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

The 3B20D Processor & DMERT Operating System:

Diagnostic Tests and Control Software

By J. L. QUINN, R. L. ENGRAM, and F. M. GOETZ

(Manuscript received March 10, 1982

Comprehensive diagnostic tests and multifeatured control software designed for execution on several host processors help craft to quickly isolate faulty hardware anywhere in the 3B20D Processor. Besides meeting the requirements for Bell System switching systems, the 3B20D diagnostics provide a high degree of modularity and portability using an operating-system-based structure. The diagnostics are used in a wide range of development, production, and maintenance activities throughout the project life cycle. Many features of the system architecture and hardware are provided to allow thorough diagnosis in a time-shared noninterfering manner. Additional features are provided in the diagnostic control structure to extend the DMERT diagnostic capabilities to application systems based on the 3B20D Processor.

I. INTRODUCTION

Many of the diagnostic principles and features embodied in the 1A Processor[1] have been incorporated in the maintenance design for the 3B20D Processor. These design principles include: (*i*) use of a special-purpose test-design language that facilitates test interpretation; (*ii*) use of a table-driven control program approach; (*iii*) use of a common test data base covering all hardware versions of the 3B20D Processor; (*iv*) partitioning of diagnostic tests into phases associated with specific hardware functions; (*v*) control features allowing selective test execution and variable degrees of detail in outputted results; and, (*vi*) optional automatic trouble location. In the 3B20D Processor design, however, a more general diagnostic design approach was followed. This approach resulted in a more portable diagnostic control structure;

it allowed diagnostic execution in several environments: factory testing using a support processor, installation testing using a remotely located processor, and in-service, on-line testing of a standby mate processor.

II. OBJECTIVES

As with earlier processor designs, the 3B20D Processor diagnostics must be effective and efficient in fault detection, provide consistent test results, protect the contents of memory, be noninterfering with normal system operation, allow automatic trouble location, and be easy to maintain and update. In addition to meeting these objectives, the 3B20D diagnostics were required to be:

(*i*) Portable—The diagnostic software must execute in several environments. Throughout this paper the execution environment is referred to as the host processor (or computer).

(*ii*) Flexible—The diagnostics would test multiple system configurations containing various vintages of circuits.

(*iii*) Modular—Standard control interfaces must accommodate differing test access facilities to the processor under test, input/output facilities, and DMERT application processes that are used to diagnose application-dependent hardware that interfaces to the 3B20D Processor.

(*iv*) DMERT compatible—Diagnostics must be integral with, rather than separate from, the operating system.[2,3]

To meet these design objectives, the diagnostic control structure was designed as an integral part of the operating system and had to support the evolutionary stages of development.

III. DIAGNOSTIC ENVIRONMENTS

As shown in Fig. 1, the 3B20D Processor can be diagnosed from several execution environments. During the early phase of processor development, a local host computer was used to support hardware, software, and diagnostic design. This access arrangement continues to be used in factory testing. Later in the development, more efficient use was made of the host computer by providing access to a remote target 3B20D Processor over a dial-up telephone line. Ultimately, in the standard duplex-system configuration, the active control unit is capable of diagnosing its own peripheral controllers and the standby control unit. Each of these access arrangements is discussed below.

3.1 Local host diagnostics

Figure 1a shows three local-host access arrangements. In the first, diagnostic programs executing in a host computer send test inputs and receive test results over a standard communications port to a Micro-level Test Set (MLTS). The MLTS connects directly to the 3B20D

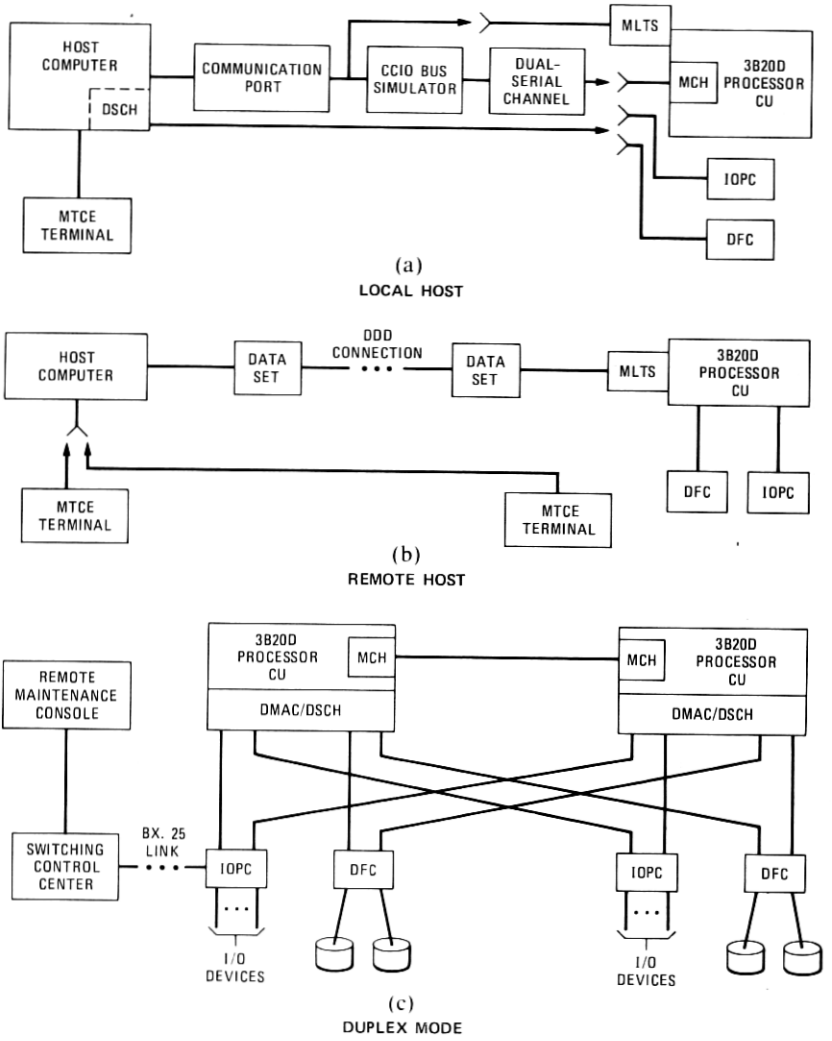

Fig. 1—The 3B20D Processor diagnostic environments.

control unit backplane, and provides complete access and control of the processor's microprogram control circuitry. For the second access path, a circuit was designed to simulate the Central Control Input/Output (CCIO) internal bus. The CCIO Bus Simulator (BS) is accessible using a standard communication input port. A Dual Serial Channel (DSCH) connected to the CCIO/BS can then communicate directly with a Maintenance Channel (MCH), the circuit designed for control-unit access. Like the MLTS the MCH can access the central control at a low level. However, only the MCH is used in the duplex configu-

ration (see Section 3.3); it communicates with either another MCH or a DSCH. As shown, the CCIO/BS–DSCH access path can also be used to diagnose the Input/Output Processor Controller (IOPC) and the Disk File Controller (DFC). Notice that when the local host is a 3B20D Processor, the path is from the DSCH of the host 3B20D Processor to the MCH, IOPC, or DFC of the target machine.

3.2 Remote host diagnostics

The DSCH is designed to communicate over distances of approximately 100 feet. Remote-host (Fig. 1b) access arrangements can be used for diagnosing over longer distances. Using data sets and a telephone line, tests stored and executed on a remote computer can be applied through the MLTS to the control unit. Peripheral controllers (IOPC and DFC) can also be diagnosed by downloading tests into the control unit and executing them. Although remote-host diagnostics are useful in cases where a local host is unavailable, execution performance is limited by the transmission facilities used.

3.3 Duplex mode diagnostics

The primary diagnostic execution environment is the 3B20D Duplex Processor (Fig. 1c). The active (on-line) processor acts as a local host for diagnosing the standby (off-line) processor. An MCH-to-MCH link provides the access path for testing the control unit. In the duplex mode, the DFC and IOPC are diagnosed from the on-line control unit using the operational interface path, a DSCH attached to the Direct Memory Access Controller (DMAC). Tests of the links from the off-line processor to the peripherals also can be run under the control of the active processor. As shown in Fig. 1c, the duplex system configuration also supports remote monitoring and control of diagnostics over a dedicated link to a Switching Control Center (SCC).

3.4 Multiple-target processors

Although the target processor is always a 3B20D Processor, it can be of many types, versions, and sizes. The diagnostic control program accounts for these differences by referencing the Equipment Configuration Database (ECD). All information relevant to the particular diagnostic tests that should be applied to each hardware unit is contained in the ECD. This information includes the name of each hardware unit within a subsystem, subunits, and their logical interconnections, equipage options, and auxiliary information, such as channel address and baud rate. Whenever a circuit design is originated or updated, diagnostic tests are designed and appropriate ECD changes are specified.

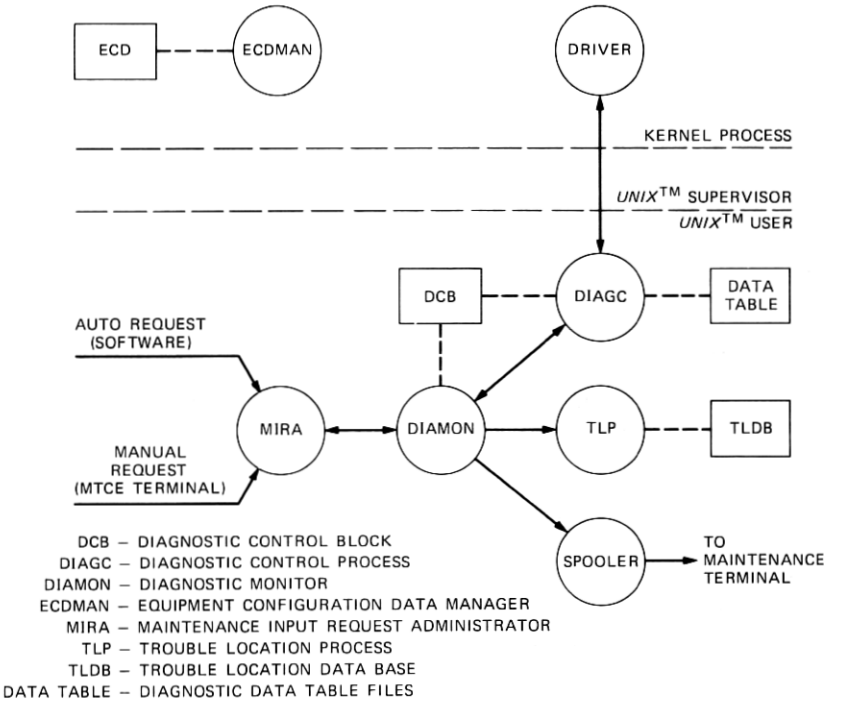

Fig. 2—Diagnostic control structure.

IV. DIAGNOSTIC CONTROL STRUCTURE

The diagnostic control structure is depicted in Fig. 2. At the kernel process level are the modules that provide access to the ECD or drive the communication links previously discussed. The *UNIX** operating system supervisor resides at the supervisor level,[3] and provides a protected environment and operating system services for the higher-level processes. The modules operating under the *UNIX* operating system that pertain exclusively to diagnostics are: the Maintenance Input Request Administrator (MIRA), the Diagnostic Monitor (DIAMON), the Diagnostic Control process (DIAGC), and the Trouble Locating Process (TLP). Output messsages from the diagnostic structure are sent to the system spooler for printing. The first three of these modules are discussed below; the TLP is described in Section VII.

4.1 MIRA

Scheduling and dispatching maintenance requests is the function of MIRA, the front-end process of the diagnostic structure. MIRA main-

* Trademark of Bell Laboratories.

tains a waiting queue and an active queue to administer each service request. Requests are serviced according to priority and resource availability; manual requests have higher priority than those initiated automatically. For each service request, MIRA spawns a DIAMON process and sends it a message. When the request is completed, DIAMON sends a message back to MIRA. Interfaces are provided in MIRA to administer routine exercise requests and inputs from the error-interrupt handler.[4]

Nine general types of request are handled by MIRA; they are described in Table I.

4.2 DIAMON

Execution of each diagnostic is directed from start to finish by DIAMON. Specifically, DIAMON will:

(*i*) Initiate and control the actual diagnostic processes specified in a message from MIRA.

(*ii*) Communicate with the Equipment Configuration Database Manager (ECDMAN) and the appropriate device driver (the software control module for a particular hardware unit) to extract control data from the ECD and retrieve path names of related utility files.

(*iii*) Build the diagnostic control block containing all the data required by a diagnostic.

(*iv*) Spawn the appropriate diagnostic control process (DIAGC); separate processes are provided for the control unit and peripherals.

(*v*) Communicate diagnostic output to MIRA and the output spooler.

(*vi*) Spawn the remove and restore processes.

(*vii*) Interface with the TLP.

Table I—Description of diagnostic requests to MIRA

Command	Description
Diagnose (DGN)	Diagnoses the unit specified in the request.
Remove (RMV)	Removes the specified unit from service.
Restore (RST)	Diagnoses the unit and restores it to service if all tests pass (ATP).
Restore Unconditional (RSTU)	Restores the unit to service without running the diagnostic.
Exercise (EX)	Starts the diagnostic in the interactive mode. This command allows stepping to a particular test, pausing, or looping over a diagnostic "phase" (ie., group of functionally related tests) segment.
Terminate (STOP)	Stops execution of a diagnostic.
Display (OP)	Displays status of queued requests in MIRA.
Inhibit (INH)	Inhibits diagnostic requests from other processes that automatically or routinely initiate diagnostics.
Allow (ALW)	Allows diagnostic sources, canceling any active INH request.

4.3 DIAGC

DIAGC is a generic name that refers to a class of processes. The DIAGC is a unit or application-dependent module that controls execution of tests. Containing all unit-dependent task routines, DIAGC translates the interpretive diagnostics and provides the interface with DIAMON. A unit's diagnostic phase table (DPT) contains the name of a particular DIAGC process to be used in the diagnosis. DIAMON imposes no limit on the number of processes that can interface with it. The following functions are provided by DIAGC:

(*i*) Opens the diagnostic driver
(*ii*) Shares the buffer (DCB) provided by DIAMON
(*iii*) Initializes the raw data buffer
(*iv*) Executes the diagnostic
(*v*) Computes the test results
(*vi*) Provides interactive control if required
(*vii*) Provides an interface to DIAMON for test results and abnormal terminations (aborts).

4.4 Portability

All diagnostic control modules are written in the C language and execute in the *UNIX* operating system environment. This facilitates the porting of the control structure to other host processors that support C and *UNIX* operating system software. Variations of processor configuration and hardware vintage can be described in the ECD. DMERT application processes can provide additional DIAGCs and Data Tables to control diagnostic test execution for interfacing hardware. Several driver processes can be supported to allow diagnostics to be executed over standard communication ports, dual-serial channels, or maintenance channels.

V. MAINTENANCE FEATURES

The combination of hardware-access circuits and modular-control programs, just discussed, provides the 3B20D Processor with considerable maintenance flexibility. Tests are selected according to the vintage of circuit under diagnosis. Displayed in Fig. 3 is a typical diagnostic input message in the PDS (Program Documentation Standards)[5] syntax, one of three command languages supported by DMERT. Requests can be made to diagnose an entire unit, a particular subunit, or all subunits in a specified community. Individual test phases or ranges of phases can be executed and the results printed with optional amounts of detail. Some diagnostic test phases—because of either their long execution time requirements or their dependency on other system hardware availability—are restricted to manual initiation. In-

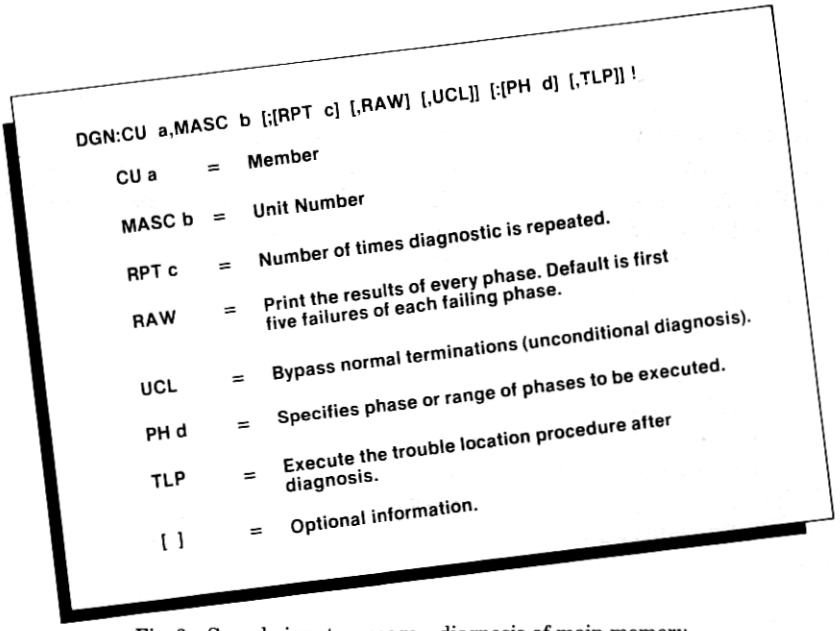

Fig. 3—Sample input message—diagnosis of main memory.

teractive features such as stepping, pausing, and looping are provided for facilitating difficult repairs. Units can be restored to service automatically if they pass all tests. Several host computer versions are supported along with application-dependent interfaces.

Diagnostics can be initiated either manually or automatically. Manual requests can be entered from either a local maintenance terminal or through a work-station terminal associated with a Switching Control Center System (SCCS) connected to the 3B20D Processor via a synchronous data link. Automatic requests originate from other software modules such as the error-interrupt handler, the routine exercise scheduler, or an application-software module.

VI. DIAGNOSTIC TEST DESIGN

6.1 General

The stringent availability requirements of Bell System applications using the 3B20D Processors had a significant impact on all aspects of system design. Diagnostic and maintenance engineers were actively involved in meeting these goals commencing with the initial architectural planning and requirements generation. Many hardware features are provided to monitor system integrity, to detect errors, to reconfigure the system, and to facilitate repair of the faulty equipment.[4,6] Although some of the features are for fault isolation during pack repairs, most are used at the system level to effect repair through pack

replacement. Diagnostics, the primary repair capability for the system, make extensive use of these hardware features for control and observation of the circuitry.

6.1.1 Circuit pack tests

The initial factory testing of circuit packs uses test vectors, which can be applied at terminals of the pack connector in a commercially available computer-controlled test set. Most of the vectors are generated independently from system-level diagnostic tests. The packs are given additional tests in a 3B20D system test bed using diagnostics and some operational sequences.

6.1.2 System-level tests

All 3B20D Processor diagnostics run under the DMERT operating system as user-level processes. To communicate with the unit being tested, the user-level process passes the test scenarios to a kernel process that interfaces to the hardware. Each of these kernel process drivers runs at its standard system priority level to perform the tests. If some time-critical tests are necessary, the priority level can be elevated to avoid interruption by other system processes.

Each of the diagnostic programs is structured to avoid any negative impact on the normal system functions. Special driver functions allow the drivers to handle error conditions generated by the diagnostic tests, thereby avoiding the normal error-handling routines. Since many fault conditions result in system errors, this capability is especially vital to allow thorough testing in the operating system environment.

In the Control Unit (CU) diagnostic, additional safeguards are implemented to assure proper handling of the system recovery and integrity hardware. Even with faults in the off-line CU integrity circuits, the system will maintain normal functionality during CU diagnostics.

The system diagnostics are organized on a unit basis, for example the Control Unit (CU), I/O processor (IOP) or Disk Controller (DFC). Each unit diagnostic is structured into test phases that pertain to a particular subunit. The phases are organized in a hierarchical fashion beginning with the more elemental operations and applying to the hard core of a subunit. Subsequent phases expand in complexity and in the totality of the circuitry exercised.

The user-level diagnostic processes, namely the control program and the test data tables, contain all of the information necessary for control and sequencing of tests. The control program has the interpretative routines for decoding each data table test statement. The program also can use various system configuration parameters, test results, and data table decision functions to modify program flow or to terminate the diagnostic.

6.1.3 Maintenance channel access

As shown in Fig. 1c, the primary interconnection between the Control Units is the maintenance channel (MCH). This circuit is an enhanced version of the dual-serial channel, with special capabilities aimed at maintenance access and control of the off-line CU. It provides the ability to run, stop, load, clear, and step the CU. The MCH allows the active CU to read some off-line CU registers directly and others indirectly using microcoded sequences. Diagnostic-test programs can be loaded through the MCH to the off-line microstore or mainstore. The MCH is controlled by a DMERT kernel process driver that carries out the diagnostic test sequences.

6.2 Control unit diagnostics

The Control Unit (CU) has seven types of subunits: Central Control (CC), Main Memory Store (MAS), Store Address Translation (SAT), Direct Memory Access (DMA), I/O Channels (DSCH), Cache (CSU), and Utility Circuit (UC). The latter three are optionally equipped; the UC is normally used for program testing and is not further discussed herein. Some 3B20D Processor applications have special circuits that are part of the CU; diagnostics for them are concatenated to the CU diagnostic. A pictorial view of the CU hierarchy is shown in Fig. 4, which depicts the multiple levels of units as defined in the ECD.

6.2.1 Central control tests

The first CU subunit tested is the CC, which contains the core of the CU. The CC diagnostics in turn test the maintenance channel, microcontrol logic and memory, registers, data-manipulation logic, memory access, timers, interrupts, I/O interface, error-control hard-

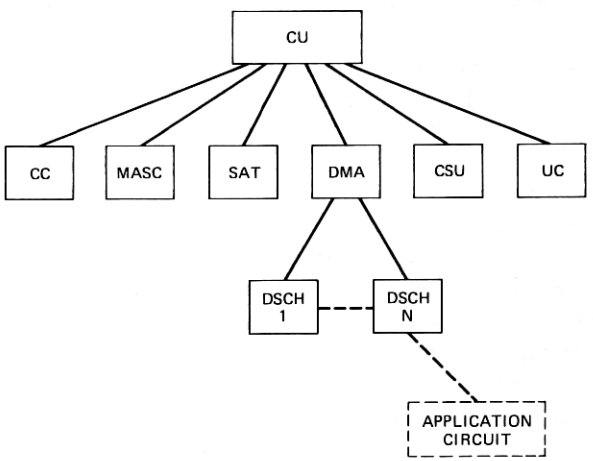

Fig. 4—Unit hierarchy of the control unit.

ware and integrity circuits. The testing uses a series of MCH operations for the basic tests. For the more complex test routines, down-loaded microcoded test programs are executed by the CU under test. The CC is extensively exercised in tests of the remaining subunits.

6.2.2 Memory tests

Initial testing checks out the basic memory-controller operations and the control and data paths from CC to the MAS. Tests are carried out on the error-detection and correction circuits in preparation for using them in array testing. Testing of the memory arrays (up to 16M bytes can be equipped) is with down-loaded microdiagnostic routines. Since the test-pattern programs are executing autonomously in the CU under test, all of its real time is used for testing. Whenever a hardware error is generated, control of the CU passes to a diagnostic error handler. The combination of self error detection and the micro-routines allows extensive pattern checking to be executed rapidly over the complete memory spectrum.

6.2.3 Store address translation tests

Functional tests are performed via MCH on all of the SAT control logic. The memory cells are then tested with various microcoded test patterns. The remainder of the tests, implemented as microdiagnostics, check out the multiplexor, compare logic, matchers, protection logic, and SAT to MAS interface.

6.2.4 Cache tests

The cache is comprised of a high-speed four-way-set associative memory and a 2K by 36-bit interrupt stack. The diagnostic performs extensive tests of the memory cells, matchers, and select logic. In addition to functional tests, a special diagnostic routine called the cache exerciser is used to stress, at high data rates, the cache interfaces to CC, MAS, and SAT. This kind of testing is effective at detecting marginal fault conditions.

6.2.5 Direct memory access tests

The DMA diagnostic checks out the CC-to-DMA communication and control paths, the internal DMA functionality and the DMA operations to MAS. Many of the tests are coded into ROM (Read Only Memory) contained in the DMA. The remainder consist of down-loaded microcoded tests and off-line, main-memory, resident test programs. The final sequence of tests verifies the DMA cache "handshaking" operations. It is noteworthy that in the DMA diagnostics, except for control and down loading through the MCH, all test sequences are executed completely by the CU under test.

6.2.6 Channel tests

The channel diagnostic carries out the remainder of the testing of the CU's I/O capability. Basic tests are performed on the communication and handshaking of the CU to all in-service system peripherals. More exhaustive tests (demand diagnostics) can be specified by the maintenance personnel for troubleshooting more elusive problems. These diagnostic phases require that a peripheral unit be configured as a "helper unit" (specified in the diagnostic input message) to allow the CU to carry out peripheral operations at a high rate.

6.3 Peripheral unit diagnostics

The Disk File Controller (DFC) and Input Output Processor (IOP) are described in Refs. 7 and 8. The DFC can control up to eight Moving Head Disks (MHD) of various capacities, types and manufacturers. The Peripheral Controllers (PC), which are under control of the IOP, are special-purpose I/O units described in Ref. 8. The testing for the DFC and IOP, which share a common front end, is primarily carried out under control of the on-line CU. Since both of these are intelligent controllers, many of the specific tests can be executed autonomously. The peripheral diagnostics utilize the DMERT kernel process drivers to interface to the hardware. Throughout these diagnostics, extensive use is made of driver-maintenance orders and special handling of error conditions.

6.3.1 IOP and DFC tests

The peripheral diagnostics use common-control programs IODIAG and DFDIAG that contain all the CU resident tests and control routines. Separate sets of data table and down-loaded microcode files are used for each unit diagnostic. The overall sequence of testing proceeds from CU/controller interface to complex internal controller operations. Most of the latter make use of the operational firmware in the controller to carry out the test sequences. The more complicated controller tests are part of the resident diagnostic firmware and are initiated by special-driver operations. At the successful conclusion of DFC or IOPC testing, the unit is restored to service to allow testing to proceed on MHD or PC circuits.

6.3.2 Moving head disk tests

Relatively limited maintenance capabilities are provided in the MHD itself. Most of the testing is carried out by the firmware routines in the DFC. To provide an overall check on MHD performance, one cylinder is devoted to diagnostic testing of each read/write head. The error-detection/correction capabilities can also be checked using this

area. As each MHD is tested successfully it can be updated from its mate copy and restored to service.

6.3.3 Peripheral-controller tests

Each controller is microprocessor controlled and can carry out most of its diagnostics autonomously. Some of the tests are firmware resident in the PC's. The remainder of the diagnostic routines are downloaded into the PCs' RAM (Random Access Memory). Some types of PCs also can exercise the units they control, for example a tape transport, and report back the results for use by the maintenance personnel.

VII. TROUBLE LOCATING PROCESS

If the diagnostic request specifies the TLP option, the TLP process is invoked at the completion of diagnostic testing. The process compares characteristics of the failures with a resident data base of fault signatures. In each data table, the designers have partitioned the tests into groups. Any test failure in a group will set a flag bit, called a key, which is permanently assigned to the group. The TLP search, based on the phase and key information, results in a rank-ordered list of closest signatures and, ultimately, into an ordered list of suspected faulty equipment. This approach makes the data base and process less sensitive than earlier methods to circuit or test changes and to marginal failures. The data base (TLDB) is generated off-line from the results of physically inserting faults into units in a test laboratory. Test engineers also can modify the TLDB directly by inserting information into the test data tables. Fig. 5 depicts a typical diagnostic output message from a faulty memory unit.

VIII. EVALUATION

Although many diagnostic tests were generated with the aid of hardware logic simulators, many tests were developed manually. To assure that the diagnostics met the objective—at least 90 percent of the simulated faults detected—an extensive evaluation process was carried out. Using physical fault insertion at the DIP (Dual In-Line Package) terminals, many thousands of faults were inserted. This approach has provided timely and effective design feedback for diagnostic test and TLDB development.

IX. CONCLUSION

In addition to providing a variety of test-control options, the 3B20D diagnostics were designed for multiple execution environments. As a result, the diagnostics have been useful throughout the development

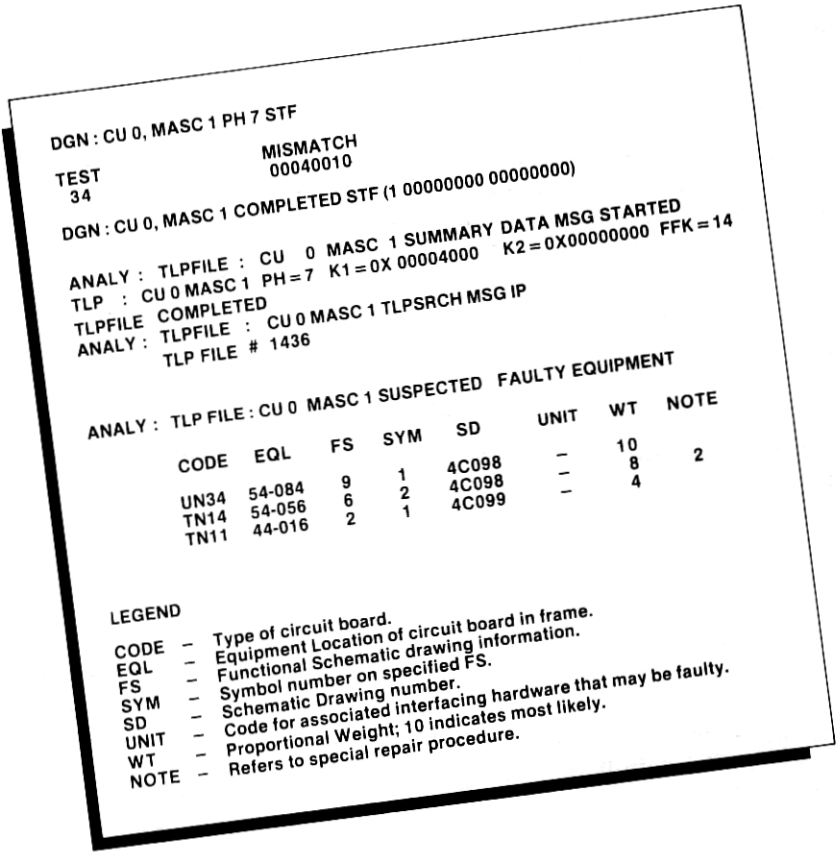

Fig. 5—Sample output message—diagnosis of memory with TLP option.

cycle and have supported the design laboratory, factory testing, installation, normal system operation, application interfaces, and field support. These diagnostics are the major tool for validating the 3B20D Processor hardware and for isolating any faults. The provision of a high degree of hardware self-checking, standby and active redundancy, self-diagnosis, microdiagnostics, and remote testing capability all have contributed to making the 3B20D Processor a high-availability real-time system. Coupled with the DMERT operating system, with its robust complement of features, the 3B20D Processor meets the needs of a wide variety of Bell System projects.

X. ACKNOWLEDGMENTS

The authors would like to acknowledge the many designers and testers at Bell Laboratories and Western Electric whose combined efforts resulted in the diagnostic package described herein.

REFERENCES

1. P. W. Bowman, M. R. Dubman, F. M. Goetz, R. F. Kranzman, E. H. Stredde, and R. J. Watters, "1A Processor: Maintenance Software," B.S.T.J., *56*, No. 2 (February 1977), pp. 255–88.
2. J. R. Kane, R. E. Anderson, and P. S. McCabe, "The 3B20D Processor & DMERT Operating System: Overview, Architecture, and Performance of DMERT," B.S.T.J., this issue.
3. M. E. Grzelakowski, J. H. Campbell, and M. R. Dubman, "The 3B20D Processor & DMERT Operating System: DMERT Operating System," B.S.T.J., this issue.
4. R. C. Hansen, R. W. Peterson, and N. O. Whittington, "The 3B20D Processor & DMERT Operating System: Fault Detection and Recovery," B.S.T.J., this issue.
5. M. E. Barton and D. A. Schmitt, "The 3B20D Processor & DMERT Operating System: Craft Interface," B.S.T.J., this issue.
6. M. W. Rolund, J. T. Beckett, and D. A. Harms, "The 3B20D Processor & DMERT Operating System: Central Processing Unit," B.S.T.J., this issue.
7. R. E. Haglund and L. D. Peterson, "The 3B20D Processor & DMERT Operating System: 3B20D File Memory Systems," B.S.T.J., this issue.
8. A. H. Budlong and F. W. Wendland, "The 3B20D Processor & DMERT Operating System: Input/Output System," B.S.T.J., this issue.