DEVELOPMENT OF DCC SOFTWARE DYNAMIC TEST FACILITY- PAST AND FUTURE

A.M. McDonald, N.D. Thai, W.J. Buijs, B.C. Wang and C.R.A. Burgis

Atomic Energy of Canada Limited
2251 Speakman Drive, Mississauga
Ontario, Canada L5K 1B2

ABSTRACT

This paper describes a test facility for future dynamic testing of DCC software used in the control computers of CANDU® nuclear power stations. It is a network of three computers: the DCC emulator, the dynamic CANDU plant simulator and the testing computer. Shared network files are used for input/output data exchange between computers. The DCC emulator runs directly on the binary image of the DCC software. The dynamic CANDU plant simulator accepts control signals from the DCC emulator and returns realistic plant behaviour. The testing computer accepts test scripts written in AECL Test Language. Both dynamic tests and static tests may be performed on the DCC software to verify control program outputs and dynamic responses

1. INTRODUCTION

The CANDU plant is controlled and monitored by two 'Digital Control Computers' commonly known as DCCX and DCCY. Each computer is capable of running the plant by itself. One is normally in control and the other is on stand-by. A third computer known as DCCZ is available for off-line software and hardware maintenance. Traditionally the computers used for the DCC's are VARIAN.

Earlier CANDU-6 DCC software was developed to run on VARIAN computers, using a VARIAN development system. For later CANDU-6 stations, although the DCC computer hardware remains relatively unchanged, new software tools have been developed by AECL for the DCC software designers to work more efficiently. The new tools include cross-development software that enables software developers to use IBM-PC compatible computers to generate code for VARIAN computers.

AECL has developed a series of tools for cross-platform software development. They include a cross-assembler, a cross-linker, a debugger and an emulator*. For the same source code, the machine code generated by the cross-assembler and linker in the PC is **identical** to the machine code generated by the VARIAN development system. Moreover, the emulator, which emulates the VARIAN computer in the PC environment, enables the DCC software to be run on a PC. The developer can debug and test the DCC software as if it were running in the VARIAN environment. The DCC software is transferred from the PC to the actual DCC computer for the final testing phase before being issued for use.

These tools open up an opportunity for the DCC software developers to take advantage of the software tools and techniques available for the PC. It is now possible to perform DCC software development and testing more effectively

* In this paper the word "emulator" is used to describe a software program that enables one computer to imitate a different computer and its peripherals. The word "simulator" is used to describe a software program that approximates a set of real hardware devices as observed from a particular interface.

by using the newly developed software tools[1]. This paper describes one such application in setting up a test facility for future dynamic functional testing of the DCC software.

2. SYSTEM TEST CONFIGURATIONS

There are three test configurations that are distinguished by the number of computers used in the set-up. The simplest configuration uses only one computer. It is most commonly used for static testing DCC software of repeat stations. The second configuration uses two computer for dynamic testing of the DCC software. The multi-computer (three or more) set-up is being developed for future application. All three configurations are discussed below.

2.1 Single-computer test set-up

In the single-computer test set-up, the testing software and the DCC software-under-test run together in the same computer, i.e., DCCX as shown in Figure 1. The advantage of this arrangement is that all I/O points are internally accessible without actual wiring. The test hardware is simple to set up. For some tests the testing software may not be needed. Off-line diagnostic utilities of the DCC software may be used as the testing software. Voltage sources, current sources and toggle switches are used for external application of test input signals. Indicators such as LEDs and meters are used for monitoring external test outputs.

The disadvantage of the single-computer arrangement is that the testing programs have to reside in the same memory space as the DCC software. Its testing functionality may be constrained by the limited memory space that can set limits on the testing demands such as demands of dynamic testing. The performance of the software-under-test may also be affected by the testing software. This is the main limitation that restricts the single-computer test set-up to simple static tests. Moreover, the testing software must be written in the same assembly language as the DCC software, which may require more effort than the use of higher level languages.

The single computer test set-up is usually reserved for static I/O manipulation and for accessing internal variables that are not available for observation on the DCC video displays. It is generally limited to static test cases where the behaviour of the program is observed one step at a time. A typical arrangement using the single computer test set-up is shown in Figure 1.

2.2 Dual-computer test set-up

In a dual-computer test set-up the testing software is installed in one computer and the DCC software is installed in a separate computer. When both control computers are available during development, it may be convenient to use DCCY as a testing computer connected to DCCX that runs the software-under-test. This arrangement is shown in Figure 2. It overcomes some constraints of the single-computer test set-up. The testing software is more easily implemented than in the single-computer configuration because the full multi-tasking capabilities of the DCCY are at the disposal of the test software developers. In fact high-level languages such as FORTRAN have been used in preparing test programs for the dual-computer test set-up[2]. Complex programs such as process simulators may be run in the DCCY to provide dynamic testing of closed-loop control programs.

However, only a subset of the DCC programs can be tested at the same time because the number of AO's available from DCCY is smaller than the number of AI's on DCCX. To test different programs the I/O has to be either physically re-wired or re-allocated by software. These changes, which may be necessary between tests, place higher demands on the test configuration control and the test schedule takes longer to complete.

2.3 Multi-computer test set-up

There is a need for a test facility that can provide automated testing to improve on test efficiency and schedule. Towards this end, the multi-computer test set-up is being developed. It takes advantage of new cost-effective computer technology and testing tool concepts developed for safety-critical software. The multi-computer test set-up uses a separate testing computer for running AECL-developed test control and display software. New advancement in information technology will make test results available for wider peer reviews and allow contributions from all technical disciplines involved in the conceptual development of the CANDU plant control programs.

The new multi-computer test set-up takes advantage of the Local Area Network for data exchange between the testing computer and the target computers. Three PC computers are used in the set-up shown in Figure 3. The first computer is the testing computer running the test control and display programs. The second computer is the DCC emulator running the DCC software to be tested. The third computer simulates the CANDU plant. Currently one work-station acts as an emulator for either DCCX or DCCY. It is possible to add separate work-stations for DCCX and DCCY so that the transfer of control between DCCX and DCCY may be tested. Additional computers can be added to this basic network for extended operator interface or system simulation.

The DCC emulator runs directly on the binary image of the DCC software captured from the actual DCC computer and transferred to the PC. The DCC emulator is written in the PC assembly language. It emulates the DCC CPU instruction set, the DCC peripheral hardware and the DCC operating environment. The dynamic CANDU plant simulator, written in FORTRAN language, is a simplified version of the CANDU-6 model. The test control and display program are written in visual programming language Labview, while the testing programs are prepared as test scripts written in AECL Test Language[1]. These features demonstrate the extensibility of the test facility to embrace diverse languages and platforms.

The DCC input and output files are shared network files as shown in Figures 4 and 5. The DCC input files contain all current DCC analogue input voltages and digital input states. The DCC output files contain all current DCC analogue output values and digital output states. The actual I/O address is used to determine the location of each I/O point in the file. The DCC display files contain the screen information normally displayed on the DCC video monitors in the Main Control Room or output to the printer. The DCC emulator file access is limited to its own I/O files.

The Plant simulator reads the Plant input files and writes its responses and other plant parameters to the Plant output file. The Plant simulator responds only to the values contained in its input files. The status of the Plant at any time is represented by its input and output files. The Plant simulator file access is limited to its own I/O files.

The testing computer has access to the I/O files of both the DCC emulator and the Plant simulator. One of its tasks is to "soft-wire" the DCC to the Plant. It performs this function by converting and copying the Plant outputs to appropriate DCC inputs and converting and copying the DCC outputs to appropriate Plant inputs. The user can disable this function for individual I/Os to simulate open-circuit signals during a test and independently controls either the DCC emulator or the Plant simulator.

The testing computer runs AECL Test Language Interpreter[1] that can automate the testing functions, generate test scenarios from test scripts, log test results and provide summaries for test reports. The use of test scripts enables the tester to prepare the test cases without the need to learn about computer programming. The use of simulated I/Os instead of hardwired I/Os greatly reduces the cost of the system. The testing computer can also select and capture a DCC display file and show it in a window at the user request. User interface to the DCC is provided by simulated keyboards with the same functionality of the actual panel-mounted keyboards.

The advantage of the multiple-computer test set-up using LAN is its cost effectiveness and flexible usage. Different development teams can work independently on different work-stations of the system. The complete system can be used in other applications such as software development, system analysis, simulation work or operator training.

3. DCC SOFTWARE TO BE TESTED

The DCC software high level design starts from the program specifications that are prepared by the system designers. These program specifications form the basis for the DCC source code. Assembly language is used for source code development to maximise real-time performance and minimise resource (e.g. memory and disk space) requirements.

The DCC binary core image or machine code is generated from the source code using conventional methods to produce the on-line and off-line software. The on-line software performs station control, data acquisition and display, alarm annunciation and data logging functions. The off-line software is used for software generation, modification or diagnostic purposes.

The on-line operating system is called the 'Executive'. It schedules or activates various tasks in accordance with their priority and manages the resource allocation for each task. Tasks signal the Executive when an action is to be initiated and the Executive or an interrupt identifies when an action is complete. The schedule of a task can be periodic, on demand or dependent on the outcome of other tasks. The Executive loads software modules into memory as required. Memory overlays are used to reduce memory requirements.

Once the software is built, it undergoes extensive testing at the module level, sub-system level and system integration level, with revisions to the source code tracked by standard software configuration control tools. The testing activities may take up to a year or more to complete.

The DCC software is loaded into the DCC emulator from a disk file containing a complete set of core image files and command files that are used to generate them. The command file is used to re-generate the core image in the PC for cross-checking with the core image on the disk to ensure that all the patches are properly installed. The Varian DOS *LIBS* program is used to capture the core image in the Varian. The Varian DOS *FLIBS* program is used to transfer core images via a data link to and from the PC computer.

In the PC computer, the PC-based FHD/MegaRam Disk Manager is used to capture the core image. This utility mimics the most useful functions of *LIBS* and *FLIBS*. It also provides file transfer functions between FHD/MegaRam disk file and MS-DOS. It enables the DCC software configuration control to be maintainable by the PC that can perform this task more effectively than the VARIAN computer.

4. DCC EMULATOR

The DCC Emulator is designed to be a generic AECL test tool for DCC software development and testing. It emulates all the hardware necessary for running the on-line programs. For generality, the DCC Emulator checks every instruction for interrupts even if interrupts are not used. This approach is selected, at the expense of reduced speed, so that it is possible to produce only one version of the DCC emulator that can support all DCC software development.

The DCC emulator emulates all instructions in the Varian V7X series instruction set including all I/O instructions. The emulation of each I/O instruction is based on the individual I/O hardware characteristics. The DCC executive or the DCC emulator itself can be patched to have matching I/O hardware addressing scheme. A generic version of the DCC emulator may be used for different stations if their I/O hardware addressing scheme can be patched to match with the emulator. On the other hand, station-dedicated DCC emulator can be generated by patching its I/O hardware addressing scheme to match with the particular station. The design of the emulator is flexible enough to

support different I/O requirements at each station. Currently the DCC emulator supports the following I/O hardware:

Moving Arm Disk (MAD),
Fixed Head Disk or MegaRam disk,
Direct Memory Access module,
Varian Priority Interrupt module,
CAE Priority Interrupt module,
Memory Protect module,
Process I/O A/I's, D/I's, A/O's, D/O's,
Real Time Clock,
Teletype,
Printer (Centronix and Statos),
Count Down Registers,
Ramtek FS-2000 Display system,
AECL Paper Tape Replacement,
Data Link (an RS 232 serial communication link),
(The Card reader and Parallel Data Link Controller may be supported in future.)

The DCC emulator has been tested with different station configurations: Gentilly-2, Pickering and Bruce B. It provides an office environment for testing DCC Executive and on line programs running in real time. The DCC emulator uses software to emulate hardware at the machine-code instruction level for easy expansion. It runs the DCC Executive 'as-is' without any patching. The Executive performs exactly the same way as it would on DCC. It can be tested with interrupts enabled since all hardware features are under software control that can single-step through interrupts. (This is not possible when the Executive runs in the Varian computer.)

The DCC emulator maintains a real time clock and count-down registers (CDR's) to ensure multi-tasking capability. It outputs dynamic display buffers for bar charts, trending, and status displays in identical format as the actual DCC. It provides full I/O support for static and dynamic I/O simulation. Because of these capabilities, the PC can replace the DCCX in the single-computer test set-up such as the one shown in Figure 1.

5. PLANT SIMULATOR

The Plant simulator performs continuous simulation of the reactor, the heavy-water circuit, the light-water circuit and the turbine-generator set. The simulation starts the initialisation by calculating the steady-state for the plant model based on the user-supplied initial boundary conditions. Subsequently it repeatedly reads the control input file (see Figure 5) and performs the dynamic calculation. It outputs the results at specified time steps to the response output file as shown in Figure 5. The Plant transient calculation intervals are controlled to match with the calculation periods of the DCC emulator so that both run on the same time scale.

Figure 6 shows the plant components included in the simulation. From the testing computer the user can control the Plant simulator by writing directly to its control file or indirectly by using the emulated DCC keyboards to issue commands to the DCC emulator. Figure 7 shows one of the simulated keyboards available at the testing computer.

The Plant simulator includes the station control programs based on the same program specification as the actual DCC software. These programs can serve as 'test oracle' against which actual control programs are compared. The user has the option of using either the test oracle or the DCC emulator to control the Plant simulator.

6. TEST CASES

The user will prepare the test as a script file written in AECL Test Language that was originally developed by AECL for testing safety-critical software[1,3]. The script file is fed to the testing computer and interpreted by an AECL Test Language Interpreter (ATLIN). Table 1 shows a simple example of a test script file.

A typical test case consists of a series of calls to activate specific test actions such as initialising test conditions, activating the test signals, acquiring the responses from the DCC program, comparing the measured responses with the expected responses. The logging of the test sequences is performed automatically by ATLIN. The test sequence can be either carried out step-by-step under user control or continuously until the end of the test script file.

The DCC program specifications are based on the works of different design or analysis groups. For some tests it is necessary to obtain expected test results from the responsible groups. The test case shown in Table 1 is an example where expected flux mapping test results are obtained from Physics Analysis group. ATLIN converts these externally supplied data into internal tables for on-line interpretation of the success or failure of each test. The comparison between measured and expected responses sometimes involves large amount of data. By automating this process, the test outcome is determined more consistently and in much shorter time than the manual process of off-line comparison-by-eye.

7. TESTING

In static tests the tester applies via the test script a set of fixed inputs to the DCC emulator AI and DI files and observes that the DCC displays the same inputs. Any mismatch will indicate possible incorrect allocation of I/O addresses. In dynamic tests the AI and DI files are controlled by the dynamic CANDU plant simulator, which in turn is controlled by the DCC emulator AO and DO files. They form a closed loop system to simulate actual installation. Through the use of test scripts the tester can manipulate manual inputs such as setpoints, gains, simulated handswitches and buttons and create dynamic testing scenarios. The system designers are able to observe the system dynamic behaviour and verify that the DCC control programs interact correctly and respond as designed under dynamic conditions. Furthermore plant upset conditions can be initiated with the Plant model and DCC software behaviour can be observed under these conditions. The objective is to ensure correct program responses and interactions under transient and steady-state conditions.

The user can access and control the DCC emulator and Plant simulator locally at their keyboards or remotely from the testing computer keyboard..

Figure 8 shows the testing computer screen where the DCC AI's are monitored and manually modified. The screen shows 16 AI's at a time but can be scrolled so that any of 2400 AI's can be accessed. A brief description is included for each AI. Similar screens are used to access and monitor the DCC DI's, AO's and DO's. Besides being manually accessible, all of the DCC I/O points can also be accessed under program control from the test script file.

Figure 7 shows the testing computer screen containing one of the simulated keyboards where the DCC functions and programs are manually activated. The button is pushed by clicking the mouse on it. The keyboard sends 3-out-of-8 codes to the DCC keyboard port in the same way as in actual installation. Other keyboards are selected by the pop-up menu at the bottom of each screen.

The response of the DCC to the keyboard entry is shown in the video screen associated with each keyboard. This video screen is displayed on the testing computer monitor. Figure 9 shows one of such screen associated with the keyboard of Figure 7. This screen is displayed when the user clicks the mouse on the button labelled "ADJUSTER ROD STATUS".

8. DEVELOPMENT RESULTS

Preliminary development results indicate that accessing I/O data in a shared network file is at least as fast as accessing the real-life hardwired I/O system. The real-time performance of the multi-computer test facility is not limited by the shared I/O file arrangement, but rather by the complexity of the DCC emulator and the Plant simulator. The system performance is improved by removing from the DCC emulator, features that are not essential for testing or by simplifying the model used in the Plant simulator. Together with the availability of newer hardware such as the Pentium CPU family with clock speed in excess of 100 MHz, it is now possible to get real-time performance with little effort and cost.

Our future plan is to expand the test facility to supplement commissioning tests, such as automatically generating data sets for use in commissioning. Currently these data sets, which are sets of input values for the DCC software to produce given outputs, are produced manually with the help of a spreadsheet program. The test facility will allow the test plan and procedures to be developed and rehearsed ahead of schedule, independent of actual hardware and equipment delivery. It will also serve as training facility for commissioning and operating the DCC. These activities can be conveniently performed from the user's workstation in the office environment.

9. CONCLUSION

AECL has developed a universal DCC test facility as the result of a successful development of cross-development tools for the Varian computer and the experience gained in setting up a test facility for validation and reliability testing of shutdown system software. The multi-computer test facility using three work-stations has been set up on the Local Area Network in our design office. The new test facility achieved its objective of being low cost, versatile and open for wide usage and expansion possibilities.

The current practice of testing DCC control software for repeat CANDU stations relies on design calculations and simulations. The finished software is subjected to static tests and limited dynamic tests to verify against the program specifications. The dynamic DCC test facility provides an efficient means to perform both static and dynamic tests to ensure high quality in the finished product.

REFERENCES

(1) N.D. Thai and A.M. McDonald, "Development of a Test Rig and its Application for Validation and Reliability Testing of Safety-critical Software", 16th CNS Annual Conference Proceeding, June 1995, Saskatoon, Saskatchewan, Canada.

(2) D. Burjorjee and W. Fieguth, "A Simulation for Testing CANDU Control Programs", International Conference on Power Plant Simulation, November 1984, Cuernavaca, Marelos, Mexico.

(3) W.D. Watson and P. Vesely, "Development of a Computerised Safety Shutdown System Test Facility", Proceedings of the Symposium on New Technology in Nuclear Power Plant Instrumentation and Control, pp.155, November 1984, Washington, D.C.

TABLE 1 EXAMPLE OF A TEST SCRIPT WRITTEN IN AECL TEST LANGUAGE

```
; SUBROUTINE FLXMAP1.SUB                ;Reference: Test Plan 86-66500-210-
                                        001, section 6.4.6. Verify Flux Map
                                        Plane #1 at 100%FP, 0 bad detectors,
                                        0 Adjusters OUT, 0 Absorbers IN
TITLE  Flux Map Data Output at Plane #1
CALL DEFINE.STD                         ; Define standard test variables.
CALL INIT.STD                           ; Put DCC SW in standard initial
                                        states.
CALL DEFINE.FLX                         ; Define individual variables or
                                        group of variables for this test,
                                        e.g. MAP1GRP.
CALL SETUP1.FLX                         ; Set up operating conditions (e.g.
                                        control rod positions) for part 1 of
                                        this test.
CALL EXPECT1.FLX                        ; Set up expected outputs e.g. for
                                        MAP1GRP (Use data supplied by Physics
                                        group)
WAIT 5 MINUTES
CALL PRNMAP1.FLX                        ; Request DCC SW to output fluxmap at
                                        plane #1
CALL READMAP1.FLX                       ; Read actual fluxmap from DCC SW
COMPARE MAP1GRP                         ; Compare actual and expected
                                        fluxmaps, The comparison results are
                                        stored in the log file.
EXIT
```

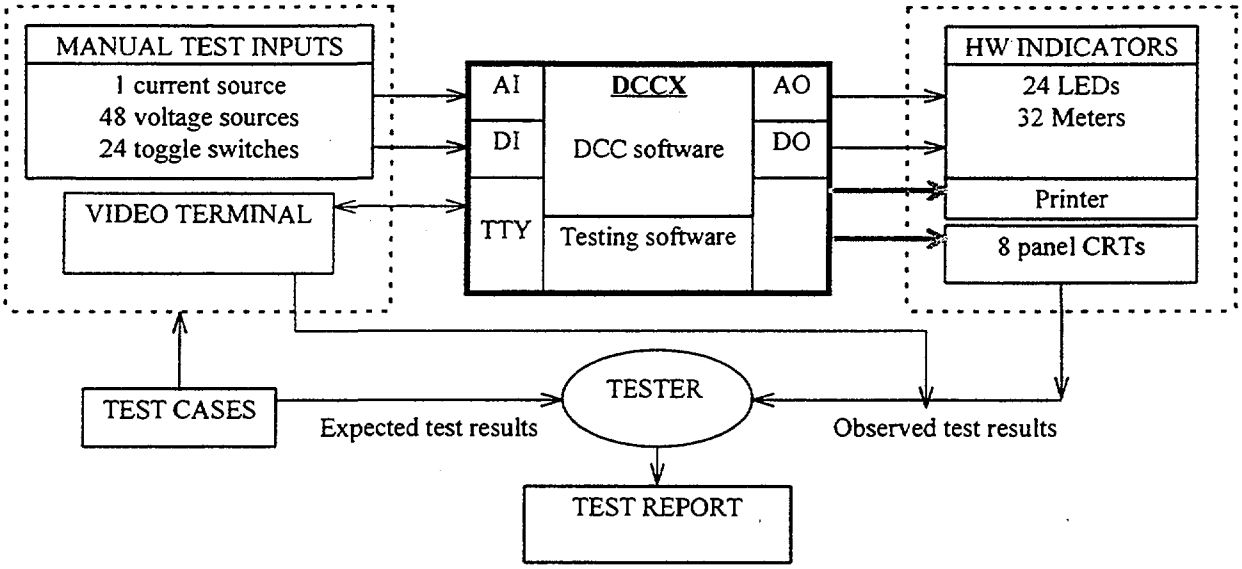

FIGURE 1. SINGLE-COMPUTER TEST SET-UP

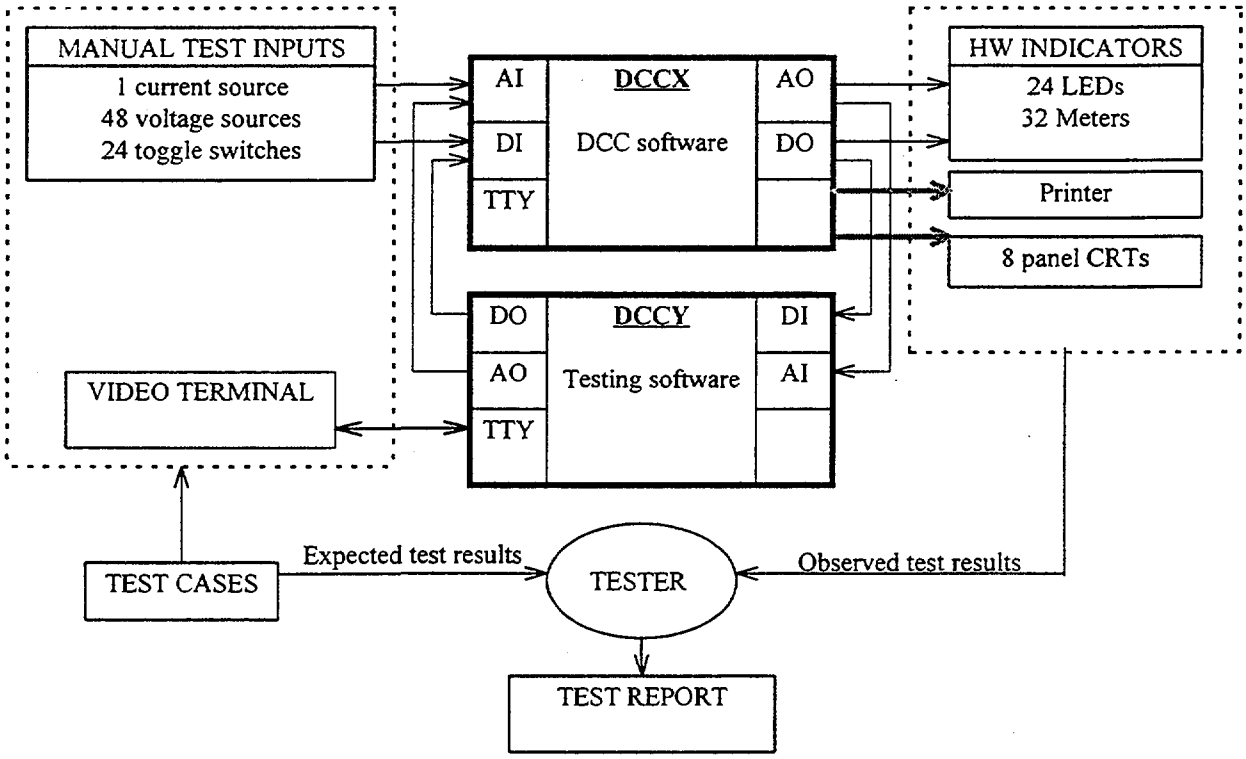

FIGURE 2. DUAL-COMPUTER TEST SET-UP

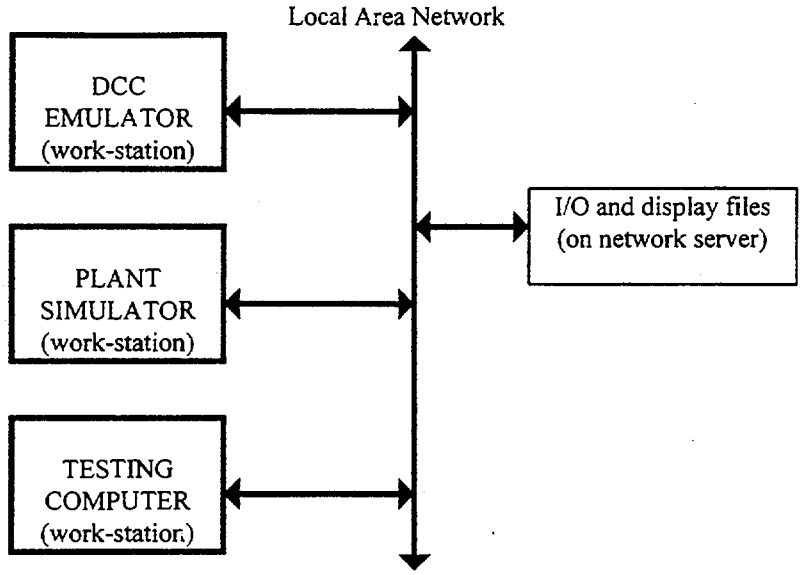

FIGURE 3. MULTI-COMPUTER DCC TEST FACILITY NETWORK

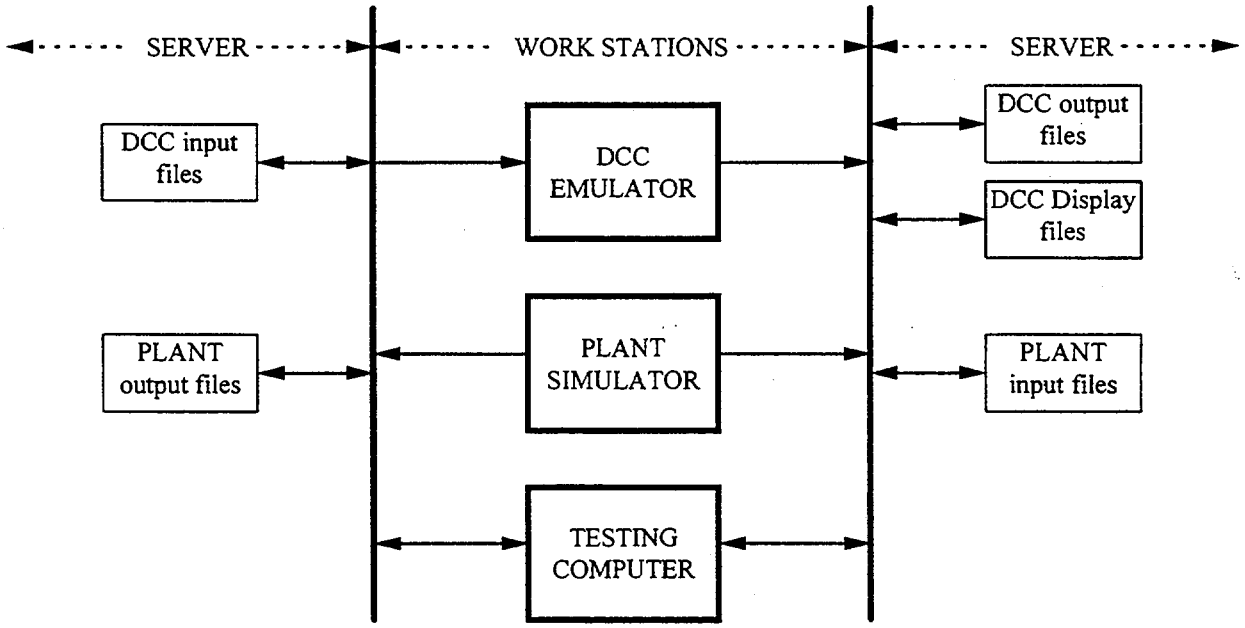

FIGURE 4 MULTI-COMPUTER DCC TEST FACILITY
WORK-STATIONS AND I/O FILES

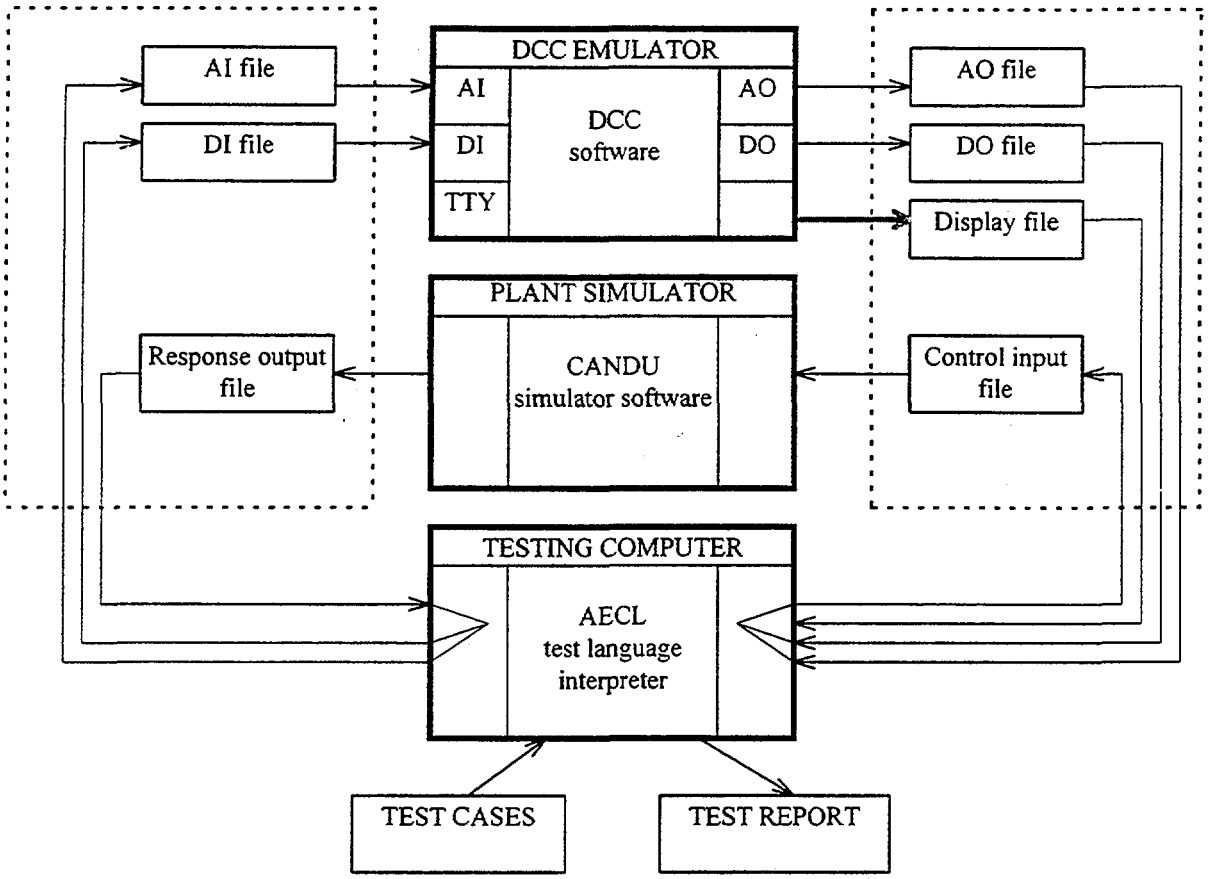

FIGURE 5. MULTI-COMPUTER TEST SET-UP

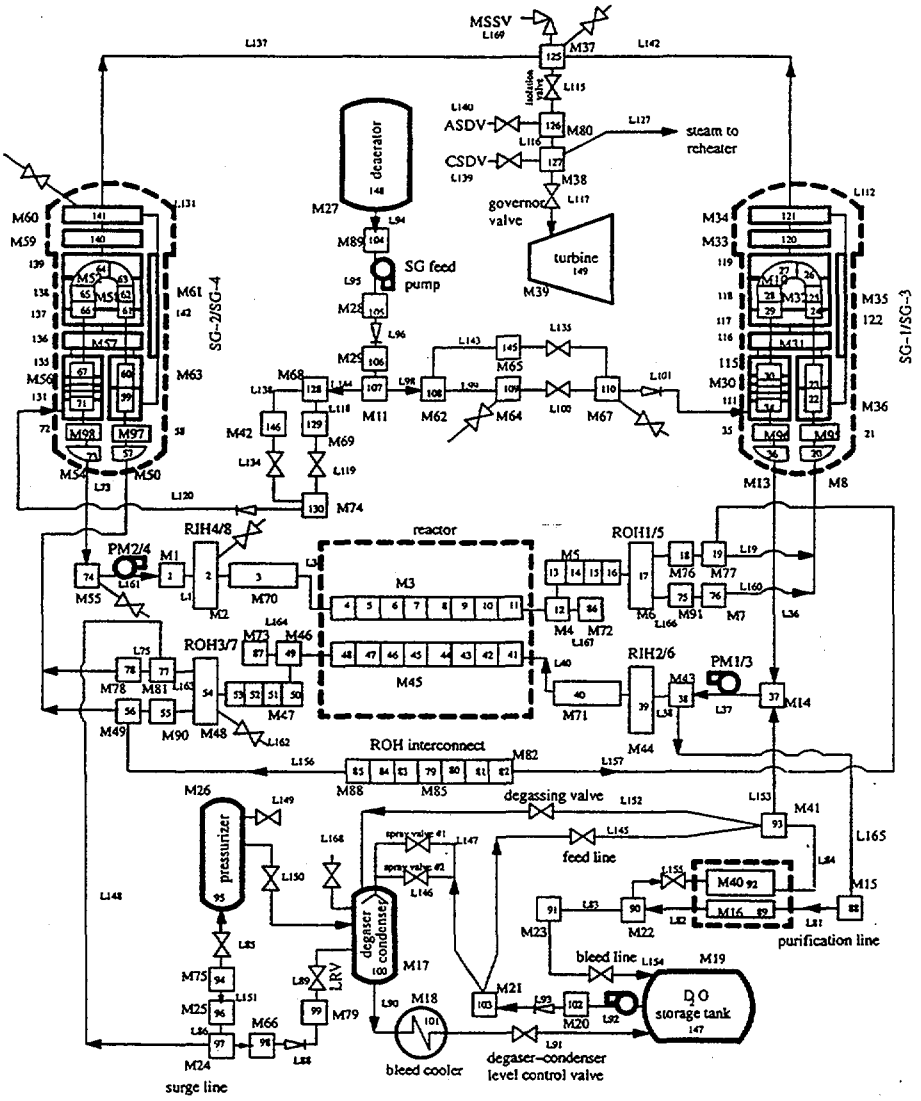

FIGURE 6. CANDU PLANT SIMULATION MODEL FOR DCC DYNAMIC TESTING

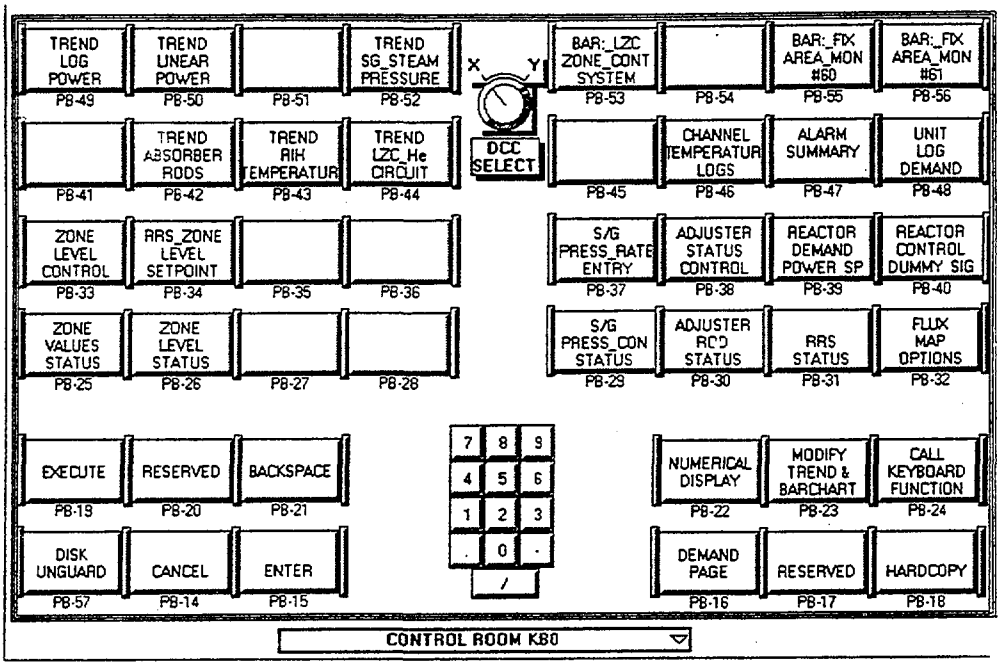

FIGURE 7. SIMULATED KEYBOARD FOR CONTROLLING THE DCC EMULATOR

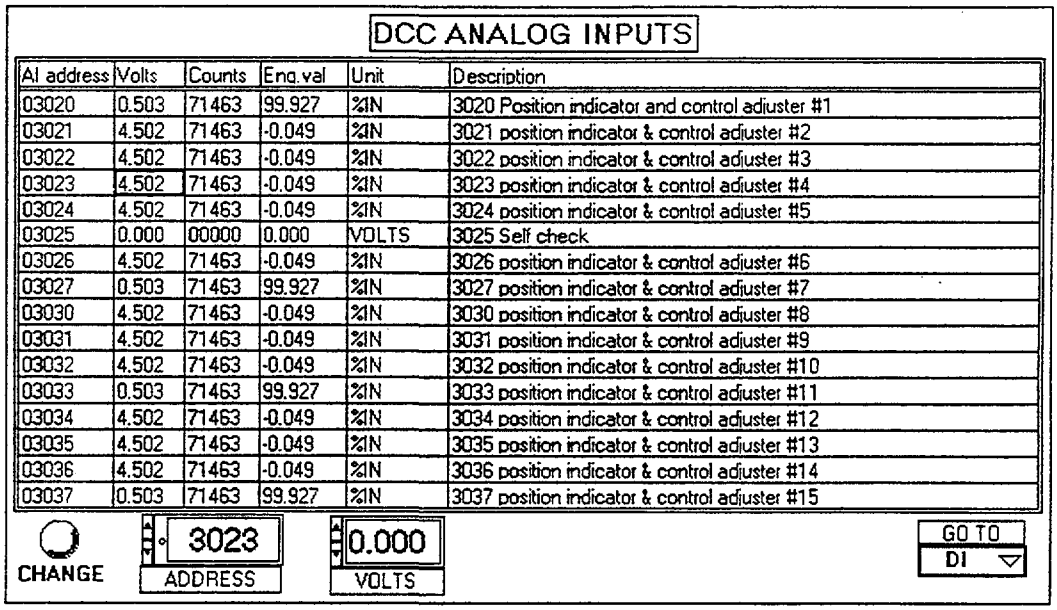

FIGURE 8. A TYPICAL USER INTERFACE FOR MANUAL CONTROL OF DCC EMULATOR ANALOG INPUTS FROM THE TESTING COMPUTER.

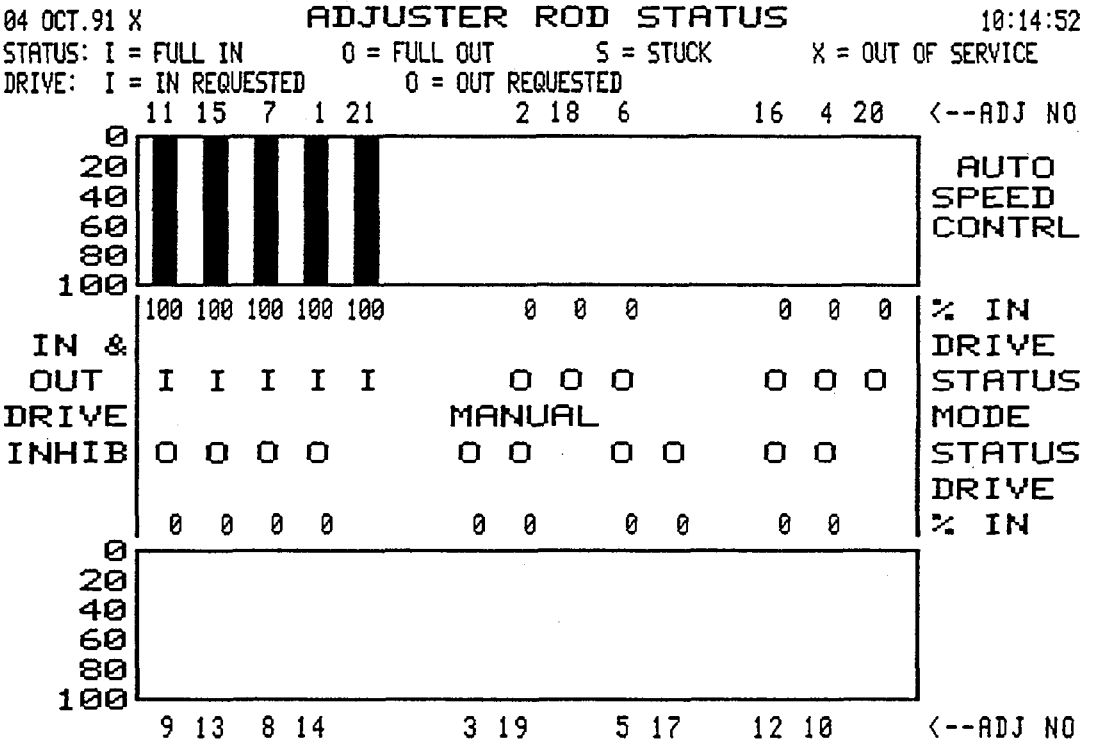

FIGURE 9. A TYPICAL DISPLAY OBTAINED FROM THE DISPLAY FILES OF THE DCC EMULATOR