intel

APPLICATION NOTE

AP-66

January 1980

Using The 8292 GPIB Controller

Tom Voll
Peripheral Components Applications

RELATED INTEL® PUBLICATIONS

8291 GPIB Talker/Listener Data Sheet
8292 GPIB Controller Data Sheet
8293 GPIB Transceiver Data Sheet
AP-41 Introduction the UPI™-41A
UPI™-41 User's Manual

RELATED ZIATECH PUBLICATIONS

ZT488 GPIB Logic Analyzer Operating Manual
ZT80/18 GPIB Controller Operating Manual
ZT7488/18 GPIB Controller Operating Manual

RELATED IEEE PUBLICATIONS

IEEE Std 488-1978 Digital Interface for Programmable Instrumentation

THE AUTHOR

Bert Forbes, Ziatech President, holds an SB from MIT (1966) and an MSEE from Stanford (1967). During the ten years prior to forming Ziatech, he worked at Hewlett-Packard where he was one of the architects of the HP3000 multi-lingual, multi-programming computer, as well as a member of the 3000's European marketing team based in Geneva, Switzerland. This innovative computer system is one of the major contributors to HP's outstanding sales record. Bert is a co-inventor named on several 3000 related patents held by Hewlett-Packard.

Also at Hewlett-Packard, Bert was the architect and Project Manager for the high-speed 16 bit CMOS/SOS microprocessor that is being used in conjunction with an HPIB controller chip in dozens of Hewlett-Packard instruments and computer peripherals. He has authored articles on logic design, IEEE 488 and computer architecture and has taught several university courses on microprocessors.

Bert has been specializing in the GPIB at Ziatech and has designed numerous GPIB controllers and interfaces. He is a nationally known consultant in this field.

Using the 8292 GPIB Controller

Contents

INTRODUCTION

The Intel® 8292 is a preprogrammed UPI™-41A that implements the Controller function of the IEEE Std 488-1978 (GPIB, HP-IB, IEC Bus, etc.). In order to function the 8292 must be used with the 8291 Talker/Listener and suitable interface and transceiver logic such as a pair of Intel 8293s. In this configuration the system has the potential to be a complete GPIB Controller when driven by the appropriate software. It has the following capabilities: System Controller, send IFC and Take Charge, send REN, Respond to SRQ, send Interface messages, Receive Control, Pass Control, Parallel Poll and Take Control Synchronously.

This application note will explain the 8292 only in the system context of an 8292, 8291, two 8293s and the driver software. If the reader wishes to learn more about the UPI-41A aspects of the 8292, Intel's Application Note AP-41 describes the hardware features and programming characteristics of the device. Additional information on the 8291 may be obtained in the data sheet. The 8293 is detailed in its data sheet. Both chips will be covered here in the details that relate to the GPIB controller.

The next section of this application note presents an overview of the GPIB in a tutorial, but comprehensive nature. The knowledgable reader may wish to skip this section; however, certain basic semantic concepts introduced there will be used throughout this note.

Additional sections cover the view of the 8292 from the CPU's data bus, the interaction of the 3 chip types (8291, 8292, 8293), the 8292's software protocol and the system level hardware/software protocol. A brief description of interrupts and DMA will be followed by an application example. Appendix A contains the source code for the system driver software.

GPIB/IEEE 488 OVERVIEW

DESIGN OBJECTIVES

What is the IEEE 488 (GPIB)?

The experience of designing systems for a variety of applications in the early 1970's caused Hewlett-Packard to define a standard intercommunication mechanism which would allow them to easily assemble instrumentation systems of varying degrees of complexity. In a typical situation each instrument designer designed his/her own interface from scratch. Each one was inconsistent in terms of electrical levels, pin-outs on a connector, and types of connectors. Every time they built a system they had to invent new cables and new documentation just to specify the cabling and interconnection procedures.

Based on this experience, Hewlett-Packard began to define a new interconnection scheme. They went further than that, however, for they wanted to specify the typical communication protocol for systems of instruments. So in 1972, Hewlett-Packard came out with the first version of the bus which since has been modified and standardized by a committee of several manufacturers, coordinated through the IEEE, to perfect what is now known as the IEEE 488 Interface Bus (also known as the HP-IB, the GPIB and the IEC bus). While this bus specification may not be perfect, it is a good compromise of the various desires and goals of instrumentation and computer peripheral manufacturers to produce a common interconnection mechanism. It fits most instrumentation systems in use today and also fits very well the microcomputer I/O bus requirements. The basic design objectives for the GPIB were to:

1. Specify a system that is easy to use, but has all of the terminology and the definitions related to that system precisely spelled out so that everyone uses the same language when discussing the GPIB.
2. Define all of the mechanical, electrical, and functional interface requirements of a system, yet not define any of the device aspects (they are left up to the instrument designer).
3. Permit a wide range of capabilities of instruments and computer peripherals to use a system simultaneously and not degrade each other's performance.
4. Allow different manufacturers' equipment to be connected together and work together on the same bus.
5. Define a system that is good for limited distance interconnections.
6. Define a system with minimum restrictions on performance of the devices.
7. Define a bus that allows asynchronous communication with a wide range of data rates.
8. Define a low cost system that does not require extensive and elaborate interface logic for the low cost instruments, yet provides higher capability for the higher cost instruments if desired.
9. Allow systems to exist that do not need a central controller; that is, communication directly from one instrument to another is possible.

Although the GPIB was originally designed for instrumentation systems, it became obvious that most of these systems would be controlled by a calculator or computer. With this in mind several modifications were made to the original proposal before its final adoption as an international standard. Figure 1 lists the salient characteristics of the

GPIB as both an instrumentation bus and as a computer I/O bus.

Data Rate
- 1M bytes/s, max
- 250k bytes/s, typ

Multiple Devices
- 15 devices, max (electrical limit)
- 8 devices, typ (interrupt flexibility)

Bus Length
- 20 m, max
- 2 m/device, typ

Byte Oriented
- 8-bit commands
- 8-bit data

Block Multiplexed
- Optimum strategy on GPIB due to setup overhead for commands

Interrupt Driven
- Serial poll (slower devices)
- Parallel poll (faster devices)

Direct Memory Access
- One DMA facility at controller serves all devices on bus

Asynchronous
- One talker
- Multiple listeners

} 3-wire handshake

I/O to I/O Transfers
- Talker and listeners need not include microcomputer/controller

Figure 1. Major Characteristics of GPIB as Microcomputer I/O Bus

The bus can be best understood by examining each of these characteristics from the viewpoint of a general microcomputer I/O bus.

Data Rate — Most microcomputer systems utilize peripherals of differing operational rates, such as floppy discs at 31k or 62k bytes/s (single or double density), tape cassettes at 5k to 10k bytes/s, and cartridge tapes at 40k to 80k bytes/s. In general, the only devices that need high speed I/O are 0.5″ (1.3-cm) magnetic tapes and hard discs, operational at 30k to 781k bytes/s, respectively. Certainly, the 250k-bytes/s data rate that can be easily achieved by the IEEE 488 bus is sufficient for microcomputers and their peripherals, and is more than needed for typical analog instruments that take only a few readings per second. The 1M-byte/s maximum data rate is not easily achieved on the GPIB and requires special attention to considerations beyond the scope of this note. Although not required, data buffering in each device will improve the overall bus performance and allow utilization of more of the bus bandwidth.

Multiple Devices — Many microcomputer systems used as computers (not as components) service from three to seven peripherals. With the GPIB, up to 8 devices can be handled easily by 1 controller; with some slowdown in interrupt handling, up to 15 devices can work together. The limit of 8 is imposed by the number of unique parallel poll responses available; the limit of 15 is set by the electrical drive characteristics of the bus. Logically, the IEEE 488 Standard is capable of accommodating more device addresses (31 primary, each potentially with 31 secondaries).

Bus Length — Physically, the majority of microcomputer systems fit easily on a desk top or in a standard 19″ (48-cm) rack, eliminating the need for extra long cables. The GPIB is designed typically to have 2 m of length per device, which accommodates most systems. A line printer might require greater cable lengths, but this can be handled at the lower speeds involved by using extra dummy terminations.

Byte Oriented — The 8-bit byte is almost universal in I/O applications; even 16-bit and 32-bit computers use byte transfers for most peripherals. The 8-bit byte matches the ASCII code for characters and is an integral submultiple of most computer word sizes. The GPIB has an 8-bit wide data path that may be used to transfer ASCII or binary data, as well as the necessary status and control bytes.

Block Multiplexed — Many peripherals are block oriented or are used in a block mode. Bytes are transferred in a fixed or variable length group; then there is a wait before another group is sent to that device, e.g., one sector of a floppy disc, one line on a printer or tape punch, etc. The GPIB is, by nature, a block multiplexed bus due to the overhead involved in addressing various devices to talk and listen. This overhead is less bothersome if it only occurs once for a large number of data bytes (once per block). This mode of operation matches the needs of microcomputers and most of their peripherals. Because of block multiplexing, the bus works best with buffered memory devices.

Interrupt Driven — Many types of interrupt systems exist, ranging from complex, fast, vectored/priority networks to simple polling schemes. The main tradeoff is usually cost versus speed of response. The GPIB has two interrupt protocols to help span the range of applications. The first is a single service request (SRQ) line that may be asserted by all interrupting devices. The controller then polls all devices to find out which wants service. The polling mechanism is well defined and can be easily

automated. For higher performance, the parallel poll capability in the IEEE 488 allows up to eight devices to be polled at once — each device is assigned to one bit of the data bus. This mechanism provides fast recognition of an interrupting device. A drawback is the frequent need for the controller to explicitly conduct a parallel poll, since there is no equivalent of the SRQ line for this mode.

Direct Memory Access (DMA) — In many applications, no imediate processing of I/O data on a byte-by-byte basis is needed or wanted. In fact, programmed transfers slow down the data transfer rate unnecessarily in these cases, and higher speed can be obtained using DMA. With the GPIB, one DMA facility at the controller serves all devices. There is no need to incorporate complex logic in each device.

Asynchronous Transfers — An asynchronous bus is desirable so that each device can transfer at its own rate. However, there is still a strong motivation to buffer the data at each device when used in large systems in order to speed up the aggregate data rate on the bus by allowing each device to transfer at top speed. The GPIB is asynchronous and uses a special 3-wire handshake that allows data transfers from one talker to many listeners.

I/O To I/O Transfers — In practice, I/O to I/O transfers are seldom done due to the need for processing data and changing formats or due to mismatched data rates. However, the GPIB can support this mode of operation where the microcomputer is neither the talker nor one of the listeners.

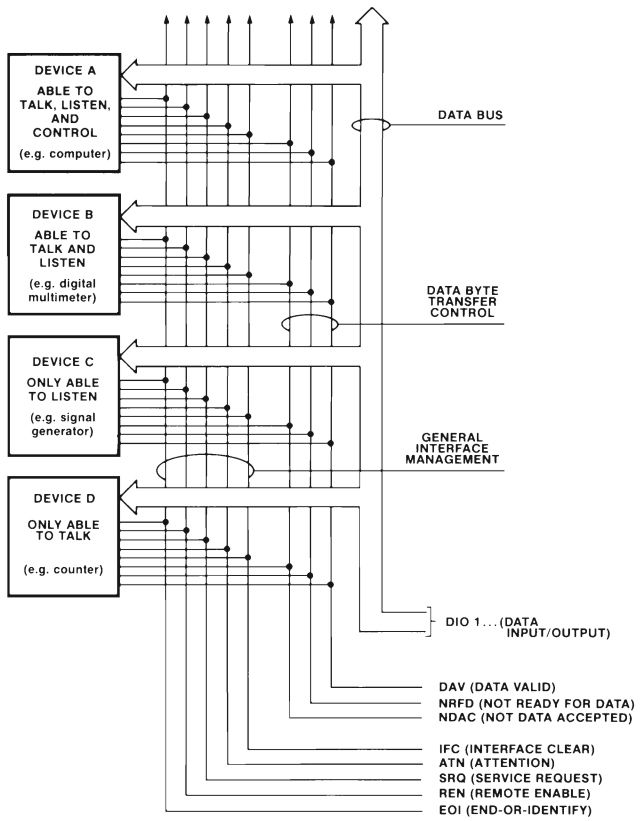


Figure 2. Interface Capabilities and Bus Structure

GPIB SIGNAL LINES

Data Bus

The lines DI01 through DI08 are used to transfer addresses, control information and data. The formats for addresses and control bytes are defined by the IEEE 488 standard (see Appendix C). Data formats are undefined and may be ASCII (with or without parity) or binary. DI01 is the Least Significant Bit (note that this will correspond to bit 0 on most computers).

Management Bus

ATN — Attention This signal is asserted by the Controller to indicate that it is placing an address or control byte on the Data Bus. ATN is de-asserted to allow the assigned Talker to place status or data on the Data Bus. The Controller regains control by re-asserting ATN; this is normally done synchronously with the handshake to avoid confusion between control and data bytes.

EOI — End or Identify This signal has two uses as its name implies. A talker may assert EOI simultaneously with the last byte of data to indicate end of data. The Controller may assert EOI along with ATN to initiate a Parallel Poll. Although many devices do not use Parallel Poll, all devices *should* use EOI to end transfers (many currently available ones do not).

SRQ — Service Request This line is like an interrupt: it may be asserted by any device to request the Controller to take some action. The Controller must determine which device is asserting SRQ by conducting a Serial Poll at its earliest convenience. The device deasserts SRQ when polled.

IFC — Interface Clear This signal is asserted only by the System Controller in order to initialize all device interfaces to a known state. After deasserting IFC, the System Controller is the active controller of the system.

REN — Remote Enable This signal is asserted only by the System Controller. Its assertion does not place devices into Remote Control mode; REN only *enables* a device to go remote when addressed to listen. When in Remote, a device should ignore its front panel controls.

Transfer Bus

NRFD — Not Ready For Data This handshake line is asserted by a listener to indicate it is not yet ready for the next data or control byte. Note that the Controller will not see NRFD deasserted (i.e., ready for data) until all devices have deasserted NRFD.

NDAC — Not Data Accepted This handshake line is asserted by a Listener to indicate it has not yet accepted the data or control byte on the DIO lines. Note that the Controller will not see NDAC deasserted (i.e., data accepted) until all devices have deasserted NDAC.

DAV — Data Valid This handshake line is asserted by the Talker to indicate that a data or control byte has been placed on the DIO lines and has had the minimum specified settling time.

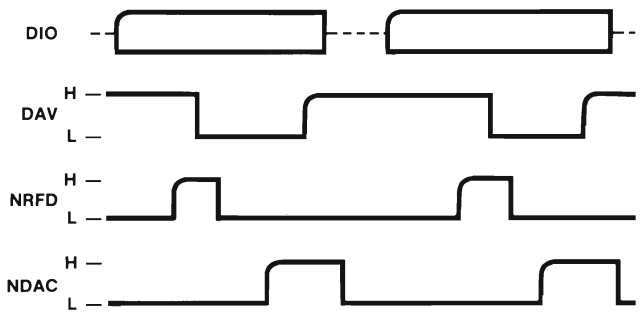


Figure 3. GPIB Handshake Sequence

GPIB INTERFACE FUNCTIONS

There are ten (10) interface functions specified by the IEEE 488 standard. Not all devices will have all functions and some may only have partial subsets. The ten functions are summarized below with the relevant section number from the IEEE document given at the beginning of each paragraph. For further information please see the IEEE standard.

1. *SH* — Source Handshake (section 2.3) This function provides a device with the ability to properly transfer data from a Talker to one or more Listeners using the three handshake lines.
2. *AH* — Acceptor Handshake (section 2.4) This function provides a device with the ability to properly receive data from the Talker using the three handshake lines. The AH function may also delay the beginning (NRFD) or end (NDAC) of any transfer.
3. *T* — Talker (section 2.5) This function allows a device to send status and data bytes when addressed to talk. An address consists of one (Primary) or two (Primary and Secondary) bytes. The latter is called an extended Talker.
4. *L* — Listener (section 2.6) This function allows a device to receive data when addressed to listen. There can be extended Listeners (analogous to extended Talkers above).
5. *SR* — Service Request (section 2.7) This function allows a device to request service (interrupt) the Controller. The SRQ line may be asserted asynchronously.
6. *RL* — Remote Local (section 2.8) This function allows a device to be operated in two modes: Remote via the GPIB or Local via the manual front panel controls.
7. *PP* — Parallel Poll (section 2.9) This function allows a device to present one bit of status to the Controller-in-charge. The device need not be addressed to talk and no handshake is required.
8. *DC* — Device Clear (section 2.10) This function allows a device to be cleared (initialized) by the Controller. Note that there is a difference between DC (*device* clear) and the IFC line (*interface* clear).
9. *DT* — Device Trigger (section 2.11) This function allows a device to have its basic operation started either individually or as part of a group. This capability is often used to synchronize several instruments.
10. *C* — Controller (section 2.12) This function allows a device to send addresses, as well as universal and addressed commands to other devices. There may be more than one controller on a system, but only one may be the controller-in-charge at any one time.

At power-on time the controller that is handwired to be the System Controller becomes the active controller-in-charge. The System Controller has several unique capabilities including the ability to send Interface Clear (IFC — clears all device interfaces and returns control to the System Controller) and to send Remote Enable (REN — allows devices to respond to bus data once they are addressed to listen). The System Controller may optionally Pass Control to another controller, if the system software has the capability to do so.

GPIB CONNECTOR

The GPIB connector is a standard 24-pin industrial connector such as Cinch or Amphenol series 57 Micro-Ribbon. The IEEE standard specifies this connector, as well as the signal connections and the mounting hardware.

The cable has 16 signal lines and 8 ground lines. The maximum length is 20 meters with no more than two meters per device.

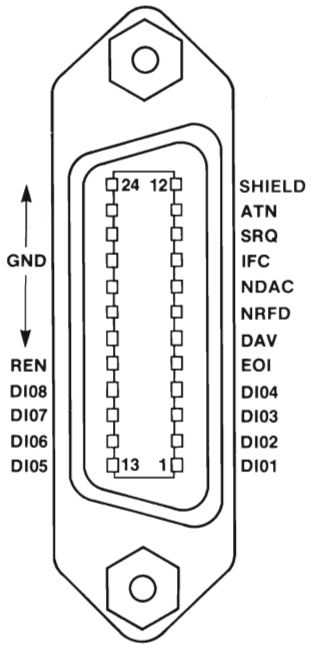


Figure 4. GPIB Connector

GPIB SIGNAL LEVELS

The GPIB signals are all TTL compatible, low true signals. A signal is asserted (true) when its electrical voltage is less than 0.5 volts and is deasserted (false) when it is greater than 2.4 volts. Be careful not to become confused with the two handshake signals, NRFD and NDAC which are also low true (i.e. > 0.5 volts implies the device is Not Ready For Data).

The Intel 8293 GPIB transceiver chips ensure that all relevant bus driver/receiver specifications are met. Detailed bus electrical specifications may be found in Section 3 of the IEEE Std 488-1978. The Standard is the ultimate reference for all GPIB questions.

GPIB MESSAGE PROTOCOLS

The GPIB is a very flexible communications medium and as such has many possible variations of protocols. To bring some order to the situation, this section will discuss a protocol similar to the one used by Ziatech's ZT80 GPIB controller for Intel's MULTIBUS™ computers. The ZT80 is a complete high-level interface processor that executes a set of high level instructions that map directly into GPIB actions. The sequences of commands, addresses and data for these instructions provide a good example of how to use the GPIB (additional information is available in the ZT80 Instruction Manual). The 'null' at the end of each instruction is for cosmetic use to remove previous information from the DIO lines.

DATA — Transfer a block of data from device A to devices B, C...
1. Device A Primary (Talk) Address
   Device A Secondary Address (if any)
2. Universal Unlisten
3. Device B Primary (Listen) Address
   Device B Secondary Address (if any)
   Device C Primary (Listen) Address
   etc.
4. First Data Byte
   Second Data Byte
   .
   .
   .
   Last Data Byte (EOI)
5. Null

TRIGR — Trigger devices A, B,...to take action
1. Universal Unlisten
2. Device A Primary (Listen) Address
   Device A Secondary Address (if any)
   Device B Primary (Listen) Address
   Device B Secondary Address (if any)
   etc.
3. Group Execute Trigger
4. Null

PSCTL — Pass control to device A
1. Device A Primary (Talk) Address
   Device A Secondary Address (if any)
2. Take Control
3. Null

CLEAR — Clear all devices
1. Device Clear
2. Null

REMAL — Remote Enable
1. Assert REN continuously

GOREM — Put devices A, B,...into Remote
1. Assert REN continuously
2. Device A Primary (Listen) Address
   Device A Secondary Address (if any)
   Device B Primary (Listen) Address
   Device B Secondary Address (if any)
   etc.
3. Null

GOLOC — Put devices A, B,...into Local
1. Device A Primary (Listen) Address
   Device A Secondary Address (if any)
   Device B Primary (Listen) Address
   Device B Secondary Address (if any)
   etc.
2. Go To Local
3. Null

LOCAL — Reset all devices to Local
1. Stop asserting REN

LLKAL — Prevent all devices from returning to Local
1. Local Lock Out
2. Null

SPOLL — Conduct a serial poll of devices A, B,...
1. Serial Poll Enable
2. Universal Unlisten
3. ZT 80 Primary (Listen) Address
   ZT 80 Secondary Address
4. Device Primary (Talk) Address
   Device Secondary Address (if any)
5. Status byte from device
6. Go to Step 4 until all devices on list have been polled
7. Serial Poll Disable
8. Null

PPUAL — Unconfigure and disable Parallel Poll response from all devices
1. Parallel Poll Unconfigure
2. Null

ENAPP — Enable Parallel Poll response in devices A, B,...
1. Universal Unlisten
2. Device Primary (Listen) Address
   Device Secondary Address (if any)
3. Parallel Poll Configure
4. Parallel Poll Enable
5. Go to Step 2 until all devices on list have been configured.
6. Null

DISPP — Disable Parallel Poll response from devices A, B,...
1. Universal Unlisten
2. Device A Primary (Listen) Address
   Device A Secondary Address (if any)
   Device B Primary (Listen) Address
   Device B Secondary Address (if any)
   etc.
3. Disable Parallel Poll
4. Null

This Ap Note will detail how to implement a useful subset of these controller instructions.

HARDWARE ASPECTS OF THE SYSTEM

8291 GPIB TALKER/LISTENER

The 8291 is a custom designed chip that implements many of the non-controller GPIB functions. It provides hooks so the user's software can implement additional features to complete the set. This chip is discussed in detail in its data sheet. The major features are summarized here:

—Designed to interface microprocessors to the GPIB
—Complete Source and Acceptor Handshake
—Complete Talker and Listener Functions with extended addressing
—Service Request, Parallel Poll, Device Clear, Device Trigger, Remote/Local functions
—Programmable data transfer rate
—Maskable interrupts
—On-chip primary and secondary address recognition
—1-8 MHz clock range
—16 registers (8 read, 8 write) for CPU interface
—DMA handshake provision
—Trigger output pin
—On-chip EOS (End of Sequence) recognition

The pinouts and block diagram are shown in Fig. 5. One of eight read registers is for data transfer to the CPU; the other seven allow the microprocessor to monitor the GPIB states and various bus and device conditions. One of the eight write registers is for data transfer from the CPU; the other seven control various features of the 8291.

The 8291 interface functions will be software configured in this application example to the following subsets for use with the 8292 as a controller that does not pass control. The 8291 is used only to provide the handshake logic and to send and receive data bytes. It is not acting as a normal device in this mode, as it never sees ATN asserted.

SH1	Source Handshake
AH1	Acceptor Handshake
T3	Basic Talk-only
L1	Basic Listen-only
SR0	No Service Requests
RL0	No Remote/Local
PP0	No Parallel Poll response
DC0	No Device Clear
DT0	No Device Trigger

If control is passed to another controller, the 8291 must be reconfigured to act as a talker/listener with the following subsets:

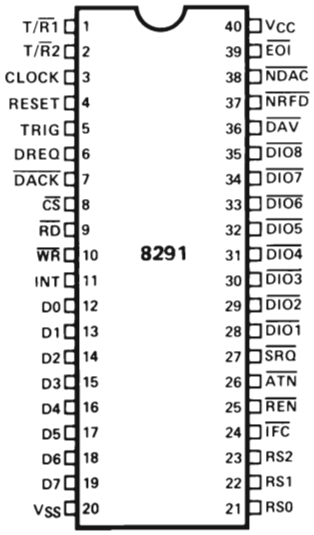
SH1	Source Handshake
AH1	Acceptor Handshake
T5	Basic Talker and Serial Poll
L3	Basic Listener
SR1	Service Requests
RL1	Remote/Local with Lockout
PP2	Preconfigured Parallel Poll
DC1	Device Clear
DT1	Device Trigger
C0	Not a Controller

Most applications do not pass control and the controller is always the system controller (see 8292 commands below).

8292 GPIB CONTROLLER

The 8292 is a preprogrammed Intel® 8041A that provides the additional functions necessary to

PIN CONFIGURATION



BLOCK DIAGRAM

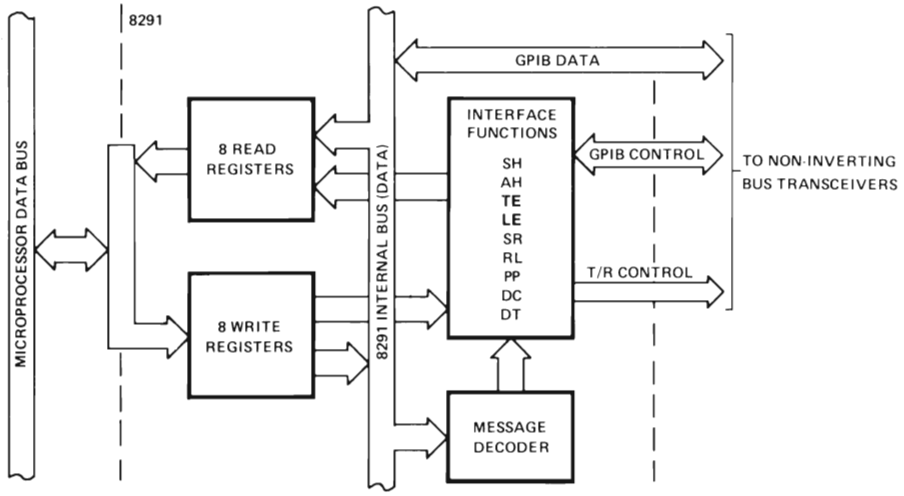


Figure 5. 8291 Pin Configuration and Block Diagram

implement a GPIB controller when used with an 8291 Talker/Listener. The 8041A is documented in both a user's manual and in AP-41. The following description will serve only as an outline to guide the later discussion.

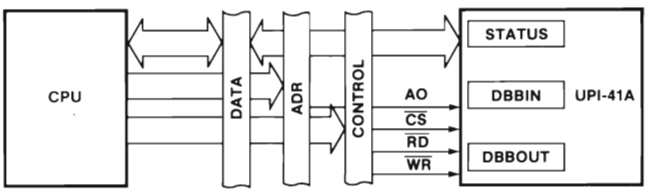
The 8292 acts as an intelligent slave processor to the main system CPU. It contains a processor, memory, I/O and is programmed to perform a variety of tasks associated with GPIB controller operation. The on-chip RAM is used to store information about the state of the Controller function, as well as a variety of local variables, the stack and certain user status information. The timer/counter may be optionally used for several time-out functions or for counting data bytes transferred. The I/O ports provide the GPIB control signals, as well as the ancillary lines necessary to make the 8291, 2, 3 work together.

The 8292 is closely coupled to the main CPU through three on-chip registers that may be independently accessed by both the master and the 8292 (UPI-41A). Figure 6 shows this Register Interface. Also refer to Figure 12.

The status register is used to pass Interrupt Status information to the master CPU (A0 = 1 on a read).

The DBBOUT register is used to pass one of five other status words to the master based on the last command written into DBBIN. DBBOUT is accessed when A0 = 0 on a Read. The five status words are Error Flag, Controller Status, GPIB Status, Event Counter Status or Time Out Status.

DBBIN receives either commands (A0 = 1 on a Write) or command related data (A0 = 0 on a write) from the master. These command related data are Interrupt Mask, Error Mask, Event Counter or Time Out.



CS	AO	RD	WR	REGISTER
0	0	0	1	READ DBBOUT
0	1	0	1	READ STATUS
0	0	1	0	WRITE DBBIN (DATA)
0	1	1	0	WRITE DBBIN (COMMAND)
1	X	X	X	NO ACTION

Figure 6. UPI-41A Registers

8293 GPIB TRANSCEIVERS

The 8293 is a multi-use HMOS chip that implements the IEEE 488 bus transceivers and contains the additional logic required to make the 8291 and 8292 work together. The two option strapping pins are used to internally configure the chip to perform the specialized gating required for use with 8291 as a device or with 8291/92 as a controller.

In this application example the two configurations used are shown in Fig. 7a and 7b. The drivers are set to open collector or three state mode as required and the special logic is enabled as required in the two modes.

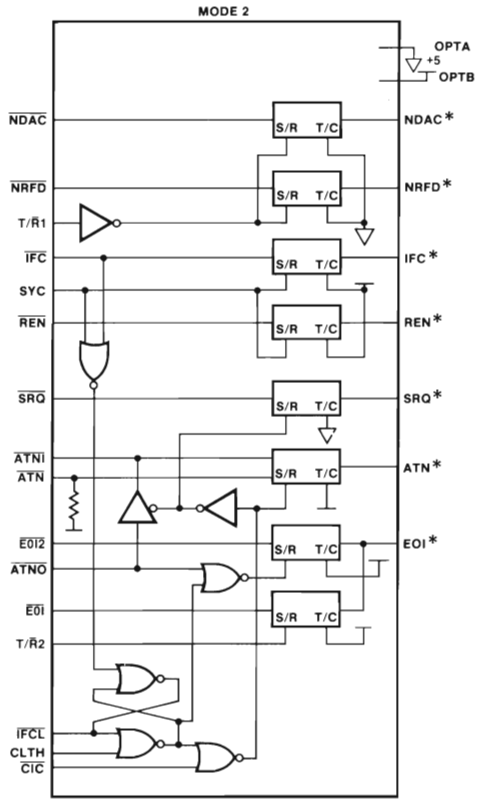


Figure 7a. 8293 Mode 2

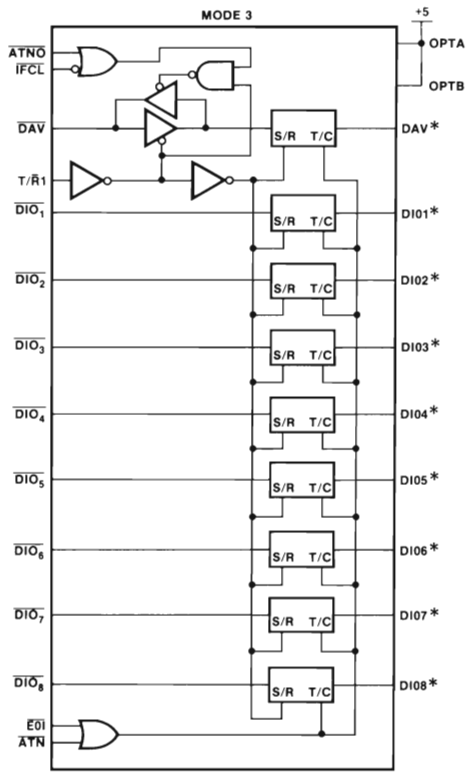


Figure 7b. 8293 Mode 3

8291/2/3 CHIP SET

Figure 8 shows the four chips interconnected with the special logic explicitly shown.

The 8291 acts only as the mechanism to put commands and addresses on the bus while the 8292 is asserting ATN. The 8291 is tricked into believing that the ATN line is not asserted by the ATN2 output of the ATN transceiver and is placed in Talk-only mode by the CPU. The 8291 then acts as though it is sending data, when in reality it is sending addresses and/or commands. When the 8292 deasserts ATN, the CPU software must place the 8291 in Talk-only, Listen-only or Idle based on the implicit knowledge of how the controller is going to participate in the data transfer. In other words, the 8291 does not respond directly to addresses or commands that it sends on the bus on behalf of the Controller. The user software, through the use of Listen-only or Talk-only, makes the 8291 behave as though it were addressed.

Although it is not a common occurrence, the GPIB specification allows the Controller to set up a data transfer between two devices and not directly participate in the exchange. The controller must know when to go active again and regain control. The chip set accomplishes this through use of the "Continuous Acceptor Handshake cycling mode" and the ability to detect EOI or EOS at the end of the transfer. See XFER in the Software Driver Outline below.

If the 8292 is not the System Controller as determined by the signal on its SYC pin, then it must be able to respond to an IFC within 100 usec. This is accomplished by the cross-coupled NORs in Fig. 7a which deassert the 8293's internal version of $\overline{\text{CIC}}$ (Not Controller-in-Charge). This condition is latched until the 8292's firmware has received the IFCL (interface clear received latch) signal by testing the $\overline{\text{IFCL}}$ input. The firmware then sets its signals to reflect the inactive condition and clears the 8293's latch.

In order for the 8292 to conduct a Parallel Poll the 8291 must be able to capture the PP response on the DIO lines. The only way to do this is to fool the 8291 by putting it into Listen-only mode and generating a DAV condition. However, the bus spec does not allow a DAV during Parallel Poll, so the back-to-back 3-state buffers (see Fig. 7b) in the 8293 isolate the bus and allow the 8292 to generate a local DAV for this purpose. Note that the 8291 cannot assert a Parallel Poll response. When the 8292 is not the controller-in-charge the 8291 may respond to PPs and the 8293 guarantees that the DIO drivers are in "open collector" mode through the OR gate (Fig. 7b).

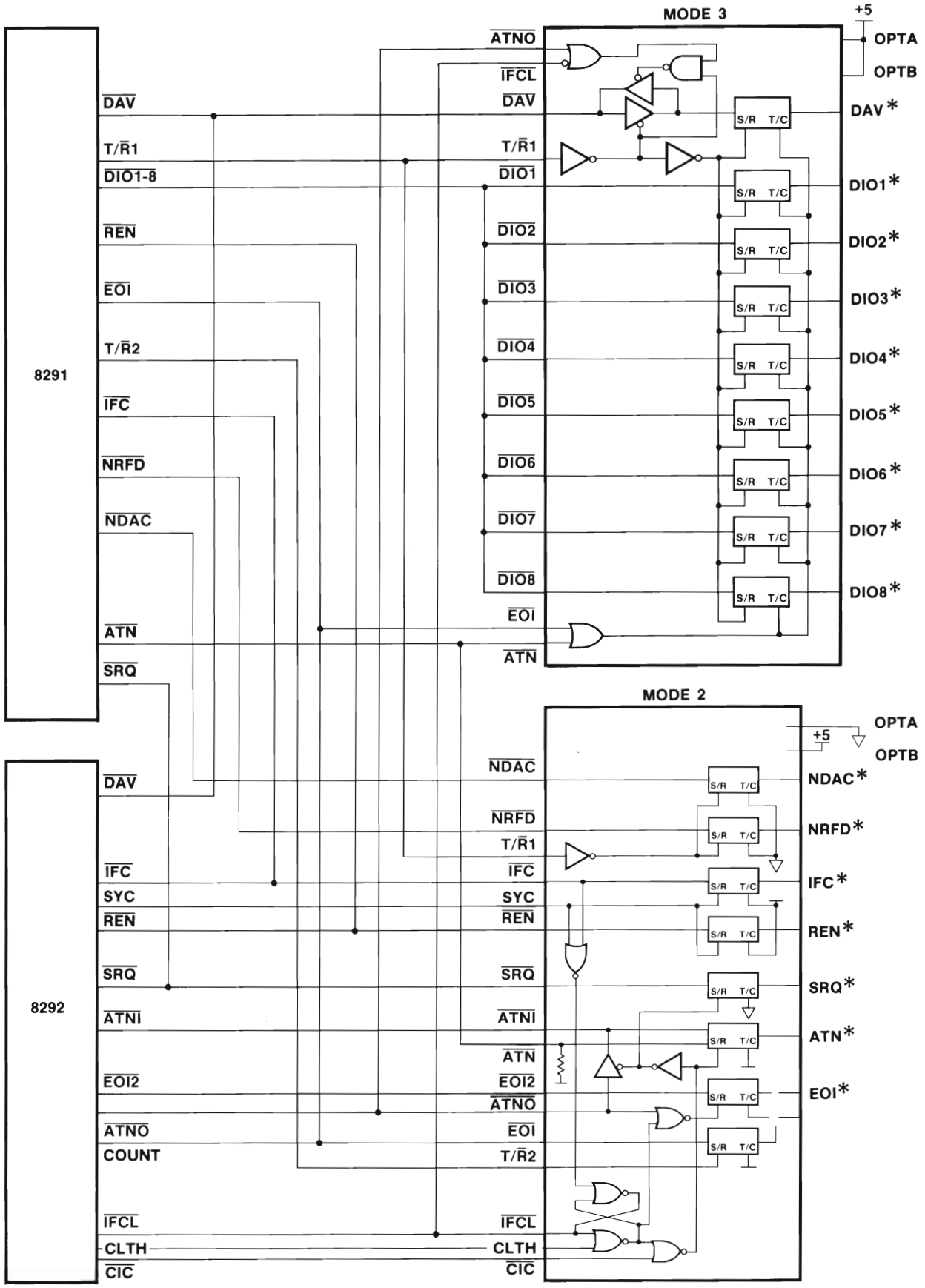


Figure 8. Talker/Listener/Controller

ZT7488/18 GPIB CONTROLLER

Ziatech's GPIB Controller, the ZT7488/18 will be used as the controller hardware in this Application Note. The controller consists of an 8291, 8292, an 8 bit input port and TTL logic equivalent to that shown in Figure 8. Figure 9 shows the card's block diagram. The ZT7488/18 plugs into the STD bus, a 56 pin 8 bit microprocessor oriented bus. An 8085 CPU card is also available on the STD bus and will be used to execute the driver software.

The 8291 uses I/O Ports 60H to 67H and the 8292 uses I/O Ports 68H and 69H. The five interrupt lines are connected to a three-state buffer at I/O Port 6FH to facilitate polling operation. This is required for the TCI, as it cannot be read internally in the 8292. The other three 8292 lines (SPI, IBF, OBF) and the 8291's INT line are also connected to minimize the number of I/O reads necessary to poll the devices.

$\overline{\text{NDAC}}$ is connected to $\overline{\text{COUNT}}$ on the 8292 to allow byte counting on data transfers. The example driver software will not use this feature, as the software is simpler and faster if an internal 8085 register is used for counting in software.

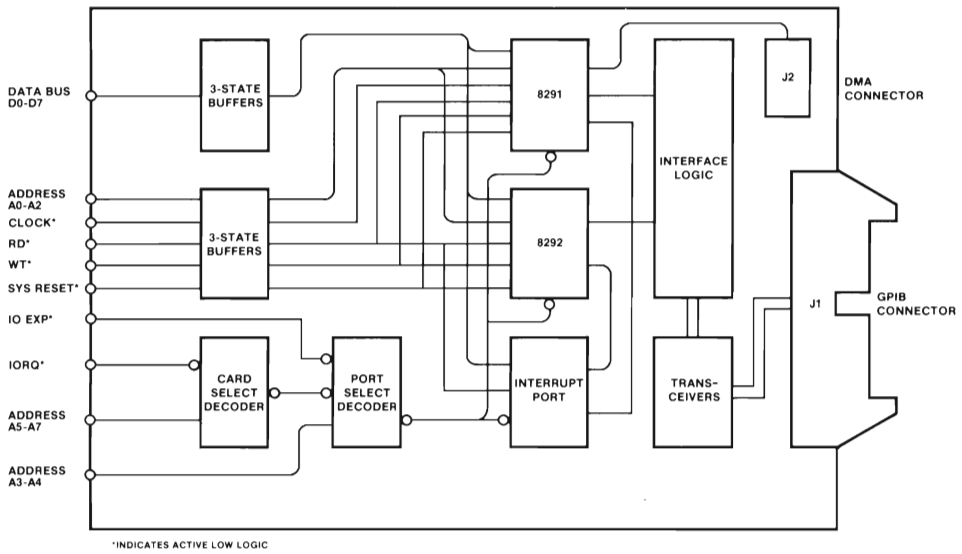


Figure 9. ZT7488/18 GPIB Controller

READ REGISTERS								PORT #	WRITE REGISTERS							
DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0	6ϕH	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0
DATA IN									DATA OUT							
CPT	APT	GET	END	DEC	ERR	BO	BI	61H	CPT	APT	GET	END	DEC	ERR	BO	BI
INTERRUPT STATUS 1									INTERRUPT MASK 1							
INT	SPAS	LLO	REM	SPASC	LLOC	REMC	ADSC	62H	0	0	DMAO	DMAI	SPASC	LLOC	REMC	ADSC
INTERRUPT STATUS 2									INTERRUPT MASK 2							
S8	SRQS	S6	S5	S4	S3	S2	S1	63H	S8	rsv	S6	S5	S4	S3	S2	S1
SERIAL POLL STATUS									SERIAL POLL MODE							
ton	lon	EOI	LPAS	TPAS	LA	TA	MJMN	64H	TO	LO	0	0	0	0	ADM1	ADM0
ADDRESS STATUS									ADDRESS MODE							
CPT7	CPT6	CPT5	CPT4	CPT3	CPT2	CPT1	CPT0	65H	CNT2	CNT1	CNT0	COM4	COM3	COM2	COM1	COM0
COMMAND PASS THROUGH									AUX MODE							
X	DT0	DL0	AD5-0	AD4-0	AD3-0	AD2-0	AD1-0	66H	ARS	DT	DL	AD5	AD4	AD3	AD2	AD1
ADDRESS 0									ADDRESS 0/1							
X	DT1	DL1	AD5-1	AD4-1	AD3-1	AD2-1	AD1-1	67H	EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0
ADDRESS 1									EOS							

Figure 10. 8291 Registers

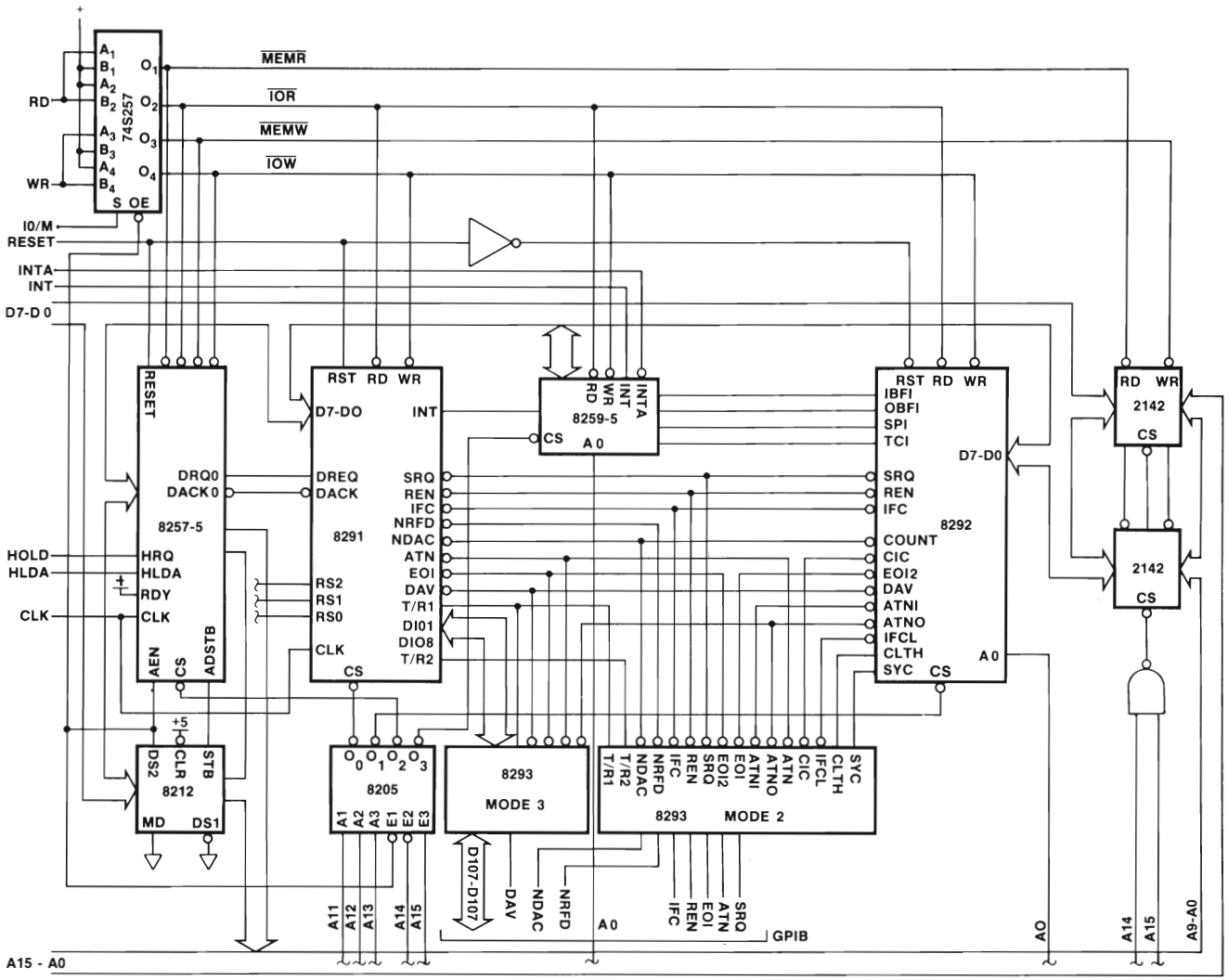


Figure 11. DMA/Interrupt GPIB Controller Block Diagram

The application example will not use DMA or interrupts; however, the Figure 11 block diagram includes these features for completeness.

The 8257-5 DMA chip can be used to transfer data between the RAM and the 8291 Talker/Listener. This mode allows a faster data rate on the GPIB and typically will depend on the 8291's EOS or EOI detection to terminate the transfer. The 8259-5 interrupt controller is used to vector the five possible interrupts for rapid software handling of the various conditions.

8292 COMMAND DESCRIPTION

This section discusses each command in detail and relates them to a particular GPIB activity. Recall that although the 8041A has only two read registers and one write register, through the magic of on-chip firmware the 8292 appears to have six read registers and five write registers. These are listed in Figure 12. Please see the 8292 data sheet for detailed definitions of each register. Note the two letter mnemonics to be used in later discussions. The CPU must not write into the 8292 while IBF (Input Buffer Full) is a one, as information will be lost.

DIRECT COMMANDS

Both the Interrupt Mask (IM) and the Error Mask (EM) register may be directly written with the LSB of the address bus (A0) a "0". The firmware uses the MSB of the data written to differentiate between IM and EM.

Load Interrupt Mask

This command loads the Interrupt Mask with D7–D0. Note that D7 must be a "1" and that interrupts are enabled by a corresponding "1" bit in this register. IFC interrupt cannot be masked off; however, when the 8292 is the System Controller, sending an ABORT command will not cause an IFC interrupt.

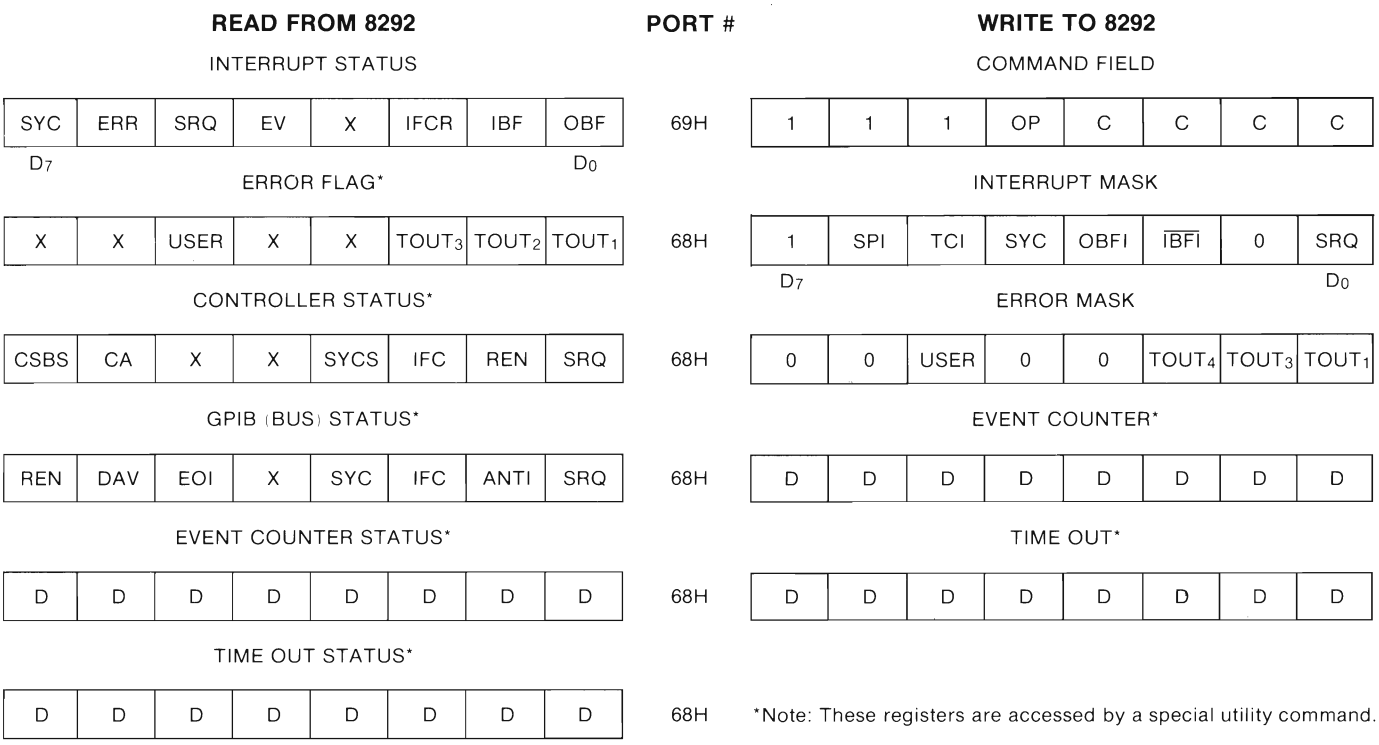


Figure 12. 8292 Registers

Load Error Mask

This command loads the Error Mask with D7–D0. Note that D7 must be a zero and that interrupts are enabled by a corresponding "1" bit in this register.

UTILITY COMMANDS

These commands are used to read or write the 8292 registers that are not directly accessible. All utility commands are written with A0 = 1, D7 = D6 = D5 = 1, D4 = 0. D3–D0 specify the particular command. For writing into registers the general sequence is:

1. wait for IBF = 0 in Interrupt Status Register
2. write the appropriate command to the 8292,
3. write the desired register value to the 8292 with A0 = 1 with no other writes intervening,
4. wait for indication of completion from 8292 (IBF = 0).

For reading a register the general sequence is:

1. wait for IBF = 0 in Interrupt Status Register
2. write the appropriate command to the 8292
3. wait for a TCI (Task Complete Interrupt)
4. Read the value of the accessed register from the 8292 with A0 = 0.

WEVC — Write to Event Counter
(Command = 0E2H)

The byte written following this command will be loaded into the event counter register and event counter status for byte counting. The internal counter is incremented on a high to low transition of the COUNT (T1) input. In this application example $\overline{\text{NDAC}}$ is connected to count. The counter is an 8 bit register and therefore can count up to 256 bytes (writing 0 to the EC implies a count of 256). If longer blocks are desired, the main CPU must handle the interrupts every 256 counts and carefully observe the timing constraints.

Because the counter has a frequency range from 0 to 133 kHz when using a 6 MHz crystal, this feature may not be usable with all devices on the GPIB. The 8291 can easily transfer data at rates up to 250 kHz and even faster with some tuning of the system. There is also a 500 ns minimum high time requirement for COUNT which can potentially be violated by the 8291 in continuous acceptor handshake mode (i.e., TNDDV1 + TDVND2–C = 350 + 350 = 700 max). When cable delays are taken into consideration, this problem will probably never occur.

When the 8292 has completed the command, IBF will become a "0" and will cause an interrupt if masked on.

WTOUT — Write to Time Out Register
(Command = 0E1H)

The byte written following this command will be used to determine the number of increments used for the time out functions. Because the register is 8 bits, the maximum time out is 256 time increments. This

is probably enough for most instruments on the GPIB but is not enough for a manually stepped operation using a GPIB logic analyzer like Ziatech's ZT488. Also, the 488 Standard does not set a lower limit on how long a device may take to do each action. Therefore, any use of a time out must be able to be overridden (this is a good general design rule for service and debugging considerations).

The time out function is implemented in the 8292's firmware and will not be an accurate time. The counter counts backwards to zero from its initial value. The function may be enabled/disabled by a bit in the Error mask register. When the command is complete IBF will be set to a "0" and will cause an interrupt if masked on.

REVC — Read Event Counter Status
(Command = 0E3H)

This command transfers the content of the Event Counter to the DBBOUT register. The firmware then sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value from the 8292 with A0 = 0.

RINM — Read Interrupt Mask Register
(Command = 0E5H)

This command transfers the content of the Interrupt Mask register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

RERM — Read Error Mask Register
(Command = 0EAH)

This command transfers the content of the Error Mask register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

RCST — Read Controller Status Register
(Command = 0E6H)

This command transfers the content of the Controller Status register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

RTOUT — Read Time Out Status Register
(Command = 0E9H)

This command transfers the content of the Time Out Status register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

If this register is read while a time-out function is in process, the value will be the time remaining before time-out occurs. If it is read after a time-out, it will be zero. If it is read when no time-out is in process, it will be the last value reached when the previous timing occurred.

RBST — Read Bus Status Register
(Command = 0E7H)

This command causes the firmware to read the GPIB management lines, DAV and the SYC pin and place a copy in DBBOUT. TCI is set to "1" and will cause an interrupt if masked on. The CPU may read the value.

RERF — Read Error Flag Register
(Command = 0E4H)

This command transfers the content of the Error Flag register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

This register is also placed in DBBOUT by an IACK command if ERR remains set. TCI is set to "1" in this case also.

IACK — Interrupt Acknowledge
(Command = A1 A2 A3 A4 1 A5 1 1)

This command is used to acknowledge any combinations of the five SPI interrupts (A1–A5): SYC, ERR, SRQ, EV, and IFCR. Each bit A1–A5 is an individual acnowledgement to the corresponding bit in the Interrupt Status Register. The command clears SPI but it will be set again if all of the pending interrupts were not acknowledged.

If A2 (ERR) is "1", the Error Flag register is placed in DBBOUT and TCI is set. The CPU may then read the Error Flag without issuing an RERF command.

OPERATION COMMANDS

The following diagram (Fig. 13) is an attempt to show the interrelationships among the various 8292 Operation Commands. It is not meant to replace the complete controller state diagram in the IEEE Standard.

RST — Reset (Command = 0F2H)

This command has the same effect as an external reset applied to the chip's pin #4. The 8292's actions are:

1. All outputs go to their electrical high state. This means that SPI, TCI, OBFI, $\overline{\text{IBFI}}$, CLTH will be TRUE and all other GPIB signals will be FALSE.
2. The 8292's firmware will cause the above mentioned five signals to go FALSE after approximately 17.5 usec. (at 6 MHz).
3. These registers will be cleared: Interrupt Status, Interrupt Mask, Error Mask, Time Out, Event Counter, Error Flag.
4. If the 8292 is the System Controller (SYC is TRUE), then IFC will be sent TRUE for approximately 100 usec and the Controller function will end up in charge of the bus. If the 8292 is not the

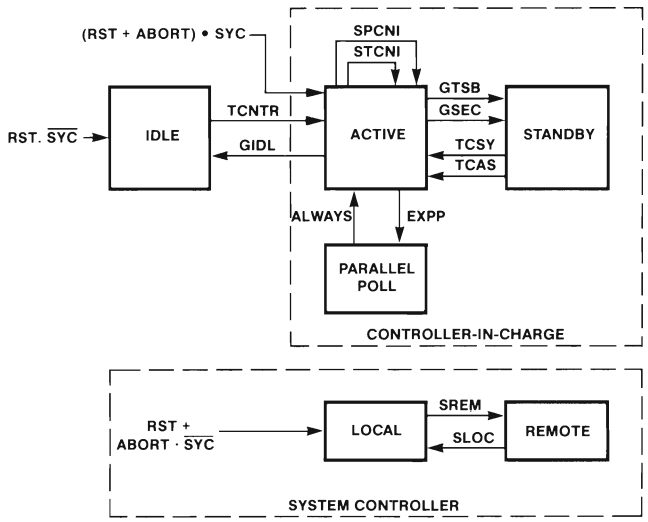


Figure 13. 8292 Command Flowchart

System Controller then it will end up in an Idle state.

5. TCI will not be set.

RSTI — Reset Interrupts (Command = 0F3)

This command clears all pending interrupts and error flags. The 8292 will stop waiting for actions to occur (e.g., waiting for ATN to go FALSE in a TCNTR command or waiting for the proper handshake state in a TCSY command). TCI will not be set.

ABORT — Abort all operations and Clear Interface (Command = 0F9H)

If the 8292 is not the System Controller this command acts like a NOP and flags a USER ERROR in the Error Flag Register. No TCI will occur.

If the 8292 is the System Controller then IFC is set TRUE for approximately 100 μsec and the 8292 becomes the Controller-in-Charge and asserts ATN. TCI will be set, only if the 8292 was NOT the CIC.

STCNI — Start Counter Interrupts (Command = 0FEH)

Enables the EV Counter Interrupt. TCI will not be set. Note that the counter must be enabled by a GSEC command.

SPCNI — Stop Counter Interrupts (Command = 0F0H)

The 8292 will not generate an EV interrupt when the counter reaches 0. Note that the counter will continue counting. TCI will not be set.

SREM — Set Interface to Remote Control (Command = 0F8H)

If the 8292 is the System Controller, it will set REN and TCI TRUE. Otherwise it only sets the User Error Flag.

SLOC — Set Interface to Local Mode (Command = 0F7H)

If the 8292 is the System Controller, it will set REN FALSE and TCI TRUE. Otherwise, it only sets the User Error Flag.

EXPP — Execute Parallel Poll (Command = 0F5H)

If not Controller-in-Charge, the 8292 will treat this as a NOP and does not set TCI. If it is the Controller-in-Charge then it sets IDY (EOI & ATN) TRUE and generates a local DAV pulse (that never reaches the GPIB because of gates in the 8293). If the 8291 is configured as a listener, it will capture the Parallel Poll Response byte in its data register. TCI is not generated, the CPU must detect the BI (Byte In) from the 8291. The 8292 will be ready to accept another command before the BI occurs; therefore the 8291's BI serves as a task complete indication.

GTSB — Go To Standby (Command = 0F6H)

If the 8292 is not the Controller-in-Charge, it will treat this command as a NOP and does not set TCI TRUE. Otherwise, it goes to Controller Standby State (CSBS), sets ATN FALSE and TCI TRUE. This command is used as part of the Send, Receive, Transfer and Serial Poll System commands (see next section) to allow the addressed talker to send data/status.

If the data transfer does not start within the specified Time-Out, the 8292 sets TOUT2 TRUE in the Error Flag Register and sets SPI (if enabled). The controller continues waiting for a new command. The CPU must decide to wait longer or to regain control and take corrective action.

GSEC — Go to Standby and Enable Counting (Command = 0F4H)

This command does the same things as GTSB but also initializes the event counter to the value previously stored in the Event Counter Register (default value is 256) and enables the counter. One may wire the count input to $\overline{\text{NDAC}}$ to count bytes. When the counter reaches zero, it sets EV (and SPI if enabled) in Interrupt Status and will set EV every 256 bytes thereafter. Note that there is a potential loss of count information if the CPU does not respond to the EV/SPI before another 256 bytes have been transferred. TCI will be set at the end of the command.

TCSY — Take Control Synchronously (Command = 0FDH)

If the 8292 is not in Standby, it treats this command as a NOP and does not set TCI. Otherwise, it waits

for the proper handshake state and sets ATN TRUE. The 8292 will set TOUT3 if the handshake never assumes the correct state and will remain in this command until the handshake is proper or a RSTI command is issued. If the 8292 successfully takes control, it sets TCI TRUE.

This is the normal way to regain control at the end of a Send, Receive, Transfer or Serial Poll System Command. If TCSY is not successful, then the controller must try TCAS (see warning below).

TCAS — Take Control Asynchronously (Command = 0FCH)

If the 8292 is not in Standby, it treats this command as a NOP and does not set TCI. Otherwise, it arbitrarily sets ATN TRUE and TCI TRUE. Note that this action may cause devices on the bus to lose a data byte or cause them to interpret a data byte as a command byte. Both Actions can result in anomalous behavior. TCAS should be used only in emergencies. If TCAS fails, then the System Controller will have to issue an ABORT to clean things up.

GIDL — Go to Idle (Command = 0F1H)

If the 8292 is not the Controller in Charge and Active, then it treats this command as a NOP and does not set TCI. Otherwise, it sets ATN FALSE, becomes Not Controller in Charge, and sets TCI TRUE. This command is used as part of the Pass Control System Command.

TCNTR — Take (Receive) Control (Command = 0FAH)

If the 8292 is not Idle, then it treats this command as a NOP and does not set TCI. Otherwise, it waits for the current Controller-in-Charge to set ATN FALSE. If this does not occur within the specified Time Out, the 8292 sets TOUT1 in the Error Flag Register and sets SPI (if enabled). it will not proceed until ATN goes false or it receives an RSTI command. Note that the Controller in Charge must previously have sent this controller (via the 8291's command pass through register) a Pass Control message. When ATN goes FALSE, the 8292 sets CIC, ATN and TCI TRUE and becomes Active.

SOFTWARE DRIVER OUTLINE

The set of system commands discussed below is shown in Figure 14. These commands are implemented in software routines executed by the main CPU.

The following section assumes that the Controller is the System Controller and will not Pass Control. This is a valid assumption for 99+% of all controllers. It also assumes that no DMA or Interrupts will be used. SYC (System Control Input) should not be changed after Power-on in any system — it adds unnecessary complexity to the CPU's software.

In order to use polling with the 8292 one must enable TCI but not connect the pin to the CPU's interrupt pin. TCI must be readable by some means. In this application example it is connected to bit 1 port 6FH on the ZT7488/18. In addition, the other three 8292 interrupt lines and the 8291 interrupt are also on that port (SPI-Bit 2, $\overline{\text{IBFI}}$-Bit 4, OBFI-Bit 3, 8291 INT-Bit 0).

These drivers assume that only primary addresses will be used on the GPIB. To use secondary addresses, one must modify the test for valid talk/listen addresses (range macro) to include secondaries.

INIT	INITIALIZATION
Talker/Listener	
SEND	SEND DATA
RECV	RECEIVE DATA
XFER	TRANSFER DATA
Controller	
TRIG	GROUP EXECUTE TRIGGER
DCLR	DEVICE CLEAR
SPOL	SERIAL POLL
PPEN	PARALLEL POLL ENABLE
PPDS	PARALLEL POLL DISABLE
PPUN	PARALLEL POLL UNCONFIGURE
PPOL	PARALLEL POLL
PCTL	PASS CONTROL
RCTL	RECEIVE CONTROL
SRQD	SERVICE REQUESTED
System Controller	
REME	REMOTE ENABLE
LOCL	LOCAL
IFCL	ABORT/INTERFACE CLEAR

Figure 14. Software Driver Routines

INITIALIZATION

8292 — Comes up in Controller Active State when SYC is TRUE. The only initialization needed is to enable the TCI interrupt mask. This is done by writing 0A0H to Port 68H.

8291 — Disable both the major and minor addresses because the 8291 will never see the 8292's commands/addresses (refer to earlier hardware discussion). This is done by writing 60H and 0E0H to Port 66H.

Set Address Mode to Talk-only by writing 80H to Port 64H.

Set internal counter to 3 MHz to match the clock input coming from the 8085 by writing 23H to Port 65H. High speed mode for the handshakes will not be used here even though the hardware uses three-state drivers.

No interrupts will be enabled now. Each routine will enable the ones it needs for ease of polling operation. The INT bit may be read through Port 6FH. Clear both interrupt mask registers.

Release the chip's initialization state by writing 0 to Port 65H.

```
INIT:
  Enable-8292                          ;Set up Int. pins for Port 6FH
    Enable TCI                         ;Task complete must be on
  Enable-8291
    Disable major address              ;In controller usage, the 8291
    Disable minor address              ;Is set to talk only and/or listen only
    ton                                ;Talk only is our rest state
    Clock frequency                    ;3 MHz in this ap note example
    All interrupts off
    Immediate execute pon              ;Releases 8291 from init. state
```

TALKER/LISTENER ROUTINES

Send Data

SEND <listener list pointer> <count> <EOS> <data buffer pointer>

This system command sends data from the CPU to one or more devices. The data is usually a string of ASCII characters, but may be binary or other forms as well. The data is device-specific.

My Talk Address (MTA) must be output to satisfy the GPIB requirement of only one talker at a time (any other talker will stop when MTA goes out). The MTA is not needed as far as the 8291 is concerned — it will be put into talk-only mode (ton).

This routine assumes a non-null listener list in that it always sends Universal Unlisten. If it is desired to send data to the listeners previously addressed, one could add a check for a null list and not send UNL. Count must be 255 or less due to an 8 bit register. This routine also always uses an EOS character to terminate the string output; this could easily be eliminated and rely on the count. Items in brackets ( ) are optional and will not be included in the actual code in Appendix A.

```
SEND:
  Output-to-8291 MTA, UNL              ;We will talk, nobody listen
  Put EOS into 8291                    ;End of string compare character
  While 20H ≤ listener ≤ 3EH           ;GPIB listen addresses are
    output-to-8291 listener            ;"space" thru " >" ASCII
    Increment listen list pointer      ;Address all listeners
  Output-to-8292 GTSB                  ;8292 stops asserting ATN, go to standby
  Enable-8291
    Output EOI on EOS sent             ;Send EOI along with EOS character
  If count < > 0 then
    While not (end or count = 0)       ;Wait for EOS or end of count
    (could check tout 2 here)          ;Optionally check for stuck bus-tout 2
      Output-to-8291 data              ;Output all data, one byte at a time
      Increment data buffer pointer    ;8085 CREG will count for us
      Decrement count
  Output-to-8292 TCSY                  ;8292 asserts ATN, take control sync.
  (If tout3 then take control async)   ;If unable to take control sync.
  Enable 8291                          ;Restore 8291 to standard condition
    No output EOI on EOS sent
  Return
```

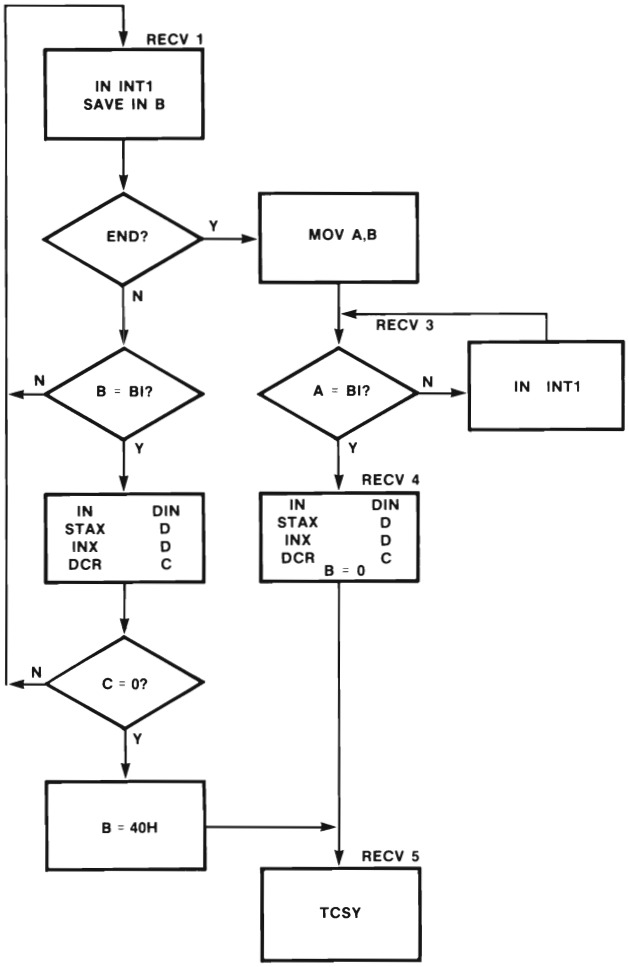


Figure 15. Flowchart For Receive Ending Conditions

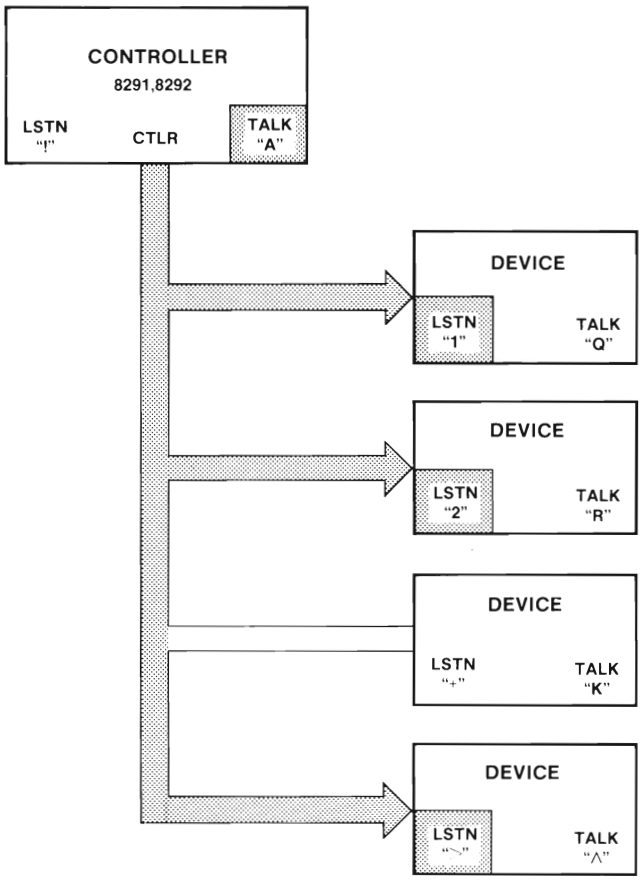


Figure 16. SEND to "1", "2", ">"; "ABCD"; EOS = "D"

Receive Data

RECV <talker> <count> <EOS> <data buffer pointer>

This system command is used to input data from a device. The data is typically a string of ASCII characters.

This routine is the dual of SEND. It assumes a new talker will be specified, a count of less than 257, and an EOS character to terminate the input. EOI received will also terminate the input. Figure 15 shows the flow chart for the RECV ending conditions. My Listen Address (MLA) is sent to keep the GPIB transactions totally regular to facilitate analysis by a GPIB logic analyzer like the Ziatech ZT488. Otherwise, the bus would appear to have no listener even though the 8291 will be listening.

Note that although the count may go to zero before the transmission ends, the talker will probably be left in a strange state and may have to be cleared by the controller. The count ending of RECV is therefore used as an error condition in most situations.

```
RECV:
  Put EOS into 8291                          ;End of string compare character
  If 40H ≤ talker ≤ 5EH then                 ;GPIB talk addresses are
    Output-to-8291 talker                    ;"@" thru "^" ASCII
  Increment talker pointer                   ;Do this for consistency's sake
  Output-to-8291 UNL, MLA                    ;Everyone except us stop listening
  Enable-8291
    Holdoff on end                           ;Stop when EOS character is
    End on EOS received                      ;Detected by 8291
    lon, reset ton                           ;Listen only (no talk)
    Immediate execute pon
  Output-to-8292 GTSB                        ;8292 stops asserting ATN, go to standby
  While not (end or count = 0 (or tout2))    ;wait for EOS or EOI or end of count
                                             ;optionally check for stuck bus-tout2
    Input-from-8291 data                     ;input data, one byte at a time
    Increment data buffer pointer
    Decrement count                          ;Use 8085 C register as counter
  (If count = 0 then error)                  ;Count should not occur before end
  Output-to-8292 TCSY                        ;8292 asserts ATN take control
  (If Tout3 then take control async.)        ;If unable to take control sync.
  Enable-8291                                ;Put 8291 back as needed for
    No holdoff on end                        ;Controller activity and
    No end on EOS received                   ;Clear holdoff due to end
    ton, reset lon
    Finish handshake                         ;Complete holdoff due to end, if any
    Immediate execute pon                    ;Needed to reset lon
  Return error-indicator
```

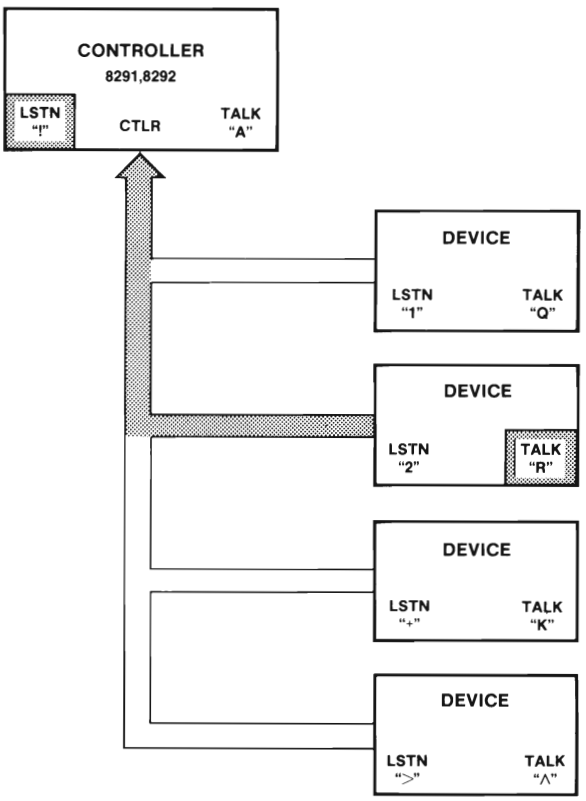


Figure 17. RECV from "R"; EOS = 0DH

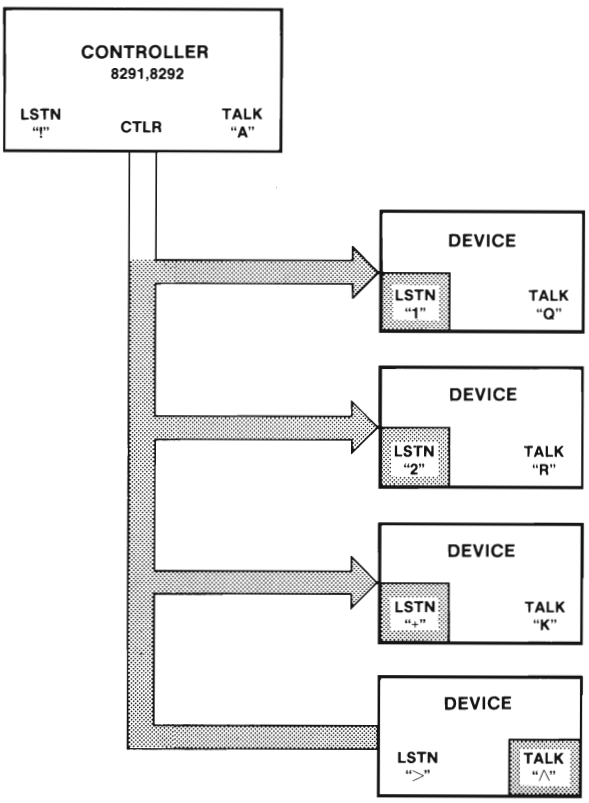


Figure 18. XFER from "^" to "1", "2", "+"; EOS = 0DH

Transfer Data

XFER < Talker > < Listener list > < EOS >

This system command is used to transfer data from a talker to one or more listeners where the controller does not participate in the transfer of the ASCII data. This is accomplished through the use of the 8291's continuous acceptor handshake mode while in listen-only.

This routine assumes a device list that has the ASCII talker address as the first byte and the string (one or more) of ASCII listener addresses following. The EOS character or an EOI will cause the controller to take control synchronously and thereby terminate the transfer.

```
XFER:
  Output-to-8291: Talker, UNL          ;Send talk address and unlisten
  While 20H ≤ listen ≤ 3EH
    Output-to-8291: Listener           ;Send listen address
    Increment listen list pointer
  Enable-8291
    lon, no ton                        ;Controller is pseudo listener
    Continuous AH mode                 ;Handshake but don't capture data
    End on EOS received                ;Capture EOS as well as EOI
    Immediate execute PON              ;Initialize the 8291
  Put EOS into 8291                    ;Set up EOS character
  Output-to-8292: GTSB                 ;Go to standby
                                       ;8292 waits for EOS or EOI and then
  Upon end (or tout2) then
    Take control synchronously         ;Regains control
  Enable-8291                          ;Go to Ready for Data
    Finish handshake
    Not continuous AH mode
    Not END on EOS received
    ton
    Immediate execute pon
  Return
```

CONTROLLER

Group Execute Trigger

TRIG < Listener list >

This system command causes a group execute trigger (GET) to be sent to all devices on the listener list. The intended use is to synchronize a number of instruments.

```
TRIG:
  Output-to-8291 UNL                   ;Everybody stop listening
  While 20H ≤ listener ≤ 3EH           ;Check for valid listen address
    Output-to-8291 Listener            ;Address each listener
    Increment listen list pointer      ;Terminate on any non-valid character
  Output-to-8291 GET                   ;Issue group execute trigger
  Return
```

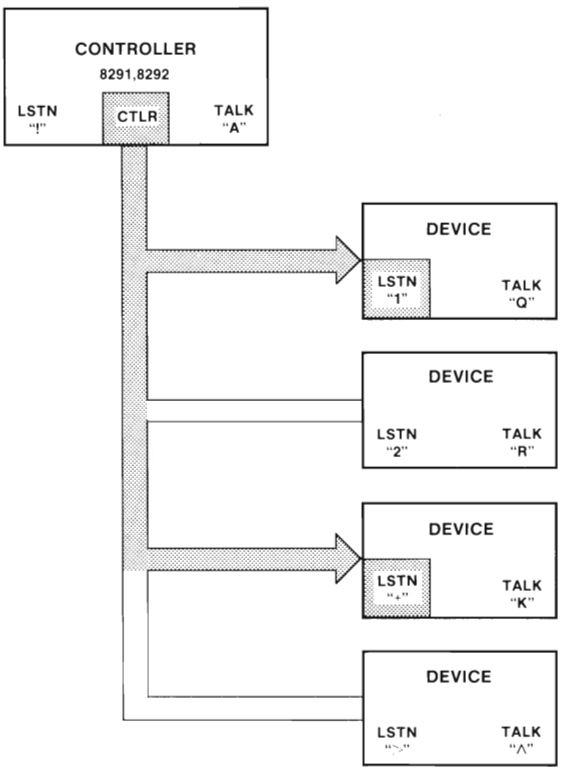


Figure 19. TRIG "1", "+"

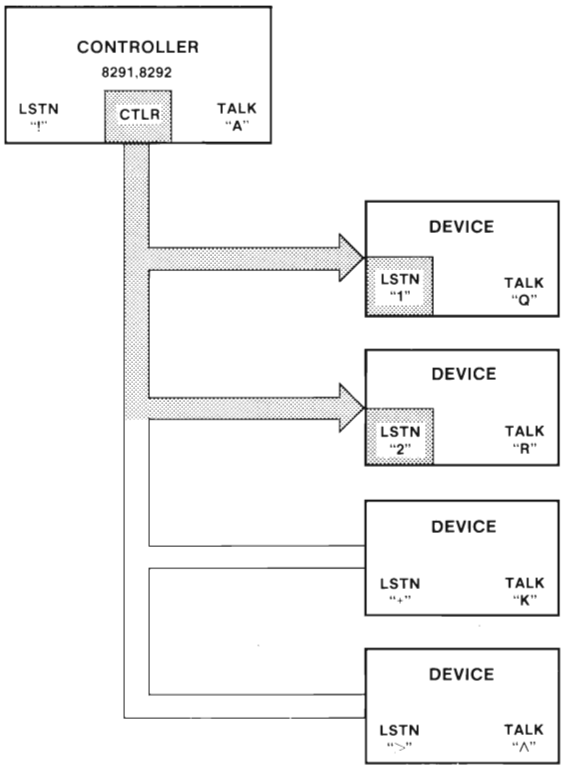


Figure 20. DCLR "1", "2"

Device Clear

DCLR < Listener list >

This system command causes a device clear (SDC) to be sent to all devices on the listener list. Note that this is not intended to clear the GPIB interface of the device, but should clear the device-specific logic.

```
DCLR:
  Output-to-8291 UNL                    ;Everybody stop listening
  While 20H ≤ Listener ≤ 3EH            ;Check for valid listen address
    Output-to-8291 listener             ;Address each listener
    Increment listen list pointer       ;Terminate on any non-valid character
  Output-to-8291 SDC                    ;Selective device clear
  Return
```

Serial Poll

SPOL < Talker list > < status buffer pointer >

This system command sequentially addresses the designated devices and receives one byte of status from each. The bytes are stored in the buffer in the same order as the devices appear on the talker list. MLA is output for completeness.

SPOL:

```
Output-to-8291 UNL, MLA, SPE        ;Unlisten, we listen, serial poll enable
                                    ;Only one byte of serial poll
                                    ;Status wanted from each talker
While 40H ≤ talker ≤ 5EH            ;Check for valid transfer
  Output-to-8291 talker             ;Address each device to talk
  Increment talker list pointer     ;One at a time
  Enable-8291
    lon, reset ton                  ;Listen only to get status
    Immediate execute pon           ;This resets ton
  Output-to-8292 GTSB               ;Go to standby
  Wait for data in (BI)             ;Serial poll status byte into 8291
  Output-to-8292 TCSY               ;Take control synchronously
  Input-from-8291 data              ;Actually get data from 8291
  Increment buffer pointer
  Enable 8291
    ton, reset lon
    Immediate execute pon           ;Resets lon
Output-to-8291 SPD                  ;Send serial poll disable after all devices polled
Return
```

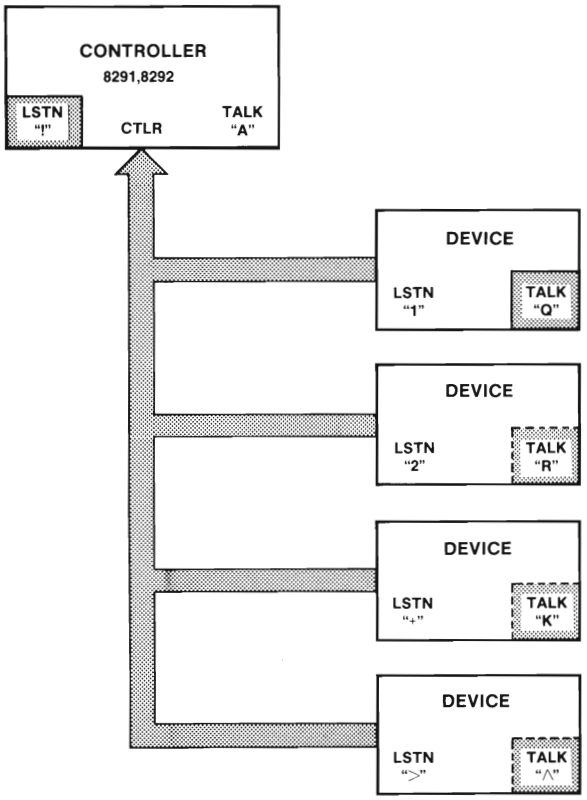


Figure 21. SPOL "Q", "R", "K", " ∧"

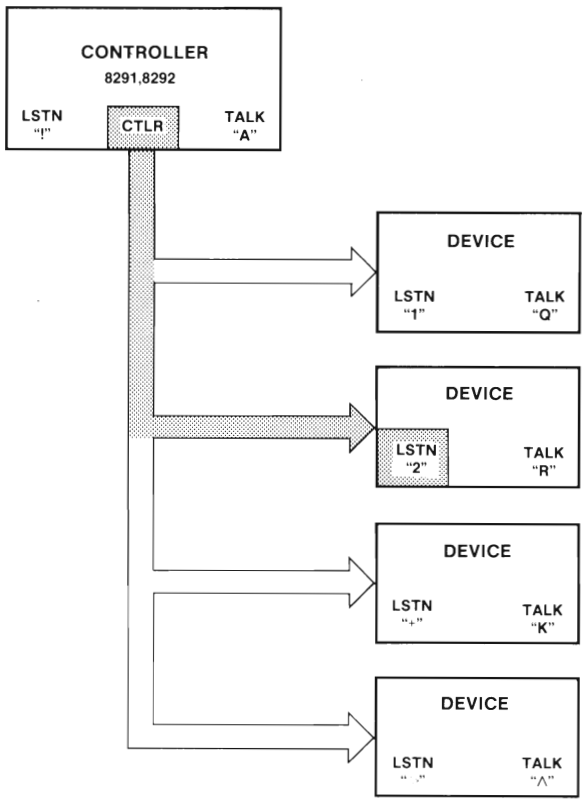


Figure 22. PPEN "2"; $iP_3P_2P_1$ = 0111B

Parallel Poll Enable

PPEN <Listener list> <Configuration Buffer pointer>

This system command configures one or more devices to respond to Parallel Poll, assuming they implement subset PP1. The configuration information is stored in a buffer with one byte per device in the same order as devices appear on the listener list. The configuration byte has the format XXXXIP3P2P1 as defined by the IEEE Std. P3P2P1 indicates the bit # to be used for a response and I indicates the assertion value. See Sec. 2.9.3.3 of the Std. for more details.

```
PPEN:
  Output-to-8291 UNL                    ;Universal unlisten
  While 20H ≤ Listener ≤ 3EH            ;Check for valid listener
    Output-to-8291 listener             ;Stop old listener, address new
    Output-to-8291 PPC, (PPE or data)   ;Send parallel poll info
    Increment listener list pointer     ;Point to next listener
    Increment buffer pointer            ;One configuration byte per listener
  Return
```

Parallel Poll Disable

PPDS < listener list >

This system command disables one or more devices from responding to a Parallel Poll by issuing a Parallel Poll Disable (PPD). It does not deconfigure the devices.

```
PPDS:
  Output-to-8291 UNL                    ;Universal Unlisten
  While 20H ≤ Listener ≤ 3EH            ;Check for valid listener
    Output-to-8291 listener             ;Address listener
    Increment listener list pointer
  Output-to-8291 PPC, PPD               ;Disable PP on all listeners
  Return
```

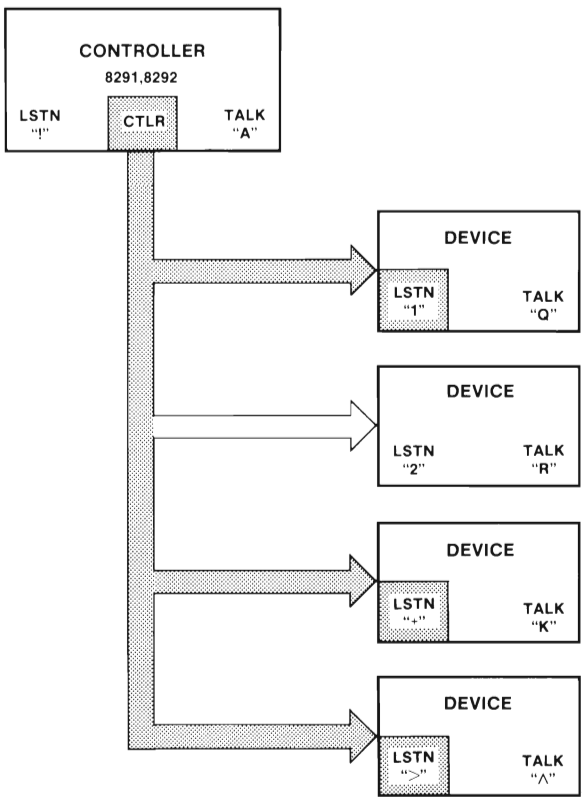


Figure 23. PPDS "1", "+", ">"

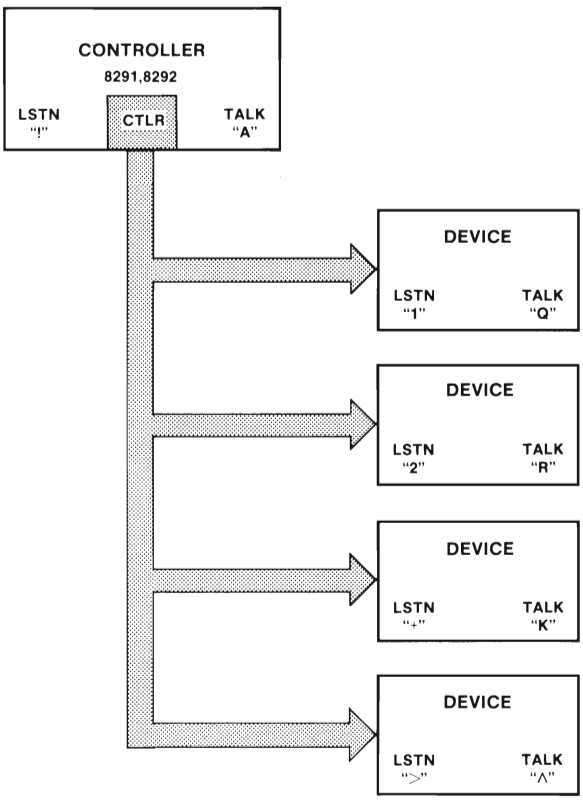


Figure 24. PPUN

Parallel Poll Unconfigure

PPUN

This system command deconfigures the Parallel Poll response of all devices by issuing a Parallel Poll Unconfigure message.

```
PPUN:
  Output-to-8291 PPU                ;Unconfigure all parallel poll
  Return
```

Conduct a Parallel Poll

PPOL

This system command causes the controller to conduct a Parallel Poll on the GPIB for approximately 12.5 usec (at 6 MHz). Note that a parallel poll does not use the handshake; therefore, to ensure that the device knows whether or not its positive response was observed by the controller, the CPU should explicitly acknowledge each device by a device-dependent data string. Otherwise, the response bit will still be set when the next Parallel Poll occurs. This command returns one byte of status.

```
PPOL:
  Enable-8291
    lon                             ;Listen only
    Immediate execute pon           ;This resets ton
  Output-to-8292 EXPP               ;Execute parallel poll
  Upon BI                           ;When byte is input
    Input-from-8291 data            ;Read it
  Enable-8291
    ton                             ;Talk only
    Immediate execute pon           ;This resets lon
  Return Data (status byte)
```

Pass Control

PCTL <talker>

This system command allows the controller to relinquish active control of the GPIB to another controller. Normally some software protocol should already have informed the controller to expect this, and under what conditions to return control. The 8291 must be set up to become a normal device and the CPU must handle all commands passed through, otherwise control cannot be returned (see Receive Control below). The controller will go idle.

```
PCTL:
  If 40H ≤ talker ≤ 5EH then
    if talker <  > MTA then         ;Cannot pass control to myself
      output-to-8291 talker, TCT    ;Take control message to talker
      Enable-8291                   ;Set up 8291 as normal device
        not ton, not lon
        Immediate execute pon       ;Reset ton and lon
        My device address, mode 1   ;Put device number in Register 6
        Undefined command pass through ;Required to receive control
        (Parallel Poll Configuration) ;Optional use of PP
      Output-to-8292 GIDL           ;Put controller in idle
  Return
```

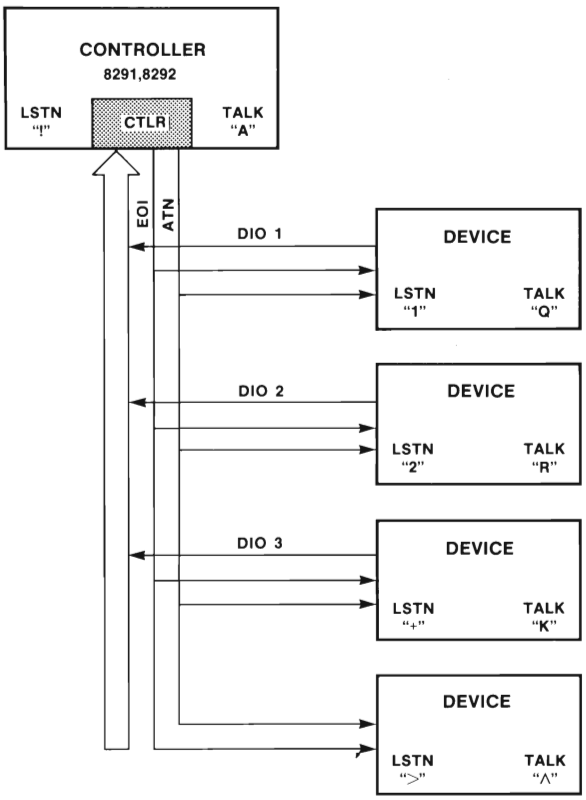


Figure 25. PPOL

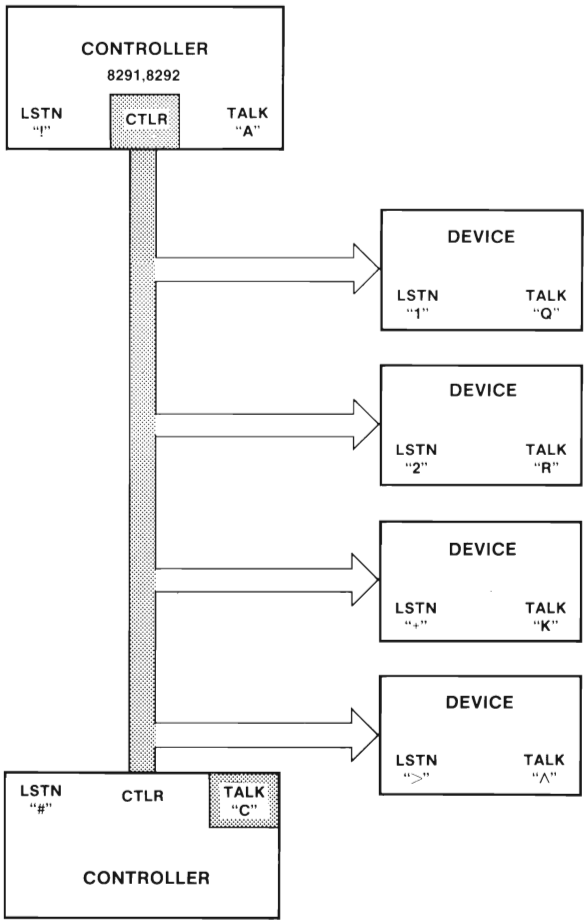


Figure 26. PCTL "C"

Receive Control

RCTL

This system command is used to get control back from the current controller-in-charge if it has passed control to this inactive controller. Most GPIB systems do not use more than one controller and therefore would not need this routine.

To make passing and receiving control a manageable event, the system designer should specify a protocol whereby the controller-in-charge sends a data message to the soon-to-be-active controller. This message should give the current state of the system, why control is being passed, what to do, and when to pass control back. Most of these issues are beyond the scope of this Ap Note.

```
RCTL:
  Upon CPT                          ;Wait for command pass through bit in 8291
    If (command=TCT) then           ;If command is take control and
      If TA then                    ;We are talker addressed
        Enable-8291

          Disable major device number   ;Controller will use ton and lon
          ton                           ;Talk only mode
          Mask off interrupts
          Immediate execute pon
```

```
        Output-to-8292 TCNTR          ;Take (receive) control
        Enable-8291
          Valid command               ;Release handshake
        Return valid
      Else
        Enable-8291
          Invalid command             ;Not talker addr. so TCT not for us
    Else
      Enable-8291
        Invalid command               ;Not TCT, so we don't care
  Return invalid
```

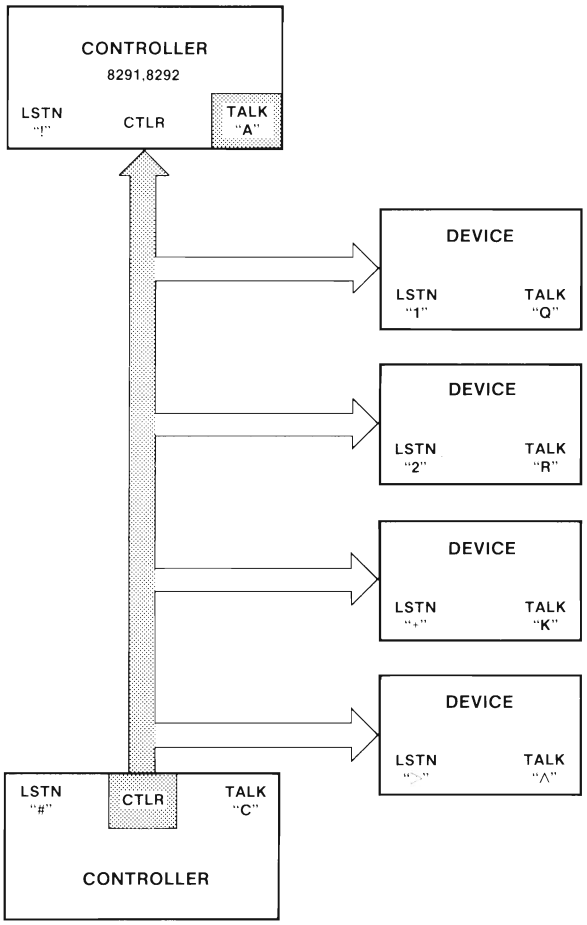


Figure 27. RCTL

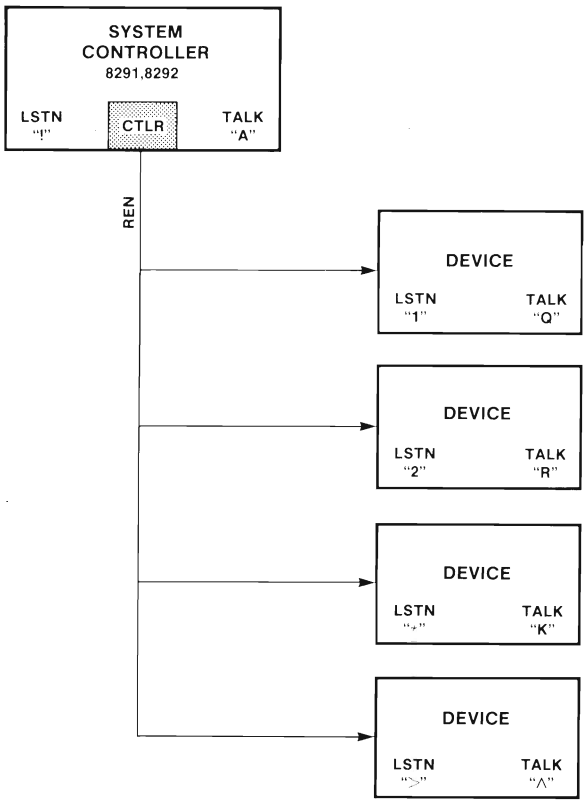


Figure 28. REME

Service Request

SRQD

This system command is used to detect the occurrence of a Service Request on the GPIB. One or more devices may assert SRQ simultaneously, and the CPU would normally conduct a Serial Poll after calling this routine to determine which devices are SRQing.

```
SRQD:
  If SRQ then                          ;Test 92 status bit
    Output-to-8292 IACK.SRQ            ;Acknowledge it
    Return SRQ
  Else return no SRQ
```

SYSTEM CONTROLLER

Remote Enable

REME

This system command asserts the Remote Enable line (REN) on the GPIB. The devices will not go remote until they are later addressed to listen by some other system command.

```
REME:
  Output-to-8292 SREM                  ;8292 asserts remote enable line
  Return
```

Local

LOCL

This system command deasserts the REN line on the GPIB. The devices will go local immediately.

```
LOCL:
  Output-to-8292 SLOC                  ;8292 stops asserting remote enable
  Return
```

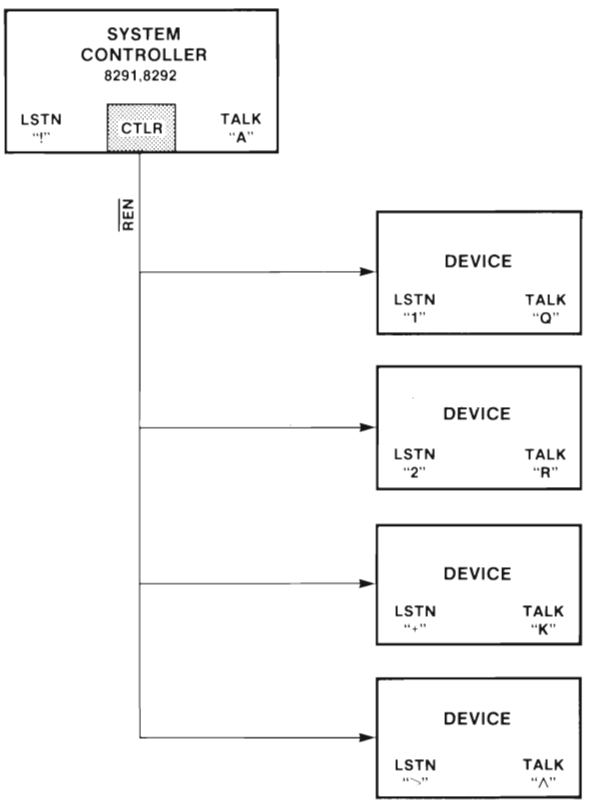


Figure 29. LOCL

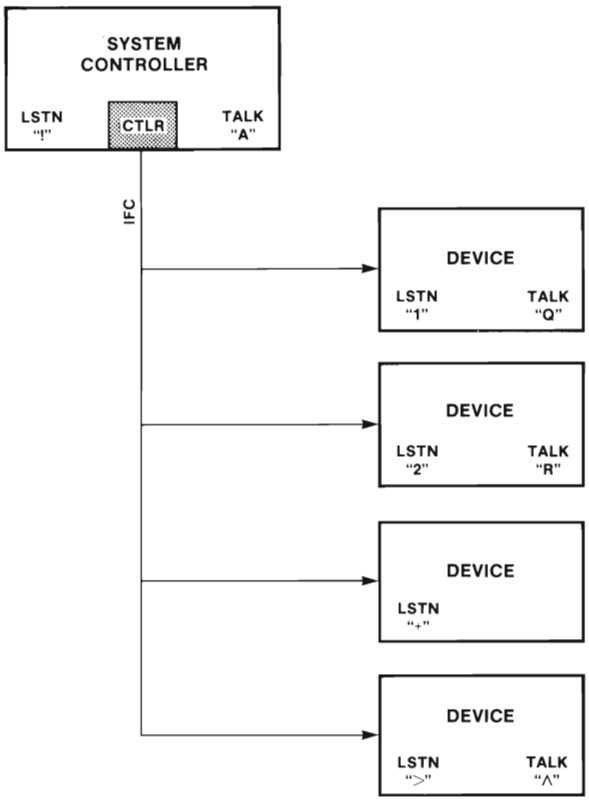


Figure 30. IFCL

Interface Clear/Abort

IFCL

This system command asserts the GPIB's Interface Clear (IFC) line for at least 100 microseconds. This causes all interface logic in all devices to go to a known state. Note that the device itself may or may not be reset, too. Most instruments do totally reset upon IFC. Some devices may require a DCLR as well as an IFCL to be completely reset. The (system) controller becomes Controller-in-Charge.

```
IFCL:
    Output-to-8292 ABORT        ;8292 asserts Interface Clear
    Return                      ;For 100 microseconds
```

INTERRUPTS AND DMA CONSIDERATIONS

The previous sections have discussed in detail how to use the 8291, 8292, 8293 chip set as a GPIB controller with the software operating in a polling mode and using programmed transfer of the data. This is the simplest mode of use, but it ties up the microprocessor for the duration of a GPIB transaction. If system design constraints do not allow this, then either Interrupts and/or DMA may be used to free up processor cycles.

The 8291 and 8292 provide sufficient interrupts that one may return to do other work while waiting for such things as 8292 Task Completion, 8291 Next Byte In, 8291 Last Byte Out, 8292 Service Request In, etc. The only difficulty lies in integrating these various interrupt sources and their matching routines into the overall system's interrupt structure. This is highly situation-specific and is beyond the scope of this Ap Note.

The strategy to follow is to replace each of the WAIT routines (see Appendix A) with a return to the main code and provide for the corresponding interrupt to bring the control back to the next section of GPIB code. For example WAITO (Wait for Byte Out of 8291) would be replaced by having the BO interrupt enabled and storing the (return) address of the next instruction in a known place. This co-routine structure will then be activated by a BO interrupt. Fig. 31 shows an example of the flow of control.

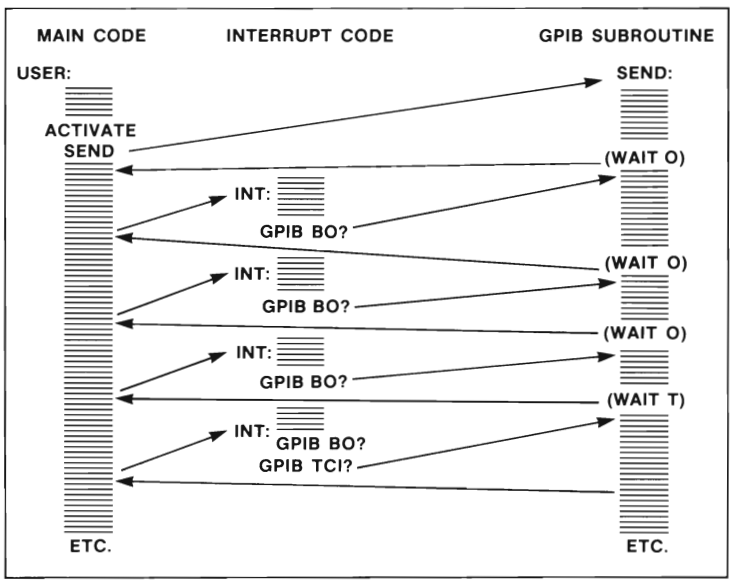


Figure 31. GPIB Interrupt & Co-Routine Flow of Control

DMA is also useful in relieving the processor if the average length of a data buffer is long enough to overcome the extra time used to set up a DMA chip. This decision will also be a function of the data rate of the instrument. The best strategy is to use the DMA to handle only the data buffer transfers on SEND and RECV and to do all the addressing and control just as shown in the driver descriptions.

Another major reason for using a DMA chip is to increase the data rate and therefore increase the overall transaction rate. In this case the limiting factor becomes the time used to do the addressing and control of the GPIB using software like that in Appendix A. The data transmission time becomes insignificant at DMA speeds unless extremely long buffers are used.

Refer to Figure 11 for a typical DMA and interrupt based design using the 8291, 8292, 8293. A system like this can achieve a 250K byte transfer rate while under DMA control.

APPLICATION EXAMPLE

This section will present the code required to operate a typical GPIB instrument set up as shown in Fig. 32. The HP5328A universal counter and the HP3325 function generator are typical of many GPIB devices; however, there are a wide variety of software protocols to be found on the GPIB. The Ziatech ZT488 GPIB analyzer is used to single step the bus to facilitate debugging the system. It also serves as a training/familiarization aid for newcomers to the bus.

This example will set up the function generator to output a specific waveform, frequency and amplitude. It will then tell the counter to measure the frequency and Request Service (SRQ) when complete. The program will then read in the data. The assembled source code will be found at the end of Appendix A.

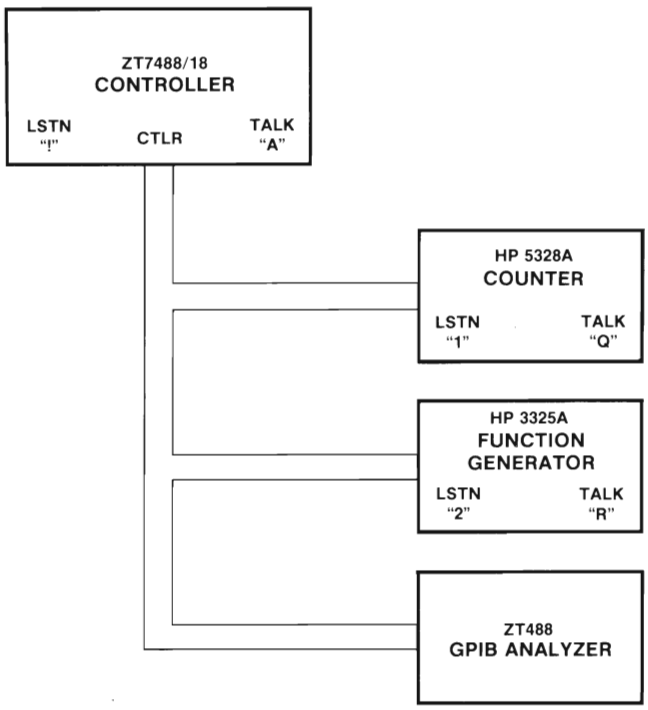


Figure 32. GPIB Example Configuration

```
SEND
  LSTN: "2", COUNT: 15, EOS: 0DH, DATA: "FU1FR37KHAM2VO (CR)"
  ;SETS UP FUNCTION GEN. TO 37 KHZ SINE, 2 VOLTS PP
  ;COUNT EQUAL TO # CHAR IN BUFFER
  ;EOS CHARACTER IS (CR) = 0DH = CARRIAGE RETURN
SEND
  LSTN: "1", COUNT: 6, EOS: "T" DATA: "PR4G7T"
  ;SETS UP COUNTER FOR P:INITIALIZE, F4: FREQ CHAN A
  ;            G7:0.1 HZ RESOLUTION, T:TRIGGER AND SRQ
  ;COUNT IS EQUAL TO # CHAR
WAIT FOR SRQ
SPOL TALK: "Q", DATA: STATUS 1
  ;CLEARS THE SRO—IN THIS EXAMPLE ONLY FREQ CTR ASSERTS SRQ
RECV TALK: "Q", COUNT: 17, EOS: 0AH,
  DATA: " +      37000.0E+0" (CR) (LF)
  ;GETS 17 BYTES OF DATA FROM COUNTER
  ;COUNT IS EXACT BUFFER LENGTH
  ;DATA SHOWN IS TYPICAL HP5328A READING THAT WOULD BE RECEIVED
```

CONCLUSION

This Application Note has shown a structured way to view the IEEE 488 bus and has given typical code sequences to make the Intel 8291, 8292, and 8293's behave as a controller of the GPIB. There are other ways to use the chip set, but whatever solution is chosen, it must be integrated into the overall system software.

The ultimate reference for GPIB questions is the IEEE Std 488, -1978 which is available from IEEE, 345 East 47th St., New York, NY, 10017. The ultimate reference for the 8292 is the source listing for it (remember it's a pre-programmed UPI-41A) which is available from INSITE, Intel Corp., 3065 Bowers Ave., Santa Clara, CA 95051.

APPENDIX A

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0
GPIB CONTROLLER SUBROUTINES

LOC  OBJ          LINE        SOURCE STATEMENT

                     1 $TITLE('GPIB CONTROLLER SUBROUTINES')
                     2 ;
                     3 ; GPIB CONTROLLER SUBROUTINES
                     4
                     5 ;
                     6 ;       for Intel 8291, 8292 on ZT 7488/18
                     7 ;       Bert Forbes, Ziatech Corporation
                     8 ;       2410 Broad Street
                     9 ;       San Luis Obispo, CA, USA 93401
                    10 ;
                    11 ;
                    12 ;       General Definitions & Equates
                    13 ;       8291 Control Values
                    14 ;
1000                15         ORG     1000H   ; For ZT7488/18 w/8085
                    16 ;
0060                17 PRT91   EQU     60H     ;8291 Base Port #
                    18 ;
                    19 ;       Reg #0 Data in & Data out
0060                20 DIN     EQU     PRT91+0 ;91 Data in reg
0060                21 DOUT    EQU     PRT91+0 ;91 Data out reg
                    22 ;
                    23 ;       Reg # 1 Interrupt 1 Constants
0061                24 INT1    EQU     PRT91+1 ;INT Reg 1
0061                25 INTM1   EQU     PRT91+1 ;INT Mask Reg. 1
0002                26 BOM     EQU     02      ;91 BO INTRP Mask
0001                27 BIM     EQU     01      ;91 BI INTRP Mask
0010                28 ENDMK   EQU     10H     ;91 END INTRP Mask
0080                29 CPT     EQU     80H     ;91 command pass thru int bit
                    30 ;
                    31 ;       Reg #2 Interrupt 2
0062                32 INT2    EQU     PRT91+2
                    33 ;
                    34 ;       Reg #4 Address Mode Constants
0064                35 ADRMD   EQU     PRT91+4 ;91 address mode register #
0080                36 TON     EQU     80H     ;91 talk only mode & not listen only
0040                37 LON     EQU     40H     ;91 listen only & not ton
00C0                38 TLON    EQU     0C0H    ;91 talk & listen only
0001                39 MODE1   EQU     01      ;mode 1 addressing for device
                    40
                    41 ;       Reg #4          (Read)  Address Status Register
0064                42 ADRST   EQU     PRT91+4 ;reg #4
0020                43 EOIST   EQU     20H
0002                44 TA      EQU     2
0001                45 LA      EQU     1       ;listener active
                    46 ;
                    47 ;       Reg #5  (Write) Auxillary Mode Register
0065                48 AUXMD   EQU     PRT91+5 ;91 auxillary mode register #
0023                49 CLKRT   EQU     23H     ;91 3 Mhz clock input
```

```
0003              50 FNHSK   EQU     03      ;91 fininsh handshake command
0006              51 SDEOI   EQU     06      ;91 send EOI with next byte
0080              52 AXRA    EQU     80H     ;91 aux. reg A pattern
0001              53 HOHSK   EQU      1      ;91 hold off handshake on all bytes
0002              54 HOEND   EQU      2      ;91 hold off handshake on end
0003              55 CAHCY   EQU      3      ;91 continuous AH cycling
0004              56 EDEOS   EQU      4      ;91 end on EOS received
0008              57 EOIS    EQU      8      ;91 output EOI on EOS sent
000F              58 VSCMD   EQU     0FH     ;91 valid command pass through
0007              59 NVCMD   EQU     07H     ;91 invalid command pass through
00A0              60 AXRB    EQU     0A0H    ;Aux. reg. B pattern
0001              61 CPTEN   EQU     01H     ;command pass thru enable
                  62 ;
                  63 ;       Reg #5  (Read)
0065              64 CPTRG   EQU     PRT91+5
                  65 ;
                  66 ;       Reg #6  Address 0/1 reg. constants
0066              67 ADR01   EQU     PRT91+6
0060              68 DTDL1   EQU     60H     ;Disable major talker & listener
00E0              69 DTDL2   EQU     0E0H    ;Disable minor talker & listener
                  70 ;
                  71 ;       Reg #7  EOS     Character Register
0067              72 EOSR    EQU     PRT91+7
                  73 ;
                  74 ;
                  75 ;       8292    CONTROL VALUES
                  76 ;
                  77 ;
                  78 ;
0068              79 PRT92   EQU     PRT91+8 ;8292 Base Port # (CS7)
                  80 ;
0068              81 INTMR   EQU     PRT92+0 ;92 INTRP Mask Reg
00A0              82 INTM    EQU     0A0H    ;TCI
                  83 ;
0068              84 ERRM    EQU     PRT92+0 ;92 Error Mask Reg
0001              85 TOUT1   EQU     01      ;92 Time Out for Pass Control
0002              86 TOUT2   EQU     02      ;92 Time Out for Standby
0004              87 TOUT3   EQU     04      ;92 Time Out for Take Control Sync
0068              88 EVREG   EQU     PRT92+0 ;92 Event Counter Pseudo Reg
0068              89 TOREG   EQU     PRT92+0 ;92 Time Out Pseudo Reg
                  90 ;
0069              91 CMD92   EQU     PRT92+1 ;92 Command Register
                  92 ;
0069              93 INTST   EQU     PRT92+1 ;92 Interrupt Status Reg
0010              94 EVBIT   EQU     10H     ;Event Counter Bit
0002              95 IBFBT   EQU     02      ;Input Buffer Full Bit
0020              96 SRQBT   EQU     20H     ;Seq bit
                  97 ;
0068              98 ERFLG   EQU     PRT92+0 ;92 Error Flag Pseudo Reg
0068              99 CLRST   EQU     PRT92+0 ;92 Controller Status Pseudo Reg
0068             100 BUSST   EQU     PRT92+0 ;92 GPIB (Bus) Status Pseudo Reg
0068             101 EVCST   EQU     PRT92+0 ;92 Event Counter Status Pseudo Reg
0068             102 TOST    EQU     PRT92+0 ;92 Time Out Status Pseudo Reg
                 103 ;
                 104 ;       8292    OPERATION COMMANDS
                 105 ;
                 106 ;
00F0             107 SPCNI   EQU     0F0H    ;Stop Counter Interrupts
00F1             108 GIDL    EQU     0F1H    ;Go to idle
00F2             109 RSET    EQU     0F2H    ;Reset
00F3             110 RSTI    EQU     0F3H    ;Reset Interrupts
00F4             111 GSEC    EQU     0F4H    ;Goto standby, enable counting
00F5             112 EXPP    EQU     0F5H    ;Execute parallel poll
00F6             113 GTSB    EQU     0F6H    ;Goto standby
00F7             114 SLOC    EQU     0F7H    ;Set local mode
00F8             115 SREM    EQU     0F8H    ;Set interface to remote
00F9             116 ABORT   EQU     0F9H    ;Abort all operation, clear interface
00FA             117 TCNTR   EQU     0FAH    ;Take control (Receive control)
00FC             118 TCASY   EQU     0FCH    ;Take control asyncronously
00FD             119 TCSY    EQU     0FDH    ;Take control syncronously
00FE             120 STCNI   EQU     0FEH    ;Start counter interrupts
                 121 ;
                 122 ;
```

```
                123 ;       8292    UTILITY COMMANDS
                124 ;
                125 ;
00E1            126 WOUT    EQU     0E1H    ;Write to timeout reg
00E2            127 WEVC    EQU     0E2H    ;Write to event counter
00E3            128 REVC    EQU     0E3H    ;Read event counter status
00E4            129 RERF    EQU     0E4H    ;Read error flag reg
00E5            130 RINM    EQU     0E5H    ;Read interrupt mask reg
00E6            131 RCST    EQU     0E6H    ;Read controller status reg
00E7            132 RBST    EQU     0E7H    ;Read GPIB Bus status reg
00E9            133 RTOUT   EQU     0E9H    ;Read timeout status reg
00EA            134 RERM    EQU     0EAH    ;Read error mask reg
000B            135 IACK    EQU     0BH     ;Interrupt Acknowledge
                136 ;
                137 ;
                138 ;       PORT F BIT ASSIGNMENTS
                139 ;
                140 ;
                141 ;
006F            142 PRTF    EQU     PRT91+0FH       ;ZT7488 port 6F for interrupts
0002            143 TCIF    EQU     02H     ;Task complete interrupt
0004            144 SPIF    EQU     04H     ;Special interrupt
0008            145 OBFF    EQU     08H     ;92 Output (to CPU) Buffer full
0010            146 IBFF    EQU     10H     ;92 Input (from CPU) Buffer empty
0001            147 BOF     EQU     01H     ;91 Int line (BO in this case)
                148 ;
                149 ;       GPIB MESSAGES (COMMANDS)
                150 ;
0001            151 MDA     EQU     1       ;My device address is 1
0041            152 MTA     EQU     MDA+40H ;My talk address is 1 ("A")
0021            153 MLA     EQU     MDA+20H ;My listen address is 1 ("!")
003F            154 UNL     EQU     3FH     ;Universal unlisten
0008            155 GET     EQU     08      ;Group Execute Trigger
0004            156 SDC     EQU     04H     ;Device Clear
0018            157 SPE     EQU     18H     ;Serial poll enable
0019            158 SPD     EQU     19H     ;Serial poll disable
0005            159 PPC     EQU     05      ;Parallel poll configure
0070            160 PPD     EQU     70H     ;Parallel poll disable
0060            161 PPE     EQU     60H     ;Parallel poll disable
0015            162 PPU     EQU     15H     ;Parallel poll unconfigured
0009            163 TCT     EQU     09      ;Take control (pass control)
                164
                165 ;       MACRO DEFINITIONS
                166 ;
                167 ;
                168 ;
                169 SETF    MACRO           ;Sets flags on A register
-               170         ORA     A
                171         ENDM
                172 ;
                173 WAITO   MACRO           ;Wait for last 91 byte to be done
                174         LOCAL   WAITL
-               175 WAITL:  IN      INT1    ;Get Int1 status
-               176         ANI     BOM     ;Check for byte out
-               177         JZ      WAITL   ;If not, try again
                178         ENDM            ;until it is
                179 ;
                180 ;
                181 WAITI   MACRO           ;Wait for 91 byte to be input
                182         LOCAL   WAITL
-               183 WAITL:  IN      INT1    ;Get INT1 status
-               184         MOV     B,A     ;Save status in B
-               185         ANI     BIM     ;Check for byte in
-               186         JZ      WAITL   ;If not, just try again
                187         ENDM            ;until it is
                188 ;
                189 WAITX   MACRO           ;Wait for 92's TCI to go false
                190         LOCAL   WAITL
-               191 WAITL:  IN      PRTF
-               192         ANI     TCIF
-               193         JNZ     WAITL
                194         ENDM
                195 ;
```

```
                196 WAITT   MACRO
                197         LOCAL   WAITL
-               198 WAITL:  IN      PRTF    ;Get task complete int,etc.
-               199         ANI     TCIF    ;Mask it
-               200         JZ      WAITL   ;Wait for task to be complete
                201         ENDM
                202
                203 RANGE   MACRO   LOWER,UPPER,LABEL
                204                         ;Checks for value in range
                205                         ;branches to label if not
                206                         ;in range. Falls through if
                207                         ;lower <= ( (H)(L) ) <= upper.
                208                         ;Get next byte.
-               209         MOV     A,M
-               210         CPI     LOWER
-               211         JM      LABEL
-               212         CPI     UPPER+1
-               213         JP      LABEL
                214         ENDM
                215 ;
                216 CLRA    MACRO
-               217         XRA     A       ;A XOR A =0
                218         ENDM
                219 ;
                220 ;       All of the following routines have these common
                221 ;       assumptions about the state of the 8291 & 8292 upon entry
                222 ;       to the routine and will exit the routine in an identical state.
                223 ;
                224 ;
                225 ;       8291:   BO is or has been set,
                226 ;               All interrupts are masked off
                227 ;               TON mode, not LA
                228 ;               No holdoffs in effect or enabled
                229 ;               No holdoffs waiting for finish command
                230 ;
                231 ;       8292:   ATN asserted (active controller)
                232 ;               note: RCTL is an exception--- it expects
                233 ;               to not be active controller
                234 ;               Any previous task is complete & 92 is
                235 ;               ready to receive next command.
                236 ;       8085:   Pointer registers (DE,HL) end one
                237 ;               beyond last legal entry
                238 ;*******************************************************
                239 ;
                240 ;
                241 ;       INITIALIZATION ROUTINE
                242 ;
                243 ;INPUTS:          None
                244 ;OUTPUTS:         None
                245 ;CALLS:           None
                246 ;DESTROYS:        A,F
                247 ;
1000 3EA0       248 INIT:   MVI     A,INTM  ;Enable TCI
1002 D368       249         OUT     INTMR   ;Output to 92's intr. mask reg
1004 3E60       250         MVI     A,DTDL1 ;Disable major talker/listener
1006 D366       251         OUT     ADR01
1008 3EE0       252         MVI     A,DTDL2 ;Disable minor talker/listener
100A D366       253         OUT     ADR01
100C 3E80       254         MVI     A,TON   ;Talk only mode
100E D364       255         OUT     ADRMD
1010 3E23       256         MVI     A,CLKRT ;3 MHZ for delay timer
1012 D365       257         OUT     AUXMD
                258         CLRA
1014 AF         259+        XRA     A       ;A XOR A =0
1015 D361       260         OUT     INT1
1017 D362       261         OUT     INT2    ;Disable all 91 mask bits
1019 D365       262         OUT     AUXMD   ;Immediate execute PON
101B C9         263         RET
                264 ;
                265 ;*******************************************************
                266 ;
                267 ;
                268 ;       SEND ROUTINE
                269 ;
```

```
                    270 ;
                    271 ;
                    272 ;       INPUTS:         HL listener list pointer
                    273 ;                       DE data buffer pointer
                    274 ;                       C  count-- 0 will cause no data to be sent
                    275 ;                       b  EOS character-- software detected
                    276 ;       OUTPUTS:        none
                    277 ;       CALLS:          none
                    278 ;       DESTROYS:       A, C, DE, HL, F
                    279 ;
                    280 ;
                    281 ;
101C 3E41           282 SEND:   MVI     A,MTA   ;Send MTA to turn off any
101E D360           283         OUT     DOUT    ;previous talker
                    284         WAITO
1020 DB61           285+??0001:  IN     INT1    ;Get Intl status
1022 E602           286+        ANI     BOM     ;Check for byte out
1024 CA2010         287+        JZ      ??0001  ;If not, try again
1027 3E3F           288         MVI     A,UNL   ;Send universal unlisten
1029 D360           289         OUT     DOUT    ;to stop previous listeners
102B 78             290         MOV     A,B     ;Get EOS character
102C D367           291         OUT     EOSR    ;Output it to 8291
                    292                         ;while listener.....
                    293 SEND1:  RANGE   20H,3EH,SEND2   ;Check next listen address
                    294+                        ;Checks for value in range
                    295+                        ;branches to label if not
                    296+                        ;in range. Falls through if
                    297+                        ;lower <= ( (H)(L) ) <= upper.
                    298+                        ;Get next byte.
102E 7E             299+        MOV     A,M
102F FE20           300+        CPI     20H
1031 FA4710         301+        JM      SEND2
1034 FE3F           302+        CPI     3EH+1
1036 F24710         303+        JP      SEND2
                    304         WAITO                   ;Wait for previous listener sent
1039 DB61           305+??0002:  IN     INT1    ;Get Intl status
103B E602           306+        ANI     BOM     ;Check for byte out
103D CA3910         307+        JZ      ??0002  ;If not, try again
1040 7E             308         MOV     A,M             ;Get this listener
1041 D360           309         OUT     DOUT            ;Output to GPIB
1043 23             310         INX     H               ;Increment listener list pointer
1044 C32E10         311         JMP     SEND1           ;Loop till non-valid listener
                    312                                 ;Enable 91 ending conditions
                    313 SEND2:  WAITO           ;Wait for lstn addr accepted
1047 DB61           314+??0003:  IN     INT1    ;Get Intl status
1049 E602           315+        ANI     BOM     ;Check for byte out
104B CA4710         316+        JZ      ??0003  ;If not, try again
                    317                         ;WAITO required for early versions
                    318                         ;of 8292 to avoid GTSB before DAC
104E 3EF6           319         MVI     A,GTSB  ;Goto standby
1050 D369           320         OUT     CMD92   ;
1052 3E88           321         MVI     A,AXRA+EOIS     ;Send EOI with EOS character
1054 D365           322         OUT     AUXMD
                    323         WAITX           ;Wait for TCI to go false
1056 DB6F           324+??0004: IN      PRTF
1058 E602           325+        ANI     TCIF
105A C25610         326+        JNZ     ??0004
                    327         WAITT           ;Wait for TCI on GTSB
105D DB6F           328+??0005: IN      PRTF    ;Get task complete int,etc.
105F E602           329+        ANI     TCIF    ;Mask it
1061 CA5D10         330+        JZ      ??0005  ;Wait for task to be complete
                    331
                    332 ;       delete next 3 instructions to make count of 0=256
                    333 ;
1064 79             334         MOV     A,C     ;Get count
                    335         SETF            ;Set flags
1065 B7             336+        ORA     A
1066 CA8810         337         JZ      SEND6   ;If count=0, send no data
1069 1A             338 SEND3:  LDAX    D       ;Get data byte
106A D360           339         OUT     DOUT    ;Output to GPIB
106C B8             340         CMP     B       ;Test EOS ...this is faster
                    341                         ;and uses less code than using
                    342                         ;91's END or EOI bits
```

```
106D CA7F10          343          JZ      SEND5    ;If char = EOS , go finish
                     344 SEND4:   WAITO
1070 DB61            345+??0006:   IN     INT1     ;Get Intl status
1072 E602            346+         ANI     BOM      ;Check for byte out
1074 CA7010          347+         JZ      ??0006   ;If not, try again
1077 13              348          INX     D        ;Increment buffer pointer
1078 0D              349          DCR     C        ;Decrement count
1079 C26910          350          JNZ     SEND3    ;If count < > 0, go send
107C C38810          351          JMP     SEND6    ;Else go finish
107F 13              352 SEND5:   INX     D        ;for consistency
1080 0D              353          DCR     C        ; "       "
                     354          WAITO
                                                   ;This ensures that the standard entry
1081 DB61            355+??0007:   IN     INT1     ;Get Intl status
1083 E602            356+         ANI     BOM      ;Check for byte out
1085 CA8110          357+         JZ      ??0007   ;If not, try again
                     358                           ;assumptions for the next subroutine are met
1088 3EFD            359 SEND6:   MVI     A,TCSY   ;Take control syncronously
108A D369            360          OUT     CMD92
108C 3E80            361          MVI     A,AXRA   ;Reset send EOI on EOS
108E D365            362          OUT     AUXMD
                     363          WAITX            ;Wait for TCI false
1090 DB6F            364+??0008:  IN      PRTF
1092 E602            365+         ANI     TCIF
1094 C29010          366+         JNZ     ??0008
                     367          WAITT            ;Wait for TCI
1097 DB6F            368+??0009:  IN      PRTF     ;Get task complete int,etc.
1099 E602            369+         ANI     TCIF     ;Mask it
109B CA9710          370+         JZ      ??0009   ;Wait for task to be complete
109E C9              371          RET
                     372 ;*****************************************************************************
                     373 ;
                     374 ;        RECEIVE ROUTINE
                     375 ;
                     376 ;
                     377 ;INPUT:           HL talker pointer
                     378 ;                 DE data buffer pointer
                     379 ;                 C  count (max buffer size) 0 implies 256
                     380 ;                 B  EOS character
                     381 ;OUTPUT:          Fills buffer pointed at by DE
                     382 ;CALLS:           None
                     383 ;DESTROYS:        A, BC, DE, HL, F
                     384 ;
                     385 ;RETURNS:         A=0 normal termination--EOS detected
                     386 ;                 A=40 Error--- count overrun
                     387 ;                 A<40 or A>5EH Error--- bad talk address
                     388 ;
                     389 ;
109F 78              390 RECV:    MOV     A,B      ;Get EOS character
10A0 D367            391          OUT     EOSR     ;Output it to 91
                     392          RANGE   40H,5EH,RECV6
                     393+                          ;Checks for value in range
                     394+                          ;branches to label if not
                     395+                          ;in range. Falls through if
                     396+                          ;lower <= ( (H)(L) ) <= upper.
                     397+                          ;Get next byte.
10A2 7E              398+         MOV     A,M
10A3 FE40            399+         CPI     40H
10A5 FA3911          400+         JM      RECV6
10A8 FE5F            401+         CPI     5EH+1
10AA F23911          402+         JP      RECV6
                     403                           ;valid if 40H<= talk <=5EH
10AD D360            404          OUT     DOUT     ;Output talker to GPIB
10AF 23              405          INX     H        ;Incr pointer for consistency
                     406          WAITO
10B0 DB61            407+??0010:   IN     INT1     ;Get Intl status
10B2 E602            408+         ANI     BOM      ;Check for byte out
10B4 CAB010          409+         JZ      ??0010   ;If not, try again
10B7 3E3F            410          MVI     A,UNL    ;Stop other listeners
10B9 D360            411          OUT     DOUT
                     412          WAITO
10BB DB61            413+??0011:   IN     INT1     ;Get Intl status
10BD E602            414+         ANI     BOM      ;Check for byte out
10BF CABB10          415+         JZ      ??0011   ;If not, try again
```

```
10C2 3E21          416          MVI     A,MLA    ;For completeness
10C4 D360          417          OUT     DOUT
10C6 3E86          418          MVI     A,AXRA+HOEND+EDEOS      ;End when
10C8 D365          419          OUT     AUXMD    ;EOS or EOI & Holdoff
                   420          WAITO
10CA DB61          421+??0012:   IN     INT1     ;Get Intl status
10CC E602          422+         ANI     BOM      ;Check for byte out
10CE CACA10        423+         JZ      ??0012   ;If not, try again
10D1 3E40          424          MVI     A,LON    ;Listen only
10D3 D364          425          OUT     ADRMD
                   426          CLRA             ;Immediate XEQ PON
10D5 AF            427+         XRA     A        ;A XOR A =0
10D6 D365          428          OUT     AUXMD
10D8 3EF6          429          MVI     A,GTSB   ;Goto standby
10DA D369          430          OUT     CMD92
                   431          WAITX            ;Wait for TCI=0
10DC DB6F          432+??0013:  IN      PRTF
10DE E602          433+         ANI     TCIF
10E0 C2DC10        434+         JNZ     ??0013
                   435          WAITT            ;Wait for TCI=1
10E3 DB6F          436+??0014:  IN      PRTF     ;Get task complete int,etc.
10E5 E602          437+         ANI     TCIF     ;Mask it
10EA DB61          439 RECV1:   IN      INT1     ;Get 91 Int status (END &/or BI)
10EC 47            440          MOV     B,A      ;Save it in B for BI check later
10ED E610          441          ANI     ENDMK    ;Check for EOS or EOI
10EF C20511        442          JNZ     RECV2    ;Yes end--- go wait for BI
10F2 78            443          MOV     A,B      ;NO, retrieve status &
10F3 E601          444          ANI     BIM      ;check for BI
10F5 CAEA10        445          JZ      RECV1    ;NO, go wait for either END or BI
10F8 DB60          446          IN      DIN      ;YES, BI--- get data
10FA 12            447          STAX    D        ;Store it in buffer
10FB 13            448          INX     D        ;Increment buffer pointer
10FC 0D            449          DCR     C        ;Decrement counter
10FD C2EA10        450          JNZ     RECV1    ;If count < > 0 go back & wait
1100 0640          451          MVI     B,40H    ;Else set error indicator
1102 C31711        452          JMP     RECV5    ;And go take control
                   453 ;
1105 78            454 RECV2:   MOV     A,B      ;Retreive status
1106 E601          455 RECV3:   ANI     BIM      ;Check for BI
1108 C21011        456          JNZ     RECV4    ;If BI then go input data
110B DB61          457          IN      INT1     ;Else wait for last BI
110D C30611        458          JMP     RECV3    ;In loop
1110 DB60          459 RECV4:   IN      DIN      ;Get data byte
1112 12            460          STAX    D        ;Store it in buffer
1113 13            461          INX     D        ;Incr data pointer
1114 0D            462          DCR     C        ;Decrement count, but ignore it
1115 0600          463          MVI     B,0      ;Set normal completion indicators
                   464 ;
1117 3EFD          465 RECV5:   MVI     A,TCSY   ;Take control synchronously
1119 D369          466          OUT     CMD92
                   467          WAITX            ;Wait for TCI=0 (7 tcy)
111B DB6F          468+??0015:  IN      PRTF
111D E602          469+         ANI     TCIF
111F C21B11        470+         JNZ     ??0015
                   471          WAITT            ;Wait for TCI=1
1122 DB6F          472+??0016:  IN      PRTF     ;Get task complete int,etc.
1124 E602          473+         ANI     TCIF     ;Mask it
1126 CA2211        474+         JZ      ??0016   ;Wait for task to be complete
                   475 ;
                   476 ;if timeout 3 is to be checked, the above WAITT should
                   477 ;be omitted & the appropriate code to look for TCI or
                   478 ;TOUT3 inserted here.
                   479 ;
1129 3E80          480          MVI     A,AXRA   ;Pattern to clear 91 END conditions
112B D365          481          OUT     AUXMD    ;
112D 3E80          482          MVI     A,TON    ;This bit pattern already in "A"
112F D364          483          OUT     ADRMD    ;Output TON
1131 3E03          484          MVI     A,FNHSK  ;Finish handshake
1133 D365          485          OUT     AUXMD
                   486          CLRA
1135 AF            487+         XRA     A        ;A XOR A =0
1136 D365          488          OUT     AUXMD    ;Immediate execute PON-Reset LON
1138 78            489          MOV     A,B      ;Get completion character
1139 C9            490 RECV6:   RET
```

```
                491 ;
                492 ;*****************************************************************
                493 ;       XFER ROUTINE
                494 ;
                495 ;
                496 ;INPUTS:        HL device list pointer
                497 ;               B  EOS character
                498 ;OUTPUTS:       None
                499 ;CALLS:         None
                500 ;DESTROYS:      A, HL, F
                501 ;RETURNS:       A=0 normal, A < > 0 bad talker
                502 ;
                503 ;
                504 ;NOTE:          XFER will not work if the talker
                505 ;               uses EOI to terminate the transfer.
                506 ;               Intel will be making hardware
                507 ;               modifications to the 8291 that will
                508 ;               correct this problem. Until that time,
                509 ;               only EOS may be used without possible
                510 ;               loss of the last data byte transfered.
                511 XFER:   RANGE   40H,5EH,XFER4   ;Check for valid talker
                512+                       ;Checks for value in range
                513+                       ;branches to label if not
                514+                       ;in range. Falls through if
                515+                       ;lower <= ( (H)(L) ) <= upper.
                516+                       ;Get next byte.
113A 7E         517+        MOV     A,M
113B FE40       518+        CPI     40H
113D FABB11     519+        JM      XFER4
1140 FE5F       520+        CPI     5EH+1
1142 F2BB11     521+        JP      XFER4
1145 D360       522         OUT     DOUT    ;Send it to GPIB
1147 23         523         INX     H       ;Incr pointer
                524         WAITO
1148 DB61       525+??0017:  IN     INT1    ;Get Intl status
114A E602       526+        ANI     BOM     ;Check for byte out
114C CA4811     527+        JZ      ??0017  ;If not, try again
114F 3E3F       528         MVI     A,UNL   ;Universal unlisten
1151 D360       529         OUT     DOUT
                530 XFER1:  RANGE   20H,3EH,XFER2   ;Check for valid listener
                531+                       ;Checks for value in range
                532+                       ;branches to label if not
                533+                       ;in range. Falls through if
                534+                       ;lower <= ( (H)(L) ) <= upper.
                535+                       ;Get next byte.
1153 7E         536+        MOV     A,M
1154 FE20       537+        CPI     20H
1156 FA6C11     538+        JM      XFER2
1159 FE3F       539+        CPI     3EH+1
115B F26C11     540+        JP      XFER2
                541         WAITO
115E DB61       542+??0018:  IN     INT1    ;Get Intl status
1160 E602       543+        ANI     BOM     ;Check for byte out
1162 CA5E11     544+        JZ      ??0018  ;If not, try again
1165 7E         545         MOV     A,M     ;Get listener
1166 D360       546         OUT     DOUT
1168 23         547         INX     H       ;Incr pointer
1169 C35311     548         JMP     XFER1   ;Loop until non-valid listener
                549 XFER2:  WAITO
116C DB61       550+??0019:  IN     INT1    ;Get Intl status
116E E602       551+        ANI     BOM     ;Check for byte out
1170 CA6C11     552+        JZ      ??0019  ;If not, try again
1173 3E87       553         MVI     A,AXRA+CAHCY+EDEOS      ;Invisible handshake
1175 D365       554         OUT     AUXMD   ;Continuous AH mode
1177 3E40       555         MVI     A,LON   ;Listen only
1179 D364       556         OUT     ADRMD
                557         CLRA
117B AF         558+        XRA     A       ;A XOR A =0
117C D365       559         OUT     AUXMD   ;Immed. XEQ PON
117E 78         560         MOV     A,B     ;Get EOS
117F D367       561         OUT     EOSR    ;Output it to 91
1181 3EF6       562         MVI     A,GTSB  ;Go to standby
1183 D369       563         OUT     CMD92
```

```
                    564           WAITX
1185 DB6F           565+??0020:   IN      PRTF
1187 E602           566+          ANI     TCIF
1189 C28511         567+          JNZ     ??0020
                    568           WAITT           ;Wait for TCS
118C DB6F           569+??0021:   IN      PRTF      ;Get task complete int,etc.
118E E602           570+          ANI     TCIF      ;Mask it
1190 CA8C11         571+          JZ      ??0021    ;Wait for task to be complete
1193 DB61           572 XFER3:    IN      INT1      ;Get END status bit
1195 E610           573           ANI     ENDMK     ;Mask it
1197 CA9311         574           JZ      XFER3
119A 3EFD           575           MVI     A,TCSY    ;Take control syncronously
119C D369           576           OUT     CMD92
                    577           WAITX
119E DB6F           578+??0022:   IN      PRTF
11A0 E602           579+          ANI     TCIF
11A2 C29E11         580+          JNZ     ??0022
                    581           WAITT           ;Wait for TCI
11A5 DB6F           582+??0023:   IN      PRTF      ;Get task complete int,etc.
11A7 E602           583+          ANI     TCIF      ;Mask it
11A9 CAA511         584+          JZ      ??0023    ;Wait for task to be complete
11AC 3E80           585           MVI     A,AXRA    ;Not cont AH or END on EOS
11AE D365           586           OUT     AUXMD
11B0 3E03           587           MVI     A,FNHSK   ;Finish handshake
11B2 D365           588           OUT     AUXMD
11B4 3E80           589           MVI     A,TON     ;Talk only
11B6 D364           590           OUT     ADRMD
                    591           CLRA            ;Normal return A=0
11B8 AF             592+          XRA     A         ;A XOR A =0
11B9 D365           593           OUT     AUXMD     ;Immediate XEQ PON
11BB C9             594 XFER4:    RET
                    595 ;
                    596 ;*************************************************
                    597 ;
                    598 ;
                    599 ;         TRIGGER ROUTINE
                    600 ;
                    601 ;
                    602 ;INPUTS:          HL listener list pointer
                    603 ;OUTPUTS:         None
                    604 ;CALLS:           None
                    605 ;DESTROYS:        A, HL, F
                    606 ;
                    607 ;
11BC 3E3F           608 TRIG:     MVI     A,UNL     ;
11BE D360           609           OUT     DOUT      ;Send universal unlisten
                    610 TRIG1:    RANGE   20H,3EH,TRIG2   ;Check for valid listen
                    611+                                  ;Checks for value in range
                    612+                                  ;branches to label if not
                    613+                                  ;in range. Falls through if
                    614+                                  ;lower <= ( (H)(L) ) <= upper.
                    615+                                  ;Get next byte.
11C0 7E             616+          MOV     A,M
11C1 FE20           617+          CPI     20H
11C3 FAD911         618+          JM      TRIG2
11C6 FE3F           619+          CPI     3EH+1
11C8 F2D911         620+          JP      TRIG2
                    621           WAITO           ;Wait for UNL to finish
11CB DB61           622+??0024:    IN     INT1      ;Get Intl status
11CD E602           623+          ANI     BOM       ;Check for byte out
11CF CACB11         624+          JZ      ??0024    ;If not, try again
11D2 7E             625           MOV     A,M       ;Get listener
11D3 D360           626           OUT     DOUT      ;Send Listener to GPIB
11D5 23             627           INX     H         ;Incr. pointer
11D6 C3C011         628           JMP     TRIG1     ;Loop until non-valid char
                    629 TRIG2:    WAITO           ;Wait for last listen to finish
11D9 DB61           630+??0025:    IN     INT1      ;Get Intl status
11DB E602           631+          ANI     BOM       ;Check for byte out
11DD CAD911         632+          JZ      ??0025    ;If not, try again
11E0 3E08           633           MVI     A,GET     ;Send group execute trigger
11E2 D360           634           OUT     DOUT      ;to all addressed listeners
                    635           WAITO
11E4 DB61           636+??0026:    IN     INT1      ;Get Intl status
11E6 E602           637+          ANI     BOM       ;Check for byte out
```

```
11E8 CAE411        638+          JZ       ??0026  ;If not, try again
11EB C9            639           RET
                   640 ;
                   641 ;*****************************************
                   642 ;
                   643 ;DEVICE CLEAR ROUTINE
                   644 ;
                   645 ;
                   646 ;
                   647 ;INPUTS:          HL listener pointer
                   648 ;OUTPUT:          None
                   649 ;CALLS:           None
                   650 ;DESTROYS:        A, HL, F
                   651 ;
11EC 3E3F          652 DCLR:     MVI      A,UNL
11EE D360          653           OUT      DOUT
                   654 DCLR1:    RANGE    20H,3EH,DCLR2
                   655+                            ;Checks for value in range
                   656+                            ;branches to label if not
                   657+                            ;in range. Falls through if
                   658+                            ;lower <= ( (H)(L) ) <= upper.
                   659+                            ;Get next byte.
11F0 7E            660+          MOV      A,M
11F1 FE20          661+          CPI      20H
11F3 FA0912        662+          JM       DCLR2
11F6 FE3F          663+          CPI      3EH+1
11F8 F20912        664+          JP       DCLR2
                   665           WAITO
11FB DB61          666+??0027:    IN      INT1     ;Get Intl status
11FD E602          667+          ANI      BOM      ;Check for byte out
11FF CAFB11        668+          JZ       ??0027   ;If not, try again
1202 7E            669           MOV      A,M
1203 D360          670           OUT      DOUT     ;Send listener to GPIB
1205 23            671           INX      H
1206 C3F011        672           JMP      DCLR1
                   673 DCLR2:    WAITO
1209 DB61          674+??0028:    IN      INT1     ;Get Intl status
120B E602          675+          ANI      BOM      ;Check for byte out
120D CA0912        676+          JZ       ??0028   ;If not, try again
1210 3E04          677           MVI      A,SDC    ;Send device clear
1212 D360          678           OUT      DOUT     ;To all addressed listeners
                   679           WAITO
1214 DB61          680+??0029:    IN      INT1     ;Get Intl status
1216 E602          681+          ANI      BOM      ;Check for byte out
1218 CA1412        682+          JZ       ??0029   ;If not, try again
121B C9            683           RET
                   684 ;
                   685 ;*************************************************
                   686 ;
                   687 ;         SERIAL POLL ROUTINE
                   688 ;
                   689 ;INPUTS:          HL talker list pointer
                   690 ;                 DE status buffer pointer
                   691 ;OUTPUTS:         Fills buffer pointed to by DE
                   692 ;CALLS:           None
                   693 ;DESTROYS:        A, BC, DE, HL, F
                   694 ;
121C 3E3F          695 SPOL:     MVI      A,UNL    ;Universal unlisten
121E D360          696           OUT      DOUT
                   697           WAITO
1220 DB61          698+??0030:    IN      INT1     ;Get Intl status
1222 E602          699+          ANI      BOM      ;Check for byte out
1224 CA2012        700+          JZ       ??0030   ;If not, try again
1227 3E21          701           MVI      A,MLA    ;My listen address
1229 D360          702           OUT      DOUT
                   703           WAITO
122B DB61          704+??0031:    IN      INT1     ;Get Intl status
122D E602          705+          ANI      BOM      ;Check for byte out
122F CA2B12        706+          JZ       ??0031   ;If not, try again
1232 3E18          707           MVI      A,SPE    ;Serial poll enable
1234 D360          708           OUT      DOUT     ;To be formal about it
                   709           WAITO
1236 DB61          710+??0032:    IN      INT1     ;Get Intl status
```

```
1238 E602          711+          ANI     BOM       ;Check for byte out
123A CA3512        712+          JZ      ??0032    ;If not, try again
                   713 SPOL1:    RANGE   40H,5EH,SPOL2   ;Check for valid talker
                   714+                                  ;Checks for value in range
                   715+                                  ;branches to label if not
                   716+                                  ;in range. Falls through if
                   717+                                  ;lower <= ( (H)(L) ) <= upper.
                   718+                                  ;Get next byte.
123D 7E            719+          MOV     A,M
123E FE40          720+          CPI     40H
1240 FA9412        721+          JM      SPOL2
1243 FE5F          722+          CPI     5EH+1
1245 F29412        723+          JP      SPOL2
1248 7E            724           MOV     A,M       ;Get talker
1249 D360          725           OUT     DOUT      ;Send to GPIB
124B 23            726           INX     H         ;Incr talker list pointer
124C 3E40          727           MVI     A,LON     ;Listen only
124E D364          728           OUT     ADRMD
                   729           WAITO             ;Wait for talk address to complete
1250 DB61          730+??0033:    IN     INT1      ;Get Int1 status
1252 E602          731+          ANI     BOM       ;Check for byte out
1254 CA5012        732+          JZ      ??0033    ;If not, try again
                   733           CLRA              ;Pattern for immediate XEQ PON
1257 AF            734+          XRA     A         ;A XOR A =0
1258 D365          735           OUT     AUXMD
125A 3EF6          736           MVI     A,GTSB    ;Goto standby
125C D369          737           OUT     CMD92
                   738           WAITX             ;Wait for TCI false
125E DB6F          739+??0034:   IN      PRTF
1260 E602          740+          ANI     TCIF
1262 C25E12        741+          JNZ     ??0034
                   742           WAITT             ;Wait for TCI
1265 DB6F          743+??0035:   IN      PRTF      ;Get task complete int,etc.
1267 E602          744+          ANI     TCIF      ;Mask it
1269 CA6512        745+          JZ      ??0035    ;Wait for task to be complete
                   746           WAITI             ;Wait for status byte input
126C DB61          747+??0036:   IN      INT1      ;Get INT1 status
126E 47            748+          MOV     B,A       ;Save status in B
126F E601          749+          ANI     BIM       ;Check for byte in
1271 CA6C12        750+          JZ      ??0036    ;If not, just try again
1274 3EFD          751           MVI     A,TCSY    ;Take control sync
1276 D369          752           OUT     CMD92
                   753           WAITX             ;Wait for TCI false
1278 DB6F          754+??0037:   IN      PRTF
127A E602          755+          ANI     TCIF
127C C27812        756+          JNZ     ??0037
                   757           WAITT             ;Wait for TCI
127F DB6F          758+??0038:   IN      PRTF      ;Get task complete int,etc.
1281 E602          759+          ANI     TCIF      ;Mask it
1283 CA7F12        760+          JZ      ??0038    ;Wait for task to be complete
1286 DB60          761           IN      DIN       ;Get serial poll status byte
1288 12            762           STAX    D         ;Store it in buffer
1289 13            763           INX     D         ;Incr pointer
128A 3E80          764           MVI     A,TON     ;Talk only for controller
128C D364          765           OUT     ADRMD     ;
                   766           CLRA
128E AF            767+          XRA     A         ;A XOR A =0
128F D365          768           OUT     AUXMD     ;Immeditate XEQ PON
                   769                             ;CLR LA
1291 C33D12        770           JMP     SPOL1     ;Go on to next device on list
                   771 ;
1294 3E19          772 SPOL2:    MVI     A,SPD     ;Serial poll disable
1296 D360          773           OUT     DOUT      ;We know BO was set (WAITO above)
                   774           WAITO
1298 DB61          775+??0039:    IN     INT1      ;Get Int1 status
129A E602          776+          ANI     BOM       ;Check for byte out
129C CA9812        777+          JZ      ??0039    ;If not, try again
                   778           CLRA
129F AF            779+          XRA     A         ;A XOR A =0
12A0 D365          780           OUT     AUXMD     ;Immediate XEQ PON to clear LA
12A2 C9            781           RET
                   782 ;
                   783 ;*****************************************************
                   784 ;
```

```
                785 ;       PARALLEL POLL ENABLE ROUTINE
                786 ;
                787 ;INPUTS:         HL listener list pointer
                788 ;                DE configuration byte pointer
                789 ;OUTPUTS:        None
                790 ;CALLS:          None
                791 ;DESTROYS:       A, DE, HL, F
                792 ;
                793 ;
12A3 3E3F       794 PPEN:    MVI     A,UNL    ;Universal unlisten
12A5 D360       795          OUT     DOUT
                796 PPEN1:   RANGE   20H,3EH,PPEN2   ;Check for valid listener
                797+                         ;Checks for value in range
                798+                         ;branches to label if not
                799+                         ;in range. Falls through if
                800+                         ;lower <= ( (H)(L) ) <= upper.
                801+                         ;Get next byte.
12A7 7E         802+         MOV     A,M
12A8 FE20       803+         CPI     20H
12AA FAD812     804+         JM      PPEN2
12AD FE3F       805+         CPI     3EH+1
12AF F2D812     806+         JP      PPEN2
                807          WAITO            ;Valid wait 91 data out req
12B2 DB61       808+??0040:   IN     INT1     ;Get Intl status
12B4 E602       809+         ANI     BOM      ;Check for byte out
12B6 CAB212     810+         JZ      ??0040   ;If not, try again
12B9 7E         811          MOV     A,M      ;Get listener
12BA D360       812          OUT     DOUT
                813          WAITO
12BC DB61       814+??0041:   IN     INT1     ;Get Intl status
12BE E602       815+         ANI     BOM      ;Check for byte out
12C0 CABC12     816+         JZ      ??0041   ;If not, try again
12C3 3E05       817          MVI     A,PPC    ;Parallel poll configure
12C5 D360       818          OUT     DOUT
                819          WAITO
12C7 DB61       820+??0042:   IN     INT1     ;Get Intl status
12C9 E602       821+         ANI     BOM      ;Check for byte out
12CB CAC712     822+         JZ      ??0042   ;If not, try again
12CE 1A         823          LDAX    D        ;Get matching configuration byte
12CF F660       824          ORI     PPE      ;Merge with parallel poll enable
12D1 D360       825          OUT     DOUT
12D3 23         826          INX     H        ;Incr pointers
12D4 13         827          INX     D
12D5 C3A712     828          JMP     PPEN1    ;Loop until invalid listener char
                829 PPEN2:   WAITO
12D8 DB61       830+??0043:   IN     INT1     ;Get Intl status
12DA E602       831+         ANI     BOM      ;Check for byte out
12DC CAD812     832+         JZ      ??0043   ;If not, try again
12DF C9         833          RET
                834 ;
                835 ;PARALLEL POLL DISABLE ROUTINE
                836 ;
                837 ;INPUTS:         HL listener list pointer
                838 ;OUTPUTS:        None
                839 ;CALLS:          None
                840 ;DESTROYS:       A, HL, F
                841 ;
12E0 3E3F       842 PPDS:    MVI     A,UNL    ;Universal unlisten
12E2 D360       843          OUT     DOUT
                844 PPDS1:   RANGE   20H,3EH,PPDS2   ;Check for valid listener
                845+                         ;Checks for value in range
                846+                         ;branches to label if not
                847+                         ;in range. Falls through if
                848+                         ;lower <= ( (H)(L) ) <= upper.
                849+                         ;Get next byte.
12E4 7E         850+         MOV     A,M
12E5 FE20       851+         CPI     20H
12E7 FAFD12     852+         JM      PPDS2
12EA FE3F       853+         CPI     3EH+1
12EC F2FD12     854+         JP      PPDS2
                855          WAITO
12EF DB61       856+??0044:   IN     INT1     ;Get Intl status
12F1 E602       857+         ANI     BOM      ;Check for byte out
12F3 CAEF12     858+         JZ      ??0044   ;If not, try again
```

```
12F6 7E          859          MOV     A,M     ;Get listener
12F7 D360        860          OUT     DOUT
12F9 23          861          INX     H       ;Incr pointer
12FA C3E412      862          JMP     PPDS1   ;Loop until invalid listener
                 863 PPDS2:   WAITO
12FD DB61        864+??0045:   IN     INT1    ;Get Intl status
12FF E602        865+         ANI     BOM     ;Check for byte out
1301 CAFD12      866+         JZ      ??0045  ;If not, try again
1304 3E05        867          MVI     A,PPC   ;Parallel poll configure
1306 D360        868          OUT     DOUT
                 869          WAITO
1308 DB61        870+??0046:   IN     INT1    ;Get Intl status
130A E602        871+         ANI     BOM     ;Check for byte out
130C CA0813      872+         JZ      ??0046  ;If not, try again
130F 3E70        873          MVI     A,PPD   ;Parallel poll disable
1311 D360        874          OUT     DOUT
                 875          WAITO
1313 DB61        876+??0047:   IN     INT1    ;Get Intl status
1315 E602        877+         ANI     BOM     ;Check for byte out
1317 CA1313      878+         JZ      ??0047  ;If not, try again
131A C9          879          RET
                 880 ;
                 881 ;        PARALLEL POLL UNCONFIGURE ALL ROUTINE
                 882 ;
                 883 ;
                 884 ;INPUTS:          None
                 885 ;OUTPUTS:         None
                 886 ;CALLS:           None
                 887 ;DESTROYS:        A, F
                 888 ;
131B 3E15        889 PPUN:    MVI     A,PPU   ;Parallel poll unconfigure
131D D360        890          OUT     DOUT
                 891          WAITO
131F DB61        892+??0048:   IN     INT1    ;Get Intl status
1321 E602        893+         ANI     BOM     ;Check for byte out
1323 CA1F13      894+         JZ      ??0048  ;If not, try again
1326 C9          895          RET
                 896 ;
                 897 ;************************************************
                 898 ;
                 899 ;CONDUCT A PARALLEL POLL
                 900 ;
                 901 ;
                 902 ;INPUTS:          None
                 903 ;OUTPUTS:         None
                 904 ;CALLS:           None
                 905 ;DESTROYS:        A, B, F
                 906 ;RETURNS:         A= parallel poll status byte
                 907 ;
1327 3E40        908 PPOL:    MVI     A,LON   ;Listen only
1329 D364        909          OUT     ADRMD
                 910          CLRA            ;Immediate XEQ PON
132B AF          911+         XRA     A       ;A XOR A =0
132C D365        912          OUT     AUXMD   ;Reset TON
132E 3EF5        913          MVI     A,EXPP  ;Execute parallel poll
1330 D369        914          OUT     CMD92
                 915          WAITI           ;Wait for completion= BI on 91
1332 DB61        916+??0049:   IN     INT1    ;Get INTl status
1334 47          917+         MOV     B,A     ;Save status in B
1335 E601        918+         ANI     BIM     ;Check for byte in
1337 CA3213      919+         JZ      ??0049  ;If not, just try again
133A 3E80        920          MVI     A,TON   ;Talk only
133C D364        921          OUT     ADRMD
                 922          CLRA            ;Immediate XEQ PON
133E AF          923+         XRA     A       ;A XOR A =0
133F D365        924          OUT     AUXMD   ;Reset LON
1341 DB60        925          IN      DIN     ;Get PP byte
1343 C9          926          RET
                 927 ;
                 928 ;************************************************
                 929 ;PASS CONTROL ROUTINE
                 930 ;
                 931 ;INPUTS:          HL pointer to talker
                 932 ;OUTPUTS:         None
```

```
                933 ;CALLS:          None
                934 ;DESTROYS:       A, HL, F
                935 PCTL:    RANGE   40H,5EH,PCTL1   ;Is it a valid talker ?
                936+                                 ;Checks for value in range
                937+                                 ;branches to label if not
                938+                                 ;in range. Falls through if
                939+                                 ;lower <= ( (H)(L) ) <= upper.
                940+                                 ;Get next byte.
1344 7E         941+            MOV     A,M
1345 FE40       942+            CPI     40H
1347 FA8A13     943+            JM      PCTL1
134A FE5F       944+            CPI     5EH+1
134C F28A13     945+            JP      PCTL1
134F FE41       946             CPI     MTA     ;Is it my talker address
1351 CA8A13     947             JZ      PCTL1   ;Yes, just return
1354 D360       948             OUT     DOUT    ;Send on GPIB
                949             WAITO
1356 DB61       950+??0050:      IN     INT1    ;Get Intl status
1358 E602       951+            ANI     BOM     ;Check for byte out
135A CA5613     952+            JZ      ??0050  ;If not, try again
135D 3E09       953             MVI     A,TCT   ;Take control message
135F D360       954             OUT     DOUT
                955             WAITO
1361 DB61       956+??0051:      IN     INT1    ;Get Intl status
1363 E602       957+            ANI     BOM     ;Check for byte out
1365 CA6113     958+            JZ      ??0051  ;If not, try again
1368 3E01       959             MVI     A,MODE1 ;Not talk only or listen only
136A D364       960             OUT     ADRMD   ;Enable 91 address mode 1
                961             CLRA
136C AF         962+            XRA     A       ;A XOR A =0
136D D365       963             OUT     AUXMD   ;Immediate XEQ PON
136F 3E01       964             MVI     A,MDA   ;My device address
1371 D366       965             OUT     ADR01   ;enabled to talk and listen
1373 3EA1       966             MVI     A,AXRB+CPTEN    ;Command pass thru enable
1375 D365       967             OUT     AUXMD
                968 ;*******optional PP configuration goes here********
1377 3EF1       969             MVI     A,GIDL  ;92 go idle command
1379 D369       970             OUT     CMD92
                971             WAITX
137B DB6F       972+??0052:      IN     PRTF
137D E602       973+            ANI     TCIF
137F C27B13     974+            JNZ     ??0052
                975             WAITT           ;Wait for TCI
1382 DB6F       976+??0053:      IN     PRTF    ;Get task complete int,etc.
1384 E602       977+            ANI     TCIF    ;Mask it
1386 CA8213     978+            JZ      ??0053  ;Wait for task to be complete
1389 23         979             INX     H
138A C9         980 PCTL1:      RET
                981 ;
                982 ;
                983 ;****************************************
                984 ;
                985 ;RECEIVE CONTROL ROUTINE
                986 ;
                987 ;INPUTS:          None
                988 ;OUTPUTS:         None
                989 ;CALLS:           None
                990 ;DESTROYS:        A, F
                991 ;RETURNS:         0= invalid (not take control to us or CPT bit not on)
                992 ;                 < > 0 = valid take control-- 92 will now be in control
                993 ;NOTE:            THIS CODE MUST BE TIGHTLY INTEGRATED INTO ANY USER
                994 ;                 SOFTWARE THAT FUNCTIONS WITH THE 8291 AS  A DEVICE.
                995 ;                 NORMALLY SOME ADVANCE WARNING OF IMPENDING PASS
                996 ;                 CONTROL SHOULD BE GIVEN TO US BY THE CONTROLLER
                997 ;                 WITH OTHER USEFUL INFO. THIS PROTOCOL IS SITUATION
                998 ;                 SPECIFIC AND WILL NOT BE COVERED HERE.
                999 ;
               1000 ;
138B DB61      1001 RCTL:       IN     INT1    ;Get INT1 reg (i.e. CPT etc.)
138D E680      1002             ANI    CPT     ;Is command pass thru on ?
138F CACF13    1003             JZ     RCTL2   ;No, invalid-- go return
1392 DB65      1004             IN     CPTRG   ;Get command
1394 FE09      1005             CPI    TCT     ;Is it take control ?
```

```
1396 C2CA13      1006          JNZ     RCTL1   ;No, go return invalid
1399 DB64        1007          IN      ADRST   ;Get address status
139B E602        1008          ANI     TA      ;Is TA on ?
139D CACA13      1009          JZ      RCTL1   ;No -- go return invalid
13A0 3E60        1010          MVI     A,DTDL1 ;Disable talker listener
13A2 D366        1011          OUT     ADR01
13A4 3E80        1012          MVI     A,TON   ;Talk only
13A6 D364        1013          OUT     ADRMD
                 1014          CLRA
13A8 AF          1015+         XRA     A       ;A XOR A =0
13A9 D361        1016          OUT     INT1    ;Mask off INT bits
13AB D362        1017          OUT     INT2
13AD D365        1018          OUT     AUXMD
13AF 3EFA        1019          MVI     A,TCNTR ;Take (receive) control 92 command
13B1 D369        1020          OUT     CMD92
13B3 3E0F        1021          MVI     A,VSCMD ;Valid command pattern for 91
13B5 D365        1022          OUT     AUXMD
                 1023 ;******** optional TOUT1 check could be put here ********
                 1024          WAITX
13B7 DB6F        1025+??0054:  IN      PRTF
13B9 E602        1026+         ANI     TCIF
13BB C2B713      1027+         JNZ     ??0054
                 1028          WAITT           ;Wait for TCI
13BE DB6F        1029+??0055:  IN      PRTF    ;Get task complete int,etc.
13C0 E602        1030+         ANI     TCIF    ;Mask it
13C2 CABE13      1031+         JZ      ??0055  ;Wait for task to be complete
13C5 3E09        1032          MVI     A,TCT   ;Valid return pattern
13C7 C3CF13      1033          JMP     RCTL2   ;Only one return per routine
13CA 3E0F        1034 RCTL1:   MVI     A,VSCMD ;Acknowledge CPT
13CC D365        1035          OUT     AUXMD
                 1036          CLRA            ;Error return pattern
13CE AF          1037+         XRA     A       ;A XOR A =0
13CF C9          1038 RCTL2:   RET
                 1039 ;
                 1040 ;************************************************
                 1041 ;
                 1042 ;        SRQ ROUTINE
                 1043 ;
                 1044 ;INPUTS:         None
                 1045 ;OUTPUTS:        None
                 1046 ;CALLS:          None
                 1047 ;RETURNS:        A= 0 no SRQ
                 1048 ;                A < > 0 SRQ occured
                 1049 ;
                 1050 ;
13D0 DB69        1051 SRQD:    IN      INTST   ;Get 92's INTRQ status
13D2 E620        1052          ANI     SRQBT   ;Mask off SRQ
13D4 CAE213      1053          JZ      SRQD2   ;Not set--- go return
13D7 F608        1054          ORI     IACK    ;Set--- must clear it with IACK
13D9 D369        1055          OUT     CMD92
13DB DB69        1056 SRQD1:   IN      INTST   ;Get IBF
13DD E602        1057          ANI     IBFBT   ;Mask it
13DF CADB13      1058          JZ      SRQD1   ;Wait if not set
13E2 C9          1059 SRQD2:   RET
                 1060 ;
                 1061 ;*******************************************
                 1062 ;
                 1063 ;REMOTE ENABLE ROUTINE
                 1064 ;
                 1065 ;INPUTS:         None
                 1066 ;OUTPUTS:        None
                 1067 ;CALLS:          NONE
                 1068 ;DESTROYS:       A, F
                 1069 ;
13E3 3EF8        1070 REME:    MVI     A,SREM
13E5 D369        1071          OUT     CMD92   ;92 asserts remote enable
                 1072          WAITX           ;Wait for TCI = 0
13E7 DB6F        1073+??0056:  IN      PRTF
13E9 E602        1074+         ANI     TCIF
13EB C2E713      1075+         JNZ     ??0056
                 1076          WAITT           ;Wait for TCI
13EE DB6F        1077+??0057:  IN      PRTF    ;Get task complete int,etc.
13F0 E602        1078+         ANI     TCIF    ;Mask it
13F2 CAEE13      1079+         JZ      ??0057  ;Wait for task to be complete
```

```
13F5 C9          1080          RET
                 1081 ;
                 1082 ;*****************************************
                 1083 ;
                 1084 ;LOCAL ROUTINE
                 1085 ;
                 1086 ;
                 1087 ;INPUTS:         None
                 1088 ;OUTPUTS:        None
                 1089 ;CALLS:          None
                 1090 ;DESTROYS:       A, F
                 1091 ;
13F6 3EF7        1092 LOCL:    MVI      A,SLOC
13F8 D369        1093          OUT      CMD92    ;92 stops asserting remote enable
                 1094          WAITX             ;Wait for TCI =0
13FA DB6F        1095+??0058: IN        PRTF
13FC E602        1096+         ANI      TCIF
13FE C2FA13      1097+         JNZ      ??0058
                 1098          WAITT             ;Wait for TCI
1401 DB6F        1099+??0059: IN        PRTF     ;Get task complete int,etc.
1403 E602        1100+         ANI      TCIF     ;Mask it
1405 CA0114      1101+         JZ       ??0059   ;Wait for task to be complete
1408 C9          1102          RET
                 1103 ;
                 1104 ;*********************************************
                 1105 ;
                 1106 ;INTERFACE CLEAR / ABORT ROUTINE
                 1107 ;
                 1108 ;
                 1109 ;INPUTS:                 None
                 1110 ;OUTPUTS:                None
                 1111 ;CALLS:                 None
                 1112 ;DESTROYS:              A, F
                 1113 ;
                 1114 ;
1409 3EF9        1115 IFCL:    MVI      A,ABORT
140B D369        1116          OUT      CMD92    ;Send IFC
                 1117          WAITX             ;Wait for TCI =0
140D DB6F        1118+??0060: IN        PRTF
140F E602        1119+         ANI      TCIF
1411 C20D14      1120+         JNZ      ??0060
                 1121          WAITT             ;Wait for TCI
1414 DB6F        1122+??0061: IN        PRTF     ;Get task complete int,etc.
1416 E602        1123+         ANI      TCIF     ;Mask it
1418 CA1414      1124+         JZ       ??0061   ;Wait for task to be complete
                 1125 ;Delete both WAITX & WAITT if this routine
                 1126 ;is to be called while the 8292 is
                 1127 ;Controller-in-Charge. If not C.I.C. then
                 1128 ;TCI is set, else nothing is set (IFC is sent)
                 1129 ;and the WAIT'S will hang forever
141B C9          1130          RET
                 1132 ;
```

```
                    1133 ;APPLICATION EXAMPLE CODE FOR 8085
                    1134 ;
0032                1135 FGDNL   EQU     '2'         ;Func gen device num "2" ASCII,lstn
0031                1136 FCDNL   EQU     '1'         ;Freq ctr device num "1" ASCII,lstn
0051                1137 FCDNT   EQU     'Q'         ;Freq ctr talk address
000D                1138 CR      EQU     0DH         ;ASCII carriage return
000A                1139 LF      EQU     0AH         ;ASCII line feed
00FF                1140 LEND    EQU     0FFH        ;List end for Talk/Listen lists
0040                1141 SRQM    EQU     40H         ;Bit indicating device sent SRQ
                    1142 ;
141C 46553146       1143 FGDATA: DB      'FU1FR37KHAM2VO',CR     ;Data to set up func. gen
1420 5233374B
1424 48414D32
1428 564F
142A 0D
000F                1144 LIM1    EQU     15                      ;Buffer length
142B 50463447       1145 FCDATA: DB      'PF4G7T'                ;Data to set up freq ctr
142F 3754
0006                1146 LIM2    EQU     6                       ;Buffer length
1431 31             1147 LL1:    DB      FCDNL,LEND              ;Listen list for freq ctr
1432 FF
1433 32             1148 LL2:    DB      FGDNL,LEND              ;Listen list for func. gen
1434 FF
1435 51             1149 TL1:    DB      FCDNT,LEND              ;Talk list for freq ctr
1436 FF
                    1150 ;
                    1151 ;SETUP FUNCTION GENERATOR
1437 060D           1152         MVI     B,CR    ;EOS
1439 0E0F           1153         MVI     C,LIM1  ;Count
143B 111C14         1154         LXI     D,FGDATA        ;Data pointer
143E 213314         1155         LXI     H,LL2   ;Listen list pointer
1441 CD1C10         1156         CALL    SEND
                    1157 ;
                    1158 ;SETUP FREQ COUNTER
                    1159 ;
1444 0654           1160         MVI     B,'T'   ;EOS
1446 0E06           1161         MVI     C,LIM2  ;Count
1448 112814         1162         LXI     D,FCDATA        ;Data pointer
144B 213114         1163         LXI     H,LL1   ;Listen list pointer
144E CD1C10         1164         CALL    SEND
                    1165 ;
                    1166 ;WAIT FOR SRQ FROM FREQ CTR
                    1167 ;
1451 CDD013         1168 LOOP:   CALL    SRQD    ;Has SRQ occurred ?
1454 CA5114         1169         JZ      LOOP    ;No, wait for it
                    1170 ;
                    1171 ;SERIAL POLL TO CLEAR SRQ
                    1172 ;
1457 11003C         1173         LXI     D,SPBYTE        ;Buffer pointer
145A 213514         1174         LXI     H,TL1           ;Talk list pointer
145D CD1C12         1175         CALL    SPOL
1460 1B             1176         DCX     D       ;Backup buffer pointer to ctr byte
1461 1A             1177         LDAX    D       ;Get status byte
1462 E640           1178         ANI     SRQM    ;Did ctr assert SRQ ?
1464 CA7714         1179         JZ      ERROR   ;Ctr should have said yes
                    1180 ;
                    1181 ;RECEIVE READING FROM COUNTER
                    1182 ;
1467 060A           1183         MVI     B,LF    ;EOS
1469 0E11           1184         MVI     C,LIM3  ;Count
146B 213514         1185         LXI     H,TL1   ;Talk list pointer
146E 11013C         1186         LXI     D,FCDATI        ;Data in buffer pointer
1471 CD9F10         1187         CALL    RECV
1474 C27714         1188         JNZ     ERROR
                    1189 ;
                    1190 ;******* rest of user processing goes here *****
                    1191 ;
                    1192 ;
1477 00             1193 ERROR:  NOP             ;User dependant error handling
                    1194 ;       ETC.
3C00                1195 ORG     3C00H
3C00                1196 SPBYTE: DS      1       ;Location for serial poll byte
0011                1197 LIM3    EQU     17      ;Max freq counter input
```

```
3C01                1198 FCDATI: DS      LIM3    ;Freq ctr input buffer
                    1199         END

PUBLIC SYMBOLS

EXTERNAL SYMBOLS


USER SYMBOLS
ABORT   A 00F9    ADR01   A 0066    ADRMD   A 0064    ADRST   A 0064    AUXMD   A 0065    AXRA    A 0080    AXRB    A 00A0
BIM     A 0001    BOF     A 0001    BOM     A 0002    BUSST   A 0068    CAHCY   A 0003    CLKRT   A 0023    CLRA    + 0007
CLRST   A 0068    CMD92   A 0069    CPT     A 0080    CPTEN   A 0001    CPTRG   A 0065    CR      A 000D    DCL     A 0014
DCLR    A 11EC    DCLR1   A 11F0    DCLR2   A 1209    DIN     A 0060    DOUT    A 0060    DTDL1   A 0060    DTDL2   A 00E0
EDEOS   A 0004    ENDMK   A 0010    EOIS    A 0008    EOIST   A 0020    EOSR    A 0067    ERFLG   A 0068    ERRM    A 0068
ERROR   A 1477    EVBIT   A 0010    EVCST   A 0068    EVREG   A 0068    EXPP    A 00F5    FCDATA  A 142B    FCDATI  A 3C01
FCDNL   A 0031    FCDNT   A 0051    FGDATA  A 141C    FGDNL   A 0032    FNHSK   A 0003    GET     A 0008    GIDL    A 00F1
GSEC    A 00F4    GTSB    A 00F6    HOEND   A 0002    HOHSK   A 0001    IACK    A 000B    IBFBT   A 0002    IBFF    A 0010
IFCL    A 1409    INIT    A 1000    INT1    A 0061    INT2    A 0062    INTM    A 00A0    INTM1   A 0061    INTMR   A 0068
INTST   A 0069    LA      A 0001    LEND    A 00FF    LF      A 000A    LIM1    A 000F    LIM2    A 0006    LIM3    A 0011
LL1     A 1431    LL2     A 1433    LOCL    A 13F6    LON     A 0040    LOOP    A 1451    MDA     A 0001    MLA     A 0021
MODE1   A 0001    MTA     A 0041    NVCMD   A 0007    OBFF    A 0008    PCTL    A 1344    PCTL1   A 138A    PPC     A 0005
PPD     A 0070    PPDS    A 12E0    PPDS1   A 12E4    PPDS2   A 12FD    PPE     A 0060    PPEN    A 12A3    PPEN1   A 12A7
PPEN2   A 12D8    PPOL    A 1327    PPU     A 0015    PPUN    A 131B    PRT91   A 0060    PRT92   A 0068    PRTF    A 006F
RANGE   + 0005    RBST    A 00E7    RCST    A 00E6    RCTL    A 138B    RCTL1   A 13CA    RCTL2   A 13CF    RECV    A 109F
RECV1   A 10EA    RECV2   A 1105    RECV3   A 1106    RECV4   A 1110    RECV5   A 1117    RECV6   A 1139    REME    A 13E3
RERF    A 00E4    RERM    A 00EA    REVC    A 00E3    RINM    A 00E5    RSET    A 00F2    RSTI    A 00F3    RTOUT   A 00E9
SDEOI   A 0006    SEND    A 101C    SEND1   A 102E    SEND2   A 1047    SEND3   A 1069    SEND4   A 1070    SEND5   A 107F
SEND6   A 1088    SETF    + 0000    SLOC    A 00F7    SPBYTE  A 3C00    SPCNI   A 00F0    SPD     A 0019    SPE     A 0018
SPIF    A 0004    SPOL    A 121C    SPOL1   A 123D    SPOL2   A 1294    SREM    A 00F8    SRQBT   A 0020    SRQD    A 13D0
SRQD1   A 13DB    SRQD2   A 13E2    SRQM    A 0040    STCNI   A 00FE    TA      A 0002    TCASY   A 00FC    TCIF    A 0002
TCNTR   A 00FA    TCSY    A 00FD    TCT     A 0009    TL1     A 1435    TLON    A 00C0    TON     A 0080    TOREG   A 0068
TOST    A 0068    TOUT1   A 0001    TOUT2   A 0002    TOUT3   A 0004    TRIG    A 11BC    TRIG1   A 11C0    TRIG2   A 11D9
UNL     A 003F    VSCMD   A 000F    WAITI   + 0002    WAITO   + 0001    WAITT   + 0004    WAITX   + 0003    WEVC    A 00E2
WOUT    A 00E1    XFER    A 113A    XFER1   A 1153    XFER2   A 116C    XFER3   A 1193    XFER4   A 11BB

ASSEMBLY COMPLETE,    NO ERRORS
```

APPENDIX B

TEST CASES FOR THE SOFTWARE DRIVERS

The following test cases were used to exercise the software routines and to check their action. To provide another device/controller on the GPIB a ZT488 GPIB Analyzer was used. This analyzer acted as a talker, listener or another controller as needed to execute the tests. The sequence of outputs are shown with each test. All numbers are hexadecimal.

SEND TEST CASES

B =	44	44	44
C =	30	2	0
DE =	3E80	3E80	3E80
HL =	3E70	3E70	3E70
3E70:	20 30 3E 3F		
3E80:	11 44		
GPIB output:	41 ATN	41 ATN	41 ATN
	3F ATN	3F ATN	3F ATN
	20 ATN	20 ATN	20 ATN
	30 ATN	30 ATN	30 ATN
	3E ATN	3E ATN	3E ATN
	11	11	
	44 EOI	44 EOI	
Ending B =	44	44	44
Ending C =	2E	0	0
Ending DE =	3E82	3E82	3E80
Ending HL =	3E73	3E73	3E73

RECEIVE TEST CASES

B =	44	44	44	44	44	44	44
C =	30	30	30	30	4	4	0=256
DE =	3E80	3E80	3E80	3E80	3E80	3E80	3E80
HL =	3E70	3E70	3E70	3E70	3E70	3E70	3E70
3E70:	40	50	5E	5F	40	40	40
GPIB output:	40 ATN	50 ATN	5E ATN		40 ATN	40 ATN	40 ATN
	3F ATN	3F ATN	3F ATN		3F ATN	3F ATN	3F ATN
	21 ATN	21 ATN	21 ATN		21 ATN	21 ATN	21 ATN
ZT488 Data	1	1	1		1	11	1
In	2	2	2		2	22	2
	3	3	3		3	33	3
	4	4	44,EOI		4	44	44
	44	5,EOI					
Ending A =	0	0	0	5F	40	0	0
Ending B =	0	0	0	44	40	0	0
Ending C =	2B	2B	2C	30	0	0	FC
Ending DE =	3E85	3E85	3E84	3E80	3E84	3E84	3E84
Ending HL =	3E71	3E71	3E71	3E70	3E71	3E71	3E71

SERIAL POLL TEST CASES

C =	30
DE =	3E80
HL =	3E70
3E70:	40
	50
	5E
	5F

C =	30
DE =	3E80
HL =	3E70
3E70:	5F
GPIB output:	3F ATN
	21 ATN
	18 ATN

GPIB output: 3F ATN 19 ATN
output: 21 ATN Ending C = 30
output: 18 ATN Ending DE = 3E80
output: 40 ATN Ending HL = 3E70
input*: 00
output: 50 ATN
input*: 41
output: 5E ATN
input*: 7F
output: 19 ATN

*NOTE: leave ZT488 in single step mode even on input

Ending C = 30
Ending DE = 3E83
Ending HL = 3E73
Ending 3E80: 00 41 7F

PASS CONTROL TEST CASES

HL =	3E70	3E70	3E70
3E70:	40	41(MTA)	5F
GPIB output:	40 ATN		
	09 ATN		
	— $\overline{\text{ATN}}$		
Ending HL =	3E71	3E70	3E70
Ending A =	02	41(MTA)	5F

RECEIVE CONTROL TEST CASES

GPIB input	10 ATN	40 ATN	41 ATN
	$\overline{\text{ATN}}$	09 ATN	09 ATN
Run Receive Control			
GPIB Input		$\overline{\text{ATN}}$	$\overline{\text{ATN}}$
Ending A =	0	0	09

PARALLEL POLL ENABLE TEST CASES

DE =	3E80	3E80
HL =	3E70	3E70
3E70:	20 30 3E 3F	3F
3E80:	01 02 03	
GPIB output:	3F ATN	3F ATN
	20 ATN	
	05 ATN	
	61 ATN	
	30 ATN	
	05 ATN	
	62 ATN	
	3E ATN	
	05 ATN	
	63 ATN	
Ending DE =	3E83	3E80
Ending HL =	3E73	3E70

PARALLEL POLL DISABLE TEST CASES

HL =	3E70	3E70
3E70:	20 30 3E 3F	3F

GPIB output:	3F ATN	3F ATN
	20 ATN	05 ATN
	30 ATN	70 ATN
	3E ATN	
	05 ATN	
	70 ATN	
Ending HL =	3E73	3E70

PARALLEL POLL UNCONFIGURE TEST CASE

GPIB output: 15 ATN

PARALLEL POLL TEST CASES

Set DIO #	1	2	3	4	5	6	7	8	None
Ending A	1	2	4	8	10	20	40	80	0

SRQ TEST

	Set SRQ momentarily	Reset SRQ
Ending A	= 02	00

TRIGGER TEST

HL = 3E70
DE = 3E80
BC = 4430
3E70: 20 30 3E 3F
GPIB output: 3F ATN
20 ATN
30 ATN
3E ATN
08 ATN
Ending HL = 3E73
DE = 3E80
BC = 4430

DEVICE CLEAR TEST

HL = 3E70
DE = 3E80
BC = 4430
3E70: 20 30 3E 3F
GPIB output: 3F ATN
20 ATN
30 ATN
3E ATN
14 ATN
Ending HL = 3E73
DE = 3E80
RC = 4430

XFER TEST

```
          B = 44
         HL = 3E70
      3E70: 40  20  30  3E  3F
GPIB output: 40 ATN
             3F ATN
             20 ATN
             30 ATN
             3E ATN
 GPIB input: 0
             1
             2
             3
             44
 Ending A = 0
          B = 44
         HL = 3E74
```

APPLICATION EXAMPLE
GPIB OUTPUT/INPUT

```
GPIB output: 41 ATN
             3F ATN
             32 ATN
             46
             55
             31
             46
             52
             33
             37
             4B
             48
             41
             4D
             32
             56
             4F
             0D EOI
             41 ATN
             3F ATN
             31 ATN
             50
             46
             34
             47
             37
             54 EOI
 GPIB input: SRQ
GPIB output: 3F ATN
             21 ATN
             18 ATN
             51 ATN
 GPIB input: 40 SRQ
GPIB output: 19 ATN
             51 ATN
```

3F ATN
21 ATN
GPIB input: 20
2B
20
20
20
33
37
30
30
30
2E
30
45
2B
30
0D
0A
GPIB output: XX ATN

APPENDIX C

REMOTE MESSAGE CODING

Mnemonic	Message Name		Type	Class	Bus Signal Line(s) and Coding That Asserts the True Value of the Message: DIO 8	7	6	5	4	3	2	1	DIO	NDAC	NRFD	ATN	EOI	SRQ	IFC	REN
ACG	addressed command group		M	AC	Y	Ø	Ø	Ø	X	X	X	X	XXX	1	X	X	X	X		
ATN	attention		U	UC	X	X	X	X	X	X	X	X	XXX	1	X	X	X	X		
DAB	data byte	(Notes 1, 9)	M	DD	D8	D7	D6	D5	D4	D3	D2	D1	XXX	Ø	X	X	X	X		
DAC	data accepted		U	HS	X	X	X	X	X	X	X	X	XXØ	X	X	X	X	X		
DAV	data valid		U	HS	X	X	X	X	X	X	X	X	1XX	X	X	X	X	X		
DCL	device clear		M	UC	Y	Ø	Ø	1	Ø	1	Ø	Ø	XXX	1	X	X	X	X		
END	end		U	ST	X	X	X	X	X	X	X	X	XXX	Ø	1	X	X	X		
EOS	end of string	(Notes 2, 9)	M	DD	E8	E7	E6	E5	E4	E3	E2	E1	XXX	Ø	X	X	X	X		
GET	group execute trigger		M	AC	Y	Ø	Ø	Ø	1	Ø	Ø	Ø	XXX	1	X	X	X	X		
GTL	go to local		M	AC	Y	Ø	Ø	Ø	Ø	Ø	Ø	1	XXX	1	X	X	X	X		
IDY	identify		U	UC	X	X	X	X	X	X	X	X	XXX	X	1	X	X	X		
IFC	interface clear		U	UC	X	X	X	X	X	X	X	X	XXX	X	X	X	1	X		
LAG	listen address group		M	AD	Y	Ø	1	X	X	X	X	X	XXX	1	X	X	X	X		
LLO	local lock out		M	UC	Y	Ø	Ø	1	Ø	Ø	Ø	1	XXX	1	X	X	X	X		
MLA	my listen address	(Note 3)	M	AD	Y	Ø	1	L5	L4	L3	L2	L1	XXX	1	X	X	X	X		
MTA	my talk address	(Note 4)	M	AD	Y	1	Ø	T5	T4	T3	T2	T1	XXX	1	X	X	X	X		
MSA	my secondary address	(Note 5)	M	SE	Y	1	1	S5	S4	S3	S2	S1	XXX	1	X	X	X	X		

Mnemonic		Message Name	Type	Class	Bus Signal Line(s) and Coding That Asserts the True Value of the Message: DIO 8	7	6	5	4	3	2	1	DIO	DAV	NRFD	NDAC	ATN	SRQ	IFC	REN	EOI
NUL	null byte		M	DD	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	XXX	X	X	X	X	X			
OSA	other secondary address		M	SE	(OSA = SCG ∧ $\overline{\text{MSA}}$)																
OTA	other talk address		M	AD	(OTA = TAG ∧ $\overline{\text{MTA}}$)																
PCG	primary command group		M	—	(PCG = ACG ∨ UCG ∨ LAG ∨ TAG)																
PPC	parallel poll configure		M	AC	Y	Ø	Ø	Ø	Ø	1	Ø	1	XXX	1	X	X	X	X			
PPE	parallel poll enable	(Note 6)	M	SE	Y	1	1	Ø	S	P3	P2	P1	XXX	1	X	X	X	X			
PPD	parallel poll disable	(Note 7)	M	SE	Y	1	1	1	D4	D3	D2	D1	XXX	1	X	X	X	X			
PPR1	parallel poll response 1		U	ST	X	X	X	X	X	X	X	1	XXX	1	1	X	X	X			
PPR2	parallel poll response 2		U	ST	X	X	X	X	X	X	1	X	XXX	1	1	X	X	X			
PPR3	parallel poll response 3	(Note 10)	U	ST	X	X	X	X	X	1	X	X	XXX	1	1	X	X	X			
PPR4	parallel poll response 4		U	ST	X	X	X	X	1	X	X	X	XXX	1	1	X	X	X			
PPR5	parallel poll response 5		U	ST	X	X	X	1	X	X	X	X	XXX	1	1	X	X	X			
PPR6	parallel poll response 6		U	ST	X	X	1	X	X	X	X	X	XXX	1	1	X	X	X			
PPR7	parallel poll response 7	(Note 10)	U	ST	X	1	X	X	X	X	X	X	XXX	1	1	X	X	X			
PPR8	parallel poll response 8		U	ST	1	X	X	X	X	X	X	X	XXX	1	1	X	X	X			
PPU	parallel poll unconfigure		M	UC	Y	Ø	Ø	1	Ø	1	Ø	1	XXX	1	X	X	X	X			
REN	remote enable		U	UC	X	X	X	X	X	X	X	X	XXX	X	X	X	X	1			
RFD	ready for data		U	HS	X	X	X	X	X	X	X	X	XØX	X	X	X	X	X			
RQS	request service	(Note 9)	U	ST	X	1	X	X	X	X	X	X	XXX	Ø	X	X	X	X			
SCG	secondary command group		M	SE	Y	1	1	X	X	X	X	X	XXX	1	X	X	X	X			
SDC	selected device clear		M	AC	Y	Ø	Ø	Ø	Ø	1	Ø	Ø	XXX	1	X	X	X	X			
SPD	serial poll disable		M	UC	Y	Ø	Ø	1	1	Ø	Ø	1	XXX	1	X	X	X	X			
SPE	serial poll enable		M	UC	Y	Ø	Ø	1	1	Ø	Ø	Ø	XXX	1	X	X	X	X			
SRQ	service request		U	ST	X	X	X	X	X	X	X	X	XXX	X	X	1	X	X			
STB	status byte	(Notes 8, 9)	M	ST	S8	X	S6	S5	S4	S3	S2	S1	XXX	Ø	X	X	X	X			
TCT	take control		M	AC	Y	Ø	Ø	Ø	1	Ø	Ø	1	XXX	1	X	X	X	X			
TAG	talk address group		M	AD	Y	1	Ø	X	X	X	X	X	XXX	1	X	X	X	X			
UCG	universal command group		M	UC	Y	Ø	Ø	1	X	X	X	X	XXX	1	X	X	X	X			
UNL	unlisten		M	AD	Y	Ø	1	1	1	1	1	1	XXX	1	X	X	X	X			
UNT	untalk	(Note 11)	M	AD	Y	1	Ø	1	1	1	1	1	XXX	1	X	X	X	X			

The 1/Ø coding on ATN when sent concurrent with multiline messages has been added to this revision for interpretive convenience.

NOTES:

(1) D1-D8 specify the device dependent data bits.

(2) E1-E8 specify the device dependent code used to indicate the EOS message.

(3) L1-L5 specify the device dependent bits of the device's listen address.

(4) T1-T5 specify the device dependent bits of the device's talk address.

(5) S1-S5 specify the device dependent bits of the device's secondary address.

(6) S specifies the sense of the PPR.

S	Response
Ø	Ø
1	1

P1-P3 specify the PPR message to be sent when a parallel poll is executed.

P3	P2	P1	PPR Message
Ø	Ø	Ø	PPR1
.	.	.	.
.	.	.	.
.	.	.	.
1	1	1	PPR8

(7) D1-D4 specify don't-care bits that shall not be decoded by the receiving device. It is recommended that all zeroes be sent.

(8) S1-S6, S8 specify the device dependent status. (DIO7 is used for the RQS message.)

(9) The source of the message on the ATN line is always the C function, whereas the messages on the DIO and EOI lines are enabled by the T function.

(10) The source of the messages on the ATN and EOI lines is always the C function, whereas the source of the messages on the DIO lines is always the PP function.

(11) This code is provided for system use, see 6.3.

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, CA 95051 • (408) 987-8080

Printed in U.S.A. T197/180/15K TL