END> is printed on the console and the user is asked to select the next function.

Function M causes the routine PPIDL in PNT488 to be called, which in turn places the P&T-488 into the Parallel Poll Idle (PPIS) state. All that this means is that the P&T-488 will not participate in a Parallel Poll.

Function N causes the routine PISTF in PNT488 to be called. This routine sets the ist (individual status) message false and then puts the appropriate response byte in the Parallel Poll response register of the P&T-488. It then puts the P&T-488 into the Parallel Poll Standby (PPSS) state. See the description of PISTF for more details.

Function Q causes the routine PISTT in PNT488 to be called. This routine sets the ist message true then puts the appropriate response byte in the Parallel Poll response register of the P&T-488. It then puts the P&T-488 into the Parallel Poll Standby (PPSS) state. See the description of PISTT for more details.

Function P causes the routine PPQRY in PNT488 to be called, which in turn executes a Parallel Poll. The response is then printed (in Hex) on the console.

Function Q sets up the P&T-488 to do a Serial Poll. The user is asked to enter a string, which should be the Talk addresses of the devices to be polled. Then the routine SPQRY in PNT488 is called, which actually performs the poll. SPQRY will return upon receipt of an affirmative response or after the string of talk addresses has been exhausted, whichever occurs first. The commands sent by the P&T-488 while it is conducting the Serial Poll are echoed on the console. The string will appear as ↑X_•••↑Y where the character ? means UNListen, ↑X is the command SPE (Serial Poll Enable), _ is the command UNTalk, the ellipsis (...) represents the Talk addresses that are sent by the P&T-488, and the ↑Y is SPD (Serial Poll Disable). If an affirmative response has been detected, DINK will print the Talk address of the device that responded affirmatively as well as the response, and then ask for the next function code. If no device responded affirmatively, DINK will print
No affirmative response to Serial Poll
Try another Serial Poll (Y/N)?
and then wait for the user to respond. If a string beginning with N is entered, DINK will ask for the next function code. If a string beginning with Y is entered, DINK will ask for another string of Talk addresses to be polled. It is important that only Talk addresses of devices which are currently connected to the 488 bus and capable of responding to a Serial Poll be entered in the string. The reason is that the P&T will send out the address and then listen for the addressed Talker to speak its poll response. If there is no Talker, there will never be a response, and the whole system will wait forever for that response.

Function R causes the routine SPSRQ in PNT488 to be called, which in turn asserts a true on the SRQ line and places the P&T-488 in the Service Request (SRQS) state. If the P&T-488 is not the Controller, the S-100 system will wait for an external Controller (i.e., some device other than the P&T-488) to assert Control and perform a Serial Poll. When the poll is made, the P&T-488 will respond affirmatively and then go into the Affirmative Poll Response (APRS) state. Then the user will be asked to select the next function.

If, on the other hand, the P&T-488 is the Controller, it will assert Control, then ask the user to enter a string of the Talk addresses of the devices to be Serial Polled. After the string has been entered the P&T-488 will poll each of these devices and then return when it has found the one requesting service or has finished polling all devices. The commands sent by the P&T-488 while it is conducting the Serial Poll are echoed on the console. The string will appear as

?fX ...fY
where the character ? means UNListen, †X is the command SPE (Serial Poll Enable), _ is the command UNTalk, the ellipsis (...) represents the Talk addresses that are sent by the P&T-488, and the †Y is SPD (Serial Poll Disable). If the user had included the P&T-488's own Talk address in the string and no other device in the string before it has responded affirmatively to the poll, the P&T-488 will respond affirmatively to the poll and go into the Affirmative Poll Response (APRS) state, then return.

As in the case of function Q, it is important that only the Talk addresses of devices actually connected to the bus and capable of responding to a Serial Poll be placed in the poll string; otherwise the S-100 system will wait forever for the response of a non-existent device.

Function S will display the state of the P&T-488, the secondary Talk and Listen addresses and the state of the 488 bus lines. All values displayed are in Hex, and the user should refer to the function STATE for a description of the meaning of the various states. The value shown on the line labelled "Abbreviated State of P&T-488" is the value that the routine STATE returned in the accumulator.

The secondary addresses shown for the Talk and Listen functions are 7F Hex if the respective function has been addressed by the 488 Controller without a secondary address (single byte addressing). Otherwise the secondary addresses shown are the characters sent by the Controller as the secondary address when the Controller last addressed the Talk and Listen functions.

The state of the eight data lines and eight command lines of the 488 bus is also displayed. The values given are in Hex, which really has no particular meaning for the eight command lines. However, the order (weighting) of the command lines is shown on the same line as a handy reminder. The weights of the command lines are shown in the following table.
Function T sets up the P&T-488 as a Talker. The user is asked whether the END message (EOI line true) is to be sent with the last character of the talk string. The only responses allowed are strings beginning with Y or N. The user is then asked for the string that the P&T-488 is to speak. Then the routine TALK of PNT488 is called and the P&T-488 speaks the string on the bus. If there are no Listeners the P&T-488 recognizes this as an error and prints a message on the console informing the user that there are no Listeners on the 488 bus. Otherwise the whole string is said and then the user is asked for the next function code.

Function V causes the routine SPIDL in PNT488 to be called, which in turn puts the P&T-488 into the No Service Requested (NPRS) state. This is equivalent to the S-100 making the local message rsriv (request service) false. The P&T-488 is also set to assert a passive false on the SRQ line. Then the routine returns and the user is asked for the next function code.

Special Considerations

The P&T-488 is heavily dependent upon the support software (in this case, PNT488) in order to communicate on the 488 bus. The S-100 system must execute one of the interface subroutines if the P&T-488 is to perform nearly any 488 bus function. This includes not only the "assertive" functions, such as Talk and Control, but the "responsive" functions, such as responding to a Serial Poll, being addressed as a Talker or Listener by the Controller, etc. The only 488 function that the P&T-488 can perform without any software support is respond to a Parallel Poll.

This limitation can create problems unless the user is aware of it and allows for it in his configuration of the 488 bus and how he uses the P&T-488. For instance, assume that some device other than the P&T-488 is the bus Controller and that it will perform a Parallel Poll periodically. The P&T-488 will respond to the poll properly, but the interface will lock up the 488 handshake function until the S-100 system releases it. This happens because the poll was done by an external Controller, so XATN was made true while the poll was performed. The P&T-488 responds to XATN true by asserting NRFD active true and by asserting all other command lines and all data lines passive false. The P&T-488 remains in this state until the S-100 system resets the XIFC bit in the ISR register. Since NRFD is active true, no handshake can proceed. The reason the P&T-488 behaves in this fashion is that if the external Controller wanted to issue commands (instead of do a Parallel Poll), it is necessary to keep it from saying anything until the S-100 system is ready to respond. The S-100 system indicates its readiness by resetting the XIFC bit in the ISR register of the P&T-488.

Another consequence of the need of the P&T-488 for software support in order to perform 488 bus functions is that something may happen on the 488 bus and the S-100 system will not find out about it until one of the PNT488 subroutines is called. For example, some device may assert an active true on the SRQ line, indicating that it wants service. The S-100 system will find out about it if any one of the routines TALK, CNTRL or LISTN are executed, but not otherwise. The P&T-488 interface card can be
set up to issue an interrupt to the S-100 system upon this and other conditions, but most customers have stated very explicitly that they do not want an interrupt driven system. Thus the P&T-488 has been strapped to defeat interrupts, and the routines in PNT488 poll the P&T-488 to find out if anything interesting is happening.

There are several things which can happen which are not a direct response to the function code the user selects. For instance, if the Listen function is selected and an External Controller asserts Control, DINK will print a message on the console informing the user of this fact and will then call the routine XCTRL in PNT488. This routine will get the commands from the External Controller and will update the states of the various interface functions of the P&T-488 as necessary. When the External Controller releases control of the bus, XCTRL will return to DINK, which in turn will ask for the next function code. At this point the user should select the SHOW function (code = 5) to find out how the state of the P&T-488 has been changed by the External Controller.

Another response the user may get is that DINK informs him that either the S-100 POC (Power-On Clear) or the S-100 Reset line has been (or is) true. Either of these conditions has the effect of putting the P&T-488 interface into its idle mode, which means that it has released all 488 data and control lines. The user should perform the Initialize (code = I) function to reset the P&T-488 to a known state.
PROGRAM LISTING

ORG 100H

DELCHR EQU 7FH ; CHARACTER TO BE ECHOED UPON RECEIPT
; OF A DELETE CODE (DELETE AND BACKSPACE
; ARE THE MOST COMMON CHOICES)

MOPT EQU 70H ; PORT ADDR OF 488 COMMAND LINES
DATPT EQU 76H ; PORT ADDR OF 488 DATA LINES

BUFSIZ EQU 128 ; NUMBER OF BYTES IN INPUT BUFFER

ENTBL EQU 8026H ; ADDRESS OF FIRST ENTRY IN PNT488 JUMP TABLE

INIT EQU ENTBL
TALK EQU ENTBL+03H
LISTN EQU ENTBL+06H
STOR EQU ENTBL+09H
CNTRL EQU ENTBL+0CH
GIM EQU ENTBL+0FH
STATE EQU ENTBL+12H
XCTRL EQU ENTBL+15H
SPOR EQU ENTBL+18H
SPORE EQU ENTBL+1BH
SREL EQU ENTBL+1EH
SPREQ EQU ENTBL+21H
SPRREQ EQU ENTBL+24H
PSPREL EQU ENTBL+27H
PPQRY EQU ENTBL+2AH
PPREQ EQU ENTBL+2CH
PPQRY EQU ENTBL+2FH
PPREL EQU ENTBL+32H

EQUATES FOR CP/M BIOS ROUTINES:

CONIN EQU @DA99H ; CONSOLE INPUT ROUTINE
CONOUT EQU @DA0CH ; CONSOLE OUTPUT ROUTINE

START: LXI SP,STAK ; INITIALIZE THE STACK POINTER
        LXI B,STAK ; PRINT ID MESSAGE
        CALL MSG
        JZ GETFN ; NOTHING IN BUFFER

GETFN: LXI SP,STAK ; RE-INITIALIZE STACK POINTER (STACK WILL BE LEFT IN
        ; DISARRAY IF 'ATN' IS MADE TRUE WHILE TALKING OR
        ; LISTENING)
        SUB A ; CLEAR ECHO FLAG SO THAT UNLESS THE FLAG
        STA ECHO ; IS SET LATER, EACH CHAR COMMUNICATED
        ; ON THE 488 BUS IS NOT ECHOED TO THE
        ; CONSOLE
        CPI 'A' ; SAVE IT FOR LATER
        JZ SETAOI ; SET NEW P&T-488 ADDRESSES
        CPI 'C' ; CLEAR 488 INTERFACE BUT DO NOT SEND IFC
        CPI 'G' ; SET GIM LINES
        JZ INITL ; INITIALIZE
        CPI 'I' ; SET NEW P&T-488 ADDRESSES
        CPI 'O' ; SET IST=W
        CPI 'G' ; CLEAR 488 INTERFACE BUT DO NOT SEND IFC
        CPI 'M' ; LISTEN
        CPI 'N' ; PUT PP IN IDLE STATE
        CPI 'O' ; SET IST=W

CS-37
@14F CA6B02 JZ  PSET ;SET IST=1
@152 FE50 CPI 'IP'
@154 EAFC01 JZ  PPOLL ;DO A PARALLEL POLL
@157 FE51 CPI 'Q'
@159 CA6B03 JZ  QRY ;DO A SERIAL POLL QUERY
@15C FE52 CPI 'R'
@15E CA1702 JZ  REQ ;DO A SERVICE REQUEST
@161 FE53 CPI 'S'
@163 CA2002 JZ  SHO ;SHOW THE STATE OF THE P&T-488
@166 FE54 CPI 'T'
@168 CA7002 JZ  TALKR ;TALK
@16B FE56 CPI 'U'
@16D CA1002 JZ  SREL ;RELEASE SRO LINE
@170 F1FB04 LXI B,RAOMS ;PRINT "INVALID FCN" MESSAGE
@173 CD7204 CALL MSGCR
@176 C31101 JMP GETFN ;GET FUNCTION AGAIN
@179 F1B204 ADROSET: LXI B,ADRMS ;SEND "GET ADDRESSES" MESSAGE
@17C D6704 CALL MSG
@17F DCE003 CALL FILBFR ;GET RESPONSE AND PUT IN BUFFER
@182 78 MOV A,B ;MAKE SURE THAT THERE ARE AT LEAST 5
@183 FE55 CPI 5 ;CHARACTERS IN THE RESPONSE
@185 FA9101 JP SETI ;5 OR MORE CHARMS
@188 01C355 LXI B,FEM5S ;PRINT TOO FEW CHARMS IN BUFFER MESSAGE
@18B CD6704 CALL MSG ;
@18E C37901 JMP ADROSET ;AND GET THE INFO AGAIN
@191 212A08 SETI: LXI H,_BUFBE ;SET UP P&T 488 LISTEN, TALK ADDRESSES
@194 CD2F05 CALL STADR ;PERFORM THE FUNCTION
@197 C9 RET ;
@198 CD7901 SETADDR: CALL ADROSET ;SET THE LISTEN, TALK ADDR, ETC
@19B C31101 JMP GETFN
@19E D6F002 CONTRO: CALL GETSTR ;FILL BUFFER AND SET POINTERS
@1AA CD2800 CALL CNTRL ;PERFORM THE FUNCTION
@1AC C31101 JMP GETFN ;GET ANOTHER FUNCTION FROM OPERATOR
@1AD C0D002 GIMSET: CALL GETCHR ;GET THE CHARACTER
@1AO C3500 CALL GIM
@1BA C31101 JMP GETFN
@1BE F1E805 INITL: LXI B,IFCM ;ASK IF IFC TO BE SENT
@1B3 D6704 CALL MSG
@1B6 DCD002 CALL YESNO ;GET RESPONSE (ZERO FLAG SET IF NO)
@1B9 0601 MV1 B,1 ;SET UP FOR NO IFC
@1BB CA0001 JZ NOIFC ;NO, SO DO NOT SEND IFC
@1BC 0600 MV1 B,0 ;YES, SO SEND IFC
@1CD CD2605 NOIFC: CALL INIT
@1CE C31101 JMP GETFN
@1CC 5E00 LSN: MV1 A,IFFH ;SET ECHO FLAG SO THAT EACH CHARACTER IS
@1CB 322808 STA ECHO ;SHOWN ON THE CONSOLE
@1CE 01E005 LXI B,EOMS ;PRINT "STOP ON EOS?"
@1D0 CD6704 CALL MSG ;
@1D1 DCD002 CALL YESNO ;GET THE RESPONSE
@1D4 C2E001 JNZ LSN1 ;STOP ON EOS BYTE
@1D7 3E00 MV1 A,0 ;NON-BUFFERED LISTENER, IGNORE EOS BYTE
@1D9 016003 LXI B,JTBL ;BC POINT TO USER JUMP TABLE
@1DC CD2C00 CALL LSTN
@1DF 018705 LXI B,ENDMS ;SHOW THAT AN END MESSAGE HAS BEEN RECEIVED
@1E2 CD7204 CALL MSGCR
@1E5 C31101 JMP GETFN
@1E8 3E02 LSN1: MV1 A,2 ;NON-BUFFERED LISTENER, STOP ON EOS BYTE
@1EA 016003 LXI B,JTBL ;POINT BC TO USER JUMP TABLE
@1ED CD2C00 CALL LSTN ;
@1F0 CD7204 CALL ORLH
@1F3 C31101 JMP GETFN ;
@1F6 CD5000 ;PIDL: CALL PRIDL ;PUT PP IN IDLE STATE
@1F9 C31101 JMP GETFN

CS-38
02A1 CD4480 CALL SPREL ;RELEASE SRQ, PUT SR FCN IN NPRS
02A4 C31101 JMP GETFN ;GET ANOTHER FUNCTION

02A7 01A607 TALKR: LXI B, TLKMS ;PRINT "SEND END WITH LAST CHAR"
02AC CD6F04 CALL MSG
02AF CDC102 CALL YESNO ;SET FLAG FOR NO END
02BF 0E00 MVI A, 0 ;SET FLAG FOR END
02C2 CA8002 JZ NOEOI ;SET FLAG FOR END
02C8 02700 INR A ;SET FLAG FOR END
02CD F5 NOEOI: PUSH PSW ;SAVE END FLAG
02D0 02A701 JMP GETFN ;GET ANOTHER FUNCTION FROM THE OPERATOR

02C1 CDC003 YESNO: CALL FILBFR
02C4 CA0202 JZ YESNI ;...BUFFER EMPTY - INVALID RESPONSE
02C7 3A4000 LDA BUFREG ;GET THE FIRST CHARACTER
02CE 0E59 CPI 'Y' ;IS IT YES?
02D0 CAD002 JZ COK ;...CHARACTER OK
02D2 FE4E CPI 'N' ;IS IT NO?
02D4 0B2006 YESNI: LXI B, NOGUD ;INVALID RESPONSE
02D7 CD7004 CALL MSGCR
02DB C31001 JMP YESNO ;TRY AGAIN

02E0 012505 GETCHR: LXI B, CHRMS ;PRINT CHARACTER PROMPT
02E3 CD6F04 CALL MSG
02ED CD003 CALL FILBFR ;GET THE CHARACTER
02F0 C2B003 JNZ GETCH1 ;...AT LEAST ONE CHARACTER IS IN THE BUFFER
02F3 01C005 LXI B, FEWMS ;POINT TO 'TOO FEW' MSG
02F6 CD7004 CALL MSGCR ;THEN PRINT IT
02FB C3D002 JMP GETCHR ;AND GET INFO FROM USER AGAIN

02F2 3A2008 GETCH1: LDA BUFREG ;PUT FIRST CHARACTER IN REG A
02F5 C9 RET

02F6 019C07 GETSTR: LXI B, STRMS ;PRINT STRING PROMPT
02F9 CD7004 CALL MSGCR
02FF CD003 CALL FILBFR ;GET A CHARACTER STRING FROM THE OPERATOR
0300 C2B003 JNZ GETS1 ;...AT LEAST ONE CHARACTER IS IN THE BUFFER
0303 01C005 LXI B, FEWMS ;POINT TO 'TOO FEW' MSG
0306 CD7004 CALL MSGCR ;THEN PRINT IT
030B C3F002 JMP GETSTR ;AND GET INFO FROM USER AGAIN

030D 2AAA08 GETS1: LHLD BUFFTR ;PUT ADDR OF LAST VALID CHAR IN HL
0310 EB XCHG ;PUT ADDR OF LAST VALID CHAR IN DE
0314 21A006 LXI H, BUFBEG ;LOAD HL WITH ADDRESS OF FIRST CHAR
0318 011603 LXI B, JTABLE ;LOAD BC WITH BEGINNING ADDR OF JUMP TABLE
031B C9 RET

USER-SUPPLIED JUMP TABLE

031C C33103 JTABLE: JMP TRGR
0319 C33003 JMP DVCL
031D C33F03 JMP BSTL
031F C34803 JMP DULR
0322 C35103 JMP BSSR
0325 C37003 JMP NLS
0328 C3A103 JMP SEQ
032B C38203 JMP POCRST
032E C38003 JMP XTN

0331 01CF07 TRGR: LXI B, TRGRS ;PRINT TRIGGER MESSAGE
0334 CD7004 CALL MSGCR
0337 C9 RET
P&T-488

DVCL: LXI B,DVMS ;PRINT DEVICE CLEAR MESSAGE
CALL MSGCR
RET

; FFL: LXI B,BMS ;WE SHOULD NEVER REACH THIS POINT, BUT
CALL MSGCR ;IF WE DO, PRINT MESSAGE
JMP GETFN

; CLR: LXI B,IFMS ;PRINT INTERFACE CLEAR MESSAGE
JMP GETFN ;ASK FOR NEW FUNCTION

; BRK: MOV B,A ;SAVE LAST CHAR COMMUNICATED ON 488 BUS
LDA ECHO ;LOOK AT THE ECHO FLAG
ORA A
MOV A,B ;GET THE LAST CHAR AGAIN
RZ ;...ECHO FCN NOT ENABLED, SO DON'T PRINT ;THE CHARACTER

; CPI 28H ;CONTROL CHARACTER?
JNC Noree ;...NO
CPI @AH ;LINE FEED?
JZ NOTCC
CPI @0H ;CR RETURN?
JZ NOTCC
ORI 40H ;MAKE THE CHAR INTO A PRINTING CHAR
PUSH PSW ;SAVE THE CHARACTER
MVI A,'t' ;PRINT UPARROW TO FLAG IT AS A
CALL PRT ;CONTROL CHARACTER
POP PSW ;PRINT THE CHARACTER

; CALL PRT ;REQUESTING DEVICE IS 

; NLS: LXI B,NLMS ;PRINT NO LISTENER MESSAGE
CALL MSGCR
JMP GETFN ;ASK FOR NEW FUNCTION

; PCRST: LXI B,POCMS ;PRINT S-100 RESET/POWER-ON CLEAR
CALL MSGCR
JMP GETFN ;ASK FOR NEW FUNCTION

; STXN: LXI B,XTNMS ;PRINT EXTERNAL CONTROLLER MESSAGE
CALL MSGCR
JMP GETFN ;ASK FOR NEW FUNCTION

; QRY: LXI B,BQFM ;SET ECHO SO THAT THE SERIAL POLL IS
STA ECHO ;SHOWN ON THE CONSOLE
CALL GETSTR ;GET STRING OF 488 DEVICES TO BE POLLED

; CALL SPORE ;FIND OUT WHICH DEVICE WANTS SERVICE
CALL CRLF ;TERMINATE THE ECHOED POLL WITH CRLF
ORA A ;SEE IF ANY AFFIRMATIVE RESPONSE
JZ AFIRM ;...YES
JZ CACD3 ;AFIRM: PUSH B ;SAVE RESPONSE BYTE
JMP SREQ ;...YES, SO REDO SERIAL POLL

; CALL ORA OF RESPONDING DEVICES TALK ADDR
LXI B,RSPM ;PRINT "REQUESTING DEVICE IS "
CALL MSG
; GET ADDR OF TALK ADDR AGAIN
POP H ;PUT DEVICE'S ADDRESS IN A REGISTER
MOV A,M ;PRINT THE DEVICE'S TALK ADDR
CALL PRT ;TERMINATE WITH A NEW LINE
CALL MSG ;PRINT RESPONSE BYTE
;GET ADDR OF TALK ADDR AGAIN
MOV A,B ;PRINT VALUE OF RESPONSE BYTE
;PRINT OCTOTHORPE AS CANCEL CHARACTER
;RE-START BUFFER FILL ROUTINE
P&T-488

DINK

; 458 E60F HEXL: ANI $08H ;STRIP HIGH NIBBLE
459 F608 ORI $08H ;CONVERT TO PRINTING CHARACTERS
45A F63A CPI $0F ;SEE IF VALUE GREATER THAN 9
45B DA5684 JC NUM ;NO
45C 6C97 ADI 7 ;*YES, SO ADD OFFSET TO GET A-F
45D C3C8 NUM: CALL PRT ;PRINT THE CHARACTER
45E C9 RET

460 0A MSG: LDAX B
461 C688 CALL PRT ;PRINT MESSAGE
462 EB60 ANI $00H ;SEE IF PARITY SET
463 D3 INX B
464 C674A JZ MSG ;NO, SO PRINT SOME MORE
465 71 C9 RET

466 CD748 MSGCR: CALL MSG ;PRINT THE MESSAGE, TERMINATE WITH CRLF
467 53 CRLF: PUSH PSW ;PRESAVE ALL REGISTERS
468 3E9D MV1 $0AH ;OUTPUT A CARRIAGE RETURN
469 47C9 C4 MSG ;PRINT
46A F1 POP PSW ;RESTORE ALL REGISTERS
46B 01 C9 RET

46C 00A456574ADRXS: DB $08H, $0AH,'Enter P&T-488 Listen and Talk addresses,'
46D 0258B17261 DB $08H, $0AH,'Serial Poll status and the End-of-String'
46E 04EE2854F5329 DB $08H, $0AH,'(EOS) bytes', $0AH
46F 00A4494E56B0DMS: DB $08H, $0AH,'INVALID FUNCTION CODE', $0AH
470 4C953445B0MS: DB $08H, $0AH,'LISTEN BUFFER FULL', $0CH
471 052546E746572CRMS: DB $08H, $0AH,'Enter a character', $0AH
472 3438382843CLMS: DB $08H, $0AH,'488 Control lines: DAV NRFD NDAC IFC ATN SRQ REN E0I', $0AH
473 546C34383284DLMS: DB $08H, $0AH,'1488 Data lines', $0AH
474 07444556549430DMS: DB $08H, $0AH,'DEVICE CLEAR', $02H
475 00A33C4566DMS: DB $08H, $0AH,'END', $00H
476 52B5747572EOEMS: DB $08H, $0AH,'Return upon receipt of EOS byte?', $0AH ; M8229
477 05AF 456E746572CRNMS: DB $08H, $0AH,'Enter function code', $0AH
478 05C3 546F5F2866F6WMS: DB $08H, $0AH,'Too few characters', $0AH ; M8229
479 05D6 00A44494E1DMS: DB $08H, $0AH,'DINK 1-2-88?', $0AH ; M1020
47A 05EB 5356565E421FCMS: DB $08H, $0AH,'Send IFC (Y/N) ?', $0AH
47B 05F8 00A4494E541FCMS: DB $08H, $0AH,'INTERFACE CLEAR', $02H
47C 062554373465LSMS: DB $08H, $0AH,'Listen Secondary Address', $0AH
47D 0622 4E4F34C49NLMS: DB $08H, $0AH,'NO LISTENE', $02H
47E 062D 00A539286FNOGUD: DB $08H, $0AH,'Y or N ONLY!!!', $0AH
47F 063E 456F20160NORSP: DB $08H, $0AH,'No affirmative response to Serial Poll', $0AH
480 0669 00A4584F3POCMS: DB $08H, $0AH,'POC/RESET TRU', $0CH
481 0675 5816712616CSS: DB $08H, $0AH,'Parallel Poll Response byte', $0AH
482 0691 5468628765RSBSM: DB $08H, $0AH,'The value of the response byte is', $0AH
483 06B3 5468E59346MPMS: DB $08H, $0AH,'The 488 device requesting service is', $0AH
484 06B8 0A42272265SOMS: DB $08H, $0AH,'Abbreviated State of P&T-488', $0AH
485 06F9 54616C68281SM: DB $08H, $0AH,'Talk State byte', $0AH
486 0795 4C69737465SSMS: DB $08H, $0AH,'Listen State byte', $0AH
487 0717 535657256593SMG: DB $08H, $0AH,'Service Request State byte', $0AH
488 0732 5256560F744MS: DB $08H, $0AH,'Remote-Local State byte', $0AH
489 074A 58017261C5SMG: DB $08H, $0AH,'Parallel Poll State byte', $0AH
48A 0753 456F747265SMG: DB $08H, $0AH,'Controller State byte', $0AH
48B 0779 4128345365SRSMS: DB $08H, $0AH,'488 device is requesting service', $0AH
48C 07C9 456E746572CRMS: DB $08H, $0AH,'Enter a string', $0AH
48D 07AB 5356566424TLKMS: DB $08H, $0AH,'Send END with last character (Y/N) ?', $0AH
48E 07CF 4455649431MS: DB $08H, $0AH,'DEVICE TRIGE', $02H
48F 07D0 5427292861TRQHMS: DB $08H, $0AH,'Try another Serial Poll (Y/N) ?', $0AH
490 07FC 54616C68281TMS: DB $08H, $0AH,'Talk Secondary Address', $0AH
491 0813 00A455854XTMS: DB $08H, $0AH,'EXTERNAL CONTROLLE', $02H

9828 00 ECHO: DB $0 ;ECHO FLAG. IF 0 DO NOT PRINT CHAR EACH
9829 00 FCN: DB $0 ;AREA TO SAVE FUNCTION CODE
982A 00 BUFBE: DS BUFPSZ ;STRING BUFFER
982B 00 2AH BUFBDR: DW BUFBEG ;STRING BUFFER POINTER

CS-43
IT IS ASSUMED THAT THE ROUTINE PRT PRINTS THE CHARACTER HELD IN THE A REGISTER, THEN RETURNS TO THE CALLING ROUTINE. ALL REGISTERS (EXCEPT THE FLAGS) ARE ASSUMED TO BE UNMODIFIED BY PRT.

SIMILARLY, IT IS ASSUMED THAT THE ROUTINE KBIN GETS A CHARACTER FROM THE KEYBOARD AND RETURNS WITH IT IN THE A REGISTER. ALL OTHER REGISTERS ARE TO BE UNAFFECTED

AN EXAMPLE OF PRT WRITTEN TO USE CP/M’S CONSOLE OUTPUT ROUTINE IN CB IOS

NOTE: THE STARTING ADDRESSES OF CONOUT AND CONIN CAN BE FOUND IN THE FOLLOWING MANNER:
1. GET THE ADDRESS STORED IN THE WORD AT LOCATION 0001 (LOW BYTE OF ADDR IN 0001, HIGH BYTE IN 0002)
2. ADD 6 TO THAT ADDRESS. THE RESULT IS THE ADDRESS OF A JUMP TO THE ROUTINE CONIN.
3. ADD 3 TO THE ADDRESS CALCULATED FOR CONIN. THIS IS THE ADDRESS OF A JUMP TO THE ROUTINE CONOUT.

PRT:

```
PUSH H
PUSH D
PUSH B
PUSH PSW ;SAVE ALL REGISTERS
ANI 7FH ;STRIP PARITY BIT
MOV C,A ;PUT CHAR INTO REG C AS NEEDED BY CB IOS
CALL CONOUT ;OUTPUT THE CHARACTER
POP PSW ;RESTORE REGISTERS
POP B
POP D
POP H
RET
```

AN EXAMPLE OF KBIN WRITTEN TO USE CB IOS CONSOLE INPUT ROUTINE

```
PUSH H ;SAVE REGISTERS
PUSH D
PUSH B
CALL CONIN ;GET THE CHAR (CP/M RETURNS WITH CHAR IN REG A)
PUSH PSW ;RESTORE REGISTERS
ANI 7FH ;STRIP PARITY BIT
MOV C,A
POP B
POP D
POP H
RET
```

END
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0482</td>
<td>ADRMS</td>
</tr>
<tr>
<td>0513</td>
<td>BMS</td>
</tr>
<tr>
<td>0525</td>
<td>CHRMS</td>
</tr>
<tr>
<td>0559</td>
<td>CONIN</td>
</tr>
<tr>
<td>007F</td>
<td>DELCHR</td>
</tr>
<tr>
<td>0597</td>
<td>ENDNS</td>
</tr>
<tr>
<td>05C3</td>
<td>FEMS</td>
</tr>
<tr>
<td>02F2</td>
<td>GETCH1</td>
</tr>
<tr>
<td>0835</td>
<td>GIM</td>
</tr>
<tr>
<td>0506</td>
<td>IDMS</td>
</tr>
<tr>
<td>0316</td>
<td>JTBL</td>
</tr>
<tr>
<td>018B</td>
<td>LS1</td>
</tr>
<tr>
<td>0379</td>
<td>NLS</td>
</tr>
<tr>
<td>0375</td>
<td>NOTCC</td>
</tr>
<tr>
<td>0211</td>
<td>PNSET</td>
</tr>
<tr>
<td>01FC</td>
<td>PROLL</td>
</tr>
<tr>
<td>020B</td>
<td>PSET</td>
</tr>
<tr>
<td>0608</td>
<td>S00MSG</td>
</tr>
<tr>
<td>074A</td>
<td>S03MSG</td>
</tr>
<tr>
<td>003E</td>
<td>SPORY</td>
</tr>
<tr>
<td>0779</td>
<td>SROMS</td>
</tr>
<tr>
<td>079C</td>
<td>STRMS</td>
</tr>
<tr>
<td>0331</td>
<td>TRGR</td>
</tr>
<tr>
<td>0813</td>
<td>XTNMS</td>
</tr>
</tbody>
</table>

Symbol Table
IEEE used the following conventions when they assigned the names used in the standard:

Lower Case names are associated with local messages (messages between a device and its interface; they MIGHT NOT appear on the 488 bus).

Upper Case names are divided into three groups:

One or two letters name interface functions,

Three letter mnemonics are remote messages (communications over the 488 bus from one interface to another) and

Four letter names ending in "S" identify the state of an interface function.

The numbers following an entry are the pages of the IEEE Standard (Apr 4, 1975) which give further information.

ACDS ACcept Data State
21, 22

ACG Addressed Command Group - multiline messages (00–0F Hex) which affect only addressed devices. The messages GTL (Go To Local), SDC (Selective Device Clear), PPC (Parallel Poll Configure) and GET (Group Execute Trigger) operate only on devices in the LADS (Listener Addressed) state. TCT (Take Control) operates on the device in the TADS (Talk Addressed) state.
48, 77

ACRS ACceptor Ready State
21, 22

Addressed Commands - Commands belonging to the Addressed Command Group (See ACG)
43

AH Acceptor Handshake - the device function which allows proper reception of data and commands appearing on the eight data lines of the 488 bus (i.e., multiline messages). The DAV (Data Available) line is sensed to determine when the multiline message is valid, and the AH function indicates its readiness for data by asserting a passive false on the NRFD (Not Ready For Data) line, and that it has received the message by asserting a passive false on the NDAC (Not Data Accepted) line. Note that it is illegal for the AH to assert both NDAC and NRFD passive false simultaneously.
20
Active False — an active false message asserted on the 488 bus is one in which it is guaranteed that a false value is received. It overrides a passive true. The standard is constructed so that it is not possible for an active true and an active false message to be asserted on the bus at the same time.

Active True — a message which when asserted on the 488 bus is guaranteed to be received as true. It overrides a passive false. The standard is constructed so that it is not possible for an active true and an active false message to be asserted on the bus at the same time.

AIDS ACceptor Idle State
20, 21

ANRS Acceptor Not Ready State
20, 21

APRS Affirmative Poll Response State
32

ATN ATtentioN — a uniline remote message indicating that a Controller is sending commands (as contrasted to a Talker sending data) over the eight data (DIO) lines.
19, 21, 24, 29, 35, 41, 48, 75–76

AWNS Acceptor Wait for New cycle State
21, 22

C Controller interface function — the interface function which allows a device to send device addresses, universal commands and addressed commands over the 488 bus. It also allows the device to conduct a Parallel Poll to determine which device needs service.
41

CACS Controller AClive State
41, 42

CADS Controller ADDressed State
41, 42

CAWS Controller Active Wait State
41, 43

CIDS Controller IDle State
41

CPPS Controller Parallel Poll State
41, 43

CPWS Controller Parallel poll Wait State
41, 43
CSBS Controller StandBy State 41,43

CSNS Controller Service Not requested State 41,44

CSRS Controller Service Requested State 41,44

CSWS Controller Synchronous Wait State 41,43

CTRS Controller Transfer State 41,44

DAB Data Byte — a multiline sent by the Source Handshake (SH) over the eight data (DIO) lines 25,48,75-76

DAC Data Accepted — the complement appears on the NDAC line. See AH, SH for further information. 19,22,48,75-76

Data Byte Transfer Control lines — the three lines (DAV, NRFD and NDAC) that are used by the Source and Acceptor functions to perform the handshake cycle. 12,18-22,67

DAV Data Valid — a uniline message sent by the Source Handshake (SH) function over the DAV line. See SH. 48,75-76

DC Device Clear interface function — the interface function which allows a device to be cleared (initialized) either individually or as part of a group. The group may be either part or all of the addressed devices in one system. 37-38

DCAS Device Clear Active State 38

DCIS Device Clear Idle State 37,38

DCL Device CLear — a multiline message (14 Hex) sent by the Controller over the eight data lines indicating that all devices are to go into the Clear state. The details are device dependent, but usually the device is left in the same state as when its power is first turned on. 38,43,48,75-77

Dense Subset — A subset of the Primary Command Group, consisting of only the Listen Address Group (LAG) and Talk Address Group (TAG). ISO codes Space through Underline, inclusive. (Values 20 Hex through 5F Hex). 77
DION  Data Input/Output line n (n goes from 1 through 8)  

DT  Device Trigger interface function - the interface function which allows a 
device to start its basic operation started either individually or as part of a 
group. This function may be used to start several devices simultaneously.  

DTAS  Device Trigger Active State  

DTIS  Device Trigger Idle State  

END  END - a uniline message sent by a Talker (EOI line active true) at the 
same time a data byte is sent on the data (DIO) lines. The message 
indicates that this is the last data byte to be sent. (See EOS for an 
alternate way of terminating a string sent by a Talker).  

EOI  End Or Identify - a uniline message which serves two purposes: if asserted 
true by a Talker it indicates that the last byte of a string is being sent. 
If asserted true by a Controller it initiates a Parallel Poll.  

EOS  End Of String - a multiline message sent by a Talker to indicate that the 
last byte of a string has been sent. Its value (ISO code) is determined by 
what the Listener(s) recognize.  

General Interface Management lines - the five lines used to perform system 
operations, such as Parallel Poll, Interface Clear, etc. Several of the 
lines are also used in data transactions: an example is EOI, which may be 
used to signal the end of a multibyte transaction. The five lines are ATN, 
EOI, IFC, REN and SRQ.  

GET  Group Execute Trigger - a multiline message (98 Hex) sent by the 
Controller indicating that all devices addressed as Listeners are to start 
performing their respective functions. This command is often used to start 
several pieces of equipment in synchronism.  

GTL  Go To Local - a multiline message (01 Hex) sent by the Controller 
indicating that all devices addressed as Listeners are to go to the Local 
state: i.e., local controls on the front or back panel (instead of device 
dependent messages on the 488 bus) control device operation. (See Local 
Control)  

gts  go to standby - a local message sent by a device to its Controller interface 
function telling it that it is finished sending commands. The response is 
that the Controller function releases the bus so that other operations (e.g., 
a Talker sending data to Listeners) may proceed.  

A-4
IDY | Identify - a uniline message sent by the Controller during a Parallel Poll telling the other devices to assert their Parallel Poll responses on the data bus.  
35, 48, 75–76

IFC | Interface Clear - a uniline message sent by the System Controller telling all other devices on the bus to go to the Idle state. This message is used to place all devices in a known state. It should be used sparingly because any bus transaction is terminated by this function.  
24, 29, 41–42, 48, 75–76

ISO | Code - a seven bit code equivalent to the American National Code for Information Interchange, ANSI X3.4-1968 (often called ASCII).  
46, 50, 77

ISR | Individual Service Request - a local message sent by a device to its Parallel Poll interface function. If the individual status (see "IST") message is equal to the S (Sense) bit received as part of the most recently received PPE (Parallel Poll Enable) command, the PPR (Parallel Poll Response) byte specified by the three bits P1–P3 of the most recent PPE command must be sent true upon receipt of an IDY (Identify) command from the Controller. Alternately, if subset PP2 (Parallel Poll function cannot be configured by the Controller) is used, local messages are substituted for S, P1–P3.  
35–37, 75

IST | Individual Status - a local message used by the Parallel Poll function to determine the proper response to an IDY (Identify) command from the Controller. See "ISR".  
35–36

L | Listen Interface function - the function which allows a device to receive data from the 488 bus.  
28

LACS | Listener Active State  
29–30

(LAD) | The listen address of a specific device (received as MLA). See "MLA".  
43

LADS | Listener Addressed State  
28–29

LAG | Listen Address Group - a subset of the ISO-7 codes, being characters SPACE through ? (20 Hex through 3F Hex).  
48, 77

LE | Listen Extended Interface function - similar to the Listen function except that a Secondary Address must be used as well as the Primary Address used for the Listen function.  
30

LIDS | Listener Idle State  
28–29
Local LockOut - a multiline command (11 Hex) sent by the Controller which tells all devices with the RL (Remote Local) interface function to obey device dependent messages sent over the 488 bus instead of their local controls (e.g., front panel).

Local control - the device is programmed by its controls instead of by the 488 interface. An example is a digital multimeter; the range, function, sample rate, etc. are set by front panel controls if it is under local control.

Local message - a message sent between a device function and an interface function. It may cause a remote message to be sent from the interface function over the 488 bus.

Listen only - a local message which causes the Listen function of the device to act as if it had been addressed by the Controller.

Listener Primary Addressed State (LPAS)

Listener Primary Idle State (LPIS)

Listen - a local message which when true and the Controller is in the active state causes the L (Listen) or LE (Listen Extended) function to go from the Idle (LIDS) to the Addressed (LADS) state.

Listen only - a local message which when true and the Controller is in the active state (CACS) causes the L (Listen) or LE (Listen Extended) function to go from the Addressed (LADS) to the Idle (LIDS) state.

Local With Lockout State (LWLS)
MLA  My Listen Address - the address which the L (Listen) or LE (Listen Extended) function will respond to. Note that the standard does not allow a 488 bus system to have both an L and an LE interface function which respond to the same primary address. MLA must belong to the LAG (Listen Address Group).
48, 75–76

MSA  My Secondary Address - the secondary address which the TE (Talk Extended) or LE (Listen Extended) functions will respond to if they are in the Primary Addressed state (TPAS or LPAS, respectively). MSA must belong to the SCG (Secondary Command Group).
24, 48, 75–76

MTA  My Talk Address - the primary address which the T (Talk) or TE (Talk Extended) function will respond to. Note that the standard does not allow a 488 bus system to have both a T and TE interface function simultaneously with the same primary address. MTA must belong to the TAG (Talk Address Group).
24, 29, 48, 75–76

multiline message - a message that is sent over two or more lines of the 488 bus. An example is Device Clear (DCL) (14 Hex sent out on the data (DIO1–DIO8) lines by the Controller).
45

nba  new byte available - a local message sent by a device to its Source Handshake (SH) function to inform it that another byte is available for it to place on the bus data (DIO1–DIO8) lines.
19, 75

NDAC Not Data ACcepted - one line of the 488 bus which carries the complement of the Data ACcepted (DAC) message. It is one of the three Data Byte Transfer Control lines. (See DAC).

NPRS Negative Poll Response State
32

NRFD Not Ready For Data - one line of the 488 bus. It carries the complement of the Ready For Data (RFD) message, and is one of the three Data Byte Transfer Control lines. (See RFD).

NUL null byte: all eight bits are false.
23, 42, 48

OSA Other Secondary Address - a secondary address which is not the same as the secondary address of the TE (Talk Extended) function while it is in the TPAS (Talk Primary Addressed state), or of the LE (Listen Extended) function while it is in the LPAS (Listen Primary Addressed state). OSA must belong to the SCG (Secondary Command Group).
48, 75–76
OTA Other Talk Address - an address other than a device's own talk address. Some devices which are capable of talking unaddress themselves if they sense that the Controller is addressing another Talker. This feature can be convenient because an UNTalk (UNT) command is not needed. OTA must belong to the TAG (Talk Address Group).

24,48,75-76

PACS Parallel poll Addressed to Configure State

35-36

Passive False - a message which when asserted on the 488 bus is NOT guaranteed to be received as false. It is overridden by an active true message.

16

Passive True - a message which when asserted on the 488 bus is NOT guaranteed to be received as true. It is overridden by an active false message.

16

PCG Primary Command Group - a subset of the ISO-7 code. It consists of all characters NUL through UNDERLINE (00 Hex through 5F Hex). It includes all of the ACG (Addressed Command Group), UCG (Universal Command Group), LAG (Listen Address Group) and TAG (Talk Address Group).

35,49,75-77

pon power on - a local message sent by the device to its own interface to inform it that power has just been applied. The interface should reset all functions (e.g., Listen, AH, Talk, etc.) to their Idle states.

75

PP Parallel Poll interface function - the function which allows a device to respond to a Parallel Poll from the Controller.

35

PPAS Parallel Poll Active State

35-36

PPC Parallel Poll Configure - a multiline message (05 Hex) sent by the Controller which causes the device presently addressed as a Listener (e.g., in the LADS state) to go into the PACS (Parallel Poll Addressed to Configure) state. While in the PACS, the PP (Parallel Poll) function is to obey the PRE (Parallel Poll Enable) and PPD (Parallel Poll Disable) messages sent by the Controller.

35,43,75-77

PPD Parallel Poll Disable - a multiline message (70 Hex) sent by the Controller which will place all devices in the PACS (Parallel Poll Addressed to Configure) state into the PPIS (Parallel Poll Idle) state.

35,43,49,75-76

PPE Parallel Poll Enable - a multiline message (60-6F Hex) sent by the Controller which will change all devices in the PPIS (Parallel Poll Idle) state to the PPSS (Parallel Poll Standby) state. It also specifies the PPRn (Parallel Poll Response byte) to be used and the S (Sense) of the PPR. The form of the message is (from most significant bit to least)
where X means don't care (may be either high or low), and the binary value formed by P3–P1 indicates which PPRn is to be used. Note that n of PPRn indicates which data line is to be made active true (i.e., D103 will be made active true when PPR3 is placed on the bus).

35, 43, 49, 75–76

**PPIS** Parallel Poll Idle State

35–36

**PPRn** Parallel Poll Response n (See PPE)

35, 49, 75–76

**PPSS** Parallel Poll Standby State

35–36

**PPU** Parallel Poll Unconfigure — a multiline message (15 Hex) sent by the Controller which takes all devices in the PPSS (Parallel Poll Standby) state and puts them into the PPIS (Parallel Poll Idle) state.

35, 43, 49, 75–77

**PUCS** Parallel poll Unaddressed to Configure State

35–36

**rdy** ready for next message — a local message sent by a device to its AH (Acceptor Handshake) interface function to indicate it is ready for another message byte from the 488 bus (i.e., another multiline remote message).

21, 75

**remote control** — a device is programmed by its 488 interface instead of by local controls. An example is a DMM whose function, range selection, etc are selected by messages sent to it over the 488 bus. See local control for contrast.

33

**REMS** REMote State

33–34

**REN** Remote ENable — one of the five General Interface Management lines. Also, a uniline message sent by the Controller to put devices addressed as Listeners into the REMS (Remote) state. When the Controller makes the REN message false, all devices are to go to the LOCS (Local) state.

33, 42, 49, 75–76

**RFD** Ready For Data — the complement appears on the NRFD line. This uniline message is used by the AH (Acceptor Handshake) function to indicate that it is ready to accept the next byte (multiline message). See AH for further information.

19, 22, 49, 75–76

**RL** Remote Local interface function — if present it allows a device to be switched from local to remote control and vice versa.

33
rpp request parallel poll - a local message sent to the Controller interface function when the device wants a Parallel Poll performed.
41,75

RQS ReQuest Service - the byte sent by the current Talker in response to a Serial Poll. Data bit 7 (D107) is true.
23,49,75-76

rsc request system control - a local message sent to the Controller interface function by the device when it wants to go to the SACS (System Control Active) state.
41,75

rsv request service - a local message sent by a device to its Service Request interface function to cause it to go to the SRQS (Service Request) state. As a consequence, the uniline message SRQ is sent active true until either rsv is sent false, or the Controller performs a Serial Poll of this device.
32,75

rtl return to local - a local message sent by a device to its Remote/Local interface function. The LOCS (Local) state is entered if neither LLO (Local Lockout) nor ACDS (Accept Data State) are true.
33,75

RWLS Remote With Lockout State
33,34

SACS System Control Active State
41,44

(SAD) Secondary AAddress - the secondary address of a specific device, and is received as either My Secondary Address (MSA) or Other Secondary Address (OSA). Its value must lie in the range 60-7E Hex. (See SCG).
43

(SBA) Status Byte, service request Acknowledged. A message sent over the 488 bus by the current Talker in response to a Serial Poll. This message indicates that this device was requesting service. Data bit 7 (D107) is true. (See RQS)
62

(SBN) Status Byte, service Not requested. Same as SBA but indicates that this device does not need service. Data bit 7 (D107) is false.
62

SCG Secondary Command Group. A subset of the ISO-7 code consisting of characters ACCENT GRAVE through TILDE (60 Hex through 7E Hex). Secondary Talk and Listen addresses must be selected from this group. (Note that DEL is not allowed as a secondary address).
49, 77
**SDC** Selected Device Clear – a multiline message (04 Hex) sent by the Controller indicating that all devices addressed as Listeners are to go into the DCAS (Device Clear Active) state. The details are device dependent, but usually the device is left in the same state as when its power is first turned on.

38, 43, 49, 75–77

**SDYS** Source Delay State
18–19

Secondary Commands – the commands PPE, PPD and (SAD).
43

**SGNS** Source Generate State
18–19

**SH** Source Handshake interface function. The function used by a Talker or Controller to insure proper communication of multiline messages. The NRFD and NDAC lines are sensed to determine whether the AH (Acceptor Handshake) function of some device is active (if both NRFD and NDAC are false simultaneously, there is no AH function on the bus, which is an error). The multiline message is placed on the eight data lines (D101–D108) and a 2 microsecond timeout is started. When NRFD is sensed false and the timeout has been completed (to insure the data lines have settled) DAV is asserted true (to show that the data is available and settled). Upon sensing NDAC false the SH asserts DAV false (to indicate that the data may no longer be valid) then removes the data. The whole cycle is repeated for subsequent bytes of data. (See AH for the other half of the handshake cycle).

18

**SIAS** System control interface clear Active State
41, 44

**sic** send interface clear – a local message which causes the devices' Controller interface function to enter the SIAS (System Control Interface Clear Active) state if it is the System Controller (i.e., it is in the SACS (System Control Active) state). As a consequence, the IFC (Interface Clear) signal is sent active true. (IFC is a uniline message sent on the IFC line).

41, 75

**SIDS** Source Idle State
18–19

**SIIS** System control interface clear Idle State
41, 44

**SINS** System control interface clear Not active State
41, 44

**SIWS** Source Idle Wait State
19–20

**SNAS** System control Not Active State
41, 44
SPAS Serial Poll Active State
24,26

SPD Serial Poll Disable – a multiline message (19 Hex) sent by the Controller. It informs all devices capable of being Talkers that they are to speak data when they are addressed to talk. (See SPE for contrast).
43,49,75–77

SPE Serial Poll Enable – a multiline message (18 Hex) sent by the Controller. It informs all devices capable of being Talkers that they are to speak their Serial Poll Status Byte (instead of data) when they are addressed to talk. See SBA, SBN, STB for further information about the status byte.
43,49,75–77

SPIS Serial Poll Idle State
24,26

SPMS Serial Poll Mode State
24,26

SR Service Request interface function. This function allows a device to asynchronously request service from the Controller-In-Charge.
31

SRAS System control Remote enable Active State
41,45

sre send remote enable – a local message sent by a device to its Control interface function. It causes the function to enter the SRAS (System Control Remote Enable Active) state only if it was already in the SACS (System Control Active) state. The uniline message REN is sent active true as long as the Controller remains in the SRAS state.
41,75

SRIS System control Remote enable Idle State
41,44

SRNS System control Remote enable Not active State
41,45

SRQ Service ReQuest – a uniline message sent on the SRQ line by the SR (Service Request) interface function. It is the duty of the Controller to provide the service needed.
49,75–76

SRQS Service ReQuest State
32

STB Status Byte. Data bits 1 through 6 and bit 8 (DIO1–DIO6, DIO8) sent in response to a Serial Poll. STB is combined with RQS to form the complete byte. (See SBA, SBN).
25,49,75–76
STRS Source Transfer State
18-19

SWNS Source Wait for New cycle State
18-19

T Talk interface function. This function allows a device to send information to other devices on the 488 bus. Only one byte (selected from the Talker Address Group) need be sent to address the Talker.
23

TACS Talker Active State
24, 26

(TAD) the Talk Address of a specific device. It is received as either My Talk Address (MTA) or Other Talk Address (OTA). It must be a member of the TAG (Talk Address Group).
43

TADS Talker Addressed State
23-24

TAG Talker Address Group. A subset of the ISO-7 code consisting of all characters from @ through UNDERLINE (40 Hex through 5F Hex). The address of a Talker (or the primary address of an Extended Talker) must be selected from this group. Note that UNDERLINE cannot be used as an address, for it is reserved as the Universal Untalk command.
49, 77

tca take control asynchronously - a local message sent by a device to its Controller interface function. It causes the function to go from the CSBS (Controller Standby) state to the CSWS (Controller Synchronous Wait) state, where it waits for at least 500 nsec (to allow the other devices on the 488 bus to respond to the active true assertion of the uniline message ATN), then proceed to the CAWS (Controller Active Wait) state. ATN is active true in both CSWS and CAWS.
41, 75

tcs take control synchronously - a local message sent by a device to its Controller interface function. It operates the same as tca EXCEPT that the function goes from CSBS to CSWS only when the AH (Acceptor Handshake) function is in the ANRS (Acceptor Not Ready) state. The effect is to insure that a message sent by a Talker is not garbled or misinterpreted as a message sent by the Controller; ATN will not become active true until the Source Handshake is complete (i.e., DAV is false, showing that the message is no longer valid).
21, 41, 75

TCT Take Control - a multiline message (09 Hex) sent by the Controller to inform the device currently addressed as a Talker that it is to become the Controller-in-Charge.
41, 43, 49, 75-77
TE  Talker Extended interface function. Similar to the Talker (T) function except that this one is addressed by two bytes. The first must be selected from the Talker Address Group (TAG) and the second from the Secondary Command Group (SCG).

TIDS Talker Idle State
23-24

Ton  talk only - a local message sent by a device to its Talk interface function. If IFC (Interface Clear) is false, the Talker function enters the TADS (Talker Addressed) state. Remember that only one Talker may be addressed at a time, so as long as ton is true no other device may have ton true or be addressed as a Talker by the Controller.

23,75

TPAS Talker Primary Addressed State
24,26

TPIS Talker Primary Idle State
24,26

UCG Universal Command Group - A subset of the ISO-7 code consisting of all characters from DLE through US (10 Hex through 1F Hex). These commands operate upon all devices which are capable of responding to a Controller; the devices are not individually addressed. For contrast see Addressed Command Group (ACG).

43,49,77

Uniline message - a message that uses only one line of the 488 bus. An example is Service ReQuest (SRQ).

Universal Command Group - See UCG

UNL UNListen - a multiline message (3F Hex or the character "?") sent by the Controller which forces the Listen function of all devices into the LIDS (Listen Idle) state.

29,43,49,75-77

UNT UNTalk - a multiline message (5F Hex or the character "_") sent by the Controller which forces the Talk function of all devices into the TIDS (Talk Idle) state.

49,77
Program Notes

The following listing of the P&T-488 Functional Test program is a version written to run under CP/M (an operating system produced by Digital Research). Only these few things need to be changed for it to run with any specific system:

1 MONITR (a name) – should be SET to the entry point of the user's monitor
2 PRINT (a routine at 03CA) The Functional test program CALLs the subroutine PRINT with the character to be printed in register A. Register pair HL must be preserved. All other registers may be trashed.
3 INSTAT (a routine at 03B5) The Functional Test routine CALLs the subroutine INSTAT. If no key has been pressed on the keyboard, INSTAT is to RETurn with the zero flag set. If a key has been pressed, INSTAT should check to see if it is a Control C. If it is a Control C, INSTAT should jump to the user's monitor, otherwise it is to RETurn to the calling program with the zero flag cleared.
4 BASPRT (a byte at 0103) The third byte of the Functional Test must contain the lowest I/O port address used by the P&T-488. If the address switch on the P&T-488 interface board has been changed from 7C Hex, the value contained in this location must also be changed.

```
P&T 488 TEST ROUTINES
RUNS UNDER CP/M

0100 ORG ¥100H
0100 # MONITR SET 0 ;CPM RE-ENTRY POINT
0105 # CPMIO SET 5 ;CPM I/O ROUTINE ENTRY POINT
010B C3502 ENTRY: JMP SELFCON ;GO TO SELECT FUNCTION ROUTINE
0103 7C BASPRT: DB 7CH ;BASE ADDR OF P&T 488 INTERFACE
0104 00 ERBYT: DB 0 ;ANY BIT SET TO 1 IS IN ERROR
0105 00 ERFLG: DB 0 ;PRINT 'NO ERRORS' IF ZERO
0106 00A STRMS: DB ¥DH,¥AH
0107 5926542834 DB 'P&T 488 Functional Test 12-29-78'
0109 00A DB ¥DH,¥AH
0110 00A DB ¥DH,¥AH
0113 4469736564F DB 'Disconnect all 488 devices from P&T 488 then',¥DH,¥AH
0113 2870726573 DB 'press any key to begin test',¥DH,¥AH
0116 2854686528 DB 'The power does not have to be turned off before',¥DH,¥AH
0118 6469736564F DB 'disconnecting 488 devices',¥DH,¥AH,¥DH,¥AH
011B 444154C1 DATMS: DB 'DATE',¥AH+00H
011D 004F004D4100MMS: DB 'COMMAND LIN',¥AH+00H
011E 584152444C01POLMS: DB 'PARALLEL POL',¥AH+00H
011E 49455445521SPMS: DB 'INTERRUPT SERVICE REGISTE',¥AH+00H
0270 00A417474PLUGMS: DB 'ADH,BAH,Attach test plug then press any key',¥DH,¥AH
0222 34363829430BMS: DB '488 CABLE',¥AH+00H
0227 45585455521XFMS: DB 'EXTERNAL INTERFACE CLEA',¥AH+00H
022F 4558545552XATMS: DB 'EXTERNAL AT',¥AH+00H
022F 464E284552NOERR: DB 'NO ERRORS',¥DH,¥AH
0266 59265428241STDUN: DB 'P&T 488 functional test complete',¥DH,¥AH
0268 2845525240BITER: DB 'ERROR - bits in error are',¥AH+00H
026A 008A CRFL: DB ¥DH,¥AH
026B ;
0265 STAK: DS 20H ;STACK AREA
```
; **************************
; TEST EACH FUNCTION IN TURN
; **************************
; 0225 31C592 SELFNC: LXI SP,STAK ;SET STACK POINTER
; 0228 97 SUB A,97
; 0229 32E501 STA ERFLG ;RESET ERROR FLAG
; 022C CD3903 CALL SETUP ;SET UP 488 PORT ROUTINES
; 022F 218601 LXI H,STRTRMS ;PRINT STARTUP MESSAGE
; 0232 D7E93 CALL PRNT8
; 0235 CD0593 STRTT8: CALL INSTAT ;SEE IF A KEY HAS BEEN PUSHED
; 0238 CD5902 JZ STRTT8 ;NO, SO WAIT UNTIL ONE IS
; 023B CDE303 CALL DATA ;CHECK DATA PORT OPERATION
; 023E 210401 LXI H,DATMS ;SET UP 488 PORT ROUTINES
; 0241 CD7203 CALL ERTST ;CHECK COMMAND PORT OPERATION
; 0244 CD0401 LXI H,OMMS
; 0247 210410 LXI H,CMOMS
; 024A CD7203 CALL ERTST
; 024D CD1004 CALL PPR ;CHECK PARALLEL POLL RESPONSE
; 024F 21E901 LXI H,POLMS
; 0252 CD7203 CALL ERTST
; 0255 CD5794 CALL ISRV ;CHECK INTERRUPT SERVICE REGISTER
; 0258 21ED01 LXI H,ISRVMS
; 025B CD2703 CALL ERTST
; 025E 217702 LXI H,PLUGMS ;TELL OPERATOR TO ATTACH PLUG
; 0261 D7E93 CALL PRNT8
; 0264 CD0503 PLUGN8: CALL INSTAT ;SEE IF A KEY HAS BEEN PRESSED
; 0267 CD5902 JZ PLUGN8 ;NO, SO WAIT UNTIL ONE HAS BEEN
; 026A CD8E02 STA CBTST ;CHECK CONTINUITY OF 488 CABLE
; 026D 21E202 LXI H,DUBLMS
; 0270 CD7203 CALL ERTST
; 0273 CD0404 CALL XIFC ;CHECK RESPONSE TO EXTERNAL IFC
; 0276 213702 LXI H,XIFMS
; 0279 CD7203 CALL ERTST
; 027C CD0404 CALL XATN ;CHECK RESPONSE TO EXTERNAL ATN
; 027F 214F02 LXI H,XATMS
; 0282 CD7203 CALL ERTST
; 0285 3A8501 LDA ERFLG ;HAVE ANY ERRORS OCCURRED?
; 0288 218602 LXI H,NOERR
; 028B CD7E03 CALL PRNT8 ;PRINT 'NO ERRORS'
; 028E CD7203 CALL ERTST
; 0291 216602 LXI H,TSTDUN
; 0294 CD7203 CALL PRNT8 ;PRINT 'TEST COMPLETE'
; 0297 D7E93 JMP MONITR
; 029A 3A8301 SETUP: LDA BASPRRT ;GET PORT ADDRESS
; 029D CD6FC CALL E63CH ;MAKE SURE IT IS A VALID ISR PORT ADDR
; 02A0 32B803 STA ISRI
; 02A3 32E501 STA ISR1
; 02A6 325E03 STA ISR0
; 02A9 3C INR A,3 ;CALCULATE COMMAND LINE PORT ADDR
; 02AB 326103 STA CMD1
; 02AE 326403 STA CMD0
; 02B2 3C INR A,3 ;CALCULATE DATA LINE PORT ADDR
; 02B5 325703 STA DAT1
; 02BB 326A03 STA DAT01
; 02BC 3C INR A,3 ;CALCULATE PARALLEL POLL RESPONSE ADDR
; 02BD 326D03 STA PPI1
; 02C0 327003 STA PPI0
; 02C3 CA RET
; 02C6 03 ISRI: DB 0DBH ;IN ISR
; 02C8 01 ISRI1: DB 0
; 02CB CA RET
; 02D0 03 ISR0: DB 0D3H ;OUT ISR
; 02D2 00 ISR01: DB 0
; 02D5 CA RET
;
**P&T - 488 Functional Test Program**

```
0360 DB  CMD1: DB $DBH ;IN CMDPORT
0361 DB  CMD1: DB $0
0362 C9  RET

0363 D3  CMD2: DB $D3H ;OUT CMDPORT
0364 00  CMD2: DB $0
0365 C9  RET

0366 DB  DAT1: DB $DBH ;IN DATPORT
0367 00  DAT1: DB $0
0368 C9  RET

0369 D3  DAT2: DB $D3H ;OUT DATPORT
036A 00  DAT2: DB $0
036B C9  RET

036C DB  PPI: DB $DBH ;IN PARPOLL
036D 00  PPI: DB $0
036E C9  RET

036F D3  PPO: DB $D3H ;OUT PARPOLL
0370 00  PPO: DB $0
0371 C9  RET

0372 3A0A11 SERTEST: LDA ERBYT ;GET CUMULATIVE ERRORS FOR THIS TEST
0375 07  ORA A
0376 C8  RZ ;NO ERRORS
0377 320511 STA ERFLG ;SET ERROR FLAG SO 'NO ERRORS' MESSAGE
0378 0A83 CALL ERPRNT ;PRINT ERROR MESSAGE
0379 C9  RET

037A 2E0A3 C07E03 ERPRNT: CALL PRNT8 ;PRINT MESSAGE POINTED TO BY HL
037B 218802 LXI H,BITER ;PRINT 'BITS IN ERROR' MESSAGE
037C 0D7E03 CALL PRNT8
037D 2E3A01 MV1 L, 'I' ;PUT ASCII 'I' IN L
037E 3A0A01 LDA ERBYT
037F 0F  PRNT8: MOV A,M ;GET THE CHAR TO BE PRINTED
0380 CDCA03 CALL PRINT ;PRINT IT ON CONSOLE DEVICE
0381 7E  MOV A,M ;GET THE CHAR AGAIN
0382 23  INX H ;POINT TO NEXT CHAR
0383 E688 ANI 0FH ;SEE IF CARRY SET
0384 CA7E03 JZ PRNT8 ;NO, SO PRINT NEXT CHARACTER
0385 C9  RET

0386 3A0E03 C07E03 ERPRNT: CALL PRNT8 ;PRINT MESSAGE POINTED TO BY HL
0387 218802 LXI H,BITER ;PRINT 'BITS IN ERROR' MESSAGE
0388 0D7E03 CALL PRNT8
0389 2E3A01 MV1 L, 'I' ;PUT ASCII 'I' IN L
038A 3A0E01 LDA ERBYT
038B 0F  PRNT8: MOV A,M ;GET THE CHAR TO BE PRINTED
038C CDCA03 CALL PRINT ;PRINT IT ON CONSOLE DEVICE
038D 7E  MOV A,M ;GET THE CHAR AGAIN
038E 23  INX H ;POINT TO NEXT CHAR
038F CDCA03 CALL PRINT
0390 3E2A01 MV1 A,'I' ;FOLLOW WITH A SPACE
0391 CDCA03 CALL PRINT
0392 2C  NOBIT: INR L ;ADVANCE BIT NUMBER
0393 3E3B01 MV1 A,'I'
0394 3E3B01 MV1 A,'I'
0395 3E3A01 INB
0396 9D  CMP L ;HAVE WE FINISHED?
0397 7C  MOV A,H ;GET BITS AGAIN
0398 C29E03 JNZ NOBIT ;NO, MORE BITS TO TEST
0399 213A02 LXI H,$EFLF ;FINISH WITH <CR><LF>
039A CD7E03 CALL PRNT8
039B C9  RET

039C 3E0E01 INSTAT: MV1 C,11D ;DO CPM CONSOLE READY FUNCTION
039D CD0500 CALL CP140
039E 0E0100 CALL CPM10
039F 6E01 ANI 1 ;LOOK AT ONLY LSB
03A0 C8  RZ ;NO CHARACTER READY
03A1 0E01 MV1 C,1 ;GET THE CHARACTER
03A2 CD0500 CALL CPM10
03A3 E67F ANI 7FH
03A4 F603 CP1 3 ;CONTROL C?
03A5 CA0000 JZ MONITR ;YES, SO ABORT
03A6 C9  RET
```

---

**B-3**
### Functional Test Program

**PRINT:** PUSH H ;Preserve HL (only registers that

;need to be preserved)

<table>
<thead>
<tr>
<th>Address</th>
<th>ASCII</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>03CA E5</td>
<td></td>
<td>PRINT: PUSH H</td>
</tr>
<tr>
<td>03CB 5F</td>
<td></td>
<td>MOV E,A</td>
</tr>
<tr>
<td>03CC 0E02</td>
<td></td>
<td>MVI C,2</td>
</tr>
<tr>
<td>03CE 0D500</td>
<td></td>
<td>CALL CPM10</td>
</tr>
<tr>
<td>03D1 E1</td>
<td></td>
<td>POP H</td>
</tr>
<tr>
<td>03D2 09</td>
<td></td>
<td>RET</td>
</tr>
</tbody>
</table>

```
****** CLEAR THE INTERRUPT SERVICE REGISTER AND RELEASE
ALL COMMAND LINES AT THE PORTS CORRESPONDING TO
THE FIRST ENTRY IN BASPRT ******
```

<table>
<thead>
<tr>
<th>Address</th>
<th>ASCII</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>03D3 3E7F</td>
<td></td>
<td>MVI A,-1</td>
</tr>
<tr>
<td>03D5 0D6303</td>
<td></td>
<td>CALL CMD0</td>
</tr>
<tr>
<td>03D8 0D6903</td>
<td></td>
<td>CALL DATO</td>
</tr>
<tr>
<td>03DB 0D6F03</td>
<td></td>
<td>CALL PPO</td>
</tr>
<tr>
<td>03DE 97</td>
<td></td>
<td>SUB A</td>
</tr>
<tr>
<td>03DF 0D5003</td>
<td></td>
<td>CALL ISRO</td>
</tr>
<tr>
<td>03E2 09</td>
<td></td>
<td>RET</td>
</tr>
</tbody>
</table>

```
****** CHECK DATA REGISTER FOR PROPER OPERATION
THE DATA REGISTER CORRESPONDING TO THE FIRST ADDRESS
IN BASPRT IS WRITTEN TO AND READ FROM. ALL BITS WHICH
ARE IN ERROR SHOW UP AS 1'S IN ERBYT (THE ERRORS ARE
CUMULATIVE) AND ARE ALSO SHOWN AS LIT BITS ON THE
PROGRAMMED OUTPUT DISPLAY ******
```

<table>
<thead>
<tr>
<th>Address</th>
<th>ASCII</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>03E3 0D3003</td>
<td></td>
<td>CALL RELCLR</td>
</tr>
<tr>
<td>03E5 3A8301</td>
<td></td>
<td>LDA BASPRT</td>
</tr>
<tr>
<td>03E8 06F601</td>
<td></td>
<td>ANI @FCH</td>
</tr>
<tr>
<td>03ED 3F</td>
<td></td>
<td>MOV E,A</td>
</tr>
<tr>
<td>03EE 57</td>
<td></td>
<td>MOV D,A</td>
</tr>
<tr>
<td>03EF 3F2003</td>
<td></td>
<td>JMP PORTST</td>
</tr>
</tbody>
</table>

```
****** PORTST PORT TEST ROUTINE
OUTPUTS 0,1,2,...,254,255 TO PORT WHOSE ADDRESS IS IN
THE E REGISTER, AND READS PORT WHOSE ADDRESS IS IN THE
D REGISTER. ANY BITS WHICH DO NOT MATCH ARE ACCUMULATED
AS CORRESPONDING 1'S IN THE C REGISTER AND IN MEMORY
LOCATION ERBYT. ******
```

<table>
<thead>
<tr>
<th>Address</th>
<th>ASCII</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>03F2 7B</td>
<td></td>
<td>PORTST: MOV A,E</td>
</tr>
<tr>
<td>03F3 320004</td>
<td></td>
<td>STA OUTDR</td>
</tr>
<tr>
<td>03F6 7A</td>
<td></td>
<td>MOV A,D</td>
</tr>
<tr>
<td>03F7 320204</td>
<td></td>
<td>STA INDR</td>
</tr>
<tr>
<td>03FA 0E00</td>
<td></td>
<td>MVI C,9</td>
</tr>
<tr>
<td>03FC 0600</td>
<td></td>
<td>MVI B,9</td>
</tr>
</tbody>
</table>

B-4
von 113FF 03 1141111 1111 114 “1 DB 114112 " 114113 A8 114114 BI IMII5 4F 114116 3211411 rI4 114aA C8 IM "C3FEr1:3 114E COD3r1:3 114E6FC 11416 F6al IM18 5F . 11419 57 1141A C3F2113 13A COO3S:3 IM2r1 3Ar13r11 IM23 E6FC IM25 F6r11 IM27 3233IM 1142A F6r12 IM2C 5F 11420 E6FE IM2F 57 1143r1 3EF6 IM32 03 (1433 "a IM34 C3F2r1:3 IM37 COO:3S3 IM3A fJErlrI IM3C lErIS Functional Test Program

DATLUP: MOV A,B ;OUTPUT TEST BYTE
0400 00 OUTDR: DB $03H
0401 08 DB $08H ;READ PORT
0402 00 INOR: DB $00B ;Determine which bits are in error
0403 A8 XOR B
0404 51 ORA C ;ADD IN PREVIOUS ERRORS
0405 4F MOV C,A ;AND SAVE UPDATED ERRORS
0406 324001 STA ERSY
0407 04 INR B ;INCREMENT TEST BYTE
0408 00 RZ ;...IF HAVE DONE ALL 256 POSSIBLE TESTS
0409 C9 JMP DATLUP

***************************************************************

; CHECK COMMAND REGISTER
; THE COMMAND REGISTER CORRESPONDING TO THE FIRST ADDRESS
; IN BASPR1 IS WRITTEN TO AND READ FROM. ALL BITS WHICH
; ARE IN ERROR SHOW UP AS '1'S IN MEMORY LOCATION ERSY
; AND ARE ALSO SHOWN ON THE PROGRAMMED OUTPUT.

*040E CDD3D3 CMND: CALL RELCLR ;CLEAR COMMAND, ISR
0411 3AD3D1 LDA BASPR1 ;GET PORT NUMBER
0414 E6FC ANI $0FCH ;MAKES IT INTO A VALID COMMAND PORT
0416 F601 ORI 1
0418 5F MOV E,A ;SET UP OUTPUT PORT
0419 57 MOV D,A ;AND INPUT PORT
041A C3F2D3 JMP PORTST ;GO TO COMMON PORT TEST ROUTINE

***************************************************************

; CHECK PARALLEL POLL RESPONSE
; THE EOI AND ATN LINES OF THE COMMAND PORT ARE PULLED
; LOW (ASSERTED TRUE) TO GET THE PARALLEL POLL RESPONSE
; ONTO THE DATA BUS. THEN TEST BYTES ARE WRITTEN INTO:
; THE PARALLEL POLL RESPONSE REGISTER AND THE DATA
; REGISTER IS READ. ANY BITS WHICH ARE IN ERROR ARE
; SAVED IN MEMORY LOCATION ERSY AND SHOWN ON THE
; PROGRAMMED OUTPUT.

041D CDD3D3 PPR: CALL RELCLR ;CLEAR COMMAND, ISR
0420 3AD3D1 LDA BASPR1 ;GET PORT NUMBER
0423 E6FC ANI $0FCH ;CHANGE IT INTO A COMMAND PORT
0425 F601 ORI 1
0427 3233D4 STA PPOUT ;STORE IT AS OPERAND OF OUTPUT INSTRUCTION
042A F602 ORI 2 ;CHANGE IT INTO A PARALLEL POLL PORT
042C 5F MOV E,A ;SET UP AS OUTPUT PORT
042D E6FE ANI $0FCH ;CHANGE IT INTO A DATA PORT
042F 57 MOV D,A ;SET UP AS AN INPUT PORT
0430 3EF6 MV1 A,$0FH ;ASSERT ATN, EOI TRUE
0432 03 DB $03H ;BY OUTPUTTING TO COMMAND PORT
0433 00 PPOUT: DB $6
0434 C3F2D3 JMP PORTST ;THEN JUMP TO COMMON PORT TEST ROUTINE

***************************************************************

; CHECK INTERRUPT SERVICE REGISTER
; WIGGLE EACH COMMAND LINE IN TURN AND CHECK
; FOR PROPER ISR RESPONSE

0437 CDD3D3 ISRV: CALL RELCLR ;CLEAR COMMAND, ISR
043A 00MM MV1 C,$0 ;INITIALIZE ERROR REGISTER
043C 1E60 MV1 E,$0 ;AND ISR RESET BYTE

B-5
Functional Test Program

************************************************************

INTERRUPT SERVICE REGISTER TEST TABLE

TABLE OF COMMANDS AND CORRESPONDING ISR CONTENTS FOR
THE ASSERTION AND THEN THE RELEASE OF THE COMMANDS.

THE FIRST BYTE IS THE NUMBER OF TESTS TO BE PERFORMED.

************************************************************

TESTS FOR TALK MODE

TSTBL1: DB 8D ; 8 TESTS ARE TO BE PERFORMED
TSTBL1: DB 07FH ; ASSERT DAY
TSTBL1: DW -1 ; LOW BYTE=RESPONSE OF ISR TO ASSERTION
TSTBL1: ; OF DAY, HIGH BYTE=RESPONSE TO RELEASE
TSTBL1: ; OF DAY

TSTBL2: DB 8D ; 8 TESTS
TSTBL2: DB 07FH ; ASSERT DAY
TSTBL2: DW -1 ; LOW BYTE=RESPONSE OF ISR TO ASSERTION
TSTBL2: ; OF DAY, HIGH BYTE=RESPONSE TO RELEASE
TSTBL2: ; OF DAY

LISTEN MODE

;
Functional Test Program

G494 FFFF DW -1
G496 F7 DB $F7H ;ATN
G497 FFFF DW -1
G499 FB DB $FBH ;SRQ
G49A FBFF DW $FFFBH
G49C FD DB $FDH ;REN
G49D FFFD DW $FDFDH
G49F FE DB $FEH ;PCC/RESET
G4A0 FFFF DW -1

*****
CHECK CABLE BY BRINGING EACH COMMAND LINE LOW
ONE AT A TIME AND OBSERVING WHETHER THE CORRESPONDING
DATA LINE IS ALSO BROUGHT LOW
*****

G4A2 C00303 CBLTST: CALL RELCLR ;RELEASE ALL DATA, COMMAND LINES
G4A5 0E88 MVI C,0 ;CLEAR CUMULATIVE ERROR REGISTER
G4A7 1EEF MVI E,$FEH ;MAKE ONLY ONE BIT TRUE IN TEST BYTE
G4A9 21C104 LXI H,CBLTBL ;POINT TO EXPECTED RESPONSES
G4AC 7B CBLUP: MOV A,E ;PUT TEST BYTE IN ACCUMULATOR
G4AD CD6303 CALL CMD0 ;AND THEN ON 488 COMMAND LINES
G4AE CD6603 CALL DAT1 ;GET BYTE FROM 488 DATA LINE PORT
G4AF AE XRA M ;SET ANY BITS WHICH DISAGREE WITH
G4B0 B1 ORA C ;EXPECTED RESPONSE
G4B5 4F MOV C,A
G4B6 328401 STA EBYTE
G4B9 23 INX H ;POINT TO NEXT EXPECTED RESPONSE
G4BA 7B MOV A,E ;GET TEST BYTE AGAIN
G4BB B7 RLC ;PREPARE TO CHECK NEXT LINE OF CABLE
G4BC 5F MOV E,A ;SAVE TEST BYTE
G4BD DACC84 JC CBLUP ;CARRY SET IF THERE ARE MORE LINES TO TEST
G4C0 C9 RET

G4C1 DF CBLTBL: DB $DFH ;D106 CORRESPONDS TO EO1
G4C2 EF DB $EFH ;D105 .. REN
G4C3 FB DB $FBH ;D103 .. SRQ
G4C4 F7 DB $F7H ;D104 .. ATN
G4C5 FD DB $FDH ;D102 .. IFC
G4C6 FE DB $FEH ;D101 .. NDAC
G4C7 7F DB $7FH ;D108 .. NRFD
G4C8 BF DB $BFH ;D107 .. DAV

*****
CHECK RESPONSE TO XIFC
(D102 IS CONNECTED TO XIFC BY SHORTING PLUG)
*****

G4C9 3E8A XIFC: MVI A,$B8H ;MAKE ALL DATA LINES (EXCEPT D102,4) TRUE
G4CA CD6903 CALL DAT0
G4CE 3E18 MVI A,18H ;MAKE ALL COMMAND LINES (EXCEPT IFC AND
G4D0 CD6303 CALL CMD0 ;ATN) TRUE
G4D3 97 SUB A
G4D4 CD5083 CALL ISR ;CLEAR ISR
G4D6 C8 MVI C,0 ;CLEAR CUMULATIVE ERROR REGISTER
G4D9 3E88 MVI A,8 ;NOW PULL DOWN D102 AS WELL
; (THIS APPLIES XIFC)

B-7
Functional Test Program

; CHECK RESPONSE TO XATN
; (D1O4 IS CONNECTED TO XATN BY THE SHORTING PLUG)

;************************************************************

0428 CD693 CALL DATO
042E CD5A3 CALL ISRI ;LOOK AT ISR
0441 EE8D XRI 8DH ;COMPARE TO EXPECTED VALUE
0443 B1 ORA C ;UPDATE CUMULATIVE ERROR REGISTER
0444 4F MOV C,A
0445 CD693 CALL CMDI ;LOOK AT COMMAND LINES
0448 EEFF XRI BFFH ;COMPARE TO EXPECTED VALUE
044A B1 ORA C ;UPDATE CUMULATIVE ERROR REGISTER
044B 4F MOV C,A
044C CD663 CALL DATI
044E EEFF XRI BFFH
0451 B1 ORA C
0452 4F MOV C,A
0453 323401 STA ERBYT
0456 C9 RET

;************************************************************

04F7 3E8A XATN: MVI A,8AH ;MAKE ALL DATA LINES (EXCEPT D1O2,4) TRUE
04F9 CD693 CALL DATO
04FC 3E58 MVI A,58H ;MAKE ALL COMMAND LINES (EXCEPT NRFD, ;ATN AND IF) TRUE
04FE CD3D3 CALL CMDI
0501 97 SUB A
0502 CD5B3 CALL ISRO ;CLEAR ISR
0505 5E99 MVI C,6 ;CLEAR CUMULATIVE ERROR REGISTER
0507 3E02 MVI A,2 ;NOW PULL DOWN D1O4 AS WELL ; (THIS APPLIES XATN)
0509 CD693 CALL DATO
050C CD5A3 CALL ISRI ;LOOK AT ISR
050E EE8D XRI 8DH ;COMPARE TO EXPECTED VALUE
0511 B1 ORA C ;UPDATE CUMULATIVE ERROR REGISTER
0512 4F MOV C,A
0513 CD663 CALL CMDI ;LOOK AT COMMAND LINES
0516 EE8F XRI BFFH ;COMPARE TO EXPECTED VALUE
0518 B1 ORA C ;UPDATE CUMULATIVE ERROR REGISTER
0519 4F MOV C,A
051A CD663 CALL DATI
051D EE7F XRI 7FH ;DATA LINES ARE FF, BUT NRFD IS CONNECTED ; TO D1O8 BY THE SHORTING PLUG
051F B1 ORA C
0522 4F MOV C,A
0521 323401 STA ERBYT
0524 C9 RET
0525 END

;************************************************************

0103 BASPRT 0288 BITER 0398 BITLP 022E CBLMS 04C1 CBLTBL
04A2 CBLSRT 04AC CBLSRP 0361 CMDI1 0360 CMDI 0104 CMDMS
0364 CMDMS 0363 CMDI 048E CMD 023A CRDF 03E3 DATA
0367 DAT11 0366 DATI 03FE DATLUP 0100 DATMS 036A DATO1
0369 DATO 0100 ENTRY 0184 ERBYT 0185 ERPLG 038A ERPT
0372 ERTEST 0482 IND 0385 INSTAT 0446 ISPAR 035B ISR1
035A ISR1 01ED ISRS 035E ISR01 035D ISR0 0437 ISRY
0341 ISTSRT 03AB NOBIT 035B NOERR 0450 OUTDR 0207 PLUGMS
0355 PLUGMS 015F PP5R 03F2 PPST 0360 PP1 0360 PP1
0370 PP01 056F PPO 0433PP 0410 PPR 03CA PRIN
037E PRNT 0303 RELCLR 044A RELARG 0230 SELFCN 0339 SETUP
0252 STAK 0106 STRMS 02B5 STRW8 0470 TSTBL1 0489 TSTBL2
0266 TSTDUN 024F XATMS 04F7 XAT 04C9 X1FC 0237 X1FMS

B-8
P&T-488 CUSTOM SOFTWARE PACKAGE

COPYRIGHT 1979 PICKLES & TROUT

CNTRL

THIS ROUTINE IS USED TO SEND COMMANDS TO THE 488 BUS
BY MEANS OF THE P&T 488. TO USE, POINT HL TO THE FIRST
BYTE OF THE STRING TO BE SENT, DE TO THE LAST BYTE, AND
BC TO THE USER-SUPPLIED JUMP TABLE. THEN CALL CNTRL;
THE ROUTINE RETURNS WITH HL POINTING TO THE LAST BYTE
SENT, ATN AND DAV FALSE. IF THE P&T 488 HAS BEEN
EITHER SELECTED AS A LISTENER OR IS TO PERFORM LISTENER
HANDSHAKE, THE ROUTINE RETURNS WITH NRFD TRUE, OTHERWISE
IT RETURNS WITH NRFD FALSE.

NOTE: THIS ROUTINE CAUSES THE P&T 488 TO EXERCISE CONTROL
IMMEDIATELY. IT IS UP TO THE USER TO INSURE THAT
CONTROL IS ASSUMED SYNCHRONOUSLY (THE INITIALIZE,
TALK AND LISTEN ROUTINES ALL RETURN OR BREAK AT
POINTS WHICH WILL GUARANTEE SYNCHRONIZATION). ALSO,
THE CONTROLLER STATE REGISTER IS LEFT IN THE STANDBY
STATE - THUS THE P&T 488 IS ASSUMED TO BE THE CONTROLLER
IN CHARGE UNTIL EITHER INIT IS CALLED, OR CSTAT IS
CLEARED BY THE USER.

GIM (GENERAL INTERFACE MANAGEMENT)

A ROUTINE WHICH ALLOWS THE USER TO CONTROL THE STATE OF
THE IFC, SQO, REN AND EOI LINES. CALL GIM WITH THE
APPROPRIATE BIT PATTERN IN THE A REGISTER.

D7 D6 D5 D4 D3 D2 D1 D0
X X X IFC X SQO REN EOI

X MEANS DON'T CARE.

IF THE BIT CORRESPONDING TO A PARTICULAR LINE IS HIGH (1)
THAT LINE IS ASSERTED TRUE ON THE 488 BUS. EG, TO SEND
OUT REN AND EOI, THE A REGISTER WOULD CONTAIN D3D (OR D3D,
D3D, ETC). IT IS UP TO THE USER TO RELEASE THE LINES.
 FOR INSTANCE, TO CLEAR (INITIALIZE) THE 488 BUS, ONE
WOULD CALL GIM WITH A=10H (ASSERT IFC TRUE) AND THEN
CALL GIM WITH A=00H. NOTE THAT THE 488 STANDARD REQUIRES
THAT IFC MUST BE ACTIVE NO LESS THAN 100 MICROSECONDS,
SO BE SURE TO WAIT BETWEEN THE TWO CALL GIM INSTRUCTIONS.

INIT

CALL INIT TO CLEAR THE P&T 488 INTERFACE. ALL 488
DATA LINES AND CONTROL LINES ARE LEFT IN THE PASSIVE
FALSE STATE, AND THE PARALLEL POLL RESPONSE IS SET TO
ALL LINES PASSIVE FALSE.

TO CLEAR THE 488 BUS, SET REGISTER B TO 02 BEFORE
CALLING INIT, AN INTERFACE CLEAR (IFC) WILL BE SENT
OUT ON THE 488 BUS AND PUT ALL DEVICES IN A KNOWN STATE.

LISTN

TO RECEIVE DEVICE-DEPENDANT MESSAGES FROM THE 488 BUS,
POINT HL TO THE BEGINNING OF A MEMORY BUFFER, DE TO ITS
END, AND BC TO THE USER-SUPPLIED JUMP TABLE, AND THEN
CALL LISTN. TWO BITS OF THE A REGISTER ARE USED AS FLAGS
FOR SPECIAL OPTIONS. IF BIT 0 (THE LEAST SIGNIFICANT
BIT) IS ZERO, THE LISTEN HANDSHAKE FUNCTION IS PERFORMED:
NO BUFFER IS USED AND THE P&T 488 PARTICIPATES IN THE HANDSHAKE PROCESS. THIS FUNCTION IS PRIMARILY TO ALLOW THE P&T TO ASSUME CONTROL SYNCHRONOUSLY, BUT CAN ALSO BE USED AS A BYTE-AT-A-TIME LISTENER, WITHOUT REQUIRING A BUFFER. IF BIT 0 OF THE A REGISTER IS NON-ZERO, THE NORMAL BUFFERED LISTENER FUNCTION IS PERFORMED. IF BIT 1 OF THE A REGISTER IS 1 THEN THE ROUTINE RETURNS WHEN EITHER END (EOI AND Dav TRUE, ATN FALSE) OR THE EOS BYTE IS SENSED. IF BIT 1 IS 0 THEN A RETURN IS MADE ONLY UPON DETECTION OF END. THE ROUTINE RETURNS WITH HL POINTING TO THE LAST BYTE RECEIVED AND NRFD IS LEFT ASSERTED TRUE.

NOTE: THIS ROUTINE PERFORMS THE LISTEN-ONLY FUNCTION. IT SETS THE LISTENER STATE TO ACTIVE WHEN CALLED, AND WHEN IT RETURNS IT LEAVES THE LISTENER STATE REGISTER IN THE LISTENER ADDRESSED STATE. USE THE ROUTINE "STATE" FIRST IF YOU WANT TO EXECUTE THE P&T 488 LISTEN FUNCTION ONLY IF THE CONTROLLER HAS ADDRESSED IT AS A LISTENER.

PISTF  SET "IST" FALSE
A ROUTINE WHICH SETS THE "IST" (INDIVIDUAL STATUS) MESSAGE FALSE. IF THE SENSE BIT OF THE MOST RECENT PARALLEL POLL ENABLE COMMAND WAS TRUE, THE PARALLEL POLL RESPONSE BYTE IS SET TO A NON-AFFIRMATIVE RESPONSE. THE PPR (PARALLEL POLL RESPONSE) MESSAGE IS DETERMINED BY THE LOW ORDER FOUR BITS OF THE BYTE STORED AT PPRSP. THE PARALLEL POLL FUNCTION IS PUT INTO THE PPSS (STANDBY) STATE.

PISTT  SET "IST" TRUE
A ROUTINE WHICH SETS THE "IST" (INDIVIDUAL STATUS) MESSAGE TRUE. IF THE SENSE BIT OF THE MOST RECENT PARALLEL POLL ENABLE COMMAND WAS TRUE, THE PARALLEL POLL RESPONSE BYTE IS SET TO AN AFFIRMATIVE RESPONSE. THE PPR (PARALLEL POLL RESPONSE) MESSAGE IS DETERMINED BY THE LOW ORDER FOUR BITS OF THE BYTE STORED AT PPRSP. THE PARALLEL POLL FUNCTION IS PUT INTO THE PPSS (STANDBY) STATE.

PPIDL  CLEAR PARALLEL POLL RESPONSE BYTE
A ROUTINE WHICH CLEARS THE PARALLEL POLL RESPONSE BYTE TO ALL ZEROS, THUS PREVENTING THE P&T-488 FROM RESPONDING TO A PARALLEL POLL. THE PARALLEL POLL FUNCTION IS PUT INTO THE PPIS (IDLE) STATE. EXECUTION OF THIS ROUTINE IS EQUIVALENT TO THE SENDING THE LOCAL MESSAGE "LPE" FALSE.

PPQRY  PARALLEL POLL
A ROUTINE WHICH PERFORMS A PARALLEL POLL AND RETURNS THE RESPONSE IN THE ACCUMULATOR. NOTE THAT NO CHECK IS MADE TO SEE IF THE P&T-488 IS THE CONTROLLER-IN-CHARGE, SO IT IS UP TO THE USER TO USE THIS ROUTINE ONLY WHEN THE P&T-488 (AND NOT SOME OTHER CONTROLLER) IS IN CHARGE. THE CONTROLLER IS LEFT IN THE STANDBY (CSBS) STATE.

SPIDL  RESET SERVICE REQUEST (SR) FCN TO IDLE
A ROUTINE WHICH PLACES A PASSIVE FALSE ON THE SERVICE REQUEST (SRQ) LINE AND PLACES THE SERVICE REQUEST FCN IN THE IDLE (NPRS) STATE.

SPQRY  SERIAL POLL QUERY: THIS ROUTINE SENDS OUT THE COMMANDS UNL (UNIVERSAL UNLISTEN) AND SPE (SERIAL POLL ENABLE). IT THEN CALLS CNTRL WITH THE TALK ADDRESSES IN ITS BUFFER CNTRL RETURNS EITHER WHEN A DEVICE REPLIES THAT IT IS
THE ONE DESIRING SERVICE, OR WHEN THE END OF BUFFER IS REACHED, THEN SPQRY SENDS THE COMMAND SPD (SERIAL POLL DISABLE), POINTS HL TO THE PRIMARY (AS CONTRASTED TO SECONDARY) TALK ADDRESS OF THE LAST DEVICE POLLED AND SETS UP THE ACCUMULATOR WITH 40 HEX IF NO DEVICE RESPONDED TO THE POLL, OR 00 HEX IF A DEVICE DID RESPOND. THE SERIAL POLL RESPONSE BYTE SENT BY THE RESPONDING DEVICE IS RETURNED IN REGISTER B. (NOTE THAT THE CONTENTS OF REGISTER B ARE MEANINGLESS IF NO DEVICE RESPONDED AFFIRMATIVELY TO THE SERIAL POLL.)

SPSQR SERIAL POLL SERVICE REQUEST
A ROUTINE WHICH SETS THE SERVICE REQUEST (SRQ) LINE TRUE THEN DETERMINES WHETHER THE P&T 488 IS THE CONTROLLER IN CHARGE. IF SO, IT JUMPS TO THE USER ROUTINE SVCRQ. OTHERWISE IT WAITS FOR AN EXTERNAL CONTROLLER TO DO A SERIAL POLL, TO WHICH IT RESPONDS THEN RETURNS TO THE CALLING PROGRAM.

STAO
CALL STAO TO SET TALKER, LISTENER ADDRESSES, SERIAL POLL STATUS AND END-OF-STRING (EOS) BYTES. HL MUST POINT TO ADDRESS OF FIRST OF FIVE BYTES.

EXAMPLE:
ADDRES: DB ' $ ' ; PRIMARY LISTENER ADDRESS = $ 
DB ' B ' ; PRIMARY TALKER ADDRESS = B 
DB 7FH ; PARALLEL POLL RESPONSE BYTE 
DB 0FFH ; SERIAL POLL STATUS BYTE 
DB 0AH ; END OF STRING BYTE = LINE FEED 

LXI H, ADDRS ; POINT HL TO BEGINNING OF ADDRESSES 
CALL STAO ; TRANSFER THEM TO 488 HANDLERS 

TALK
TO SEND DEVICE-DEPENDANT MESSAGES ON THE 488 BUS, POINT HL TO THE BEGINNING OF THE STRING OF BYTES TO BE SENT, POINT DE TO THE LAST BYTE OF THE STRING, AND POINT BC TO THE BEGINNING OF THE USER-SUPPLIED JUMP TABLE. CALL TALK; THE ROUTINE RETURNS WITH DAV FALSE. IF THERE IS NO INTERRUPTION, HL WILL POINT TO LAST BYTE OF STRING, BUT IF AN INTERRUPTION OCCURRED (SUCH AS SOME DEVICE REQUESTING SERVICE AND THE P&T 488 IS CONFIGURED AS THE SYSTEM CONTROLLER OR NO LISTENERS ON THE BUS), HL POINTS TO THE LAST BYTE SENT. IF THE A REGISTER IS NON-ZERO WHEN THE ROUTINE IS CALLED, THE LAST BYTE IN THE BUFFER WILL BE SENT WITH EOI ACTIVE TRUE.

NOTE: THIS ROUTINE PERFORMS THE TALK-ONLY FUNCTION (IT DOES NOT CHECK TO SEE WHETHER THE P&T 488 HAS BEEN ADDRESSED AS A TALKER BY THE CONTROLLER). EXECUTION OF THIS ROUTINE AUTOMATICALLY SETS THE TALK STATUS REGISTER TO ADDRESSED, AND WHEN THE BUFFER IS EMTPTIED THE TALK STATUS REGISTER IS LEFT SET TO TALKER ADDRESSED. IF YOU WANT TO GO TO THE TALK MODE ONLY IF THE CONTROLLER HAS ADDRESSED THE P&T 488 AS A TALKER, USE THE ROUTINE "STATE" TO DETERMINE WHETHER THE TALK FUNCTION HAS BEEN ADDRESSED.
; XCTRL  EXTERNAL CONTROLLER RESPONSE ROUTINE
; THIS ROUTINE ACCEPTS THE COMMANDS PRESENTED ON THE 488
; BUS BY AN EXTERNAL CONTROLLER (THAT IS, SOME DEVICE
; OTHER THAN THE P&T 488 IS THE CONTROLLER) AND UPDATES
; THE VARIOUS STATE REGISTERS AS NECESSARY. IT RETURNS TO
; THE CALLING PROGRAM WHEN THE EXTERNAL CONTROLLER CEASES
; SENDING COMMANDS (WHEN ATN BECOMES FALSE). BOTH NRFD
; AND NDAC ARE LEFT TRUE (LOW) TO PREVENT THE TALKER
; FROM SAYING ANYTHING UNTIL THE S-100 SYSTEM IS READY
; TO LISTEN.

USER SUPPLIED JUMP TABLE
; THIS TABLE PROVIDES THE ENTRY POINTS TO SPECIAL ROUTINES
; REQUIRED BY THE P&T 488 INTERFACE. IT IS THE USER'S
; RESPONSIBILITY TO PURGE THE STACK IF HE DOES NOT TERMINATE
; ANY OF THESE ROUTINES WITH A RETURN. THE TABLE MUST
; BE ORGANIZED IN THE ORDER SHOWN. THE USER NEED NOT RESTORE
; ANY OF THE REGISTERS BEFORE RETURNING.

EXAMPLE:

JMTBL: JMP TRIGR ;DETECTED DEVICE TRIGGER
        JMP DVCLR ;DETECTED DEVICE CLEAR
        JMP BUFUL ;LISTEN BUFFER IS FULL
        JMP ICCLR ;DETECTED INTERFACE CLEAR
        JMP BREAK ;AFTER EACH BYTE TRANSFER ON THE
                     ; 488 BUS, A CALL IS MADE TO BREAK.
                     ; THIS ALLOWS THE USER TO REGAIN
                     ; CONTROL OF THE S-100 SYSTEM BEFORE
                     ; A COMPLETE BUFFERFUL OF BYTES
                     ; HAS BEEN SENT OVER THE 488 BUS.
                     ; IF THE USER DOES NOT WANT TO
                     ; INTERRUPT 488 OPERATION, HE MERELY
                     ; EXECUTES A RETURN. THE A REGISTER
                     ; CONTAINS THE LAST BYTE COMMUNICATED
                     ; OVER THE 488 BUS, AND HL POINT
                     ; TO THE BUFFER ADDRESS CONTAINING
                     ; THAT BYTE. THUS THE USER CAN
                     ; TERMINATE LISTENING ON A PARTICULAR
                     ; ASCII CODE OR NUMBER OF CHARACTERS
                     ; COMMUNICATED.
        JMP NOLSN ;NOBODY'S LISTENING!
        JMP SVCRQ ;DETECTED SERVICE REQUEST AND P&T 488
                     ; IS THE CONTROLLER. HL POINTS
                     ; TO THE LAST BYTE IN THE BUFFER THAT
                     ; HAS BEEN INPUT/OUTPUT.
        JMP POC ;DETECTED S-100 RESET/POWER-ON-CLEAR
        JMP XATN ;SOMEBODY ELSE ASSERTED ATN TRUE!

STATE
; THIS ROUTINE PASSES INFORMATION TO THE USER ABOUT THE
; STATE OF THE 488 INTERFACE. AFTER A 'CALL STATE' THE
; BIT PATTERN IN THE A REGISTER HAS THE FOLLOWING MEANING:

   . . . . . . . BOTH TALK AND LISTEN ARE IDLE
   . . . . . . . .0 TIDS- (NOT TALKER IDLE STATE)
   . . . . . . . .1 LIDS- (NOT LISTENER IDLE STATE)
   . . . . . . .0 PPS (PARALLEL POLL IDLE STATE)
   . . . . . . .1 PPSS (PARALLEL POLL STANDBY STATE)
   . . . . . . .0 LOCS (LOCAL STATE)
   . . . . . . .1 LWLS (LOCAL WITH LOCKOUT)
   . . . . . . .0 REMS (REMOTE STATE)
   . . . . . . .1 RWLS (REMOTE WITH LOCKOUT)
   . . . . . . CIDS (CONTROLLER IDLE STATE)
   . . . . . . CIDS- (CONTROLLER NOT IDLE STATE)
THE HL REGISTER PAIR IS LEFT POINTING TO THE FIRST ENTRY OF THE STATE TABLE, THEREFORE THE USER MAY GET MORE DETAILED STATE INFORMATION BY ACCESSING THE TABLE HIMSELF.

ORG 8033H

ISRPT EQU 7CH ; ADDR OF 488 INTERRUPT STATUS PORT
CMOPT EQU 7DH ; ... OF COMMAND PORT
DATPT EQU 7EH ; ... OF DATA PORT
PORT EQU 7FH ; ... OF PARALLEL POLL RESPONSE PORT

GTL EQU 1 ; ISO-7 BIT CODE FOR "GO TO LOCAL" COMMAND
SOC EQU 4 ; SELECTIVE DEVICE CLEAR
PPC EQU 5 ; PARALLEL POLL CONFIGURE
GET EQU 8 ; GROUP EXECUTE TRIGGER
TCT EQU 9 ; TAKE CONTROL
LLO EQU 11H ; LOCAL LOCKOUT
DOL EQU 14H ; DEVICE CLEAR
PPU EQU 15H ; PARALLEL POLL UNCONFIGURE
SPE EQU 18H ; SERIAL POLL ENABLE
SPO EQU 19H ; SERIAL POLL DISABLE
UNL EQU 3FH ; UNIVERSAL UNTALK
UNT EQU 5FH ; UNIVERSAL UNTALK

VARIABLE AREA

LSTNP: DB C0FFH ; PRIMARY LISTEN ADDRESS
TALKP: DB OAFFH ; PRIMARY TALK ADDRESS
PPRSP: DB 0FFH ; PARALLEL POLL RESPONSE
SPSTS: DB 0FFH ; SERIAL POLL STATUS BYTE
EOB: DB 0AH ; END OF STRING CHARACTER

TSTAT: DB 0 ; TALK STATE (INITIALIZE TO TIDS)

...... 0 TIDS TALK IDLE STATE
...... 0 TADS TALKER ADDRESSED STATE
...... 0 TAGS TALKER ACTIVE STATE*
...... 0 SPAS SERIAL POLL ACTIVE STATE*
...... 0 SPIS SERIAL POLL IDLE STATE
...... 0 SPMS SERIAL POLL MODE STATE
...... 0 TPIS TALKER PRIMARY IDLE STATE
...... 1 TPAS TALKER PRIMARY ADDRESSED STATE

LSTAT: DB 0 ; LISTEN STATE

...... 0 LIDS LISTENER IDLE STATE
...... 0 LADS LISTENER ADDRESSED STATE
...... 0 LACS LISTENER ACTIVE STATE*
...... 0 LPIIS LISTENER PRIMARY IDLE STATE
...... 0 LPAS LISTENER PRIMARY ADDRESSED STATE

LISTEN HANDSHAKE - PARTICIPATES IN 488 COMMUNICATIONS BUT DOES NOT PLACE BYTE INTO BUFFER. MAINLY USED TO ALLOW SYNCHRONIZATION OF ASSUMPTION OF CONTROL BY THE P&T 488. MAY ALSO BE USED TO READ A BYTE AT A TIME (A CALL TO THE USER SUPPLIED ROUTINE "BREAK" IS EXECUTED AFTER EACH BYTE IS HEARD).
BUFFER ORIENTED LISTENER

IGNORE EOS

RETURN UPON RECEIPT OF EOS

;SERVICE REQUEST STATE

;REMOTE-LOCAL STATE

;PARALLEL POLL STATE

;CONTROLLER STATE

;STORAGE AREA FOR SECONDARY LISTEN ADDRESS

;FOR SECONDARY TALK ADDRESS

;MOST RECENT OUTPUT TO COMMAND LINES

;BEGINNING ADDRESS OF USER JUMP TABLE

;POINTER OF BUFFER PRESENTLY IN USE

;ADDRESS OF END OF TALK BUFFER
Custom Software Package  
Version 1.4

<table>
<thead>
<tr>
<th>Offset</th>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0016</td>
<td>D8</td>
<td>LBPTR: DW = 0; LISTEN BUFFER POINTER</td>
</tr>
<tr>
<td>0018</td>
<td>D8</td>
<td>LSEND: DW = 0; ADDRESS OF LISTEN BUFFER END</td>
</tr>
<tr>
<td>001A</td>
<td>D8</td>
<td>CBPTR: DW = 0; CONTROLLER BUFFER POINTER</td>
</tr>
<tr>
<td>001C</td>
<td>D8</td>
<td>CBEND: DW = 0; ADDRESS OF CONTROLLER BUFFER END</td>
</tr>
<tr>
<td>001E</td>
<td>D8</td>
<td>SBPTR: DW = 0; SERIAL POLL BUFFER POINTER</td>
</tr>
<tr>
<td>0020</td>
<td>D8</td>
<td>SBEND: DW = 0; ADDRESS OF SERIAL POLL BUFFER END</td>
</tr>
<tr>
<td>0022</td>
<td>D8</td>
<td>SBRSP: DB = 0; SERIAL POLL RESPONSE BYTE</td>
</tr>
<tr>
<td>0023</td>
<td>D8</td>
<td>LBYTE: DB = 0; CONTAINS BYTE MOST RECENTLY COMMUNICATED</td>
</tr>
<tr>
<td>0024</td>
<td>D8</td>
<td>TEO1: DB = 0; MAKE EOI TRUE ON LAST TALKER BYTE IF &lt;&gt; 0</td>
</tr>
<tr>
<td>0025</td>
<td>D8</td>
<td>XSPRS: DB = 0; BUFFER FOR SERIAL POLL RESPONSE TO AN EXTERNAL CONTROLLER</td>
</tr>
</tbody>
</table>

**FIXED AREA - PROMMABLE**

**JUMP TABLE OF ENTRY POINTS**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0026</td>
<td>C357BG</td>
<td>ENTBL: JMP INIT; CLEAR P&amp;T 488 (SEND IFC IF B=0)</td>
</tr>
<tr>
<td>0029</td>
<td>C39888</td>
<td>JMP TALK; TALK ONLY ROUTINE</td>
</tr>
<tr>
<td>002C</td>
<td>C353B1</td>
<td>JMP LISTN; LISTEN ONLY ROUTINE</td>
</tr>
<tr>
<td>002F</td>
<td>C387BG</td>
<td>JMP STADR; COPY LISTEN, TALK ADDRESSES</td>
</tr>
<tr>
<td>0032</td>
<td>C37CB2</td>
<td>JMP CNTRL; CONTROLLER FUNCTION</td>
</tr>
<tr>
<td>0035</td>
<td>C35FB4</td>
<td>JMP GIM; SET IFC, SRO, REN, EOI</td>
</tr>
<tr>
<td>0038</td>
<td>C3F6B4</td>
<td>JMP STATE; DETERMINE THE STATE OF THE INTERFACE</td>
</tr>
<tr>
<td>003B</td>
<td>C329B5</td>
<td>JMP XCTRL; EXTERNAL CONTROLLER SERVICE ROUTINE</td>
</tr>
<tr>
<td>003E</td>
<td>C30B66</td>
<td>JMP SQRY; SERIAL POLL QUERY ROUTINE</td>
</tr>
<tr>
<td>0041</td>
<td>C331B7</td>
<td>JMP SPSRQ; SERIAL POLL REQUEST ROUTINE</td>
</tr>
<tr>
<td>0044</td>
<td>C36FB7</td>
<td>JMP SPIOL; PUT SERVICE REQUEST FCN IN IDLE STATE</td>
</tr>
<tr>
<td>0047</td>
<td>C37CB7</td>
<td>JMP PPQRY; PARALLEL POLL ROUTINE</td>
</tr>
<tr>
<td>004A</td>
<td>C3A0B7</td>
<td>JMP PISTT; SET THE &quot;1ST&quot; MESSAGE TRUE</td>
</tr>
<tr>
<td>004D</td>
<td>C39FB7</td>
<td>JMP PISTF; SET THE &quot;1ST&quot; MESSAGE FALSE</td>
</tr>
<tr>
<td>0050</td>
<td>C374B4</td>
<td>JMP PPIDL; DISABLE PARALLEL POLL RESPONSE</td>
</tr>
</tbody>
</table>

**CONSTANTS**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0053</td>
<td>3F</td>
<td>BSPE: DB = UNL; COMMANDS UNLISTEN, SERIAL POLL ENABLE</td>
</tr>
<tr>
<td>0054</td>
<td>18</td>
<td>DB SPE</td>
</tr>
<tr>
<td>0055</td>
<td>19</td>
<td>BSPE: DB = SPD; COMMAND SERIAL POLL DISABLE</td>
</tr>
<tr>
<td>0056</td>
<td>5F</td>
<td>BUNT: DB = UNT; COMMAND ANY TALKER TO UNADDRESS ITSELF</td>
</tr>
</tbody>
</table>

**INIT - INITIALIZE P&T 488 AND 488 BUS**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0057</td>
<td>3EFF</td>
<td>INIT: MVI A, $0FFH; CLEAR ALL DATA, CONTROL LINES</td>
</tr>
<tr>
<td>0059</td>
<td>D37E</td>
<td>OUT DATPT</td>
</tr>
<tr>
<td>005B</td>
<td>CD3B82</td>
<td>CALL COMND</td>
</tr>
<tr>
<td>005E</td>
<td>D37F</td>
<td>OUT PPORT; CLEAR PARALLEL POLL RESPONSE PORT</td>
</tr>
<tr>
<td>0060</td>
<td>97</td>
<td>SUB A; ZERO A REGISTER</td>
</tr>
<tr>
<td>0061</td>
<td>D37C</td>
<td>OUT ISRPT; CLEAR ALL INTERRUPT LATCHES, SET; P&amp;T 488 TO NON-INTERRUPT MODE</td>
</tr>
<tr>
<td>0063</td>
<td>3206B0</td>
<td>STA LSTAT; UNADDRESS LISTEN FUNCTION</td>
</tr>
<tr>
<td>0066</td>
<td>3203B0</td>
<td>STA TSTAT; UNADDRESS TALK FUNCTION</td>
</tr>
<tr>
<td>0069</td>
<td>3207B0</td>
<td>STA SSTAT; NEGATIVE POLL RESPONSE (SERVICE REQUEST)</td>
</tr>
</tbody>
</table>
805C 320880 STA RSTAT ;LOCAL STATE (REMOTE-LOCAL)
805F 320880 STA PSTAT ;PARALLEL POLL IDLE STATE
8062 320480 STA CSTAT ;CONTROLLER IDLE STATE
8065 88 CMP B ;B=0? (IF SO, DO INTERFACE CLEAR)
8067 C9 RNZ
8067 3E0F MV1 A,8EH ;NOW DO A INTERFACE CLEAR
8069 D370 OUT CM0PT
806B E3 TWIDL: XTHL ;TWIDDLE THUMBS FOR AWHILE
806C E3 XTHL ;TO ALLOW OTHER DEVICES TO RESPOND
806D 30 DCR A
806E 27880 JNZ TWIDL
806F 3EFF MV1 A,8FH ;REMOVE IFC
8070 CD382 CALL COMND ;OUTPUT NEW COMMAND
8072 C9 RET

;**************************************************************
; STORE ADDRESSES - SETS TALKER, LISTENER ADDRESSES,
; PARALLEL POLL AND SERIAL POLL RESPONSE
; BYTES AND THE END-OF-STRING BYTE TO
; USER DEFINED VALUES
;**************************************************************

8078 1E05 STADR: MV1 E,5 ;SET BYTE COUNTER TO 5
8079 010080 LXI B,LISTNP ;POINT TO CONTROLLER ADDRESS TABLE
807C 7E NXTAD: MOV A,M ;GET USER-SUPPLIED ADDRESS
807D 02 STAX B ;SAVE IT IN CONTROLLER ADDRESS TABLE
807E 23 INX H ;POINT TO NEXT USER-SUPPLIED ADDR LOCATION
807F 03 INX B ; AND TO NEXT CONTROLLER ADDR LOCATION
8080 10 DCR E ;DECREMENT BYTE COUNT
8081 C28C80 JNZ NXTAD ;...THERE'S MORE TO TRANSFER
8082 CDC587 CALL PPNBL ;UPDATE PARALLEL POLL RESPONSE
8083 C9 RET

;**************************************************************
; TALK-ONLY FUNCTION
;**************************************************************

8088 322480 TALK: STA TEO1 ;SAVE EOI FLAG
808B 3A0580 LDA TSTAT ;GET TALK STATUS
808E 604 ANI 4 ;KEEP ONLY SERIAL POLL MODE STATE
808F 3C INR A ;SHOW TALKER IS ADDRESSED
8091 320580 STA TSTAT
8094 97 SUB A ;CLEAR A REGISTER
8095 320680 STA LSTAT ;UNADDRESS LISTENER
8098 221080 SHLD BPTR ;INITIALIZE BUFFER POINTER
809A 221280 SHLD TBPTR ; AS WELL AS TALK BUFFER POINTER
809C 8B XCHG
809D 221480 SHLD TBEND ;STORE END ADDRESS OF TALK BUFFER
809E 60 MOV H,B
809F 69 MOV L,C
80A0 220880 SHLD JMPAD ;STORE USER JUMP TABLE ADDRESS
80A3 3A0880 LDA GIMTC ;GET IFC, ATN, SRO, REN AND EOI STATE
80A6 9E08 ORI E0BH ;MAKE DRY, NRFD, NDAC PASSIVE FALSE
80A8 C370 OUT CM0PT ;OUTPUT COMMAND
80AA 320D80 STA GIMTC ;UPDATE MEMORY IMAGE OF MOST RECENT COMMAND
80AC DB7C TALK1: IN ISRPT ;CHECK FOR POC, ATN AND IFC
80AD 2F QMA
80AA E619 ANI 19H
80AB C44182 CNZ PAI
80AC DB7D IN CM0PT ;FIRST SEE IF THERE ARE ANY LISTENERS
80CB 2F CHA ;488 USES NEGATIVE LOGIC
80CC E660 ANI 60H ;KEEP ONLY RFD, DAC
80CD C0888 CZ UNLSN ;...NO LISTENERS, I REFUSE TO TALK TO MYSELF.
80DE E640 ANI 40H ;WAIT UNTIL READY FOR DATA IS TRUE
80DF 2C180 JNZ TALK1
80E0 2A1280 LHLD TBPTR ;GET THE DATA BYTE
80E1 7E MOV A,M
80E2 32258 STA LBYTE ;UPDATE MOST RECENT BYTE REGISTER
80E3 2F CMA ;488 HAS NEGATIVE TRUE LOGIC
80E4 037E OUT QATPT
80E5 2A1480 LHLD TBEND ;IS THIS THE LAST BYTE IN THE TALK BUFFER?
80E6 EB XCHG
80E7 2A1280 LHLD TBPTR
80E8 7C MOV A,H
80E9 8A CMP D
80EA 2B281 JNZ NTLST ;NO
80EB 7D MOV A,L
80EC 88 CMP E
80ED 2B281 JNZ NTLST ;NO
80EE 3A2480 LDA TEO1 ;IS EOI SUPPOSED TO BE TRUE?
80F0 87 ORA A
80F1 CA821 JZ NTLST ;NO
80F2 3A0800 LDA GIMTC
80F3 E6FE ANI 0FEH ;FORCE EOI ACTIVE TRUE
80F4 D37D OUT CMDPT ;OUTPUT COMMAND
80F5 320800 STA GIMTC ;UPDATE MEMORY IMAGE OF MOST RECENT COMMAND
80F6 3A0800 NTLST: LDA GIMTC ;NOW SET DATA ACTIVE TRUE
80F7 E67F ANI 7FH
80F8 F660 ORI 60H ; BUT SET NRFD, NDAC PASSIVE FALSE
80F9 D37D OUT CMDPT ;OUTPUT COMMAND
80FA 320800 STA GIMTC ;UPDATE MEMORY IMAGE OF MOST RECENT COMMAND
80FB 087C TALK2: IN ISRPT ;CHECK FOR POC, ATN, IFC
80FC 2F CMA
80FD E619 ANI 19H
80FE C4182 CNZ PA1
80FF D87D IN CMDPT ;WAIT FOR DATA ACCEPTED
8100 E620 ANI 20H ;LOOK AT DAC BIT
8101 CA8E81 JZ TALK2 ;DATA NOT ACCEPTED YET
8102 3A0800 LDA GIMTC ;GET STATE OF IFC, ATN, SRQ, REN, EOI
8103 F661 ORI 0E1H ;MAKE DAV, NRFD, NDAC, EOI PASSIVE FALSE
8104 D37D OUT CMDPT ;OUTPUT COMMAND
8105 320800 STA GIMTC ;UPDATE MEMORY IMAGE OF MOST RECENT COMMAND
8106 3EFF MVI A,0FFH ;REMOVE DATA FROM LINES
8107 037E OUT QATPT
8108 212881 TCNTU: LXI H,TCNTU ;GET TALK CONTINUATION ENTRY ADDRESS
8109 C00184 CALL SRVIS
810A C00184 CALL UBRAK ;SEE IF THE USER WANTS CONTROL OF S-100
810B 2A1480 LHLD TBEND ;SEE IF LAST BYTE WAS SENT
810C EB XCHG
810D 2A1280 LHLD TBPTR
810E 7C MOV A,H
810F 8A CMP D
8110 CA821 JNZ NTEND ;...NOT TALK BUFFER END
8111 7D MOV A,L
8112 88 CMP E
8113 CA821 JNZ NTEND ;...HAVE NOT FINISHED TALK BUFFER
8114 C9 RET
8115 E620 INX H ;POINT TO NEXT BYTE
8116 221280 SHLD TBPTR
8117 221000 SHLD SPTR ;UPDATE TALK BUFFER AND COMMON BUFFER POINTER
8118 C3C180 JMP TALK1 ;KEEP TALKING UNTIL INTERRUPTED OR FINISHED

D - 9
LISTEN-ONLY FUNCTION

0153 C5 LISTN: PUSH B ;SAVE BC FOR LATER
0154 1F RAR ;SEE IF BIT 0 OF A REG IS 0
0155 D25081 JNC BYTL ;YES, SO SET UP BYTE LISTENER
0158 080A MV1 B,100 ;SET LSTAT TO ACTIVE, BUFFERED
015A C36281 JMP EAST
015D 211880 BYTL: LXI H,LBEND ;USE THIS LOCATION AS THE "BUFFER"
0160 0601 MV1 B,1 ;SET LSTAT TO ADDRESSED/ACTIVE, NON-BUFFERED
0162 1F EAST: RAR ;TEST FOR EOS OPTION
0163 78 MOV A,B
0164 D25981 JNC LSET ;LEAVE OPTION FLAG CLEARED
0167 F610 ORI 10H ;SET OPTION FLAG IN LSTAT
0169 320680 LSET: STA LSTAT ;AND STORE IN LISTENER STATE BYTE
016C 01 POP B ;RESTORE BC REGISTERS
016D 3EFF MV1 A,FFH ;ASSERT DATA LINES PASSIVE FALSE
016F D37E OUT DATPT
0171 3A0800 LDA GIMTC ;GET STATE OF IFC, ATN, SRQ, REN AND EO1
0174 E69F ANI 9FH ;MAKE NRFD ACTIVE TRUE
0176 F6A0 ORI 0A0H ;MAKE DAY, NDAC PASSIVE FALSE
0178 D370 OUT CMPRT ;OUTPUT COMMAND
017A 320800 STA GIMTC ;UPDATE MEMORY IMAGE OF MOST RECENT COMMAND
017D 3A0580 LDA TSTAT ;UNADDRESS TALKER, BUT LEAVE SERIAL POLL
0180 E604 ANI 4 ;MODE STATE ALONE
0182 320580 STA TSTAT
0185 221080 SHLD BPTR ;INITIALIZE BUFFER POINTER TO BEGINNING
0188 221880 SHLD LBPTR ;DO THE SAME FOR THE LISTEN BUFFER
018B E6 XCHG
018C 221880 SHLD LBEND ;STORE ADDRESS OF LISTEN BUFFER END
018F 6B MOV H,B
0190 69 MOV L,C
0191 220800 SHLD JMPAD ;STORE USER JUMP TABLE ADDRESS
0194 DB7C DAVH: IN 1SRPT ;CHECK FOR POC, ATN AND IFC
0196 2F CMA
0197 619 ANI 19H
0199 C4582 CNZ LPAI
019C D670 LSNI: IN CMPRT ;WAIT UNTIL DAV IS HIGH (PASSIVE FALSE)
019E E680 ANI B0H
01A0 C4981 JZ DAVH
01A3 3A0800 LDA GIMTC ;SET NDAC, NRFD LOW
01A6 E69F ANI 9FH
01A8 F6A0 ORI 80H ;SET DAY PASSIVE FALSE (HIGH)
01AA D370 CMPRT ;OUTPUT COMMAND
01AC 320800 STA GIMTC ;UPDATE MEMORY IMAGE OF MOST RECENT COMMAND
01AF 21AF81 LCNTU: LXI H,LCNTU ;GET LISTEN CONTINUATION ADDRESS
01B2 CD184 CALL SRVIS
01B5 3A0800 LDA GIMTC ;GET LOW BITS OF CONTROL WORD
01B8 FC60 ORI 0C0H ;SET "NDAC" LOW, "NRFD" HIGH
01BA E6DF ANI 0DFH
01BC D370 OUT CMPRT ;OUTPUT COMMAND
01BE 320800 STA GIMTC ;UPDATE MEMORY IMAGE OF MOST RECENT COMMAND
01C1 DB7C DAVL: IN 1SRPT ;CHECK FOR ATN, POC OR IFC
01C3 2F CMA
01C4 E619 ANI 19H
01C6 C4582 CNZ LPAI
01C9 D670 CMPRT ;NOW WAIT FOR "DAV" LOW (ASSERTED TRUE)
Custom Software Package
Version 1.4

;**********************************************************************************************
; CQMN: ROUTINE TO OUTPUT BYTE IN A REGISTER
; TO GENERAL INTERFACE MANAGEMENT AND DATA TRANSFER CONTROL PORT. IT ALSO
; UPDATES GMTC, A MEMORY IMAGE OF THE MOST RECENT COMMAND PLACED ON THE
; GM & TC LINES.
;**********************************************************************************************

81D3 E69F  ANI  9FH
81D5 D37D  OUT CMDPT ;OUTPUT COMMAND
81D7 3200B0  STA GMTC ;UPDATE MEMORY IMAGE OF MOST RECENT COMMAND
81DA D87E  IN DATPT ;GET THE DATA
81DC 2F  CMP 488 USES ACTIVE LOW LOGIC
81DD 2A1680  LHLD LBPTR ;STORE BYTE IN BUFFER
81DE 77  MOV M,A
81E1 322380  STA LBYTE ;AND IN FIXED MEMORY LOCATION (FOR
; BYTE ORIENTED LISTENER)
81E4 D87D  IN CMDPT
81E6 F5  PUSH PSW ;KEEP IMAGE OF 488 CMD LINES SO CAN CHECK
; FOR END
81E7 3A0DB0  LDA GMTC
81EA F6A8  ORI 0A0H ;ASSERT ONLY "NRFD" (NDAC SET HIGH)
81EE 3200B0  OUT CMDPT ;OUTPUT COMMAND
81F1 3200B0  STA GMTC ;UPDATE MEMORY IMAGE OF MOST RECENT COMMAND
81F4 CDC184  CALL UBRK ;SEE IF USER WANTS CONTROL OF S-100
81F4 F1  POP PSW ;CHECK FOR EDI ACTIVE TRUE
81F5 E681  ANI 1
81F7 CA1282  JZ LDUN ;LAST BYTE HAS EDI TRUE.
81FA 3A0680  LDA LSTAT ;TERMINATE ON EOS?
81FD E681  ANI 18H
81FF CA0C82  JZ NEOS ;NO
8202 3A0480  LDA EOS ;GET END-OF-STRING BYTE
8205 2A1688  LHLD LBPTR
8206 BE  CMP M ;COMPARE TO BYTE JUST RECEIVED-ARE THEY
; THE SAME?
8209 CA1282  JZ LDUN ;YES
820C CDC184 NEOS: CALL BFCHK ;CHECK FOR FULL BUFFER
820F C39481  JMP DIVH ;REPEAT LOOP FOREVER
8212 3A0680  LDUN: LDA LSTAT ;KEEP HANDSHAKE AND EOS FLAGS
8215 E618  ANI 18H ;SHOW LISTEN STATE IS ADDRESSED (NOT ACTIVE)
8217 3C  INR A
8218 3200B0  STA LSTAT
821B C9  RET
821C 2A1680 BFCHK: LHLD LBEND ;PUT BUFFER END ADDRESS IN DE
821F EB  XCHG ; POINTER IN HL
8222 2A1688  LHLD LBPTR
8223 3A0680  LDA LSTAT ;DETERMINE IF BYTE OR BUFFER ORIENTED LISTENER
8226 E680  ANI 8 ;BYTE ORIENTED
8228 C8  RZ ;
8229 7D  MOV A,L ;CHECK FOR END OF BUFFER
822A BB  CMP E
822B C23382  JNZ NOFLO ;MORE BUFFER AVAILABLE
822E 7C  MOV A,H
822F BA  CMP D
8238 CAB284  JZ UBFUL ;BUFFER FULL, GO TO USER FOR INSTRUCTIONS
8233 25  NOFLO: INX H ;POINT TO NEXT BUFFER LOCATION
8234 221680  SHLD BPTR ;UPDATE BUFFER POINTER
8237 221680  SHLD LBPTR
823A C9  RET

;**********************************************************************************************
P3.T-488

823B D37D  COMND: OUT COMPT ;OUTPUT COMMAND
823D 320080  STA GIMTC ;UPDATE MEMORY IMAGE OF MOST RECENT COMMAND
8240 C9  RET

******************************************************************************

PAIL  CHECK FOR PAIL 488 LOCKOUT DUE TO S-100 RESET
PAIL1 EXTERNAL CONTROLLER assert ATN
PAIL2 488 EXTERNAL CONTROLLER asserting ATN
PAIL3 ONLY IF THE INTERFACE IS NOT LOCKED OUT DUE TO AN S-100 POC
PAIL4 OR 488 IFC (INTERFACE CLEAR).

******************************************************************************

8241 214182  PAI:  LXI H,PAIL ;RE-ENTER THIS ROUTINE UNTIL EACH OF
8244  E5  PAI1:  PUSH H ;POC, ATN AND IFC HAVE BEEN CLEARED
8245  D87C  IN  ISRPT
8247 1F  RAR  ;PUT POC BIT IN CARRY
8248 D2BE84  JNC UPOC ;IF POC ACTIVE TRUE
824B 1F  RAR  ;REN > CARRY
824C 1F  RAR  ;SRQ > CARRY
824D 1F  RAR  ;XATN > CARRY
824E 1F  RAR  ;XIFC > CARRY
824F D29F84  JNC UIFC ;XIFC IS ACTIVE TRUE
8252 17  RAL  ;XATN > CARRY
8253 D26F82  JNC PUATN ;XATN HAS CHANGED STATES
8256  E1  POP H ;CLEAR RE-ENTRY ADDRESS ON STACK
8257  C9  RET

8258 219C81  LPAI:  LXI H,LSN1 ;PUT COMMON LISTEN RETURN ADDRESS IN HL
LPAI1 IN  ISRPT
825B  E3  XTHL
825C  D87C  IN  ISRPT
825E 1F  RAR
825F D2BE84  JNC UPOC
8262 1F  RAR
8263 1F  RAR
8264 1F  RAR
8265 1F  RAR
8266 D29F84  JNC UIFC
8269 17  RAL
826A D26F82  JNC PUATN
826D  E3  XTHL ;RETURN TO CALLING PROGRAM
826E  C9  RET

826F D87D  PUATN: IN COMPT ;SEE IF ATN HAS BEEN Asserted OR RELEASED
8271  E608  ANI 8
8273 C8B84  JZ UATN ;asserted, keep RETURN ADDRESS
8276 3EF7  MV1 A,8F7H ;RELEASED, so reset XATN BIT IN ISR
8278 D37C  OUT ISRPT
8279 E3  XTHL ;PUT NORMAL RETURN ADDR BACK ON STACK
827B C9  RET

0 - 12
OUTLINE OF OPERATION:

- SET NRFD, NDAC LOW (TRUE)  \rightarrow SET UP AH (ACCEPTOR HANDSHAKE)
- SET DAV HIGH, ATN LOW  \rightarrow TAKE CONTROL OF THE BUS
- CLUP: SET NRFD HIGH  \rightarrow AH READY
- WAIT UNTIL NRFD HIGH  \rightarrow WAIT FOR OTHER DEVICES
- PLACE BYTE ON DATA LINES  \rightarrow CONTROLLER TELLS IT LIKE IT IS
- SET DAY LOW  \rightarrow AND CLAIMS THE DATA LINES ARE VALID
- SET NRFD LOW  \rightarrow AH PREPARES TO GET BYTE
- READ THE DATA LINES  \rightarrow AH GOT THE BYTE AND CHEWS IT
- SET NDAC HIGH  \rightarrow SET UP APPROPRIATE STATES
- WAIT FOR NDAC HIGH  \rightarrow CONTROLLER WAITS FOR OTHER DEVICES
- SET DAY HIGH  \rightarrow CONTROLLER PLANS TO CHANGE DATA LINES
- END OF CONTROLLER BUFFER?
  - YES: SET ALL DATA LINES HIGH
  - NO: CALL BREAK
  - ADVANCE BUFFER POINTER
  - JMP CLUP

SEE IF USER WANTS SOMETHING
SEND NEXT BYTE

CNTRL: CALL CTRL ;DO THE CONTROLLER THING
LDA CSTAT ;PUT CONTROLLER INTO STANDBY (CSBS)
ANI @FP0H ;BUT KEEP OTHER STATE INFO
ORI 6
STA CSTAT
LDA GIM1C ;RELEASE ATN LINE
ORI 8
CALL COMND
LOA CSTAT
TAKE CONTROLLER OUT OF IDLE STATE
ANI 9FH ;PULL NRFD, NDAC LOW (ACTIVE TRUE)
ORI 80H ;MAKE DAV HIGH (PASSIVE FALSE)
CALL COMND
ANI @F7H ;ASSERT ATN TRUE (LOW)
CALL COMND
LDA GIM1C
CLUP: CALL COMND
LDA GIM1C
ORI @OH ;SHOW ACCEPTOR HANDSHAKE READY
CALL COMND
82C2 CD4182  CTRL1: CALL PAI ;CHECK FOR P&T 488 LOCKOUT DUE TO
  ;EXTERNAL IFC OR S-100 POC
  ;>> NOTE << THE ATN WE ARE ASSERTING MASKS
  ;ANY EXTERNAL APPLICATION OF ATN, SO WE
  ;NEED NOT WORRY ABOUT SOME OTHER CONTROLLER
  ;SENDING ATN ACTIVE TRUE.

82C5 DB7D  IN CMOPT ;SEE IF ALL DEVICES READY FOR BYTE
82C7 E64F  ANI 40H
82C9 CAC282  JZ CTRL1 ;NOT READY YET
82CC 2A1A80  LHLD CBPTR ;GET THE BYTE
82CF 7E  MOV A,M
82D0 2F  QMA ;488 HAS NEGATIVE LOGIC
82D1 D37E  OUT DATPT
82D3 3A0D80  LDA GIMTC
82D6 E67F  ANI 7FH ;MAKE DAV ACTIVE TRUE (LOW)
82D8 CD3B82  CALL COMMD
82D8 E68F  ANI 0BFH ;MAKE NRFD TRUE (LOW)
82DD CD3B82  CALL COMMD
82E0 DB7E  IN DATPT ;READ THE BYTE
82E2 2F  QMA ;488 USES NEGATIVE LOGIC
82E3 47  MOV B,A ;SAVE IT FOR NOW
82E4 32238C1  STA LBYTE ;SAVE IT IN THE LAST BYTE REGISTER
82E7 3A0D80  LDA GIMTC
82EA F62I  ORI 02H ;SHOW CONTROLLER WE GOT IT
  ;(MAKE NDAC PASSIVE FALSE)
82EC CD3B82  CALL COMMD
82EF CD7383  CALL UPDB ;LOOK THIS COMMAND OVER AND SEE IF ANY
  ;OF THE INTERFACE FUNCTIONS ARE
  ;AFFECTED. UPDATE THE FUNCTION STATES
  ;AS NECESSARY. THE COMMAND IS IN REG B.

82F2 CD4182  CTRL2: CALL PAI ;CHECK FOR LOCKOUT DUE TO POC, XATN OR XIFC
82F5 DB7D  IN CMOPT ;WAIT FOR NDAC HIGH (FALSE)
82F7 E62F  ANI 20H
82F9 CAF282  JZ CTRL2 ;NDAC LOW (TRUE)
82FC 3A0D80  LDA GIMTC ;SET DAV HIGH (FALSE)
82FF F680  ORI 80H
8301 CD3B82  CALL COMMD
8304 3EFF  MV1 A,8FFH ;RELEASE THE 488 DATA LINES
8306 D37E  OUT DATPT
8308 210883  LXI H,CTRNTU ;SET UP SRQ RE-ENTRY ADDRESS
830B CD0184  CALL SRVIS ;CHECK FOR SRQ (SERVICE REQUEST)
830E CDC184  CTRL6: CALL UBRK ;IF USER WANTS CONTROL OF S-100
8311 2A1C80  LHLD CBEND ;GET CONTROLLER BUFFER END ADDRESS
8314 8B  XCHG
8315 2A1A80  LHLD CBPTR ;AND POINTER ADDRESS
8318 7C  MOV A,H
8319 BA  CMP D
831A C22683  JNZ NCEND ;NOT AT END OF CONTROLLER BUFFER
831D 7D  MOV A,L
831E 8B  CMP E
831F C22683  JNZ NCEND ;NOT AT END OF CONTROLLER BUFFER
8322 CD3383  CALL ADDRES ;FINISH ADDRESSING OF TALK, LISTEN
  ;OF P&T-488
8325 C9  RET

8326 2A1A80  NCEND: LHLD CBPTR ;GET CONTROLLER BUFFER POINTER
8329 23  INX H ;POINT TO NEXT ENTRY IN BUFFER
832A 221A80  SHLD CBPTR
832D 221B80  SHLD BPTR ;UPDATE COMMON BUFFER POINTER
8330 CB3A82  JMP CLUP ;AND SEND NEXT BYTE

D - 14
; THIS ROUTINE CHECKS TO SEE IF THE TALK OR LISTEN FUNCTION IS IN THE PRIMARY ADDRESSED STATE. IF IT IS, THIS ROUTINE CHANGES THE STATE TO ADDRESSED AND PUTS A DUMMY SECONDARY ADDRESS IN THE SECONDARY ADDRESS STORAGE LOCATION.

8333 3A0580 ADDRESS: LDA TSTAT ;SEE IF TALKER IN PRIMARY ADDRESSED STATE
8336 E600 ANI 8
833B CA583 JZ NTPRI ;NO, SO LEAVE IT ALONE
833B 3A0580 LDA TSTAT ;GET TALKER STATE AGAIN
833E F601 ORI 1 ;SHOW IT AS ADDRESSED
8340 E6F7 ANI 0F7H ;AND NO LONGER PRIMARY ADDRESSED
8342 320580 STA TSTAT
8345 3E7F MSI A,7FH ;PUT IN DUMMY SECONDARY ADDRESS
8347 320C80 STA TALKS ;TO SHOW NON-EXTENDED TALKER
834A 3A0680 LNADR: LDA LSTAT ;KEEP THE HANDSHAKE AND EOS FLAGS,
834E E618 ANI 18H ;BUT UNADDRESS THE LISTEN FUNCTION
8350 320680 STA LSTAT ;SINCE THE TALK FUNCTION IS ADDRESSED
8352 C9 RET

8353 3A0680 NTPRI: LDA LSTAT ;NOW CHECK LISTENER STATE
8356 E604 ANI 4 ;TO SEE IF IN PRIMARY ADDRESSED STATE
835B CA7283 JZ NLPR1 ;NO, GO ON TO NEXT FUNCTION
835F 3A0680 LDA LSTAT ;GET LISTEN STATE AGAIN
8363 F601 ORI 1 ;SHOW IT AS ADDRESSED
8366 E6FB ANI 0FBH ;BUT NOT PRIMARY ADDRESSED
8368 320680 STA LSTAT
836B 3E7F MSI A,7FH ;AND PUT DUMMY SECONDARY ADDRESS TO
836D 320B80 STA LSTNS ;SHOW NON-EXTENDED LISTENER
836F 3A0580 TNADR: LDA TSTAT ;KEEP THE SERIAL POLL STATE BUT
8371 E604 ANI 4 ;UNADDRESS THE TALK FUNCTION SINCE THE
8373 320580 STA TSTAT ;LISTEN FUNCTION IS ADDRESSED
8375 C9 NLPR1: RET

; UPD8 THE COMMAND FROM THE CONTROLLER-IN-CHARGE IS IN THE B REGISTER. LOOK AT THE COMMAND AND UPDATE THE FUNCTIONAL STATE OF THE INTERFACE AS IS NECESSARY.

8378 7B UPD8: MOV A,B ;PUT THE COMMAND IN REGISTER A
837B E6F7 ANI 7FH ;STRIP THE PARITY BIT
837E 47 MOV B,A ;SAVE IT IN B FOR LATER USE
837F F600 CPI 6BH ;BELONGS TO SECONDARY COMMAND GROUP
8380 F2C683 JP RSCG ;(SECONDARY ADDRESS, ETC)
8383 CD3383 CALL ADDRESS ;IF TALK OR LISTEN IS PRIMARY ADDRESSED
8386 3A0580 ;CHANGE IT TO ADDRESSED AND PUT IN
8389 3A0580 DUMMY SECONDARY ADDRESS

; PRIMARY COMMAND GROUP <<<<<

838F 78 MOV A,B ;GET COMMAND AGAIN
8390 FE05 CPI PPC ;IS IT PARALLEL POLL CONFIGURE?
8395 CA583 JZ RPPC ;YES, SO UPDATE THE PP STATE
8398 3A0580 LDA PSTAT ;NO, SO PUT PP STATE INTO PUCS
839B E6FB ANI 0FBH

D - 15
; 83A8 329980 STA PSTAT ;GET THE COMMAND AGAIN
83B0 78 MOV A,B
83BE FE40 CPI 40H
8398 2AFB84 JP RTAG ;TALK ADDRESS GROUP
8393 FE28 CPI 28H
8395 F22884 JP RLAG ;LISTEN ADDRESS GROUP
8398 FE81 CPI GTL
839A CA4D84 JZ RGLT ;GO TO LOCAL
839D FE08 CPI SDC
839F CA4E84 JZ RSCG ;SEE IF EXECUTE TRIGGER
83A2 FE88 CPI Cd
83A4 CA6684 JZ RGET ;GROUP EXECUTE TRIGGER
83A7 FE09 CPI TCT
83A9 CA6F84 JZ RTCT ;TAKE CONTROL
83AC EF11 CPI LLO
83AE CA7084 JZ RLOL ;LOCAL LOCKOUT
83B1 FE14 CPI DCL
83B3 CA7184 JZ RDCL ;UNIVERSAL DEVICE CLEAR
83B6 FE15 CPI PPU
83B8 CA7484 JZ RPPU ;PARALLEL POLL UNCONFIGURED
83BB FE18 CPI SPE
83BD CA7684 JZ RSPE ;SERIAL POLL ENABLE
83C0 FE19 CPI SPD
83C2 CA8684 JZ RSPD ;SERIAL POLL DISABLE
83C5 C9 RET ;DON'T RECOGNIZE THE COMMAND
83C6 3A0580 RSCG: LDA TSTAT ;SEE IF IN TALKER PRIMARY ADDRESS STATE
83C9 E608 ANI 8
83CB CAF883 JZ RSCG1 ;NO
83CE 78 MOV A,B ;GET SECONDARY ADDRESS AGAIN
83CF 320C80 STA TALKS ;AND SHOW IT AS TALK SECONDARY ADDRESS
83D0 3A0580 LDA TSTAT ;GET TALKER STATE AGAIN
83D3 E604 ANI 4 ;KEEP ONLY SERIAL POLL MODE STATE
83D7 C0 INR A ;SHOW TALKER IS ADDRESSED
83D8 320580 STA TSTAT
83DB CD4A83 CALL LNADR ;UNADDRESS DUE TO MY TALK ADDRESS
83DE C9 RET ;DONE INTERPRETING THE COMMAND
83DF 3A0680 RSCG1: LDA LSTAT ;SEE IF IN LISTENER PRIMARY ADDRESSED STATE
83E2 E604 ANI 4
83E4 CAF883 JZ RSCG2 ;NO
83E7 78 MOV A,B ;SAVE LISTENER SECONDARY ADDRESS
83E8 320B80 STA LSTNS
83EB 3A0680 LDA LSTAT ;GET LISTENER STATE AGAIN
83EE E618 ANI 18H ;KEEP LISTEN HANDSHAKE AND EOS FLAGS
83F0 3C INR A ;SHOW STATE AS ADDRESSED LISTENER
83F3 C6A83 CALL TNAQR ;UNADDRESS DUE TO MY LISTEN ADDRESS
83F7 C9 RET ;DONE INTERPRETING COMMAND
83F8 3A0980 RSCG2: LDA PSTAT ;SEE IF PARALLEL POLL IS TO BE CONFIGURED
83FB E604 ANI 4 ;PARALLEL POLL IN PACS?
83FD CA4E84 JZ RSCG3 ;NO
8400 78 MOV A,B ;GET THE COMMAND AGAIN
8401 E610 ANI 10H ;IS IT PPD (PARALLEL POLL DISABLE)?
8403 C24784 JNZ PPPD ;YES, SO PUT PP INTO PPS
8406 78 MOV A,B ;NO, SO SAVE PPE MESSAGE
8407 322880 STA PPRSP
840A CDC587 CALL PPNBL ;PUT THE APPROPRIATE PPR MESSAGE IN
841D C9 RET ;THE PARALLEL POLL RESPONSE REGISTER
841E C9 RSCG3: RET ;NO OTHER FUNCTIONS DECODED YET++++
P&T-488

Custom Software Package
Version 1.4

840F 3A0580 RTAG: LDA TSTAT ;GET TALK STATUS
8412 E604 ANI 4 ;KEEP SERIAL POLL MODE STATE
8414 4F MOV C,A ;SAVE IT IN REGISTER C
8415 3A0180 LDA TALKP ;GET PRIMARY TALK ADDRESS
8416 88 CMP B
8419 C22384 JNZ NTLK ;;COMMAND DOES NOT MATCH PRIMARY TALK ADDRESS
841C 79 MOV A,C ;GET TALK STATE AGAIN
841D F608 ORI B ;SHOW PRIMARY ADDRESS STATE
841F 320580 STA TSTAT
8422 C9 RET ;DONE INTERPRETING THE COMMAND

8423 79 NTLK: MOV A,C ;GET TALK STATE AGAIN
8424 320580 STA TSTAT ;SHOW IT AS UNADDRESSED (BECAUSE THIS COMMAND WAS EITHER UNIVERSAL UNTALK ; OR OTHER TALK ADDRESS)
8427 C9 RET ;DONE INTERPRETING THE COMMAND

8428 3A0680 RLAG: LDA LSTAT ;GET LISTEN STATE
842B E618 ANI 1BH ;KEEP ONLY HANDSHAKE AND EOS FLAGS
842D 4F MOV C,A ;SAVE IT IN REGISTER C FOR LATER
842E 3A0080 LDA LSTNP ;GET PRIMARY LISTEN ADDRESS
8431 88 CMP B
8432 C23C84 JNZ NLSN ;;DOES NOT MATCH COMMAND
8435 79 MOV A,C ;GET UNADDRESSedListEN AGAIN
8436 F604 ORI 4 ;SHOW IT AS PRIMARY ADDRESS STATE
8438 320680 STA LSTAT
843B C9 RET

8443 3A0680 NLSN: LDA LSTAT ;THIS IS NOT MY LISTEN ADDRESS, SO
8445 E6FB ANI 0FBH ;INSURE THAT P&T-488 IS IN LPIS STATE
8448 320680 STA LSTAT
844A 78 MOV A,B ;GET COMMAND
844E FE3F CPI 3FH ;UNIVERSAL UNLISTEN?
844F 08 RNZ ;OTHER LISTEN ADDRESS, SO LEAVE LSTAT ALONE
8447 C0 MOV A,C ;GET UNADDRESSedListEN STATE
8449 320680 STA LSTAT ;DUE TO UNIVERSAL UNLISTEN COMMAND
844C C9 RET

844D C9 RGTL: RET ;GO TO LOCAL FUNCTION NOT IMPLEMENTED

844E 3A0680 RSDC: LDA LSTAT ;SELECTIVE DEVICE CLEAR
8451 E603 ANI 3 ;IS THE LISTEN MODE ADDRESSED?
8453 C0 RZ ;;NO
8454 C3AF84 JMP UDVCL ;YES, SO CLEAR THE DEVICE

8457 3A0680 R PPC: LDA LSTAT ;PARALLEL POLL CONFIGURE
845A E601 ANI 1 ;SEE IF LISTEN FCN IN LADS
845C C0 RZ ;;NO, SO IGNORE PPC COMMAND
845D 3A0980 LDA PSTAT ;;YES, SO PUT PP INTO PACS
8460 F604 ORI 4
8462 320980 STA PSTAT
8465 C9 RET

8466 3A0680 RGET: LDA LSTAT ;GROUP EXECUTE TRIGGER
8469 E603 ANI 3 ;SEE IF FUNCTION ADDRESS
846B C0 RZ ;;NO
846C C3AC84 JMP UTRGR ;YES, SO PERFORM DEVICE TRIGGER

846F C9 RTCT: RET ;TAKE CONTROL - NOT IMPLEMENTED

8470 C9 RLLO: RET ;LOCAL LOCKOUT - NOT IMPLEMENTED
8471 C3AF84 RDCL: JMP UDVCL ;UNIVERSAL DEVICE CLEAR

D - 17
; PPI DL: SUB A ; PARALLEL POLL UNCONFIGURE
8475 320980 STA PSTAT ; PUT PP INTO PPI/PUCS
8478 3EFF MVI A,OFH ; CLEAR RESPONSE BYTE REGISTER
847A D37F OUT PPORT
847C C9 RET

847D 3A0580 RSPE: LDA TSTAT ; SET SERIAL POLL MODE BIT IN TALKER
8482 320580 STA TSTAT
848C C9 RET

8486 3A0580 RPG: LDA TSTAT ; CLEAR SERIAL POLL MODE BIT IN
848F 40F8 ANI OFFH ; TALKER STATE REGISTER
848B 320580 STA TSTAT
848E C9 RET

****** GIM - GENERAL INTERFACE MANAGEMENT ******
A ROUTINE WHICH ALLOWS THE USER TO SET THE
STATE OF THE IFC, SRQ, REN AND EOI LINES
******

848F C5 GIM: PUSH B ; IMMEDIATELY SET IFC, SRQ, REN AND EOI LINES.
8490 6617 ANI 17H ; STRIP OUT DON'T CARES
8492 2F CMA ; 848 USES NEGATIVE LOGIC
8493 47 MOV B,A
8494 3A0680 LDA GIMTC ; GET STATE OF LOCAL ATN, ETC
849F 4617 ORI 17H ; STRIP OUT IFC, SRQ, ETC
8500 A0 ANA B ; COMBINE INTO NEW COMMAND
8502 DA3882 CALL COMND ; OUTPUT NEW COMMAND
8505 C1 POP B ; RESTORE BC
850C C9 RET

****** CALCULATE AND JUMP TO APPROPRIATE ENTRY IN ******
USER-SUPPLIED JUMP TABLE
******

849F 97 UIFC: SUB A ; ZERO REG A
84A0 320680 STA LSTAT ; PUT LISTEN FCN IN IDLE
84A3 320680 STA TSTAT ; PUT TALK FCN IN IDLE
84A6 320A80 STA CSTAT ; PUT CONTROLLER FCN IN IDLE
84A9 1E09 MVI E,9
84AB 01 DB 1
84AC 1E00 UTRGR: MVI E,Ø ; E=Difference between User Jump Table
84AE 01 DB 1 ; Base Address and Desired Entry Point
84AF 1E03 UDVCL: MVI E,3 ; BC=GARbage
84B1 01 DB 1
84B2 1E06 UBFUL: MVI E,6
84B4 01 DB 1
84B5 1E18 UATN: MVI E,24D
84B7 01 DB 1
84BB 1E0F UNLSN: MVI E,15D
84BC 01 DB 1
84BB 1E12 USRO: MVI E,18D
**P&T-488**

**Custom Software Package**

**Version 1.4**

---

840D 01  DB 1
840E 1E15 UPOC: MVI E,21D
840F 01  DB 1
8410 1E0C UBRAC: MVI E,12D
8411 1E0D  MVI D,0
8412 2A0E80 LHLD JMPAD ;GET BASE ADDRESS OF USER JUMP TABLE
8413 19  DAD D ;CALCULATE ACTUAL ADDRESS
8414 1E0E  PUSH H ;AND PUT IT ON THE STACK
8415 3A2380 LDA LBYTE ;PUT LAST BYTE HEARD IN A REG
8416 2A1008 LHLD BPTR ;AND POINTER OF CURRENT BUFFER IN HL
8417 29  RET ;THEN "RET" TO USER JUMP TABLE

---

SRVIS: CHECK FOR SERVICE REQUEST. IF SRQ IS TRUE,
THE STATE TABLE IS CHECKED TO DETERMINE IF
THE P&T 488 IS THE CONTROLLER-IN-CHARGE. IF
IT IS, THE ADDRESS IN REGISTERS HL IS SUBSTITUTED
FOR THE RETURN ADDRESS AND A BRANCH IS MADE TO
THE USER-SUPPLIED ROUTINE SVCRQ. IF THE CONDITIONS
ARE NOT MET, A RETURN IS MADE TO THE CALLING ROUTINE.

---

SRVIS: IN CMOPT ;CHECK FOR SRQ TRUE (LOW)
84D3 6E04 ANI 4
84D5 CA184 JZ SRVI ;...SRQ TRUE: SHOULD WE IGNORE IT?
84DB 3A0A80 LDA CSTAT
84DB E56F ANI 0EFH ;PUT CONTROLLER INTO CSNS STATE
84DD 320A80 STA CSTAT
84DE C9  RET

SRVI: LDA CSTAT ;IGNORE SRQ LINE IF IT HAS ALREADY BEEN
84E1 3A0A80 ANI 10H ; DETECTED
84E4 5E10 RNZ

STATE - RETURNS WITH ABBREVIATED STATE INFORMATION
IN THE A REGISTER AND HL PointING TO THE
FIRST ENTRY OF THE STATE TABLE. THUS IF
THE USER REQUIRES DETAILED STATE INFORMATION,
HE CAN LOOK INTO THE STATE TABLE.

---

STATE: PUSH B ; PRESERVE BC REGISTERS
84F7 3A0580 LDA TSTAT ;GET TALKER STATE
84FA 6E01 ANI 1 ;SEE IF ADDRESS
84FC 47 TIDL: MOV B,A
84FD 3A0580 LDA LSTAT ;GET LISTENER STATE
8500 6E03 ANI 3 ;SEE IF ADDRESS OR ACTIVE
8502 CA0785 JZ LIDL ;...LISTENER IDLE
8505 3E02 MVI A,2 ;PUT LISTENER NOT-IDLE STATE IN BIT 1
XCTRL: PUSH H; SAVE USER JUMP TABLE ADDRESS

8529 E5  XCTRL:  PUSH  H  ;SAVE USER JUMP TABLE ADDRESS
852A 60  MOVLH, $B
852B 69  MOV  L,C
852C 22 C8  SHLD JMPAD
852F E1  POP  H

8530 30D80  XCTRL:  LDA  GIMTC ;SET UP ACCEPTOR HANDSHAKE
8533 E69F  ANI  $9F ;BY SETTING NRFD, NDAC LOW (TRUE)
8535 F680  ORI  $80 ;AND DAV HIGH (PASSIVE FALSE)
8537 CD8B 2 CALL COMND
853A 3E F6  MVI A, $8F6H ;CLEAR DATA LINES
853C 03 7E  OUT DATPT
853E 3F E6  MVI B, $06E6H ;CLEAR XATN BIT IN ISR, LEAVE INTERRUPTS
8540 D37C  OUT 1SRPT; DISABLED
8542 CDF985  XCTRL1:  CALL PI ;CHECK FOR LOCKUP DUE TO POC OR IFC
8545 DB7D  IN  QMOPT ;NOW CHECK FOR ATN
8547 E680  ANI  $8
8549 2A085  JNZ  XCDUN
854C DB7D  IN  QMOPT ;WAIT UNTIL DAV IS HIGH (PASSIVE FALSE)
854E E680  ANI  $80 ;
8550 CA4285  JZ  XCTRL1
8553 30D80  LDA  GIMTC ;NOW SET NRFD HIGH (WE'RE READY)
8556 F47F  ORI  $40
8558 CD3B 2 CALL COMND
855B CDF985  DAVT:  CALL PI ;WAIT FOR DAV LOW (TRUE)
855E DB7D  IN  QMOPT
8560 E680  ANI  $8 ;XATN TRUE?
8562 C2A85  JNZ  XCDUN ;NO, SO QUIT THIS ROUTINE
8565 DB7D  IN  QMOPT
8567 E680  ANI  $80
8569 25B85  JNZ  DAVT
856C 30D80  LDA  GIMTC ;SET NRFD, NDAC LOW (TRUE)
856F E69F  ANI  $9F

D - 20
P&T-488

CALL COMND ;GET THE COMMAND FROM THE EXTERNAL CONTROLLER
8574 2F CMA ;THE 488 BUS USES NEGATIVE LOGIC
8577 E6F ANI 7FH ;STRIP PARITY BIT
8579 47 MOV B,A ;SAVE THE COMMAND IN REGISTER B
857A 3229B0 STA LBYTE ;AND IN LAST BYTE REGISTER
857D 3A09B0 LDA GIMTC ;TELL THE CONTROLLER WE GOT IT
857E 66F ORI 2FH ;BY SETTING NDAC HIGH (FALSE)
857F CD3B82 CALL COMND
8585 CDF985 DA VF: CALL PI ;NOW WAIT FOR DA VF FALSE (HANDSHAKE COMPLETE)
8588 DB7D IN CMDPT
858A 3A09B0 ANI 80H
858C CA92B0 JZ DA VF
858F 3A09B0 LDA GIMTC ;COMPLETE HANDSHAKE BY SETTING NDAC LOW
8592 E6F ANI 9FH
8594 CD3B82 CALL COMND
8597 CD7383 CALL UPD8 ;FIGURE OUT WHAT THE COMMAND MEANS
859A CDC184 CALL UBRAK ;SEE IF USER WANTS CONTROL OF S-100
859C 32811 JMP XCTL1 ;GET THE NEXT COMMAND
859F CD3383 XCDUN: CALL ADDRES ;_FINISH ADDRESSING TALK/LISTEN FCNS
; OF P&T-488
85A3 E6F6 MVI A,0F6H ;CLEAR THE XATN BIT IN THE ISR
85A5 D37C OUT ISRPT
85A7 3A09B0 LDA TSTAT ;CHECK TO SEE IF IN SERIAL POLL MODE
85AA E6B4 ANI 4
85AC 48 RZ ;...NO, GO BACK TO CALLING ROUTINE
85AD 3A09B0 LDA TSTAT ;ARE WE ADDRESSED AS THE TALKER?
85BE E601 ANI 1
85B2 CA92B0 JZ NTLKR ;...NO, WAIT FOR NEXT COMMAND
85B5 2A08B0 LHLD JMPAD ;PUT USER JUMP TABLE ADDRESS IN BC
85B8 44 MOV B,H
85B9 40 MOV C,L
85BA 21298D LXI H,XSPRS ;POINT TO EXTERNAL CONTROLLER SERIAL
; POLL RESPONSE BYTE BUFFER
85BD 3A09B0 LDA SPSTS ;GET SERIAL POLL STATUS BYTE
85C0 E6F MVI A,0F6H ;CLEAR THE XATN BIT IN THE ISR
85C2 77 MOV M,A ;AND PUT INTO BUFFER
85C5 3A07B0 LDA SSTAT ;ARE WE REQUESTING SERVICE?
85C6 E630 ANI 3FH
85C9 CADF85 JZ SRSP ;...NO
85D3 3A08B0 LDA GIMTC ;CLEAR SRQ LINE
85DC F604 ORI 4
85DF CD3B82 CALL COMND
85D0 3E20 MVI A,20FH ;AND PUT INTO THE AFFIRMATIVE POLL
; RESPONSE (APRS) STATE
85D3 32078D STA SSTAT
85D5 3E40 MVI A,40FH ;SET SERIAL RESPONSE TO SERVICE Request
; ACKNOWLEDGED
85DA 86 ORA M
85DB 77 MOV M,A
85DC 330F85 JMP SRSP

85DF 54 SRSP: MOV D,H ;MESSAGE IS ONLY THE ONE BYTE
85E0 50 MOV E,L
85E1 CD98B0 CALL TALK ;SAY THE RESPONSE MESSAGE
85E4 3A08B0 NTLKR: LDA GIMTC ;RELEASE NRFD, NDAC SO THE ADDRESSED
85E7 F606 ORI 60H ; TALKER CAN RESPOND WITH ITS SERIAL POLL
85E9 CD3B82 CALL COMND ; RESPONSE BYTE
85EC CDF985 NTLKL: CALL PI ;CHECK FOR IFC OR POC
85EF D7D IN CMDPT ;WAIT FOR RE-APPLICATION OF EXTERNAL ATN
85F1 E608 ANI 8 ;LOOK AT ONLY XATN
85F3 CA92B0 JZ XCTL0 ;...XATN TRUE, SO GO TO EXTERNAL
; CONTROLLER ROUTINE
85F6 C3EC85 JMP NTLKL1 ;REPEAT LOOP UNTIL NEXT COMMAND COMES

D - 21
85F9 DB7C  ; PI: IN ISRPT ; CHECK FOR POC OR IFC
85F9 E601  ; ANI I ; LOOK AT ONLY POC
85FD C4F84  ; JZ UPOC
86G0 DB7C  ; IN ISRPT
86G2 E610  ; ANI 10H ; LOOK AT ONLY IFC
86G4 CA9F84  ; JZ UIFC
86G7 C9  ; RET

******************************************************************************
 ; SPQRY SERIAL POLL QUERY
 ; SENDS OUT THE COMMANDS UNL (UNIVERSAL UNLISTEN),
 ; SPE (SERIAL POLL ENABLE), THEN THE TALK ADDRESSES
 ; THAT ARE IN ITS BUFFER (BY CALLING CTRL). UPON
 ; RETURN TO SPQRY, THE COMMAND SPD (SERIAL POLL
 ; DISABLE) IS SENT, THEN RETURNS TO THE CALLING PROGRAM

******************************************************************************

86G8 221E80  ; SPQRY: SHLD SBPTR ; INITIALIZE SERIAL POLL BUFFER POINTER
86G8 8B  ; XCHG
86G0 222080  ; SHLD SBEND ; STORE END ADDRESS OF SERIAL POLL BUFFER
86G0 60  ; MOV H,B
8610 60  ; MOV L,C
8611 220E80  ; SHLD JMPAD ; STORE USER JUMP TABLE ADDRESS
8614 215B80  ; LXI H,BSPE ; POINT TO UNL, SPE MESSAGE
8617 54  ; MOV D,H
8618 50  ; MOV E,L
8619 13  ; INX D ; MESSAGE IS ONLY THE TWO BYTES
861A C9 29B2  ; CALL CTRL ; SEND THE TWO COMMANDS BUT DO NOT
     ; RELEASE THE ATN LINE
861D 2A0E80  ; SPQ1: LHLD JMPAD ; GET ADDRESS OF USER'S JUMP TABLE
8622 44  ; MOV B,H
8622 4D  ; MOV C,L ; AND PUT INTO BC
8622 215680  ; LXI H,BUNT ; POINT TO "UNT" MESSAGE
8625 54  ; MOV D,H
8626 50  ; MOV E,L
8627 C9 29B2  ; CALL CTRL ; AND SEND IT BEFORE THE TALK ADDR
862A 2A0E80  ; LHLD JMPAD ; GET THE ADDR OF THE USER'S JUMP TABLE
862D 44  ; MOV B,H
862E 4D  ; MOV C,L
862F 2A2880  ; LHLD SBEND ; POINT TO SERIAL POLL BUFFER
8632 8B  ; XCHG
8633 2A1E80  ; LHLD SBPTR ; (TALK ADDRESSES) AND SEND THEM
     ; ONE BY ONE

 ; SEE IF THERE IS ANOTHER ADDRESS IN THE SERIAL POLL BUFFER

8636 7C  ; MOV A,H
8637 8A  ; CMP D
8638 C24986  ; JNZ NSPEND ; NOT END OF BUFFER
8639 7D  ; MOV A,L
863C 8B  ; CMP E
863D CA4686  ; JZ NSPSEC ; END OF BUFFER, THUS THERE IS NO
     ; SECONDARY ADDRESS

 ; THERE IS ANOTHER ADDRESS: NOW SEE IF IT IS A SECONDARY ADDR

8640 23  ; NSPEND: INX H
8641 7E  ; MOV A,M

D - 22
P&T-488

8642 FE58 CPI 60H ;NOT A SECONDARY ADDRESS
8644 DA486 JC NSPSEC ;IT IS A SECONDARY ADDR, SO SEND IT ALSO
8647 EB XCHG SENDSP ;SEND ONLY THE ONE BYTE
8648 C34F68 JMP SENDSP

; 864B 2A1E80 NSPSEC: LHLD SBPTR ;POINT TO FIRST BYTE TO BE SENT
864C EB XCHG SENDSP ;ACTUALLY SEND THE ADDRESS(ES)

; NOW LISTEN TO THE RESPONSE SENT BY THE ADDRESSED TALKER

8655 3A0B80 LDA TSTAT ;ARE WE ADDRESSED TO TALK?
8656 EB01 ANI 1
8657 C2986 JNZ WETLK ;YES
8658 3A0B80 LDA GIMTC ;NO, SO BECOME A LISTENER
8659 EB0F ORI 008H ;SET DAV, NRFD, IFC, ATN FALSE
865A EB58 CALL COMND
865B CD3B82 CALL CNTRL SPQ2: CALL PAI ;WAIT UNTIL DAV IS TRUE (LOW)
865D EB70 IN CMDPT
865E EB90 ANI 80H
865F C2B8E6 JNZ SPQ2

8662 3A0B80 CALL PAI ;SENDSP: PUSH D ;SAVE ADDR OF LAST BYTE SENT
8663 EB58 CALL COMND
8664 CD3B82 CALL CNTRL

8665 DB7E IN CMDPT ;GET THE SERIAL POLL STATUS BYTE
8666 EB02 JNZ WETLK ;GET DATEPT ;488 USES NEGATIVE LOGIC
8667 EB06 STA SPRSP ;AND SAVE IT FOR LATER USE
8668 3A0B80 LDA GIMTC
8669 F520 ORI 20H ;NOW MAKE NDAC FALSE (HIGH)
866A EB58 CALL COMND
866B CD4182 SPQ3: CALL PAI ;WAIT FOR DAV FALSE (HIGH)
866C EB70 IN CMDPT
866D EB90 ANI 80H
866E EB98 JNZ WETLK ;488 USES NEGATIVE LOGIC
866F EB58 CALL COMND
8670 CD3B82 CALL CNTRL

8671 3A0B80 LDA GIMTC ;RESPOND TO TALKER WITH NRFD
8672 EB58 CALL COMND
8673 DB7E IN DATPT ;GET THE SERIAL POLL STATUS BYTE
8674 EB02 JNZ WETLK ;488 USES NEGATIVE LOGIC
8675 EB06 STA SPRSP
8676 EB58 CALL COMND
8677 CD4182 SPQ3: CALL PAI ;WAIT FOR DAV FALSE (HIGH)
8678 EB70 IN CMDPT
8679 EB90 ANI 80H
867A EB98 JNZ WETLK ;488 USES NEGATIVE LOGIC
867B EB58 CALL COMND
867C CD3B82 CALL CNTRL

867D 3A0B80 LDA GIMTC ;CLEAR THE SQ BIT
867E EB58 CALL COMND
867F 3E20 MVI A,20H ;AND PUT INTO THE AFFIRMATIVE POLL
8680 3A0B80 LDA GIMTC ;RESPONSE (APRS) STATE
8681 EB58 CALL COMND
8682 3B8E86 JMP WTLK1

8683 3A0B80 NSPQR: LDA SPSTS ;GET SERIAL POLL STATUS BYTE
8684 EB02 JNZ WETLK ;488 USES NEGATIVE LOGIC
8685 F640 ORI 40H ;MAKE BIT 6 NON-ZERO (WE DON'T NEED SERVICE)
8686 EB58 CALL COMND
8687 3228E6 STA SPRSP ;AND SAVE SERIAL POLL RESPONSE
8688 DB7E IN DATPT ;GET SERIAL POLL RESPONSE FROM 488 BUS
8689 EB02 JNZ WETLK ;488 USES NEGATIVE LOGIC
868A EB06 STA SPRSP
868B EB58 CALL COMND
868C 3A0B80 LDA GIMTC
868D 2F MVI B,48H ;SHOW LISTENER READY, TALKER NOT
868E CB3B86 CALL COMND
86C9 CD4182 WTLK2: CALL PAl ;CHECK FOR POC/IFC
86CC DB7D IN CMDPT ;WAIT UNTIL NRFD FALSE
86CE E640 ANI 40H
86D0 CA986 JZ WTLK2 ;SOMEBODY ELSE IS SLOWING US DOWN
86D3 3A080 LDA 01MTC
86D6 E67F ANI 7FH ;MAKE DAV TRUE (TALKER SAYING IT)
86D8 CD382 CALL COMND
86D9 E66F ANI 0BFH ;THEN NRFD TRUE (LISTENER GETTING IT)
86D0 CD382 CALL COMND
86E0 F620 ORI 20H ;NDAC FALSE (LISTENER GOT IT)
86E2 CD382 CALL COMND
86E5 CD4182 WTLK3: CALL PAl ;CHECK FOR POC/IFC
86E8 DB7D IN CMDPT ;WAIT UNTIL NDAC FALSE
86E9 E620 ANI 20H
86EA CA386 JZ WTLK3
86E8 E640 LDA 01MTC
86E9 F660 ORI 20H ;DAV FALSE (TALKER REMOVING DATA)
86F0 CD382 CALL COMND
86F7 3EFF MVP A,0FFH ;RELEASE THE 488 DATA LINES
86F9 D37E OUT DATPT
86FB D1 SPQ7: POP D ;GET BFR ADDR OF LAST BYTE SENT
86FC 3A220 LDA SPRSP ;HAVE WE FOUND THE NEEDY DEVICE YET?
86FF E640 ANI 40H
8701 C21987 JNZ SPQ9 ;•• YES, SO TERMINATE POLL
8704 2A2080 LHLD SBEND ;ANYTHING LEFT IN THE BUFFER?
8707 7A MOV A,D
8708 BC CMP H
8709 C21187 JNZ SPQ8 ;•• YES, SO CONTINUE THE POLL
870C 7E MOV A,E
870D 8D CMP L
870E CA1987 JZ SPQ9 ;•• NO, SO TERMINATE POLL
8711 13 SPQ8: INX D ;POINT TO NEXT ENTRY IN BUFFER
8712 6B XCHG
8713 221E80 SHLD SBPTR ;AND UPDATE THE BUFFER POINTER
8716 C31D86 JMP SPQ1 ;THEN POLL NEXT DEVICE
8719 2A680 SPQ9: UHLD JMPAD ;GET ADDRESS OF USER'S JUMP TABLE
871C 44 MOV B,H
871D 40 MOV C,L ;AND PUT INTO BC
871E 215580 LXI H,BSPR ;POINT TO 'SPR MESSAGE
8721 54 MOV D,H
8722 5D MOV E,L ;COMMAND IS ONLY ONE BYTE
8723 C7C82 CALL CNTRL
8726 2A1E80 LHLD SBPTR ;POINT TO WINNING ENTRY IN SERIAL POLL BFR
8729 3A220 LDA SPRSP ;GET RESPONSE TO SERIAL POLL
872C 47 MOV B,A ;AND SAVE IN REG B FOR THE USER
872D 2F CMW
872E E640 ANI 40H ;SET A REGISTER
8730 C9 RET ;GO BACK TO CALLING PROGRAM

;******************************************************
; SPSRQ SERIAL POLL REQUEST
; SET THE SRQ (SERVICE REQUEST) LINE TRUE (LOW)
; THEN DETERMINE IF PAT 488 IS THE CONTROLLER-IN-CHARGE.
; IF SO, JMP TO SVCRQ.
; IF NOT, WAIT UNTIL EXTERNAL CONTROLLER RESPONES WITH
; A SERIAL POLL. ANSWER THE POLL, THEN RETURN TO THE
; CALLING PROGRAM.
;******************************************************
8731 60 SPSRQ: MOV H,B
8732 69 MOV L,C ;SAVE USER JUMP TABLE ADDRESS
8733 220E30 SHLD JMPAD
8736 3E10 MVI A,10H ;UPDATE SERVICE REQUEST STATE BYTE TO
8738 320B80 STA SSTAT ; THE SERVICE REQUEST (SRQS) STATE
8739 3A0A80 LDA CSTAT ;PUT THE CONTROLLER IN THE CSRS STATE
873E 62F ANI 2FH ; IF THE P&T-488 IS THE CONTROLLER
8744 CA8887 JZ NCTRL
8743 F610 ORI 10H
8744 320A80 STA CSTAT
8748 DB7C NCTRL: IN ISRPT ;SEE IF LOCKED UP DUE TO CHANGE IN XATN
874A 6588 ANI 8
874C C25C87 JNZ NLOK ;...NOT LOCKED
874F 3EFF MVI A,OFH ;PRESERVE HANDSHAKE LOCK, BUT RELEASE
8751 CD3B82 CALL COMMD ; XATN BIT IN ISR
8754 3EBF MVI A,OFBH ;MAKE ONLY NRFD TRUE
8756 D37D OUT COMP
8759 3EF7 MVI A,OF7H ;RELEASE XATN BIT IN ISR
875A D37C OUT ISRPT
875C 3A0B80 NLOK: LDA GIMTC
875F 66F8 ANI OFBH ;MAKE SRQ TRUE (LOW)
8761 CD3B82 CALL COMMD
8764 3A0A80 LDA CSTAT ;CONTROLLER IN IDLE STATE AND NOT SYSTEM
876A COMMD ; SYSTEM CONTROL ACTIVE?
876C C3E485 JMP NTLKR ;...YES, SO WAIT FOR CONTROLLER TO DO
876D DB7C ; A SERIAL POLL
8771 60 SPIDL: PUT SRQ FCN IN IDLE STATE
8773 61 ; SPIDL
8775 62 ; PUT SRQ FCN IN IDLE MODE
8777 63 ; PUT SRQ FCN IN IDLE MODE
8779 64 ; PUT SRQ FCN IN IDLE MODE
877B 65 ; PUT SRQ FCN IN IDLE MODE
877D 66 ; PUT SRQ FCN IN IDLE MODE
877F 67 ; PUT SRQ FCN IN IDLE MODE
8781 68 ; PUT SRQ FCN IN IDLE MODE
8783 69 ; PUT SRQ FCN IN IDLE MODE
8785 6A ; PUT SRQ FCN IN IDLE MODE
8787 6B ; PUT SRQ FCN IN IDLE MODE
8789 6C ; PUT SRQ FCN IN IDLE MODE
878B 6D ; PUT SRQ FCN IN IDLE MODE
878D 6E ; PUT SRQ FCN IN IDLE MODE
878F 6F ; PUT SRQ FCN IN IDLE MODE
8791 70 ; PUT SRQ FCN IN IDLE MODE
8793 71 ; PUT SRQ FCN IN IDLE MODE
8795 72 ; PUT SRQ FCN IN IDLE MODE
8797 73 ; PUT SRQ FCN IN IDLE MODE
8799 74 ; PUT SRQ FCN IN IDLE MODE
879B 75 ; PUT SRQ FCN IN IDLE MODE
879D 76 ; PUT SRQ FCN IN IDLE MODE
879F 77 ; PUT SRQ FCN IN IDLE MODE
87A1 60 PPI: CALL PAI ;CHECK FOR POC, XIFC, XATN
87A3 61 ; RESET XATN IF ATN NO LONGER TRUE
87A5 62 LDA GIMTC ;GET IMAGE OF WHAT'S ON COMMAND LINES
87A7 63 ANI OF7H ;MAKE ATN TRUE
87A9 64 CD3B82 CALL COMMD ; DO IT
87AB 65 66F6 ANI OF6H ;NOW MAKE ED1 TRUE ALSO
87AD 67 CD3B82 CALL COMMD
87AF 68 67F7 IN DATPT ;GET THE RESPONSE TO THE PARALLEL POLL
87B1 69 2F CMA ;488 USES NEGATIVE LOGIC
87B3 6A 322380 STA LBYTE ;SAVE THE RESPONSE
87B5 6B 3A0B80 LDA GIMTC
87B7 6C 6581 ORI 1 ;MAKE ED1 FALSE
87B9 6D CD3B82 CALL COMMD
87BB 6E 6588 ORI 8 ;MAKE ATN FALSE
87BD 6F CD3B82 CALL COMMD
87BF 66F8 LDA CSTAT ;PUT CONTROLLER IN STANDBY
87C1 67 3A0A80 LDA CSTAT ;PUT CONTROLLER IN STANDBY
87C3 68 E6F0 ANI OFBH ;KEEP SYSTEM CONTROL/SRQ STATES

D - 25
PISTT SET "1ST" MESSAGE TRUE AND PUT THE PROPER PARALLEL POLL RESPONSE MESSAGE IN THE PARALLEL POLL RESPONSE REGISTER

PISTF SET "1ST" MESSAGE FALSE AND PUT THE PROPER PARALLEL POLL RESPONSE MESSAGE IN THE PARALLEL POLL RESPONSE REGISTER

PPNBL THIS ROUTINE CALCULATES THE PARALLEL POLL RESPONSE WHICH CORRESPONDS TO THE FOUR LOW-ORDER BITS OF PPRSP. IT THEN DETERMINES WHETHER THE PARALLEL POLL RESPONSE IS TO BE TRUE OR FALSE ON THE BASIS OF THE 1ST BIT OF PSTAT. THE PROPER RESPONSE BYTE IS PLACED IN THE PARALLEL POLL RESPONSE REGISTER OF THE P&T-488, AND THE PP FUNCTION IS PUT INTO THE STANDBY (PPSS) STATE.

PPNBL: LDA PSTAT ;PUT PP INTO PARALLEL POLL STANDBY (PPSS)
87C8 F601 ORI 1
87CA 320980 STA PSTAT
87CD E602 ANI 2 ;LOOK AT THE 1ST BIT
87CF 17 RAL ;AND PUT RESULT IN FOURTH BIT POSITION
87D0 17 RAL
87D1 4F MOV C,A ;AND SAVE RESULT IN REG C
87D2 3A0280 LDA PPRSP ;GET THE PPE BYTE
87D5 E608 ANI 8 ;KEEP ONLY THE SENSE BIT
87D7 A9 XRA C ;COMPARE SENSE BIT AND 1ST BIT
87D8 C2EB87 JNZ PPCLR ;•• IST<>S, THUS PPR MESSAGE IS FALSE
87DB 3A0280 LDA PPRSP ;GET THE PPE BYTE
87DE E607 ANI 7 ;KEEP THE LOW THREE BITS
87E0 4F MOV C,A
87E1 3E01 MVI A,1 ;CALCULATE THE PPR MESSAGE
Custom Software Package
Version 1.4

87E3 9D  PPRCAL: DOR  C  ;DECREMENT SHIFT COUNT
87E4 FAEC87  JM  PPRDUN  ;DONE SHIFING
87E7 87  ADD  A  ;LEFT SHIFT ONE MORE
87E8 C3E87  JMP  PPRCAL  ;DO UNTIL DONE

87EB 97  PPCLR:  SUB  A  ;ZERO A REGISTER
87EC 2F  PPRDUN: CMA  ;PUT INTO PPR PORT (REMEMBER 488 USES
                      ;NEGATIVE LOGIC, SO THE VALUE WE PUT
                      ;IN THE PPR PORT IS THE COMPLEMENT OF
                      ;WHAT THE CONTROLLER WILL SEE WHEN IT
                      ;DOES A PARALLEL POLL.)

87ED D37F  OUT  PPORT  ;PUT BYTE IN PARALLEL RESPONSE PORT
87EF C9  RET

87F8  END

>>>>>  SYMBOL TABLE  <<<<

8333  ADDRESS  821C  BFCHK  8818  BPTR  8055  BSRD  8053  BSPE
8056  BUNT  815D  BYTE  801C  CBEND  801A  CPTR  8326  CCNTU
8523  CIDL  82BA  CLUP  0070  CMPT  827C  CNTRL  823B  COMND
800A  CSTAT  8292  CTRL  82C2  CTRL1  82F2  CTRL2  830E  CTRL6
807E  DAPT  8585  DAYF  8194  DAVH  81C1  DAVL  835B  DAVT
8014  DCL  8026  ENTBL  8004  EOB  8162  EOST  9008  GET
848F  GIIM  8000  GSINT  0001  GTR  8057  INIT  807C  ISRIPT
808E  JMPAD  8018  LBEND  8D16  LBPTR  8923  LCDYTE  81AF  LCNTU
8212  LDUN  8597  LIDL  8153  LIT  8011  LLD  834A  LNADR
8258  LPA1  8169  LSET  819C  LSNN  8906  LSTAT  8000  LSTNP
82B0  LSTNS  8326  NCEND  8748  NCTR  820C  NEOS  875C  NLIN
8372  NLPI  843C  NSNI  8233  NFOLO  864F  NSPEN  86B3  NSPQR
864B  NSEPDC  8146  NTEND  8423  NTLK  852D  NTLK1  852E  NTLKR
8192  NTLST  8353  NTPR  890C  NXTAD  8241  PA1  824A  PA11
8789  PISTF  87AD  PIST  85F9  P1  0005  PPC  878B  PPCR
8474  PPI1DL  87C5  PPNBL  007F  PPORT  877C  PPQR  87E3  PPRCAL
87EC  PPADD  8002  PPRS  0015  PPU  0809  PSTAT  826F  PUATN
8471  RDCL  8466  RGET  8440  RGTI  8428  RLAG  847D  RLLD
8457  RRPC  8474  RPPU  83C6  RSCG  830F  RSCG1  83F8  RSCG2
840E  RSCG3  844E  RSDC  8468  RSDP  847D  RSPE  8006  RSTAT
840F  RTAG  846F  RTCT  8028  SBEND  881E  SBPTR  0084  SDC
864F  SENSOP  8019  SPO  0718  SPE  876F  SPO1  8610  SPO1
8668  SP02  8688  SPO3  86F8  SPO7  8711  SPOB  8719  SPO9
8668  SP0R  8922  SPRSP  8731  SPRSR  8003  SPRTS  850F  SRSP
84E1  SRV1  84D1  SRVIS  0007  SSTAT  8007  STDAR  8466  STATE
8098  TALK  0001  TALK1  010E  TALK2  8001  TALKP  000C  TALKS
8814  TBEND  8012  TBPTR  8128  TCNTU  0009  TCT  8024  TE01
84FC  TIDL  836A  TNADR  0005  TSTAT  007B  TWIN  8485  UATN
8482  UDFUL  84C1  UBRK  84AF  UDVC  849F  UIFC  0035  UNL
8488  ULLSN  003F  UNT  8373  UPD0  848E  UPDC  84BB  USRQ
84AC  UTRGR  8695  WETLK  8699  WETLK1  86C9  WETLK2  86E5  WETLK3
85A8  XCDUN  8530  XCTLO  8542  XCT1  8529  XCTRL  8025  XSRPS
# Code Assignments for "Command Mode" of Operation.

(SENT AND RECEIVED WITH ATN TRUE)

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>MSG</td>
<td>0</td>
<td>MSG</td>
<td>0</td>
<td>MSG</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NUL</td>
<td>DLE</td>
<td>SP</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>SOH</td>
<td>GTl</td>
<td>DC1</td>
<td>LLO</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>STX</td>
<td>DC2</td>
<td>&quot;</td>
<td>2</td>
<td>B</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>ETX</td>
<td>DC3</td>
<td>3</td>
<td>C</td>
<td>S</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>EOW</td>
<td>DC4</td>
<td>DCL</td>
<td>4</td>
<td>D</td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>ENQ</td>
<td>PFC</td>
<td>NAK</td>
<td>PPU</td>
<td>%</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>ACK</td>
<td>SYN</td>
<td>6</td>
<td>6</td>
<td>V</td>
<td>f</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>BEL</td>
<td>ETB</td>
<td>7</td>
<td>7</td>
<td>G</td>
<td>g</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>BS</td>
<td>CAN</td>
<td>8</td>
<td>8</td>
<td>H</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>HT</td>
<td>SPD</td>
<td>9</td>
<td>9</td>
<td>I</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>LF</td>
<td>SUB</td>
<td>10</td>
<td>10</td>
<td>J</td>
<td>z</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>VT</td>
<td>ESC</td>
<td>11</td>
<td>11</td>
<td>K</td>
<td>k</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>FF</td>
<td>FS</td>
<td>12</td>
<td>12</td>
<td>L</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>CR</td>
<td>GS</td>
<td>13</td>
<td>13</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>SO</td>
<td>RS</td>
<td>14</td>
<td>14</td>
<td>N</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>SI</td>
<td>US</td>
<td>15</td>
<td>15</td>
<td>O</td>
<td>1</td>
</tr>
</tbody>
</table>

## Notes:
1. MSG = INTERFACE MESSAGE
2. \( b_1 = \text{DIO1} \ldots b_7 = \text{DIO7} \)
3. REQUIRES SECONDARY COMMAND
4. DENSE SUBSET (COLUMN 2 THROUGH 5). ALL CHARACTERS USED IN BOTH COMMAND & DATA MODES.

---

Copied from Hewlett-Packard Co.
The program BUSMON monitors and reports all transactions on the IEEE-488 bus. 488TODSK records data sent over the 488 bus into a disk file. DSKTO488 sends the contents of a disk file over the bus as data. HANDSHAK.ASM contains the source code for routines which perform the Source and Acceptor Handshake functions. An example of how to use HANDSHAK.ASM is given in the program SAMPLHS.ASM.

**BUSMON**

The program BUSMON monitors and reports all transactions which occur on the IEEE-488 bus. The operator can choose two different forms for the report. The **normal** form displays the transactions without any special handling. The other form is **expanded**, which means that non-printing characters are replaced with strings of printable characters. This form is especially useful for those cases where one is trying to distinguish between tabs and spaces, or determine whether line feed precedes carriage return, etc. The form of the report can be selected by typing a character on the console keyboard while the program is running. Once the form has been selected, its action may be repeated by typing any key on the keyboard.

The operator can set BUSMON to stop on one of three different conditions: on each carriage return, line feed, or each character. The condition is selected by using one of the four **stop code** keys. The stop code can be changed at any time by typing the appropriate stop code key. The stop code keys and the corresponding stop conditions are shown in the following table. Note that typing a stop code key will **NOT** cause a repeat of the previous stop condition, but will invoke a new stop condition. The program starts in the Carriage Return mode.

**Expand/Normal Option**

N or n Show characters normally
X or x Expand the non-printing characters. Space (20 Hex), Horizontal Tab (9) and Line Feed (0A Hex) are replaced by the strings <SPACE>, <HT> and <LF> respectively. The non-printing character Carriage Return (0D Hex) causes the message <CR> to be printed followed by a carriage return and a line feed. All other non-printing characters are replaced with the two character string of an up arrow followed by a capital letter. Thus the non-printing character 01 Hex is replaced by the string ↑A, while the character 1A Hex is printed as ↑Z.

**Stop Codes**

<table>
<thead>
<tr>
<th>Carriage Return</th>
<th>Display all transactions up to and including the next carriage return.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line Feed</td>
<td>Display all transactions up to and including the next line feed.</td>
</tr>
<tr>
<td>Space</td>
<td>Display the next transaction (allows stepping one byte at a time).</td>
</tr>
<tr>
<td>G or g</td>
<td>Go. Display all transactions continuously without stopping on Line Feed, Carriage Return or next byte.</td>
</tr>
</tbody>
</table>

† CP/M is a trademark of Digital Research
Abort
Control C Abort. Go back to the CP/M command mode.

Console/Printer Switch
Ø Direct all output to the console.
1-9 Direct all output to the system printer.

NOTE: to direct output to both the console and printer, select the console and then press Control P.

IEEE-488 Functions
I or i Assert IFC (perform an Interface Clear).
R or r Make REN true (assert Remote Enable).
L or l Make REN false (all instruments will go to Local mode).
Q or q Make SRQ true (request service).
W or w Make SRQ false (cease requesting service).
P or p Perform a Parallel Poll and report the results.
S or s Show the state of the IEEE-488 lines.
T or t Talk - collect a string of characters from the operator then send it over the bus as a Talker.
C or c Control - collect a string and send it over the bus as a Controller.

NOTE: While collecting a string for Talk or Control the following keys have special meaning:
Control X Delete the string and restart collection. This allows errors to be corrected.
RETURN Terminate the collection of the string. The carriage return is not included in the string.
ESCAPE Put the next character into the string. This allows ESCAPE, RETURN and Control X to be put into the string. For instance, to get the string ?A<ESCAPE>12<RETURN><LINE FEED>, you would type ?A<ESCAPE><ESCAPE><ESCAPE><RETURN><LINE FEED><RETURN>. In this example, the string <ESCAPE> means that the ESCAPE key is pressed, not that the 8 keys <, E, S, C, A, P, E and > are pressed. Similarly, <RETURN> and <LINE FEED> mean that the RETURN and LINE FEED keys are used.

Each time the Controller becomes active (asserts ATN active true), a carriage return-line feed is sent to the console, followed by the string COMMAND:, followed by another carriage return-line feed pair. Similarly, each time the Controller becomes inactive (ATN is false), a carriage return, line feed, the string DATA:, carriage return and a line feed is sent to the console. Thus all characters printed after COMMAND: and before DATA: are instructions sent by the Controller, (for example, "?n means
All characters printed after DATA: and before COMMAND: are data (otherwise known as device-dependant messages). Examples are readings from a DVM which has been commanded to be a Talker, etc.

Messages are also printed on the console to indicate occurrences of IFC (Interface Clear), indicate a change of the state of the REN (Remote Enable) line, and of the SRQ (Service Request) line. The message >>> S-100 POC/RESET TRUE <<< is printed whenever the Power On Clear or the RESET line of the S-100 system becomes true.

Whenever the Controller is active, a descriptive string is substituted for special non-printing messages. For example, >> GO TO LOCAL << is printed when 01 Hex is received and ATN is true. The list of messages and the corresponding non-printing characters is as follows:

<table>
<thead>
<tr>
<th>Character</th>
<th>Message</th>
<th>Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>&gt;&gt; GO TO LOCAL &lt;&lt;</td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>&gt;&gt; SELECTIVE DEVICE CLEAR &lt;&lt;</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>&gt;&gt; PARALLEL POLL CONFIGURE &lt;&lt;</td>
<td></td>
</tr>
<tr>
<td>08</td>
<td>&gt;&gt; GROUP EXECUTE TRIGGER &lt;&lt;</td>
<td></td>
</tr>
<tr>
<td>09</td>
<td>&gt;&gt; TAKE CONTROL &lt;&lt;</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>&gt;&gt; LOCAL LOCKOUT &lt;&lt;</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>&gt;&gt; UNIVERSAL DEVICE CLEAR &lt;&lt;</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>&gt;&gt; PARALLEL POLL UNCONFIGURE &lt;&lt;</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>&gt;&gt; SERIAL POLL ENABLE &lt;&lt;</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>&gt;&gt; SERIAL POLL DISABLE &lt;&lt;</td>
<td></td>
</tr>
</tbody>
</table>

The results of this program can be misleading for the following reasons:

1. This program functions as a Listener on the 488 bus. If there were no Listeners on the bus before this routine was run, any Talker would have been unable to say a thing. However, when this routine is run, the Talker has someone to talk to. Thus the operation of the 488 system may be changed by the fact that the Bus Monitor routine is run.

2. This routine is slow compared to the speed that communication on the 488 bus is capable of attaining. Thus 488 throughput may be drastically slowed by using the bus monitor.

3. This routine is incapable of sensing a Parallel Poll issued by another controller, or the response to that Parallel Poll. If it happens that this routine tests the EOI line at the time of a Parallel Poll, it will show the message <END>, even though ATN is true.

488TODSK

The program 488TODSK is used to record all data transactions directly into a CP/M disk file. To use the program type

488TODSK filename.ext x<CR>

where filename.ext is the file name and extension of the file into which the data is to be recorded, and x is the option code. Note that there must be one and only one space
between 488TODSK and the file name, and also one and only one space between the file name and the option code. The characters <CR> mean that the Carriage Return key is pressed, not that the four keys <, C, R and > are pressed.

Three different options are available: none, Z and E. The option E means that the file will be closed and control passed back to the console upon receipt of the 488 END message. The option Z means that the file will be closed and control passed back to the console upon receipt of a Control Z in the data stream (the Control Z is also placed in the file). This option can be useful because CP/M text files are terminated by a Control Z. If no option is selected (that is, a Carriage Return follows the file name), the file can be closed only by pressing Control C on the console. Note that Control C can be used at any time to abort the program; all data received up to the time the Control C was pressed is saved in the file. Some garbage will also appear at the end of the file because the whole buffer is saved in the disk file, and the buffer probably was not filled at the time Control C is pressed.

Error messages are printed on the console if the disk directory is full, the data area is full, or any other disk write error occurs. In each case the function is aborted. If the name of the file is the same as one which is already on the disk, the operator is asked if it is OK to replace the old file. If the operator responds by typing any character other than "Y" or "y", the function is aborted and the old file is left untouched. If the operator responds with either "Y" or "y", the old file is erased and the new one takes its place.

DSKT0488

The program DSKT0488 sends the contents of a CP/M disk file over the 488 bus. The program is called by the string

DSKT0488 filename.ext x

where filename.ext is the name of the file that is to be sent and x is the option code. Only two options are available: none and Z. The Z option causes the Control Z to be sent with the 488 END message when a Control Z is found in the file, then the program returns control to the console. This can be useful for text files that are terminated by a Control Z. If no option code is selected, the entire file is sent followed by a null with the 488 END message, then control is returned to the console. The program may be aborted at any time by typing Control C on the console.

Error messages are printed on the console if there is no Listener on the bus, if the file is not on the disk, or if an invalid option code is selected. In each case the program is aborted and control is returned to the console.

If you have two systems and want to send a file from one to the other via the 488 bus, you would type

488TODSK filename.ext E<CR>
on the system which is to receive the file, and

DSKT0488 filename.ext<CR>
on the one which is sending the file. (It is not necessary to use the same file name or extension.) Note that the system receiving the file must be started first, otherwise the first byte of the file will be lost or the sending system will complain that there are no listeners.
HANDSHAK

The source file HANDSHAK.ASM is actually two subroutines: a routine for Source handshake and a routine for Acceptor handshake. These routines can be useful in special applications where it is desired to use the S-100 system as a Talk Only or Listen Only device, or where increased data rate on the 488 bus is needed. These routines are capable of running much faster than the larger Custom System, CPM488 or 488BAS routines because the larger routines check for the existence of another Controller on the bus, check for excessive time in the handshake cycle, and many other things.

Refer to the chapter titled Hardware Description in the P&T-488 manual for information about the bit mapping of the ports and the 488 bus lines.

SAMPLHS

This file contains the source code for a routine which uses the Source, Acceptor and Initialization subroutines in HANDSHAK to take data from the IEEE-488 bus and display it on the console.
; Source and Acceptor Handshake listings

ISRPT EQU 7CH
CMDPT EQU ISRPT+1
DATPT EQU ISRPT+2
PPORT EQU ISRPT+3

MONITR SET 0 ;CP/M warmstart entry
CPMIO SET 5 ;CP/M I/O entry point

CR SET 0DH ;ASCII carriage return
LF SET 0AH ;ASCII line feed
ES SET ' $ ;CP/M buffered print string terminator

BUFPRN SET 9 ;CP/M fcn. number for buffered print

TALK

TLKT: LDA GIMTC ;get the image of the byte last sent
ORI 8 ;make sure that ATN is false (high)
STA GIMTC when do source handshake

SOURCE HANDSHAKE

This routine takes the byte in memory location CHAR and says it on the 488 bus as a Talker. If either the S-100 RESET or Power On Clear line is or has been true, or if the 488 ATN or IFC lines are or have been true, then an error message is printed and the routine jumps to the system monitor.

SRCHS: LDA GIMTC ;get 488 command line image
ORI 60H ;set NRFD, NDAC high (false)
CALL COMND

SRC1: CALL INTRPT ;check for POC, ATN or IFC
JNZ BYE ;;...abort if POC, ATN or IFC true
IN CMDPT ;see if there are any listeners
CMA
ANI 60H ;check only NRFD, NDAC
JZ NOLSN ;;no listeners error
ANI 40H ;wait until NRFD is high (false)
JNZ SRC1
LDA CHAR ;get the data byte
CMA ;488 uses negative logic
This routine gets one byte from the 488 bus and returns with it in register A. If either the S-100 RESET or Power On Clear line is or has been true, or if the 488 ATN or IFC lines are or have been true, then an error message is printed and the routine jumps to the system monitor.

PENDANT: LDA GIMTC
       ORI 8 ; make ATN false
       ANI 9FH ; and NRFD true, NDAC true
       CALL COMND
       LDA GIMTC
       ORI 40H ; now make NRFD false
       CALL COMND

PENDANT: CALL INTRPT ; see if received POC, ATN or IFC
       JNZ BYE ; ... abort
       IN CMDPT ; look at DAV
       ANI 80H
       JNZ PENDANT ; ... DAV still false
       IN DATPT ; get the data
       CMA ; 488 uses negative logic
       MOV D, A ; keep the data in register D
       LDA GIMTC
       ORI 20H ; NDAC false
       ANI 0BFH ; NRFD true
       CALL COMND

PENDANT: CALL INTRPT ; ... abort
       JNZ BYE ; wait for DAV false
       IN CMDPT ; ...
       ANI 80H
       JZ PENDANT ; ... DAV still true
       LDA GIMTC
       ANI 9FH ; NRFD true, NDAC true

CP/M AUX-7
CALL  COMND
MOV  A,D   ;put the data back in register A
RET

; Initialize 488 board
; This routine should be called after every S-100 RESET or Power On Clear
;
INIT:  MVI  A,0FFH
OUT  PPORT  ;clear parallel poll response port
OUT  DATPT  ; and 488 data port
CALL  COMND  ; and 488 control lines and image byte
SUB  A
OUT  ISRPT  ;clear Interrupt Service Register
STA  RETCOD  ; clear return code
STA  CHAR  ; and CHAR
RET

; COMND keeps track of the last byte that was output to the command port. It is necessary to keep track of what the P&T-488 interface board is asserting on the bus because the 488 bus is an open-collector wire-or system, so it is not possible to determine what the P&T-488 is asserting on the 488 bus by merely sensing the 488 lines.
;
COMND:  STA  GIMTC  ;update the 488 command line image
OUT  CMDPT  ;put it on the command lines
RET

; Check for interrupt due to ATN, IFC or POC
;
NOTE:  This function does not reset the interrupts in the Interrupt Service Register (ISR)
;
INTRPT:  IN  ISRPT  ;look at the interrupt service register
RAR  ;put POC bit in carry
CNC  IPOC  ;..set POC bit in return code byte if no carry
RAR  ;REN > CARRY
RAR  ;SRQ > CARRY
RAR  ;ATN > CARRY
CNC  IATN  ;..set the XATN bit
RAR  ;IFC > CARRY
CNC  IIFC  ;..set the XIFC bit
LDA  RETCOD
ANI  0F0H  ;look at only POC, IFC and ATN
RET

IPOC:  PUSH  A
LDA  RETCOD
ORI  80H

ICOM:  STA  RETCOD
POP  A  ;restore reg A and carry

CP/M AUX-8
RET

IATN:  PUSH  A
       LDA  RETCOD
       ORI  20H
       JMP  ICOM

IIFC:  PUSH  A
       LDA  RETCOD
       ORI  40H
       JMP  ICOM

;  Print the reason for aborting then jump to the monitor
;
BYE:   PUSH  PSW  ;save the error code
       LXI  D,MS2  ;power on clear
       ANI  80H
       CNZ  PRINT
       POP  PSW  ;get the error code again
       PUSH  PSW
       LXI  D,MS3  ;XIFC
       ANI  40H
       CNZ  PRINT
       POP  PSW
       LXI  D,MS4  ;XATN
       ANI  20H
       CNZ  PRINT
       JMP  MONITR

;  No listeners present - print error message then
;  jump to the monitor
;
NOLSN: LXI  D,MS1  ;print no listener msg
;
;  Print error message and return to monitor
;
ERROR: CALL  PRINT
       JMP  MONITR

;  print the line pointed to by DE
;
PRINT: MVI  C,BUFPRN
       CALL  CPMIO
       RET

;  GIMTC:  DB  0  ;image of last byte sent to CMDPT
CHAR:   DB  0
RETCOD: DB  0  ;a byte containing the error code

MS1:    DB  'No listeners on the bus',CR,LF,ES
MS2:    DB  'S-100 POWER ON CLEAR or RESET',CR,LF,ES
MS3:    DB  'Another 488 Controller is asserting IFC true',CR,LF,ES
MS4:    DB  'Another 488 Controller is asserting ATN true',CR,LF,ES
This program uses the Acceptor handsahke routine to get a data byte from the IEEE-488 bus and display it on the system console.

ORG 100H

MONITR SET 0 ;CP/M warmstart entry point
CPMIO SET 5 ;CP/M I/O routine entry point
GETCHR SET 1 ;CP/M function code for console input
PUTCHR SET 2 ;CP/M function code for console output
CONSTAT SET 11 ;CP/M function code for console status

LXI SP,2000H ;initialize stack pointer
CALL INIT ;initialize the P&T-488 card

LOOP:
CALL ACEPTR ;get a byte from the 488 bus
MOV E,A ;put it in register E for CP/M
MVI C,PUTCHR ;function to print on console
CALL CPMIO ;CP/M I/O routine entry point
MVI C,CONSTAT ;look to see if a key is pressed
CALL CPMIO
ANI 1
JZ LOOP ;...no key pressed
MVI C,GETCHR ;get the key
CALL CPMIO
CPI 3 ;CONTROL C?
JNZ LOOP ;...no, so continue getting data from the bus
JMP MONITR ;...yes, so do a warmstart

Insert the Handshake routines here

END