THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 62, No. 1, January 1983
Printed in U.S.A.

The 3B20D Processor & DMERT Operating System:

3B20D Input/Output System

By A. H. BUDLONG and F. W. WENDLAND

(Manuscript received March 18, 1982)

The 3B20D Input/Output system supports a large number of peripheral devices of various types. The 3B20D Input/Output Processor was designed for maximum flexibility because of the diversity of the peripheral interface requirements. Because of this diversity the basic structure was developed around a high-speed bipolar multiplexor. This multiplexor interfaces with a standard microprocessor bus to numerous specialized one- or two-board computers, which interface to the peripheral devices. Power and unit designs of the input/output processor also provide flexibility in power control and ground isolation of the peripherals. The variety of peripheral units include slow- and medium-speed tape drives (including streamers), low- and high-speed data links, medium to very-high-speed printers, and scanner and signal distributor circuits.

I. INTRODUCTION

The 3B20D Input/Output system was planned from conception to be very flexible, and to support a large number of different peripheral devices. To accomplish this, several difficult system and design-level problems had to be solved.

(*i*) Processor real time—Previous experience in system input/output had shown that even low-speed input/output could be very real-time intensive. Terminal input/output, in particular, requires considerable per-character processing. The 3B20D Input/Output system, therefore, had to provide for front-end processing.

(*ii*) Lack of interface standards—Input/output standards at the physical, electrical, and protocol levels for certain peripheral devices, even those of the same functional class, i.e., tapes, floppy disks, and so

on, are relatively nonexistent. Where there are standards, such as Electronics Industries Association's RS 232 (electrical and physical standards for data link and terminal access), they often are interpreted differently by various vendors, cover only a portion of the market for a class of devices, or are subject to periodic revision. The input/output system, therefore, had to provide a standard interface that could easily support a wide range of nonstandard interfaces and power/ground systems.

(*iii*) Variations in operation and maintenance processing—Error handling, as well as operational processing, is significantly different for each peripheral device.

(*iv*) Noninterruptive growth and maintenance—The input/output system had to permit easy addition of new or replacement devices without interrupting service.

The sheer number of various devices available, their mismatch in performance to the central processing unit bus structures, and their often low cost and rapid obsolescence required an extremely flexible and adaptive interface structure.

II. THE 3B20D INPUT/OUTPUT PROCESSOR

The 3B20D input/output processor resolves the many problems stated above with five major hardware components, and a modular software structure in the host. The hardware components are: duplicated access to each 3B20D half over the two dual-serial channel interfaces; a common high-speed memory access multiplexor, the peripheral interface controller; communities of single-board computers, called peripheral controllers, which provide preprocessing intelligence and specialized device input/output interfaces; the input/output microprocessor interface bus, which provides a common Peripheral Controller interface to the Peripheral Interface Controller; and unit power design structured for flexibility and growth (see Fig. 1).

2.1 Circuit partitioning

The dual-serial channel is described in detail in this issue including the Duplex Serial Bus Selector.[1,2] The duplex serial bus selector acts in concert with the bus interface controller circuit pack to communicate with the peripheral interface controller. The bus interface controller buffers processor data and commands to the peripheral interface controller, as well as data and status information from the peripheral interface controller. It also performs the mandatory "handshaking" to communicate with the duplex serial bus selector. The bus interface controller allows the 16-bit peripheral interface controller to transmit and receive data from the higher-capacity 32-bit duplex serial bus selector at a rate the peripheral interface controller is able to accept.

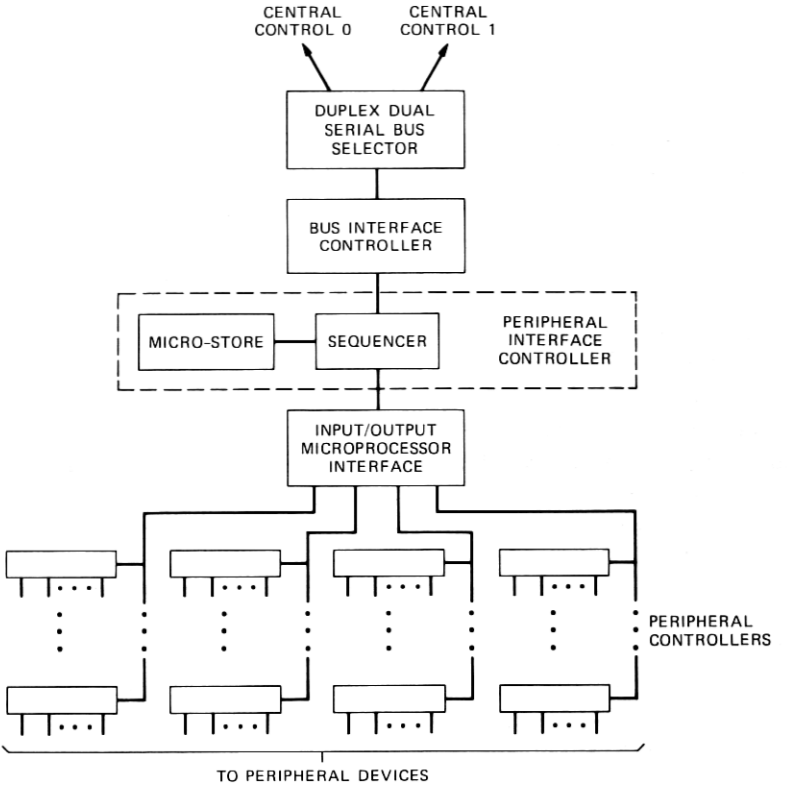


Fig. 1—The 3B20D peripheral interface.

The bus interface controller verifies the integrity of data transferred between the duplex serial bus selector and the peripheral interface controller via per-byte data-parity checks. The bus interface controller uses a 16-word by 32-bit data first in-first out register that can be alternately accessed by the peripheral interface controller and duplex serial bus selector for the transfer of data. The bus interface controller contains a 32-bit command register that records 3B20D processor commands to the peripheral interface controller, and a 16-bit status flag register and a 16-bit error flag register. The status flags are used to request and interrupt direct memory access service from the 3B20D Processor, and inform the peripheral interface controller of 3B20D commands or requests for data transfer. The error flags record the occurrence of errors and aid in fault resolution and recovery. Peripheral interface controller sanity and interval timing functions are also provided by the bus interface controller.

The peripheral interface controller is a simplex, high-speed, bipolar microprocessor. The peripheral interface controller, with its associated

bus interface circuits, handles all common maintenance and operational input/output functions between the various peripheral controllers, and the 3B20D direct memory access.

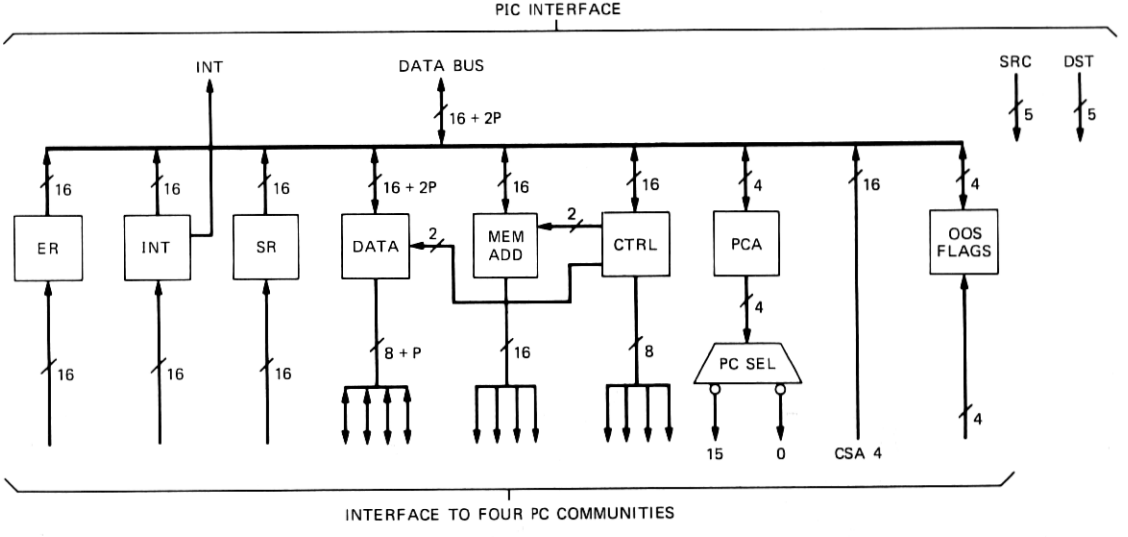
The peripheral interface controller consists of two circuit packs, a controller and one 8K-word micro control memory. The peripheral interface controller contains an arithmetic and logic unit, a 16-word general register file, a 4K-word data store, an 8-level interrupt controller, a microcontroller sequencer, and a pipe-line register. The 8K-by 40-bit micro control memory pack stores the peripheral interface controller operational and diagnostic firmware.

The input/output microprocessor interface circuit serves to interface the 16-bit peripheral interface controller with up to four communities of four peripheral controllers each. The interface between the input/output microprocessor interface and each community consists of a 16-bit memory address, an 8-data plus 1-parity-bit bidirectional data bus, and eight control pulses. Figure 2 details the input/output microprocessor interface bus structure. In addition, a private peripheral controller select signal is connected to each of the 16 peripheral controllers. The selected peripheral controller acknowledges the receipt of a control pulse by setting its control signal acknowledge. The 16 control signal acknowledge bits are compared to the 16 peripheral controller select signals to indicate to the peripheral interface controller when the acknowledgment has been received.

Each peripheral controller also has three request leads that are employed to report errors or to request service. Each peripheral controller request lead is assigned a bit in each of three 16-bit request registers: peripheral controller error, interrupt, and service request. A summary bit is provided over the interrupt register contents to trigger a peripheral interface controller interrupt whenever any peripheral controller interrupts are set. The peripheral interface controller can read each of the request registers to resolve requests to the peripheral controller level.

The four fanout branches of data, address, and control signals (one per community) are driven from three common registers. The 16-bit address register is implemented using binary up/down counters to allow autoincrement and autodecrement capability. The mode of operation is specified by bits in the control signal register. Special input/output microprocessor interface circuitry has been provided to guarantee address setup and hold times of the peripheral controller memory read and write control signals.

The peripheral interface controller data register is segmented by the input/output microprocessor interface into high and low bytes to be shipped a byte at a time to and from the 8-data and 1-parity-bit peripheral controller data buses. A flag in the control register selects



ER – ERROR REGISTER
INT – INTERRUPT REGISTER
MEM ADD – MEMORY ADDRESS
P – PARITY

PC – PERIPHERAL CONTROLLER
PIC – PERIPHERAL INTERFACE CONTROLLER
SR – SERVICE REQUEST REGISTER

Fig. 2—Microprocessor interface.

the high or low byte as the source or destination of the data transfer between the input/output microprocessor interface and the selected peripheral controller. An additional bit in the control register is used to toggle the data byte selector flag after a peripheral controller memory read or write cycle. The circuit incorporates a delay timing chain to guarantee data hold time during peripheral controller memory write operations.

The control register is partitioned into two fields; an 8-bit register, the outputs of which directly drive the control signal buses; and an 8-bit field that controls internal functions in the input/output microprocessor interface.

An out-of-service bit is provided for each of the four peripheral controller communities. This 4-bit register in the input/output microprocessor interface can be read and written by the peripheral interface controller. Additionally, a power-down condition within a community will automatically set its out-of-service flag. When out of service, the interrupt requests of the four peripheral controllers within that community are not ORed into the interrupt register summary.

2.2 Peripheral controllers

The peripheral controller is a microcomputer system that serves as an intelligent interface between the peripheral interface controller and the slow- to medium-speed peripheral units. The peripheral controller is responsible for all device-specific front-end processing and per-byte interrupt handling. This relieves the 3B20D central processing unit from low-level tasks and greatly increases the number of peripheral devices the 3B20D can support. Figure 3 shows a typical one-board peripheral controller design. Up to four subdevices can be separately addressed within a peripheral controller by the peripheral interface controller.

2.2.1 Peripheral interface controller-to-peripheral controller communications

Subdevices are capable of initiating several types of work requests. A peripheral controller may initiate a service request that results in the peripheral interface controller reading service-request status information to determine the type of work required. A subdevice activates a service request to transfer data to or from 3B20D main memory or to indicate a job is completed. In addition to service requests, a subdevice may at a high priority level, initiate an interrupt request that results in interrupt-request status information being read by the peripheral interface controller. The peripheral controller interrupt request is activated by a subdevice to transfer high-priority informa-

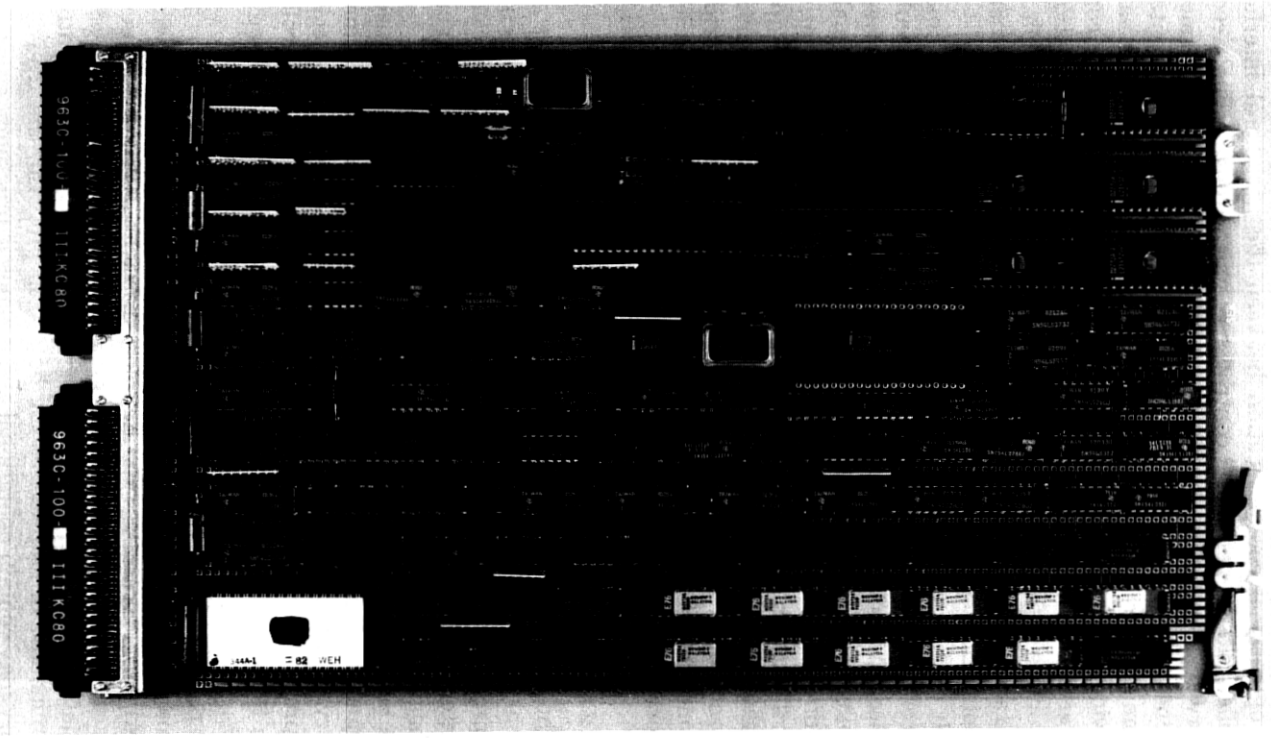


Fig. 3—Typical one-board peripheral controller design.

tion to or from 3B20D main memory or to report a high-priority job has completed.

All operational communication between the peripheral interface controller and the peripheral controller is done through a dual-access memory resident on each peripheral controller. Dedicated memory locations are reserved within the dual access memory that contain pointers used by the peripheral interface controller to locate each subdevice data buffers, data transfer parameters, and status and command areas. The dual access memory also contains memory locations that contain information at the peripheral controller level such as service and interrupt request data and peripheral controller command area pointers.

Each of the peripheral controllers is an independent single-board microcomputer. The hardware is specialized to handle the particular interface and is not necessarily the same as any other peripheral controller. However, all peripheral controllers have a functionally common front-end design and maintain the same communication protocol with the peripheral interface controller. The input/output processor has been designed to allow the peripheral controllers to exist as autonomously as possible. This capability will enable future peripheral controller designers to use state of the art microprocessors and microprocessor-controlled peripherals.

2.2.2 Peripheral controller hardware architecture

Although peripheral controllers designed for specific interfaces are different, all peripheral controllers have the following components: microprocessor, read-only memory, random access memory, service and interrupt sources, isolation circuitry, error detection logic, and scan back circuitry.

The peripheral controller employs a microprocessor to administer the transfer of data between its peripheral units and 3B20D main memory. There are no requirements dictating the use of a specific microprocessor type. The intent is to allow a peripheral controller design to take advantage of the microprocessor that most efficiently handles a particular peripheral unit.

The peripheral controller houses a bootstrap program stored in read-only memory that is entered when power is initially applied or when the peripheral controller reset function is activated by the peripheral interface controller. The bootstrap program causes the microprocessor to initialize its periphery and to recognize peripheral interface controller commands. In addition to the bootstrap program, the read-only memory may contain operational and diagnostic programs.

The peripheral controller uses random access memory to store the operational program, and buffer transient data, and to provide storage

for peripheral interface controller communication parameters. The portion of random access memory containing the data buffers and communication parameters is accessed by both the peripheral interface controller and peripheral controller microprocessor. Therefore, this portion of random access memory must provide dual-access capability.

To prevent an insane microprocessor condition from babbling on the common bus, the peripheral controller is provided with an isolation state. In this state, isolation circuitry removes all response signals generated by the associated peripheral controller from the bus. In some peripheral controller applications the isolation state also prevents data transmission to the peripheral units connected to the peripheral controller.

2.2.3 Service and interrupt request generation

The peripheral controller generates a service request or an interrupt request whenever a data transfer to or from 3B20D main memory is required, or to indicate that a command completion response is available. The peripheral controller provides a separate pulse source for the interrupt request and service request indication. The peripheral controller must refrain from sending a service request or interrupt request until the previous one has been recognized by the peripheral interface controller. The peripheral interface controller indicates recognition of a service request or interrupt request by clearing the associated request pending flag in the peripheral controller dual-access memory.

Each subdevice within the peripheral controller is assigned 4 bytes of dual-access memory, which contain the status information for the associated subdevice. The status information is comprised of generic and user-defined data, and is read as a result of subdevice initiated service requests and interrupt requests.

2.2.4 Hardware error detection

The peripheral controller provides parity checking and generation for data contained in the dual-access memory. Since most microprocessor and microprocessor peripherals available today do not carry parity within the device, routine diagnostics must take over the responsibility of error detection for these devices. Parity over read-only memory data is not a requirement. This enables peripheral controller designs to take advantage of the available large, 8-bit-wide, read-only memories. However, a program sanity check must be provided to detect a read-only memory data failure. A parity error flip/flop is used to store the indication of a dual-access memory parity failure. The peripheral controller also provides control logic to invert parity generation in order to check the parity circuits.

Peripheral controllers operate autonomously with regard to peripheral interface controller activity and, therefore, provide their own system clock. A system clock check circuit is provided by the peripheral controller to detect clock activity abnormalities. A clock error flip-flop is used to store the clock failure indication. The error lead is employed to report the detection of such failures.

Due to the complexity of the peripheral controller and the limited amount of parity checking provided, the peripheral controller uses routine maintenance diagnostics to enhance error-detection capabilities. Redundant software, rather than redundant hardware, is employed to detect fault conditions. The peripheral controller microprocessor periodically executes routine maintenance diagnostics during nonpeak data transmission time intervals. A routine maintenance diagnostics error flip-flop is provided to store the routine maintenance diagnostics failure.

To assure program execution sanity, the peripheral controller provides a sanity check mechanism. The sanity check complexity is directly proportional to the destructive powers of an insane peripheral controller microprocessor. A sanity failure flip-flop is used to store the sanity failure indication.

2.2.5 Scan back circuitry

The peripheral controller provides four directly addressable read-only functions: status and error information, address lower loop-around data, address upper loop-around data, and peripheral controller type-identity code.

The peripheral interface controller uses the scan back address data in a loop-around mode to check the integrity of the direct memory access address and data buses and to verify that a particular peripheral controller is receiving all the address information.

III. PERIPHERAL INTERFACE CONTROLLER AND PERIPHERAL CONTROLLER FIRMWARE AND SOFTWARE

3.1 Peripheral interface controller firmware

The functions performed by the peripheral interface controller are carried out at two levels: base level and interrupt level. Deferrable tasks are performed during base level processing and nondeferrable at interrupt level (see Fig. 4).

Peripheral controller service requests are handled during the peripheral interface controller base level loop. The peripheral interface controller interrogates the peripheral controller service request register to ascertain which peripheral controllers are requesting service. The peripheral interface controller then reads the peripheral controller status bytes in dual-access memory to determine the type of job to be

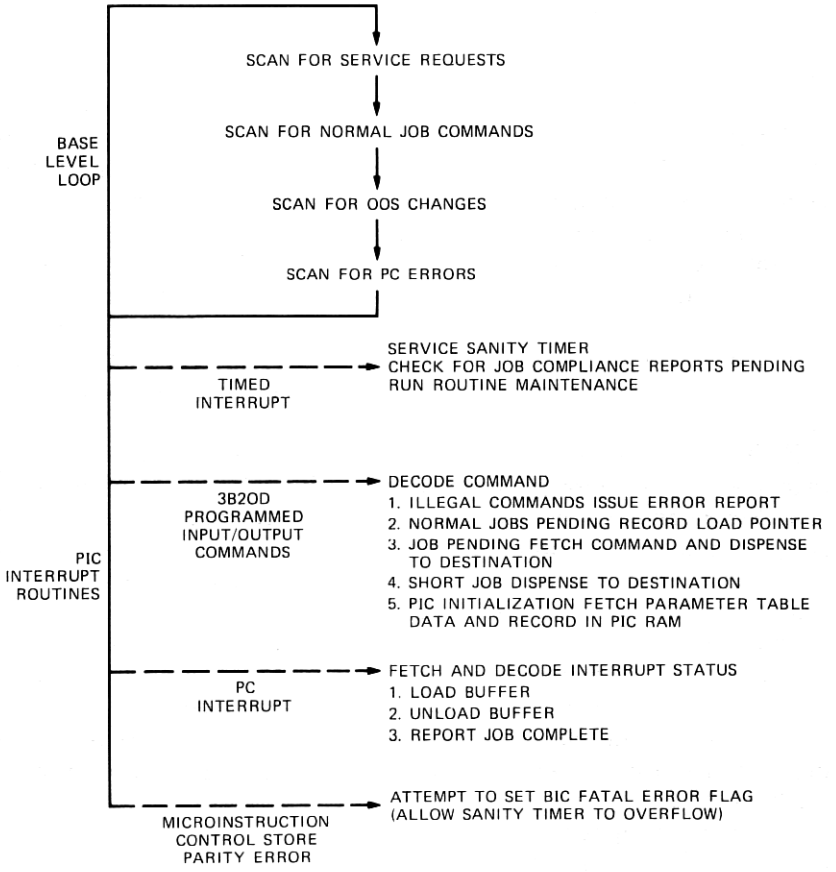


Fig. 4—Peripheral interface controller firmware structure.

performed and executes that job. Job sizes are restricted to a maximum transfer size of 256 bytes to guarantee the maximum time the peripheral controller is held out of dual-access memory. The peripheral controller breaks the transfer of a large block into small blocks to satisfy this requirement. Reports of completed jobs are buffered in an area of the peripheral interface controller data store.

After servicing peripheral controller requests, the peripheral interface controller compares the load and unload pointers in a job queue stored in 3B20D main memory. If the queue is not empty, the peripheral interface controller unloads up to eight jobs and issues them to the appropriate peripheral controllers or to the peripheral interface controller itself.

The peripheral interface controller next scans for changes in the peripheral controller community out-of-service status. A power con-

verter out of tolerance causes an out-of-service indicator to be automatically set in the input/output microprocessor interface. A high-priority error response is issued to report such a condition.

The peripheral interface controller next scans for peripheral controller error reports. Again, the peripheral interface controller reports this via a high-priority error response. The loop is complete at this point and the peripheral interface controller begins scanning for peripheral controller service requests once again.

Interrupts are employed to initiate time-critical tasks. A timed interrupt routine is entered at a fixed rate. The sanity timer is serviced at this time. Additionally, the job-completion queue is interrogated and, if not empty, is loaded into the response buffer in 3B20D main memory. The peripheral interface controller then interrupts the 3B20D.

High-priority jobs are handled using the interrupt mechanism. Upon registration of the message in the high-priority buffer in the peripheral controller dual-access memory, the peripheral controller interrupts the peripheral interface controller. The peripheral interface controller moves the buffer contents into 3B20D main memory. This process continues until the entire message has been transferred into 3B20D main memory. The peripheral controller then enters a job-completion report into the appropriate entry in the peripheral controller dual-access memory and again interrupts the peripheral interface controller. The peripheral interface controller then moves the report into the high-priority response buffer in 3B20D main memory and interrupts the 3B20D.

Programmed input/output commands arriving from the 3B20D also trigger a peripheral interface controller interrupt. The peripheral interface controller decodes the command to determine its type. New entries in the 3B20D main memory job queue are reported to the peripheral interface controller in this manner. A portion of the command contains the 3B20D load pointer for the job queue. The peripheral interface controller records this entry into the peripheral interface controller random access memory to be used during base level processing.

The 3B20D also uses programmed input/output commands to alert the peripheral interface controller of the presence of a high-priority job awaiting execution in the 3B20D main memory high-priority job register. The peripheral interface controller retrieves the command and immediately issues it to the specified peripheral controller.

Additionally, short jobs can be transferred to the peripheral interface controller via programmed input/output. Jobs of this sort are constrained to contain no more than 28 bits of command information. Only a single 32-bit programmed input/output command is used.

3.2 Peripheral controller software

3.2.1 Initialization

Peripheral controller initialization is initiated by a 3B20D Processor resident input/output processor fault recovery program. The fault recovery program can direct the peripheral interface controller, via a configuration command, to generate a clear signal for any one of the 16 addressable peripheral controllers within the input/output processor. The peripheral controller responds to the clear signal by transferring control to a bootstrap program resident in on-board read-only memory. Execution of the bootstrap program results in initialization of address pointers, flags, timers, interrupt control circuitry, and other parameters as required. In addition to the above mentioned items, the peripheral controller must initialize associated work pointers. On-board peripheral controller diagnostics are downloaded from the central processing unit during this stage.

Upon completion of initialization the peripheral controller microprocessor monitors sanity and responds to any commands placed in the peripheral controller work queues.

3.2.2 Peripheral interface controller/ peripheral controller microprocessor command execution

A peripheral controller microprocessor can be directed to perform certain tasks through the use of a command queue system. A peripheral controller can have as many as 15 queues, three of which are for generic type commands. The remaining queues are double ended (deques) and provide communication to peripheral controller subdevices. Each of the four peripheral controller subdevices has a transmit, receive, and asychronous report deque. Jobs are sent to the transmit and receive deques informing the peripheral controller subdevice to send and receive data respectively. Unlike the transmit and receive deques the asychronous report deque only sends subdevice reports, it never receives jobs for execution. The command queues and their associated pointers—load, unload, and current—reside in the peripheral controller dual-access memory. The peripheral interface controller, after entering a command in the queue, informs the peripheral controller by updating the queue load pointer. The peripheral controller detects the entry, processes the job, and inserts a completion report in the queue. The completion report includes a 2-byte completion code. The peripheral controller receives knowledge of the completion report via a service request. Retrieving the report, the peripheral interface controller sends it to 3B20D main memory and updates the unload pointer. The peripheral controller maintains a current job pointer, which allows the peripheral interface controller knowledge of which peripheral controller job is currently running.

A unique completion code has been reserved as an indication of an invalid command. Upon detection of this condition, the microprocessor immediately loads the invalid-command completion code byte into the appropriate field in the command/response entry and signals a job-completion report via a service request or interrupt request. At this point, the microprocessor is ready to accept the next entry in this command deque.

For high-priority command entries, the microprocessor may set an interrupt request indicator instead of setting a command completion service request indicator. The peripheral interface controller recognizes and responds to all peripheral controller interrupt request work before responding to peripheral controller service request work.

3.2.3 Peripheral controller microprocessor base level and interrupt level work

There are three modes of communication between the peripheral controller microprocessor, subdevices, and peripheral interface controller:

(*i*) Demand interrupt mode—The peripheral interface controller interrupts the peripheral controller microprocessor to indicate a high-priority command has been placed in the high-priority work deque. The peripheral controller microprocessor immediately responds to the command interrupt by accepting the command and executing the associated code as required. The subdevice may interrupt the peripheral controller microprocessor, indicating receive data is present. In this mode the microprocessor checks the data for errors and, if none are found, moves the data to an associated input buffer. When the input-buffer-full threshold is reached, the microprocessor uses a command address pointer to locate a receive command. The receive command contents are required to permit the peripheral controller microprocessor to initiate a direct memory access transfer of data.

(*ii*) Base level polling mode—In the base level program, all subdevices are polled in sequence for status and any required processing is performed. The status of all currently active direct memory access transfers is updated and new direct memory access jobs are initiated as required. Typically, all subdevices transmit data, and low-priority maintenance work is done during the base level loop.

(*iii*) Time interrupt mode—A real-time clock provides periodic interrupts to the peripheral controller microprocessor for the purpose of handling low-level periodic functions. Some examples of the functions processed in this mode are: checking subdevice status, maintenance of a sanity timer, and administration of routine maintenance diagnostics.

3.3 Peripheral interface controller firmware development tools

Available for development of peripheral interface controller firmware are (*i*) the Read-Only Memory Emulator and Trace Control Unit circuit packs and (*ii*) the Read-Only Memory Emulator and Trace System software system. These function jointly to support the peripheral interface controller as a test tool during hardware and firmware debugging. The read-only memory emulator and trace system has hardware control over the peripheral interface controller and provides program single-stepping and breakpointing capabilities. Program memory can be examined, modified, and downloaded, via a commercial microprocessor based system. A trace system is provided with three 16-bit by 256-word trace memories.

IV. INPUT/OUTPUT PROCESSOR UNIT AND POWER

The input/output processor unit and power is structured around the peripheral interface controller and the four communities of peripheral controllers as separate items. Each is located within its own housing and is separately powered. An overall power control switch is provided with the peripheral interface controller; and each peripheral controller community may be individually switched without affecting the others (see Fig. 5). Although several standard voltages are supplied in the backplane, other voltages can be supplied as part of the peripheral controller design, if required.

Since several of the problems of interfacing to various peripherals are related to the wide range of electrical and physical interfaces required, the input/output microprocessor interface bus connector field was separated from the peripheral device defined connector area for each peripheral controller slot. The entire upper half of each peripheral controller's backplane connector was reserved for definition by the particular application. Standard Bell System connectors are then used to generate a cable or harness that performs the translation to the specific connector of each physical device. The foundation peripheral controller for No. 5 ESS and the programmable link controller for the signal transfer point system used in the Common Channel Interoffice Signaling network are two examples of application peripheral controllers made possible by the flexibility of the input/output processor. Multiple board peripheral controllers have also been designed for special applications using connectorized straps in the upper connector area to interconnect the circuit packs that comprise the multiple board peripheral controller. Circuit pack slots are thus system defined, depending upon the peripherals needed by each application.

Because all four peripheral controller communities are on separate

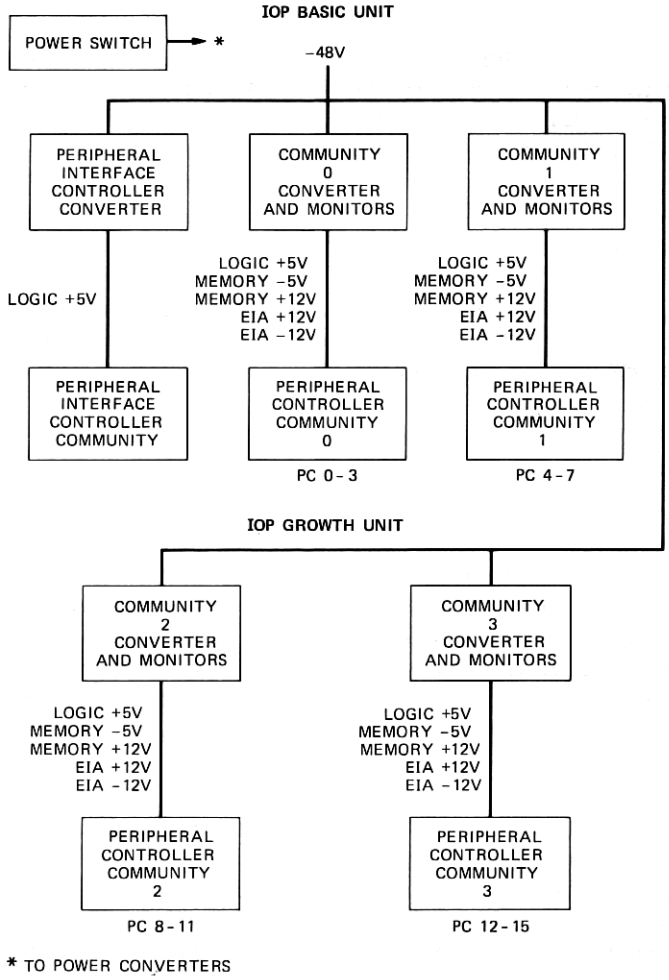


Fig. 5—Input/output processor power diagram.

power systems, peripheral controllers with special grounding or power requirements can be assigned to separate peripheral controller communities, effectively isolating them from the remaining communities, and providing separate power control.

V. 3B20D PERIPHERALS CURRENTLY SUPPORTED

The peripheral controllers and peripheral devices available on the 3B20D Processor are discussed in the following paragraphs.

5.1 Nine-track magnetic tape controller

Up to four 25-inches-per-second (IPS), 1600-bits-per-inch (BPI), phase-encoded (PE) nine-track tape transports can be accommodated per peripheral controller (only one transport may be accessed at a time via a shared formatter). The drive accepts up to 10.5-inch-diameter reels (2400 feet by 0.5-inch tapes). The interface is an RS422 differential bus up to 250 feet long using an industry standard data and control format. The controller can read and write any block size up to 2K bytes long, write file marks, seek forward or reverse, and can rewrite and reread blocks of data that exhibit errors. The instantaneous data rate is 40K bytes per second. The average data transfer rate for 2K data blocks is 26K bytes per second.

5.2 Scanner-signal distributor controller

Forty-eight scan points and 32 signal distributor points—which may be located up to 1000 feet away (assuming 26-gauge twisted pair)—are provided by this controller. Scanning is carried on autonomously or it is directed. A change of a scan state must remain in the new state for two consecutive scan cycles before it is recognized. The signal distributor can operate or release a point, or it can flash a point indefinitely or for a timed period. Both the scanner and the signal distributor are self protected to 140 volts, and can detect a foreign potential in the same range. A point scanned must be a floating contact or saturated transistor switch. Points are scanned once every 48 ms. The signal distributor can supply up to 5 mA into a floating load or loop. The scanner signal distributor was designed primarily to operate optical couplers and low-power relays. Scan or distribute points that are remote to the 3B20D Processor are interfaced through an optical coupler isolation circuit.

5.3 High speed nine-track magnetic tape controller

Up to four 12.5- to 125-IPS, 1600-BPI, phase encoded, start-stop transports, or up to four 25-IPS streaming transports with embedded formatter, in any mix, may be accommodated per peripheral controller. Access is via an RS 322 bus limited to 20 feet using an industry standard data and control format. Only one transport may be accessed at a time. The controller can write or read any block size up to 6K bytes as well as perform the same control functions as the peripheral controller described in 5.1. The instantaneous data transfer rate is 200K bytes per second. The average data transfer rate when reading 6K blocks of data is 150K bytes per second.

5.4 Synchronous data link peripheral controller

This circuit pack is a BX.25 level-2 synchronous link arranged for full duplex private line or dial backup operation, and it provides two independent channels at bit rates up to 9.6K b/s. The channels differ in that one supplies an automatic call unit port for dial-up and dial-in operations. The backup configuration is supported when equipped with Dataphone II generation data sets. The peripheral controller capacity is 9600 b/s full duplex. Each of the two BX.25 level-2 channels has an associated level-1 interface that is RS449/RS232C compatible.

5.5 Asynchronous data link peripheral controller

This peripheral controller provides two independently programmable full-duplex or half-duplex, asynchronous or isochronous channels at data rates up to 9.6K b/s. It can communicate with a local terminal directly or via a "null modem" connection, with a remote terminal via a modem and private line, or with a remote terminal via a modem and the public switched telephone facility. The user can specify the data rate, parity and stop code format in addition to input, output, and line discipline processing options. Each channel drives an optically isolated EIA RS232C standard port.

5.6 Maintenance teletypewriter peripheral controller

This peripheral controller is designed to permit manual intervention and control of the 3B20D control unit and peripherals through the maintenance terminal. It contains sufficient code in read-only memory to allow control of the system during routine or emergency maintenance procedures. Read-only memory code performs memory and device initialization, parsing of the 3B20D emergency action interface input and support for the maintenance terminal display and a remote maintenance center data link using BX.25 protocol. Down-loaded random access memory code provides additional functions such as support for teleterminal input/output syntax and other teleterminal display capabilities.

5.7 BX.25 data link interface

The data link controller provides a general-purpose high-speed synchronous BX.25 level-2 interface and a special high-capacity software interface to the 3B20D. The peripheral controller is equipped with an RS232C and a CCITT V.35 interface having data rates of up to 56K b/s.

VI. SUMMARY

The 3B20D Processor's input/output structure was designed around a high-speed direct memory access input/output channel, a high speed

multiplexor and direct memory access interface, and single- and multiple-board intelligent controllers, which provide device-specific front-end processing and interrupt handling. These components, associated with a common driver with device-specific handlers in the 3B20D Processor's operating system, successfully provide a flexible and effective interface to a wide range of peripheral devices.

VII. ACKNOWLEDGMENTS

The 3B20D Input/Output Processor and its peripherals are the result of a team effort of circuit, physical, and software design groups. The authors wish to acknowledge T. S. Greenwood, J. O. Becker, and R. T. Watters for their management support of the project, and the particular design efforts of T. T. Butler, D. J. Brahm, W. V. Lindquist, N. E. Stohs, J. L. Snapp, T. T. Towle, S. P. Michiels, M. G. Bradac, S. P. Davison, W. A. Hommowun, W. R. Hudgins, and R. M. Venzon, C. A. Mennitt, C. B. Kelley, W. G. Reichert, R. Huisman, all of whom helped to make the concepts become a reality.

REFERENCES

1. W. N. Toy and L. E. Gallaher, "Overview and Architecture of the 3B20D Processor," B.S.T.J., this issue.
2. M. W. Rolund, J. T. Beckett, and D. A. Harms, "3B20D Central Processing Unit," B.S.T.J., this issue.