ESD-TR-90-141

Annual Report

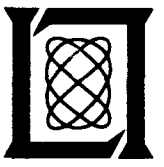
Knowledge-Based System Analysis and Control Defense Switched Network Task Areas

30 September 1989

Lincoln Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LEXINGTON, MASSACHUSETTS



Prepared for the Defense Communications Agency
under Air Force Contract F19628-90-C-0002.

This report is based on studies performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. The work was sponsored by the Defense Communications Engineering Center of the Defense Communications Agency under Air Force Contract F19628-90-C-0002.

The ESD Public Affairs Office has reviewed this report, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

Hugh L. Southall

Hugh L. Southall, Lt. Col., USAF
Chief, ESD Lincoln Laboratory Project Office

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

KNOWLEDGE-BASED SYSTEM ANALYSIS AND CONTROL DEFENSE SWITCHED NETWORK TASK AREAS

H.M. HEGGESTAD
Group 21

ANNUAL REPORT SUBMITTED TO
DR. SYED SHAH
DCEC R610
1860 WIEHLE AVENUE
RESTON, VA 22090-5500

1 OCTOBER 1988 – 30 SEPTEMBER 1989

ISSUED 10 DECEMBER 1990

LEXINGTON MASSACHUSETTS

ABSTRACT

The primary thrust of the FY89 program was installation and demonstration of the Network Management Expert System (NMES) at DCA Headquarters in Stuttgart, Germany. Two major efforts were involved: completion of improvements and extensions to NMES itself, and performance of numerous tasks in matching the hardware and software to the DCA-Europe environment so that it could be meaningfully operated there. Additional components of the FY89 program were implementation of an Operator Trainer to develop operator skills in the use of the current manual Network Management Support System, as well as execution of a set of enhancements to, and investigations with, the Call-by-Call Simulator (CCSIM).

In July 1989 two Sun 3/260 workstations and an IBM PS/2 Model 80 computer were installed at DCA-Europe, to support NMES, CCSIM, and the Trainer. A Lincoln contract staff member took up an extended assignment at DCA-Europe at the same time. Between July and September the software was installed, checked out, modified and debugged as necessary, and prepared for the demonstrations. The demonstrations were performed by two very competent military ACOC personnel offered for the purpose by ACOC Management, during the period 18-22 September 1989. Both the Expert System and the Trainer were demonstrated repeatedly to site and visitor personnel. The scenarios are described in the body of the report. The ACOC community recognized the valuable potential of these systems, and offered encouragement to carry out the significant integration and completion that must be done before this new technology can become part of their working inventory.

The FY89 SOW also called for substantial effort to begin to scope and understand the problems of correlating and interpreting DCS transmission system alarms and presenting filtered results to Tech Control and Network Management decision makers. By analogy with the success of the Call-by-Call Simulator (CCSIM) as a portrayal of realistic network behavior to use in developing network management knowledge, it has been conjectured that TRAMCON and DPAS alarm pattern simulation systems could make up for the lack of access to real transmission networks for MITEC software developers, as well as the fact that real networks seldom produce interesting alarm patterns. In FY89 a specification was developed for a TRAMCON alarm generator for this purpose, with an eye to implementing it in FY90. The complete specification is given in Appendix D.

Table of Contents

LIST OF ILLUSTRATIONS

LIST OF TABLES

1.0 EXECUTIVE SUMMARY AND INTRODUCTION

1.1 Executive Summary

The primary thrust of the FY89 program was installation and demonstration of the Network Management Expert System (NMES) at DCA Headquarters in Stuttgart, Germany. Two major efforts were involved: completion of improvements and extensions to NMES itself, and performance of numerous tasks in matching the hardware and software to the DCA-Europe environment so that it could be meaningfully operated there. Additional components of the FY89 program were implementation of a training system to develop operator skills in the use of the current manual Network Management Support Systems, as well as execution of a set of enhancements to, and investigations with, the Call-by-Call Simulator (CCSIM).

The first major undertaking with NMES was to address memory management problems that had been observed in FY88 as the size and complexity of the system gradually increased. The software development environment then in use was the ART (tm Inference Corporation) expert system shell running in Lucid Common LISP under the Sun UNIX operating system in our Sun 3/260 workstations. The vendor of each of these entities said that the problem had to be with one of the other two. The solution was to transfer NMES to the CLIPS shell which runs in the C language, was developed by NASA Johnson Space Center, and is free to Government users. The transition went smoothly, and was completed in the early spring. This solved the memory problems, and also removed the need to pay sizeable software license fees for each computer in which NMES would be installed in the future.

A new graphics interface was implemented for NMES by borrowing and modifying the sophisticated system that had been developed for the Call-by-Call Simulator (CCSIM). This allowed both interactive and automatic operation of the NMES.

Numerous changes were required to enable NMES to operate with real data from the telephone switches in the European DSN. These included modifying the system to accept "comma-separated" switch report data as currently produced by the Network Management Support System (NMSS) in use at DCA-Europe. It was also necessary to provide mechanisms for the NMES to operate successfully when damaged or incomplete switch report data is interspersed with good reports - a circumstance which was happening frequently in DCA-Europe at that time due to noise on the long-distance modem circuits collecting the switch data. Up to that time the NMES had operated with switch reports received every five minutes from the simulated switches, which matches the behavior of real Northern Telecom DMS-200 switches; however, at that time the NMSS was limited by data circuit bandwidth to 15-minute reporting cycles, and NMES changes had to be made to

deal with this. Another major difference is that only part of the switches in the European DSN have been upgraded to DMS-200s, and only those switches provide data to the NMSS; all the other switches are affecting the network traffic and performance but not reporting data on which NMES will be able to base its reasoning. This is in contrast to our simulation experiments, in which we were dealing with a future DSN in which switch upgrades were complete. Another issue was that the switch reports from CCSIM (which matched those of real DMS switches in content, in accordance with Northern Telecom documentation) contained certain useful data fields which were not yet being collected by the real NMSS in the field.

All of the above problems required substantial changes in NMES rule structures, monitors, and other areas. Difficulties were compounded by the fact that we could not test the system with real data until it was brought to Europe late in the FY.

A collateral DCA-Europe oriented project covered by our Statement of Work was development of an NMSS Operator Trainer. This was to consist of an IBM PS/2 Model 80 computer identical to that in the real NMSS, and driven by the CCSIM with data and scenarios that closely match those of the real DSN. This system was installed and demonstrated at DCA-Europe in September 1989, together with a user's guide and documentation that would permit it to be used for training purposes.

The FY89 SOW called for continuing functional development of CCSIM in a number of areas, besides a substantial set of changes necessary to support the Trainer. This included modification of CCSIM to produce the various kinds of real-world DSN behavior noted above. We needed to make changes in the internal labelling of trunk groups; the form, number and contents of switch report fields; local traffic models; routing mechanisms; trunk noise and switch damage models; and numerous other areas. In addition we modified CCSIM to allow simulation of various candidate routing strategies and to model data calls (e.g., long-term modem connections by data users). A common-channel signalling simulation design was developed and presented to DCEC for approval, with the intention of implementing it in CCSIM in the coming year. Also, numerous enhancements were made to the graphics interface.

In July 1989 two Sun 3/260 workstations and an IBM PS/2 Model 80 computer were installed at DCA-Europe, to support NMES, CCSIM, and the Trainer. A Lincoln contract staff member took up an extended assignment at DCA-Europe at the same time. Between July and September the software was installed, checked out, modified and debugged as necessary, and prepared for the demonstrations.

The demonstrations themselves were performed by two highly

competent military ACOC personnel offered for the purpose by ACOC Management: Sergeant First Class Nancy Fuller, USA, and Gunnery Sergeant Steven Waynick, USMC. They delivered expert performances of the demos on the following timetable:

12 Sept: arrival of Lincoln demonstration team on site

12-15 Sept: equipment checkout, plans and rehearsals

18 Sept: Briefings and demos for site personnel

19 Sept: Demo for DCA-Europe management and CONUS visitor
(Col. Waterman, Deputy Director, DCEC)

20 Sept: Demo for Tactical USAFE (Ramstein)

21 Sept: Demo for Army 5th Signal Command (Worms)

22 Sept: wrap-up, future plans

The following is a synopsis of the NMES demo that was repeatedly performed by Sgts. Waynick and Fuller:

- Connect NMES to live switch data; show displays, data arrival, colors and meanings

- Connect to simulated data, so that NMES response to network problems can be shown; Sgt Waynick on CCSIM, Sgt Fuller on NMES

- Run CCSIM with normal conditions for 30 simulated minutes, explaining displays and capabilities

- Destroy two key switches and a major trunk group simultaneously

- Show NMES correctly identifying all three, recommending controls

- Take a snapshot before control application

- Apply controls, compare before-and-after network performance

The following is a synopsis of the NMSS Trainer demo:

- Connect CCSIM/Trainer (Sgt Waynick) to NMSS (Sgt Fuller)

- Run 30 simulated minutes, describing functions

- Sgt Waynick secretly destroys a key trunk group (NVL-MHL)

- Sgt Fuller (the "trainee") detects the failure; confirms it

over one more reporting cycle; and applies SKIP controls for NVL-MHL trunks

- Sgt Waynick secretly takes down the switch at ABE

- Sgt Fuller detects and confirms the failure, and applies DCC and SKIP controls

The demonstrations clearly exhibited useful and effective technology, and it was plainly evident that the ACOC community wants and needs the benefits of the technology. Analysis and pattern recognition in the masses of DSN switch statistics data is difficult even for experienced network managers, and by the time ACOC personnel acquire significant experience they are transferred, and new people come in. Consequently the prospect of automatically obtaining situation assessments and control action recommendations is highly appealing. The demonstrations showed bona fide evidence that these functions were being done.

Not surprisingly, however, the ACOC community recognized that significant integration and completion must be done before this new technology can become part of their working inventory. The operations staff is already burdened with learning and using numerous different user interfaces for systems they use in their work, and there is no rationale for requiring them to learn two additional, unique systems that each do a part of the DSN network management functions. Clearly the systems should be integrated into a single workstation as noted above. Also, whereas the examples demonstrated looked right to all concerned, DSN network management spans a wide range of functions and conditions, and it will be necessary to do extensive evaluation and analysis of the integrated workstation to establish that it performs correctly throughout this range. This process will undoubtedly entail redesign and correction in some areas.

The FY89 SOW also called for substantial effort to begin to scope and understand the problems of correlating and interpreting DCS transmission system alarms and presenting filtered results to Tech Control and Network Management decision makers. In particular, it is recognized that these decision makers will be able to function more rapidly and effectively if they can obtain immediate information about transmission system problems from TRAMCON and DPAS systems, rather than having to wait while the user community slowly reacts to transmission impairments and produces traffic pattern distortions that allow the managers to recognize and correct the problems. By analogy with the success of the Call-by-Call Simulator (CCSIM) as a portrayal of realistic network behavior to use in developing network management knowledge, it has been conjectured that TRAMCON and DPAS alarm pattern simulation systems could make up for the lack of access to real transmission networks for MITEC software developers, as well as the fact that real networks seldom produce interesting

alarm patterns. In FY89 a specification was developed for a TRAMCON alarm generator for this purpose, with an eye to implementing it in FY90. The complete specification is given in Appendix D.

1.2 Background

The DCEC-sponsored Knowledge-Based System Analysis and Control Program was established at Lincoln Laboratory to address an important problem area associated with the ongoing implementation of the Defense Switched Network (DSN), namely learning and applying network management techniques to correct traffic overload and network outage problems and to avoid disasters. The keystone of this new Program is to develop such techniques and to capture the knowledge in a flexible, portable Network Management Expert System which will allow DSN field personnel to deliver state-of-the-art network management performance.

It was recognized by DCEC in FY86 that the Call-by-Call Simulator (CCSIM), a powerful DSN simulation tool built by Lincoln Laboratory under a DCEC-sponsored effort ending in FY85, could provide a means for approaching these goals. The original purpose of CCSIM was to study and evaluate candidate call routing and preemption techniques for the DSN, and it was a large software system which had a variety of features well-tailored to that purpose. If suitably modified and extended, CCSIM could become an experimental testbed for learning to do DSN network management. Accordingly, the new DCEC-sponsored program was established in FY87 with three major goals:

1. Implement the new features and capabilities needed to suit CCSIM to these new purposes;

2. Conduct a variety of experimental exercises with CCSIM to derive DSN network management methodologies; and

3. Develop a prototype Network Management Expert System to serve as a repository of this near-term knowledge as well as future skills and techniques to be derived from both simulated and actual DSN experience.

These goals were accomplished, as described in the Knowledge-Based System Analysis and Control, Defense Switched Network Task Areas FY87 Annual Report, providing a very interesting initial look at the potential of these techniques for developing genuinely valuable tools for future field deployment.

The program was continued in FY88, with a Statement of Work (SOW) that was intended to reap the benefits of the FY87 work by consolidating and extending the initial results. Task I of the SOW envisioned an Interactive DSN Simulation (IDSIM) consisting

of an enhanced CCSIM in a Sun 3/260 workstation, interfaced with an enhanced NMES in a second Sun 3/260 workstation. The primary function of the latter is to exercise control and network management upon the simulated DSN running in CCSIM. Task II of the FY88 SOW provided for performing simulation studies with CCSIM on a variety of traffic overload, congestion and damage scenarios, aimed at broadening the knowledge base of network management remedies for problem conditions of concern to DSN operators. Task III was the precursor of the NMSS Operator Trainer that was demonstrated in FY89 as noted above; this Task was to demonstrate the feasibility of such a trainer and to develop its design, as well as to begin the actual implementation. Task IV provided for analysis and review of future requirements of the Defense Data Network for monitoring and control, and for identification of expert system techniques and technology that could be applied to improve the effectiveness of DDN management. All of these tasks were carried out, as described in the Knowledge-Based System Analysis and Control, Defense Switched Network Task Areas FY88 Annual Report.

During FY88 DCEC asked for a field test and evaluation of NMES to be planned for late FY89 at DCA-Europe, using the same live input data that is available to controllers using the NMSS. This new requirement considerably influenced Lincoln's FY88 activity, in terms of both making NMES implementation choices and planning FY89 work toward the demonstration objectives. Two Lincoln staff spent a week at DCA-Europe in September 1988 coordinating needs and plans with the site personnel, for both the NMES demo and the NMSS Controller Trainer, which became a closely related activity because the Trainer would be used to provide inputs for NMES during the development of the field-test version.

The outcome of all this activity was an FY89 program that heavily emphasized the field demonstration, as described in the body of this Report.

2.0 DCA-Europe Demonstrations

2.1 Implementation of the Demonstration Expert System

2.1.1 Background

At the end of FY88 the Network Management Expert System (NMES) was a functioning component of the Interactive Defense Switched Network Simulator (IDSIM). It was a combination of a rule based expert system using the ART expert system development shell and routines called 'monitors' written in SUN Common LISP that processed switch reports from the Call-by-Call Simulator (CCSIM) and asserted facts into the ART database. It had a graphics interface that made use of unsupported Sun View graphics LISP macros. Routines written in the 'C' language were used to communicate with the other components of IDSIM.

Through a combination of monitors and rules, the FY88 NMES could recognize switch outages and overflowing and overloaded links. It had a planning module that could apply appropriate controls to treat the recognized troubles, and the effectiveness of the control actions could be evaluated through CCSIM. Implementation of control actions 'turned on' new monitors that would look for indications that the network had returned to normal. Other rules would then fire to remove the related controls.

Independent of any considerations relative to a demonstration at DCA-Europe, there was a strongly felt need to improve the system. Experience had shown that NMES could not handle large networks effectively due to problems with memory management in the ART/LISP environment. Even for the case of small networks, memory problems (an excessive frequency of LISP garbage collections) were seen when NMES ran for several hours of simulation time. It was also clear that a new graphics interface was needed to enhance the information available to the operator and to overcome concerns about the lack of vendor support for the LISP macros used in the FY88 interface.

2.1.2 Conversion of NMES to CLIPS

At the start of FY89, work was undertaken to address the memory management problems observed with the ART/LISP environment. Progress was discouraging, and significant improvement appeared to be unlikely unless the environment itself could be altered, a task ill-matched to our resources. Instead, we chose to investigate the use of an expert system shell developed by the Artificial Intelligence Section at NASA/Johnson Space Center called CLIPS ('C' Language Integrated Production System). Our investigations showed that conversion to CLIPS would be relatively straightforward and highly beneficial.

CLIPS contains many of the basic functions of ART that we had been using. It is designed to provide high portability, low cost and easy integration with other systems. Problems involving licensing and availability on more than one machine that had been of concern with ART are avoided because CLIPS is available at no cost for U.S. government work. CLIPS has been used to develop expert systems that have proven to operate reliably in the field over a long period of time. Using CLIPS simplifies the interface problems to other components of our system because it is written in C and developed to interact directly with programs written in the C language.

The transition from the ART/LISP/C environment of the FY88 NMES to the CLIPS/C environment went smoothly and was completed during early spring of 1989. The structure of the FY89 NMES is essentially the same as that described for the FY88 NMES in the previous Annual Report. The lowest level is the network representation made up of C structures that are created when NMES is initialized and later filled in with processed data from the switch reports as they arrive. At initialization time, the network representation is also entered, by assertion, into the CLIPS database and stored as facts. On each cycle of NMES operation, the C structures are scanned by monitors, implemented now in C, for interesting features and useful information. The outputs of the monitors are stored in the CLIPS database as the abstract state of the network. Potential problems are identified by a set of CLIPS rules which look for patterns in the abstract state of the network, and these problems are confirmed over time by another set of rules. A planning module devises control actions to improve or correct a confirmed problem situation. An observation module watches the effects of the applied controls over time and removes the controls when the problem no longer exists.

2.1.3 New NMES Graphics Interface

A new graphics interface for NMES was realized by adapting the CCSIM graphics interface to use different information for coloring the switches and links. For NMES, the colors represent levels of anomaly and problem detection and show that action has been taken. By mousing buttons the operator can set the mode of NMES to be either automatic or interactive. In automatic mode, NMES sends control commands to CCSIM immediately upon the detection of network damage or restoration. In interactive mode the operator must act to approve, ignore, or defer the implementation of controls. In either mode the operator, by mousing the switches and links, can get text describing the problems and any recommended actions. Also by mouse action, the operator can see all the switch report information available to NMES about the switches and trunk groups. Other information, such as the true extent of network damage, that is available to

the CCSIM user from its graphics interface, is not presented to the NMES user.

2.1.4 Changes Needed for the Demonstration NMES

In order to operate with real network data in the demonstrations it was necessary to change NMES to work with switch reports in a format called 'comma-separated' data used by the NMSS workstation (see Section 2.2). In some respects, these reports contained less information than the reports from CCSIM used in the FY88 work, and they were generated less frequently (every 15 minutes as opposed to every 5 minutes). For example, at the time of the demonstrations, the real switch reports did not have any information about call blockage at the switches or the activity of precedence calls in the network. On the other hand, they did have other information about the internal state of the switches (see Section 3.2.3) that was not available from CCSIM in FY88. They also provided information about individual trunk groups rather than the links between switches that was reported by CCSIM in FY88.

To handle the different switch report format, a program module called the 'translator' was written to convert the comma-separated data into the format used by CCSIM. We chose to keep using the CCSIM format since it could handle the union of the information available from simulations and that obtainable from the real network. The translator has two modes of operation. It can feed data to NMES either from archived data files or from the live network when located on site. In addition to running with the translator, NMES can be run with simulated switch reports from CCSIM.

In addition to reporting on inter-switch trunks, the real network sends reports on other trunk groups going to PBXs and 4-wire users. These other groups are of little interest to network managers because problems associated with these groups have little affect on network performance, and in most cases there is nothing that a network manager can do about them. Because of the complexity involved in simulating and representing these trunks, it was decided that only inter-switch trunks would be monitored by NMES. Data for other groups is discarded by the translator as it processes the switch reports.

In the real network, switch reports are subject to both loss and damage due to interruptions of and noise on the communication lines between the switches and the ACOC. Some of the damage situations can be detected and removed by the Data Acquisition Interface (DAI) processor that polls the switches, with the consequence that partial reports can be received. The NMES monitors had to be changed to deal with these partial reports.

NMES requires a complete set of switch reports for a sample period to have arrived before starting its processing cycle. Of course, there will be no report from a switch that is damaged, and there may be none from others from which communication with the DAI was lost. In the FY88 NMES, we used the arrival of the first report for the next sample period to trigger the processing cycle. The same technique was first used for the demonstration NMES, but because of the 15 minute polling period, a considerable delay was introduced while waiting for the first poll of the next period. Since the DAI makes a certain number of attempts to poll a switch and if all fail then sends a special message called an empty switch report, we changed NMES to start its processing cycle when one report (normal or empty) had been received from each switch for the current period. This technique worked satisfactorily most of the time and was used for the demonstrations and tests that followed. However, we later learned that the DAI in some cases would decide to re-poll a switch after sending the 'empty' report. If the re-poll was then successful, NMES would get a new report which it was not expecting and would become confused until the next complete cycle.

Noise on the lines between the switches and the DAI can cause data corruptions that cannot be detected by the DAI. These can cause errors in the behavior of client programs. We chose to have NMES do some smoothing over time to reduce the probability of mis-recognition due to such noise. Hind-sight says that this decision was a mistake. The data errors did not occur frequently enough to justify the delay introduced by the time smoothing. We had based our choice on some early data samples obtained from Europe that showed a number of errors. However, by the time of the demonstrations, the error characteristics of the communication lines had improved significantly, and the smoothing was probably not needed.

Another major difference between the demonstration environment and the earlier simulations is that only a sub-set of switches in the network report to the ACOC. Consequently, less information is available to NMES about some trunk and switch outages because reports from both ends of a trunk group or from all neighboring switches are often lacking. The lack of information does not cause any fundamental change in the methodology for diagnosing problems, but it does increase the probability of making an error in diagnosis, particularly in the 'false alarm' sense. The probability of error is greatest at times of low network traffic since the only source of information available directly to NMES comes from the response of the network to attempts to route calls. If no attempts are made in some time period, NMES gets no information. If only a few calls are attempted, NMES gets some information, but it is 'noisy' in the sense that there tend to be large percentage fluctuations from period to period. In the demonstration system we attempted to

compensate for both the network sub-set and low traffic effects by requiring an apparent problem to persist over time before announcing a diagnosis. We would have preferred to take traffic statistics into account in the diagnosis process and to generate statistical confidence measures for NMES diagnoses, but it was clear that the data neccessary to generate the statistics and the software to incorporate them would not be available in time for the demonstrations.

The FY88 NMES had rules for recognizing some traffic problems (overloaded trunks) and applying controls to improve the call failure rate. Since these FY88 rules made use of switch report data that was not to be available from the real network at demonstration time, and the control actions had only a small affect on the call failure rate in our experiments, we decided not to attempt to include this capability in the demonstration NMES. Instead we chose to limit NMES, as far as a complete problem recognition and control application package was concerned, to dealing with switch and trunk outages only. Since the demonstration was to include watching the real network as well as interacting with CCSIM, and there was only a small probability of a switch or trunk outage occurring on the real network during the demonstration, we decided to add the capability to recognize some other anomaly situations for which NMES would have no control recommendations but which might be expected to occur occasionally during a period of watching the real network. These included switch problems such as failures to find a free MF receiver or a free Call Condense Block (CCB), and trunk group problems such as low holding time and high incoming connections per circuit per hour.

Appendix B contains a detailed description of the monitor routines used in the demonstration NMES. It also contains a description of the CLIPS rules involved in recognizing switch outages and restorations.

2.2 The NMSS Operator Trainer

Work on the NMSS Operator Trainer was brought to completion during FY89. It was demonstrated at DCEC in August 1989 and at DCA-Europe in September. The goal of the work was to adapt CCSIM so that it could serve as a training aid for users of the Network Management Support System (NMSS) developed by GTE for DCA-Europe. NMSS is a PC-based system designed to provide centralized network management control of a small set of Northern Telecom switches. It uses two IBM PS/2 computers. One, called the Data Acquisition Interface (DAI) polls the switches, reformats and archives the Operational Measurement (OM) reports returned by the switches, and sends them to the other computer called the NMSS workstation. It provides graphical and alphanumeric displays from which a skilled operator can deduce the presence of anomalies or problems in the network and, by mouse and keyboard interaction, can cause

network management controls to be sent back to the switches by the DAI.

Since considerable skill is required to use NMSS effectively, it was felt that there was a need for a training capability so that potential operators could practice taking network management actions in a simulated network situation. Some kinds of training scenarios can easily be carried out while observing the behavior of the real network, but others involving serious damage or major control interventions cannot safely be carried out except in simulation. Under Task III of our FY88 SOW we began work on the adaptation of CCSIM to perform this simulation task. The work involved the extension of CCSIM to model a number of aspects of switch behavior that were new to it (see Section 3.2 of this report) plus the development of a new module called the 'trainer' that handled the data reformatting and communication protocol issues associated with connection to the NMSS workstation.

To support the effort we procured two IBM PS/2 computer systems with graphics capabilities to match the NMSS workstations. We used one for our software development and testing. We loaded it with NMSS workstation software provided by DCA-Europe and confirmed operation using archived files of the 'comma-separated' data also provided by DCA-Europe. We were then in position to begin testing the 'comma-separated' data produced by CCSIM through the trainer module. For the demonstration of the trainer in August, we moved the PS/2 to DCEC and connected it to a SUN workstation there.

The other PS/2 was sent to DCA-Europe for use in the demonstrations in September. It was sent early so that people there would have an additional machine to help with their NMSS development work that was proceeding in parallel with our trainer work. Not surprisingly, the parallel developments led to some additional work because the system we were trying to simulate was evolving as we worked, and very little NMSS documentation was available. Fortunately, evolution slowed sufficiently to allow a sucessful demonstration in September. Some of the details of the formats for control invocations and reponses were not worked out until we arrived at DCA-Europe to prepare for the demonstration.

Further detail on the NMSS operator trainer can be found in Appendix C of this report which reproduces the User's Manual and Software Description documents delivered separately to the sponsor during the year.

2.3 Specification of the Simulated Network

For the NMES and Operator Trainer demonstrations in September, we needed to define a network for CCSIM that would represent the subset of DSN-EUR that would be monitored by NMSS at that time. The demonstration network had to contain all of the switches that would report to NMSS. In addition it had to have enough other switches to account for the interswitch trunks connecting the NMSS sub-set with the rest of the network. At a minimum, these other switches would be the immediate neighbors of the NMSS-reporting switches. We hoped to be able to aggregate the traffic in and out of the NMSS sub-set and attribute it to the immediate neighbor switches. We were confident that such aggregation could be made to match real network data for any particular steady-state situation. However, when damage or major traffic pattern shifts occurred, we thought it unlikely that the aggregated traffic would continue to match real-net behavior. Accordingly, we planned to define both a large network with a full set of DSN-EUR backbone switches and a small network with just the NMSS sub-set and its immediate neighbors and to compare their behavior under scenarios of interest. As it turned out, the effort necessary to define the large network was beyond what could be accomplished in the available time, and we elected to focus on the small network for the demonstrations.

Since the current DSN-EUR is quite different from the networks we had been simulating, we requested detailed information on the current configuration from DCA B-521. On 29 June we received twelve disks and several maps. The maps showed the current trunking, and the disks contained complete routing information by end-office codes. A few weeks later we received another disk with the trunking information in machine-readable form. We also had a single page representation of Autovon tandem switch routing that had been provided by DCEC, and, of course, we had the trunking data from the NMSS XREF files for the NMSS reporting switches and their neighbors. Unfortunately, the routing tables on the twelve disks did not constitute a complete and consistent set of data. Some tables applied to the current network, others to a future configuration, and there was nothing on the disks to say which was which. Still other tables were unreadable or incomplete. As we were beginning to consider various alternatives, we came upon another source of information. While on a visit to the European ACOC, we were offered copies of the complete routing tables for the six Autovon 490L switches in Europe. With this solid detailed information as a guide, we were able to decide which portions of the information on the twelve disks were consistent with the current network, and putting it all together, to generate CCSIM routing tables for the demonstration network.

In addition to information about the network topology (switches and trunks) and the routing tables used in the network, CCSIM must have a traffic matrix representing the source to destination traffic for all switches in the simulation. DCEC provided us with printouts of measured data for busy-hour and busy-day traffic out of some of the switches in the NMSS sub-set. There was no information on incoming calls to the NMSS sub-set switches in the measured data. Since the measurements were taken over a short time period, there were many source-destination pairs that showed no traffic at all during the sample. They also provided us with estimated traffic matrices for the entire European DSN. Unfortunately, that data existed only as a hard-to-read copy of a lengthy printout, and efforts to obtain it in machine-readable form were unsuccessful. If we were to have used it in detail, it would have been necessary to enter it manually, a slow and error-prone process. We would also have had to translate the units of traffic intensity and the source/destination place names. In the end, we decided that we could make no practical use of this potentially valuable information.

We chose to deal with the traffic problem by assuming an approximate symmetry in the distributions between outgoing and incoming calls, i.e, if a switch was a source of traffic for n other switches in the measured data, we would assume that it was a destination for the same n other switches acting as sources. The symmetry assumption would not extend to the traffic intensity, the trunk statistics for which showed to be different in the two directions. Further we would arbitrarily spread the distribution somewhat by replacing some of the zeros in the pairwise matrices with small values. We would then use a trial and error procedure supported by spread sheet calculations to adjust traffic intensities until trunk statistics from CCSIM satisfactorily matched those from archived NMSS busy-hour data. Simply matching the trunk statistics for a normal busy hour, which can be done with very many different traffic matrices, does not give any assurance that the behavior of the simulation will be realistic for a damage situation. For example, the trunk statistics can be most easily matched by assuming that all traffic from a switch is destined for its immediate neighbors, but that traffic pattern is clearly very unrealistic for many neighbor pairs.

The procedure described in the previous paragraph produced traffic matrices that matched reasonably well to both archived NMSS trunk data and the one-way end-to-end measured traffic data. As in all work associated with an event by event simulation, there is a random component that must be treated statistically. It is not practical to attempt to get a good match on all trunk groups, particularly those of low capacity which have a high variance from run to run of the simulation. The behavior of the simulated network when damage is introduced does not appear to be

obviously incorrect, although we know that it cannot be totally correct because of our assumptions, particularly the attribution of traffic to and from portions of the overall network that are outside the NMSS sub-set to immediate neighbors of the sub-set.

Even for a small network, the effort required to produce a traffic matrix using the trial and error procedure described above is quite large. If CCSIM is to be used for detailed investigations pertaining to the behavior of a real network under damage conditions, more complete measured traffic data is needed as input to the simulations. With complete source to destination traffic data the procedure for generating a CCSIM traffic matrix would be algorithmic. With machine-readable data it could be reduced to a computer program with little difficulty.

At the time of the demonstrations, we used a 26-node network in the simulations. Of the 26 nodes, 25 represented real switches (14 tandem, 11 end-office). The 26th, labelled 'CON', served as a destination node for all traffic to CONUS. Traffic from CONUS was simulated as having originated at two of the 14 tandem switches that acted as gateways. Twelve of the 25 nodes representing real switches were members of the NMSS reporting sub-set. Shortly after the demonstrations were completed, growth in the NMSS sub-set required an increase in the simulated network to 32 nodes.

2.4 September 1989 Demonstration Results.

Preparations.

In July 1989 two Sun 3/260 workstations and an IBM PS/2 Model 80 computer were installed at DCA-Europe, to support NMES, CCSIM, and a replica of the NMSS Workstation, respectively. A Lincoln contract staff member (Mr. Paul Gesswein) took up an extended assignment at DCA-Europe at the same time. Through the efforts of Mr. Gesswein and visiting staff from Lexington, between July and September the software was installed, checked out, modified and debugged as necessary, and prepared for the demonstrations.

It was deemed desirable to have the actual demonstrations performed by military ACOC personnel, and two highly competent people were offered for the purpose by ACOC Management: Sergeant First Class Nancy Fuller, USA, and Gunnery Sergeant Steven Waynick, USMC. These two were quick, enthusiastic and computer-literate, and they worked long hours in developing and practicing demo scenarios with the Lincoln people. They then delivered expert performances of the demos on the following timetable:

12 Sept: arrival of CONUS demonstration team on site

12-15 Sept: equipment checkout, plans and rehearsals

18 Sept: Briefings and demos for site personnel

19 Sept: Demo for DCA-Europe management and CONUS visitor
(Col. Waterman, Deputy Director, DCEC)

20 Sept: Demo for Tactical USAFE (Ramstein)

21 Sept: Demo for Army 5th Signal Command (Worms)

22 Sept: wrap-up, future plans

Demo Scenarios.

The following is a synopsis of the NMES demo that was repeatedly performed by Sgts Waynick and Fuller:

- Connect NMES to live switch data; show displays, data arrival, colors and meanings

- Connect to simulated data, so that NMES response to network problems can be shown; Sgt Waynick on CCSIM, Sgt Fuller on NMES

- Run CCSIM with normal conditions for 30 simulated minutes, explaining displays and capabilities

- Destroy two key switches and a major trunk group simultaneously

- Show NMES correctly identifying all three, recommending controls

- Take a snapshot before control application

- Apply controls, compare before-and-after network performance

The following is a synopsis of the NMSS Trainer demo:

- Connect CCSIM/Trainer (Sgt Waynick) to NMSS (Sgt Fuller)

- Run 30 simulated minutes, describing functions

- Sgt Waynick secretly destroys a key trunk group (NVL-MHL)

- Sgt Fuller (the "trainee") detects the failure; confirms it over one more reporting cycle; and applies SKIP controls for NVL-MHL trunks

- Sgt Waynick secretly takes down the switch at ABE

- Sgt Fuller detects and confirms the failure, and applies DCC and SKIP controls

Sgts. Fuller and Waynick were also trained by GTE representatives to demonstrate the LARS neural net system for diagnosing DSN network problems on the basis of information contained in the statistics reports from the switches. During the week of 25-29 September they skilfully demonstrated LARS for the same groups of visitors who saw NMES and the Trainer.

Assessment of the Demo Experiences.

The demonstrations clearly exhibited useful and effective technology, and it was plainly evident that the ACOC community wants and needs the benefits of the technology. Analysis and pattern recognition in the masses of DSN switch statistics data is difficult even for experienced network managers, and by the time ACOC personnel acquire significant experience they are transferred, and new people come in. Consequently the prospect of automatically obtaining situation assessments and control action recommendations is highly appealing. The demonstrations showed bona fide evidence that these functions were being done.

To nobody's surprise, however, the ACOC community recognized that significant integration and completion must be done before this new technology can become part of their working inventory. The operations staff is already burdened with learning and using numerous different user interfaces for systems they use in their work, and there is no rationale for requiring them to learn two additional, unique systems that each implement a part of the DSN network management function suite. Clearly the systems should be integrated into a single workstation as noted above. Also, whereas the examples demonstrated looked right to all concerned, DSN network management spans a wide range of functions and conditions, and it will be necessary to do extensive evaluation and analysis of the integrated workstation to establish that it performs correctly throughout this range. This process will undoubtedly entail redesign and correction in some areas.

3.0 CCSIM Development

Task II of the FY89 SOW called for functional development of the Lincoln Laboratory Call-by-Call Simulator (CCSIM) in a number of different areas. As modified in March 1989, the SOW identified the work needed in CCSIM to support the planned demonstrations at DCA-Europe at the end of the FY as the primary goal with other functional development to proceed as far as resources permitted. Other development was to include the simulation of candidate DSN routing strategies, the modeling of calls having different holding times and arrival rates (e.g., data calls), and the extension of CCSIM to model both in-band and Common Channel Signaling (CCS). During FY89 progress was made in all of these areas, but not all were complete at the end of the FY. In the following paragraphs we present a discussion of the status of CCSIM at the end of FY89 in each of the functional areas plus a little background information for readers unfamiliar with the history and function of CCSIM.

3.1 Background

CCSIM was originally developed at Lincoln Laboratory to support a study of routing and preemption strategies for the DSN and was delivered to DCEC in September 1985 at the close of that study. Starting in FY87 and continuing through FY88, extensive modifications were carried out to allow CCSIM to support the investigation of network management (NM) strategies for the evolving DSN. Unnecessary features of the original CCSIM were removed or deactivated. Models for switch processing and signaling times were added, and a mechanism was provided to account for resources tied up by calls that block at points in the network beyond the source switch. Periodic switch reports and network management controls were added using the DSN Generic Switch Specifications provided by DCEC as well as Northern Telecom Practices as guides. Routing was changed to allow the engineered routes used in the Pacific DSN, and in-band signaling was modeled with respect to signaling delays and damage effects. A sophisticated graphics interface was created to allow an experimenter to view color displays of network status and performance as well as to introduce NM control actions and control all simulator activity.

CCSIM uses an event-by-event simulation technique to work out the fate of individual calls generated randomly with a Poisson arrival-time distribution and exponentially distributed holding times so as to achieve an average offered traffic pattern matching a given point-to-point erlang traffic matrix. The calls so generated are taken as a set of call intentions and are allowed to retry after blocking, preemption, damage, or reaching a busy destination according to parameters set by the experimenter. Statistics produced during the simulation allow the experimenter to observe both the performance of the network

as seen by the switches and presented in the periodic switch reports, and also the performance as seen by the users and presented as Grade-of Service (GOS) and other performance matrices. The latter information is not available for the real DSN or other telephone systems.

CCSIM has been constantly evolving since work resumed in FY87. The initial goal of supporting research in network management has been extended to support the development and field demonstration of NMES and the training of network controllers in the use of the Network Management Support System (NMSS) at DCA-Europe Headquarters (see Section 2). To satisfy the new goals, it has been necessary to increase the level of detail embodied in the simulation. As a consequence, use of CCSIM by a network researcher has been made somewhat more difficult because of the necessity to specify more details about the network to be simulated. We have attempted to minimize this difficulty by providing suitable default values wherever possible.

3.2 Extensions for DCA-Europe Demonstrations

In order to make CCSIM useful in supporting the NMES demonstration and the NMSS trainer at DCA-Europe, it was necessary to make several major changes in the way CCSIM functions and in the way that networks are defined for the simulations. These changes affected the way that trunk groups are represented, the formats of NM controls, and the content of switch reports. In addition, a new mechanism was needed to simulate local traffic so that its effect on the switch reports could be simulated. Switch and trunk group damage models were also changed, and a new capability was introduced to simulate one type of noisy trunk problem. The following sub-sections provide details on these changes and additions.

3.2.1 Trunk Group Representation

In CCSIM at the end of FY88 trunk groups were identified by a source switch name, a destination switch name, and a type that could be either land or satellite. There could thus be at most two trunk groups between any pair of switches. In the real network there is no limit on the number of trunk groups between switch pairs and there are several dimensions in the trunk type space. Trunks are identified by names called CLLIs (Common Language Link Identifiers). We chose to extend CCSIM by replacing the 'net.link' file that previously defined trunking with a new 'net.clli' file. This file has two lines of text for each trunk group. (The group has a different CLLI name as seen by each terminating switch.) Fields on the lines specify the source and destination switches, the equipped capacity, the CLLI name used in the 'other' direction, the type of signaling associated with the group, a satellite/terrestrial type used for signaling delay computations, and a directionality field allowing

CCSIM to handle one-way trunk groups. Other fields specify a trunk group number needed for switch report generation and type information that is not used by CCSIM but is displayed for the experimenter (see Appendix D for details). An optional field allows some control of the order in which parallel trunk groups are searched during routing (see Section 3.2.5).

The process of creating a net.clli file for a network can be very tedious. There is quite a lot of information to transcribe from other sources if a real network is to be simulated. In the case of the sub-set of the European DSN that we simulated for the demonstrations, we had 'XREF' files from NMSS that provided part of the information needed for the 'net.clli' file. We developed a set of UNIX tools to produce a first cut at the 'net.clli' file and to provide consistency checks when the 'XREF' files are changed to match real network changes. The tools, however, cannot do the entire job of creating a correct 'net.clli' because the 'XREF' files do not contain all the required information, and some manual file editing is necessary to achieve a satisfactory simulation.

For the case in which a simulation is desired for an abstract network for which there are no real CLLI names for trunk groups, we have created a utility called 'link_to_clli' that will make a usable 'net.clli' file from the 'net.link' file previously used in CCSIM. This program manufactures CLLI names for the trunks, makes the proper associations, and assures that consistency requirements are met.

3.2.2 NM Control Changes

In the FY88 CCSIM the trunk group controls SKIP, CANT, and CANF were applied to the links between switches, not to individual trunk groups. In the real network they apply to individual groups. We have changed CCSIM to follow the real network practice and to use the CLLI names for the trunk groups. (See Appendix E for a complete description of all CCSIM NM controls as of the time of the demonstrations.)

A consequence of the change is that an experimenter working with an abstract network has to change the way he sets up experiments. For example, if he wanted to see the effect of causing 50% of the traffic being routed from switch A to switch B to skip over that route, and there were more than one trunk group between the switches, he would have to put SKIP controls on all the groups such that the product of the fractions skipped would be 50%. The extra computation is not really burdensome, but the experimenter must take care to carry it out correctly, and consequently there are increased possibilities for error.

3.2.3 New Switch Report Fields

NMSS polls the Northern Telecom DMS switches in the European DSN for certain Operational Measurement (OM) data. It gets five reports which contain status information about the switch itself and one each for the trunk groups at the switch. Some of the trunk groups are inter-switch trunks, but many go to PBXs and 4-wire subscribers. CCSIM generates reportable data for inter-switch trunks only. Of course, the PBXs and 4-wire subscribers could be added to the simulation as switches, but the complete simulation would become unwieldy, and we lack traffic data at the PBX/subscriber level to use as input to CCSIM. Accordingly, we chose to report zero activity on all but inter-switch trunks for the demonstrations and controller training exercises. This simplification seems warranted, since there is little or nothing that a controller can do about any problems that might appear on PBX or subscriber trunks.

The FY88 CCSIM already generated the information needed for all but one of the trunk OM report fields. That field shows the number of trunks in the group that are viewed by the switch as being 'In Service'. We chose to report as 'In Service' the number of undamaged trunks in the group. At the time of the demonstrations we learned that 'In Service' was a number controlled by switch maintenance personnel and is completely independent of the actual state of the trunks (see Sections 3.2.7 and 3.2.8). The other fields show the number of incoming and outgoing attempts, overflows, and trunk usage. The values of these numbers were available in CCSIM, but it was necessary to combine the numbers for different precedences to produce the NMSS OM fields.

The five switch status reports are called 'CPU', 'CP', 'RCVR', 'RADR', and 'DTSR'. The CPU report shows the number of times that mismatches were observed between the redundant CPUs in the switch and the number of warm restarts that have occurred. This information can be useful to a network manager because it shows that the switch is experiencing problems, but these problems do not arise from the network activity that CCSIM is simulating. Consequently, CCSIM reports zeros for the fields in this report. For training purposes it is desirable to introduce other values via the graphics interface, but these are not produced by CCSIM itself.

The CP report shows statistics on Call Condense Block (CCB) activity in the switch. CCBs are storage blocks used by the switch program to remember essential information about the calls it is handling. There are seven numbers in the report. Four are constant and show the quantity of resources available to the switch. In order to produce these numbers, we added a new input file called 'net.sw' for CCSIM to hold these and other switch provisioning numbers which vary from switch to switch and change

from time to time as the network evolves. (See Appendix D for details about 'net.sw'.)

Three numbers in the CP report vary with network activity, and new code has been added to CCSIM to generate them. One is a peg count of CCBs seized in the reporting period. CCSIM calculates this number by accumulating the number of inter-switch call attempts made during the period and adding a number representing local traffic (see Section 3.2.4) and a housekeeping factor said by the NTI Practices Documentation to represent approximately one third of the total. A second reported field shows the number of overflows, i.e., failures to find a free CCB when one is needed. A call experiencing such a failure is "left high and dry" according to the documentation. CCSIM assumes that any such calls are local calls and does not count them in any of its other statistics. The third field represents CCB usage, i.e., the number that are busy as sampled at 100 sec intervals and accumulated over the reporting period. As for the seizures, the usage by inter-switch calls is calculated exactly while the usage by local traffic is approximated. No usage is attributed to the housekeeping factor associated with the CCB seizure calculation.

The RCVR report shows statistics about the activity of the peripheral devices called 'receivers' that are used to receive in-band signaling. There are two types of receivers. One accepts the MF signaling used on inter-switch trunks, the other accepts tone or pulse dialing from users and some PBXs. For each type there is a field showing the number of receivers available to the switch. CCSIM obtains this constant value from the 'net.sw' file. As for the CP report there are three fields that vary with activity. These again represent seizures, overflows, and usage except that usage is based on a ten second scan. The MF receiver seizures are calculated using the actual counts of incoming call attempts while the other seizures use the local traffic approximation. In both cases, the numbers are increased by the number of test calls reported in the RADR and DTSR reports (see below). Overflows are calculated by testing the number of seizures against an estimate of the number that could be handled in the reporting interval. If the seizure peg count exceeds the estimate, the difference is reported as the overflow value. In a real network, the overflows would cause calls to fail, but in CCSIM we cannot relate the overflow number to individual calls and do not show call failures in the statistics as a consequence of receiver overflows.

The RCVR report also has fields for usage of each receiver type. CCSIM calculates these values by multiplying the respective seizure peg counts by a constant factor. At demonstration time this factor was equal to one half. Experience has shown that the factors should be different for the two

receiver types and should vary from switch to switch (see Section 3.2.8).

The RADR (Receiver Attachment Delay Recorder) report shows the results of periodic test calls made to determine whether the switch is providing receivers sufficiently rapidly to maintain proper service. Again there are fields for each of the two receiver types. There are fixed fields that CCSIM gets from the 'net.sw' file that show the requested number of test calls per hour (usually 900) and two delay thresholds (usually 3 and 7 seconds). Data fields report the number of test calls actually made during the reporting period and the number which exceeded either or both of the delay thresholds. CCSIM calculates the number of test calls that would have been made in the reporting period from the requested number per hour. It arbitrarily sets the numbers exceeding the lower delay thresholds equal to one half the number of the corresponding receiver overflows, if any. The number exceeding the higher threshold is always set to zero. This simulation is very simplistic, but we lack data that can be used to develop a more realistic model. It is rare to observe any indication of a failure to complete the test calls within the threshold times.

The DTSR (Dial Tone Speed Recorder) report is similar in intent to the RADR report. It shows failures to provide dial tone to subscribers within a 3 second threshold time. Numbers are reported for both rotary and tone dial subscribers. Fields show the number of tests carried out and the number of times the threshold was exceeded. CCSIM did not simulate these tests at the end of FY88. It reported zeros for all values.

3.2.4 Local Traffic Model

The CP, RCVR, and RADR reports have data fields that depend upon local traffic at a switch. In CCSIM, local traffic includes traffic that is truly local to the switch as well as traffic that goes between PBXs and 4-wire subscribers that are homed on a switch but that does not register as outgoing or incoming traffic on the inter-switch trunks that are being simulated. In the FY88 version of CCSIM an entry on the diagonal of the traffic matrix would cause such traffic to be generated on a call-by-call basis. Since such traffic did not use any resources that CCSIM modeled, all the calls would complete successfully, and there would be no retries or preemptions. For the demonstration, we wanted to be able to set local traffic to sufficiently high levels that evidence of switch problems could be made to appear in the new reports. Generating traffic on a call-by-call basis at those levels would choke CCSIM's CPU and memory resources. We decided instead to introduce a new model that would more economically simulate the effects of local traffic.

The program to model local traffic is executed every switch report sampling interval (100 seconds). It computes the number of new calls that would have arrived and that would have hung up during the previous interval. It uses the same setting for call duration that the experimenter specified for routine inter-switch calls and computes the traffic levels to match the erlang values found in the diagonal (self-to-self) of the traffic matrix. This computation is much less expensive than generating individual calls and maintaining the bookkeeping for them.

3.2.5 Routing Extension

In real networks, routing tables consist of lists of trunk groups to search in a particular order. When parallel trunk groups exist between switch pairs, the engineer building the routing tables can arrange for these groups to be searched in different orders for different routing destinations even though the next switch in the routes is the same. Using this strategy, traffic can be balanced among the parallel groups so that as traffic to a switch increases the overflows will be more or less balanced among the parallel groups. In CCSIM, routing tables are made up of lists of switches rather than trunk groups to try in a particular order. Without any additional mechanisms, this routing scheme will always search parallel trunk groups in the same order. Consequently, the first group searched will tend to exhibit overflows, perhaps at a high percentage, before the second becomes full. There is no difference in the call blocking probabilities or ultimate path through the network (in terms of a list of switches visited) that a call will follow, but the switch reports will look different.

In order to make CCSIM's reports more closely resemble those from the real switches, we added a feature that allows an experimenter to set a probability value associated with the first of the parallel trunk groups that allows the second group to be tried first some fraction of the time. An optional field in the 'net.clli' file line carries the value. If the field is present, the first group will be tried second with a probability corresponding to the field value. Experience has shown that with a modest amount of trial-and-error tailoring, CCSIM switch reports can be made to resemble real switch reports to a satisfactory degree for normal traffic patterns. For abnormal patterns, there can be noticeable differences. The ideal solution to the problem would be to change to trunk group routing, but the change would be very expensive in terms of changes to CCSIM, memory space, and effort on the part of experimenters to get and transcribe the detailed routing information from the real switches.

3.2.6 Noisy Trunk Simulation

Simulation of one kind of noisy trunk problem was added to

CCSIM in FY89. The kind chosen was noise that would cause a user to hang up after a short time because communication was unsatisfactory and then to try the call again, in the hope of getting a better connection. The noise in this model does not cause signaling problems on the trunks. Such a case would require a different model to produce appropriate anomaly symptoms.

A trunk made noisy in the simulator will exhibit short holding times and increased attempt rates in both directions, and call failures will register in the statistics to an extent that depends upon the traffic level and availability of alternate routes and/or parallel trunk groups. The noisy state applies to an entire group and may be turned on and off from the graphics interface or the CCSIM command file.

3.2.7 Damage Model

At the time of the demonstrations at the end of FY89 CCSIM had a model for trunk damage that, as mentioned above, reported as 'In-Service' the number of undamaged trunks in the group. This value should have been independent of actual damage and set arbitrarily by switch maintenance personnel. In the case of a switch outage, the trunks to the switch from its neighbors lost an 'In-Service' trunk each time an attempt was made to route a call to the damaged switch. Further, the attempts from the neighbors failed without reporting overflows until the 'In-Service' value fell to zero. We learned in discussions with field personnel associated with the demonstrations that the non-overflow-reporting situation should occur only in some special cases of switch failure. In most cases, the trunks would exhibit overflows immediately.

It was clear that our damage models at demonstration time were incorrect, at least in so far as the switch reports were concerned, and we acquired some useful information in our discussions. It appears, however, that after the initial transient effects had died out, the damage models showed correct behavior in so far as their effects on the network's traffic handling capabilities were concerned.

3.2.8 Further Work

There are three areas in which further work is needed to complete the extension of CCSIM to support the DCA-Europe Demonstrations and continuing NM workstation development in FY90. The first is the damage model. We gathered useful information during the FY89 demonstrations, but more is needed to get the model to behave properly. Unfortunately, we do not have examples of all the real damage situations that we would like to simulate. We must depend on a mixture of documentation and input from field personnel. For example, we need to know how a switch behaves

when it experiences signaling failure on attempting to use a trunk to a neighboring switch. We have a flow chart in the NTI Practices document which shows how their commercial DMS switches handle this situation. We had two experts at DCA-Europe that told us that document did not apply to the military switches, and they each gave us slightly different versions of how they thought the military switches handled the case. We are proceeding with the development of an improved model, and we expect that several iterations may be required before it performs to the satisfaction of the interested parties.

The second area where further work is clearly indicated is the need to add the Destination Code Cancellation (DCC) control to CCSIM. This is a control used in the old AUTOVON switches that has been provided in the military DMS switches for compatibility. It is not in the DSN Generic Switch Specification and consequently did not get into CCSIM in our earlier work. Its action is similar to the behavior of Code Block (CB) in CCSIM except that it acts on tandem as well as originating traffic at the switch at which it is applied. It is the preferred control to use in the event of switch damage because it is easier to use than CB and it can be applied at fewer switches because of its action on tandem calls. Since it's action is very similar to that of CB in CCSIM, it will be easy to add.

The third area concerns the introduction of numbering plan (code) information into CCSIM. CCSIM generates simulated calls between source and destination switches. In the real network, calls go between individual telephones which have numbers. A Code Block or Call Gap control can be applied to affect calls to individual numbers, sets of numbers, offices, and area codes. It is not practical to change CCSIM to generate calls to individual phone numbers, but it would be possible to do so for office codes, allowing several such codes to a switch. Requiring the use of codes would make CCSIM use by experimenters interested in abstract network design issues more complex. Consequently, any addition of codes to CCSIM should allow for continued use without requiring the experimenter to come up with all the detail associated with the relationship between codes and switches. A code capability was not needed for the FY89 demonstrations. It is desirable for the sake of completeness, but it is not clear that there will be a real need for it in FY90 CCSIM applications.

3.3 Simulation of Candidate DSN Routing Strategies

The original version of CCSIM had a variety of routing and preemption options that were disabled when the Engineered Routing mechanisms were installed to allow simulation of the DSN during FY87-88. Task II of the FY89 SOW called for the restoration of these capabilities. This work has been accomplished. CCSIM now works with Forward, Modified Forward, and Engineered routing tables. Crankback can be used with any of those tables. Primary

Path Only routing can be obtained by setting the value of the route control digits to 'one' in the Engineered routing table or by the use of the Alternate Route Cancellation (ARC-B) control. Adaptive Routing can be realized by use of the CHNG-ROUTE-TABLE command to CCSIM, but the experimenter must provide the tables that are appropriate to the changed network conditions.

The 'genrt' utility has been updated to work in the SUN workstation. It generates Forward and Modified Forward routing tables from 'net.node', 'net.cost' and 'net.conn' files. The 'cost' file provides a metric that is used to determine which route will yield the shortest 'distance' to the destination node from any node in the network. The 'conn' file tells what connectivity exists among the switches. Another utility was updated that can be used to calculate the cost file from the latitudes and longitudes of the switches that are found in the node file. We were not pleased with the performance of the routing tables that we built using the cost files generated by that utility. Either hand tailoring of the cost file or a better distance calculation seems to be needed to get a good routing table. To facilitate the generation of routing tables, a new utility was written to build 'net.conn' files from the 'net.link' files that are more readily available to users at DCEC.

To aid in the modification of Engineered routing tables we wrote a set of utilities that produces human-readable and easily editable forms from the form used internally by CCSIM and provided by DCEC. A matching set of utilities converts the forms back to the internal form after editing. The latter make a set of consistency checks but do not check for routing loops or shuttles. These utilities were very helpful in creating the routing tables needed to simulate the sub-set of the European DSN that we simulated for the demonstrations.

Crankback has been restored to operation and checked out in CCSIM. It may be used in conjunction with any of the routing table types. When crankback is activated, CCSIM ignores the route controls associated with the routing tables and explores the possible routes according to its algorithm. Two types are available. One explores all routes at each switch. The other explores primary paths only at nodes not marked as 'spill' nodes in the 'net.node' file. Loops and shuttles are avoided by a node trace that prevents a switch from being revisited in the search. Limits on the number of crankbacks allowed are provided independently for each precedence level. The resources used by the 'stubs' where blocking occurs during the search are modeled by dummy calls that are hung up after an appropriate time derived from the signaling and switch processing time parameters.

Preemption options in CCSIM have been expanded so that two types can be simulated in either of two moods. The types are 'out' and 'back', and the moods are 'ruthless' and 'friendly'.

Preemption on the way out means that calls are preempted during the routing process whether or not the preempting call will succeed. This is the type used in the real DSN. Preemption on the way back means that preemption will take place only when a call can complete to its destination. In a real network, back preemption would require Common Channel Signaling for implementation, but CCSIM does not force that constraint. Ruthless preemption means that a preemptive search is made whenever a free trunk is not found on the next route in the route list. Friendly preemption means that preemptive searches are made only when the direct and all alternate routes have been searched for a free trunk. The real DSN uses the ruthless mode. Our experiments show that the friendly mode results in fewer preemptions at moderate traffic densities but offers no advantage under either low or high traffic conditions.

3.4 Modeling Data Calls

CCSIM has been extended to provide a second set of traffic generators that can be used to generate traffic with different holding time and arrival rate statistics. Data calls are the typical source of such traffic. Lacking information on data call statistics, we simply replicated the voice call traffic generators already in use in CCSIM and provided for a separate traffic file (in erlangs) called 'net.dtraf' and global variables to set the average message lengths and distribution as a function of precedence. Another global variable in the net.inval file indicates whether or not data calls are desired in a simulation. If not, CCSIM does not look for the data traffic matrix and generates no data calls.

If data calls are turned on, their traffic level can be changed dynamically from either the graphics interface or the command file. The data calls are mixed with the voice calls in the GOS and other statistics generated by CCSIM.

3.5 Developing a CCS Model

A model for Common Channel Signaling (CCS) in CCSIM was developed and presented to DCEC for approval. It allows the experimenter to set arbitrary message sizes for call setup, takedown, preemption, etc. messages. From these it calculates the CCS message traffic on each CCS link involved in a call processing event. From the average over a sampling period settable by the experimenter, it estimates the expected queuing delay on each link and uses this value in computing call setup time and the time that resources are held up when calls block. Statistics are generated showing the overall average signaling rate and the peak observed in any sampling interval.

In this model, whether a trunk group uses CCS or In-band signaling is specified by a field in the 'net.clli' file. Two

other files provide the capacities and delays that are to be used for the associated CCS channel.

Implementation of the CCS model was one of the activities that suffered when priority was given to the DCA-Europe demonstrations. Work was begun on this model in FY89, but it was not complete by the end of the year. Work continues into FY90, and CCS will be discussed in more detail in the FY90 Annual Report.

3.6 New CCSIM Network Management Control

During FY89 an experimental network management control was added to CCSIM. Called 'CAN-EOC-OVF', it cancels routine precedence calls that overflow the last trunk group in their routing chain at the switch at which it is applied. The canceled calls are given the treatment associated with Emergency Announcements, i.e., they are not allowed to retry and are considered to be failed calls. To our knowledge, the control is not found in any real switch. It was implemented in CCSIM to investigate the benefits, if any, that would occur if the traffic that was having the greatest difficulty getting out of a switch under heavy load conditions could be canceled. The effect is similar to that obtained by putting CANF on final trunk groups, but is easier to apply and potentially more effective because a particular trunk group may be a final group for some destinations and not final for others.

Only a few experiments with CAN-EOC-OVF were carried out in FY89, and little can be said about the potential utility of such a control from the results. In general, we have found that controls that cancel calls rarely have a beneficial effect on network performance. Their use almost always reduces the number of successful calls in the network as compared with taking no action at all. They can have application in a real network as a temporary measure to avoid routing loops and shuttles while routing table changes are being introduced, and they could be helpful in reducing switch congestion effects, but those applications cannot be readily evaluated in simulations with CCSIM since routing changes there are instantaneous, and congestion effects are not modeled.

3.7 Graphics Interface Enhancements

Many enhancements to the CCSIM Graphics Interface were carried out in FY89. A more user-friendly interface was achieved by making the mouse button usage consistent in all contexts, by displaying all pertinent information when a user response is needed, and by checking all command entries to preclude illegal commands, thereby avoiding error messages.

Switch coloring was added to the previously available trunk coloring, and the mechanism for selecting and identifying the colorings was improved. Coloring is now controlled either by stepping through a list of options via a mouse button or by using a pull-down menu to show all the options and allow a direct selection to be made. The control buttons are displayed in a line across the top of the screen together with labeled color bars showing the relationship between numerical values and colors. The interface software allows a value of zero to be represented by a completely different color than that used for the smallest non-zero value in the range. For example, when the lines representing trunk groups are colored to show overflows, a color range from yellow to red is used for the numerical range of zero to 100% in the ratio of overflows to attempts, with the exception that the absence of any overflows is indicated by a green line. The currently available coloring options are:

For Switch Colorings:
- Overall Incomplete Calls
- Tandem Incomplete Calls
- Originating Incomplete Calls
- Controls
- Monitoring Off

For Trunk Colorings:
- Usage
- Precedence Usage
- Overflows
- End Chain Overflows
- (ACH) - Attempts per Circuit per Hour
- (OCCH) - Outgoing Connections per Circuit per Hour
- (ICCH) - Incoming Connections per Circuit per Hour
- Monitoring Off

Pop-up windows were also added in FY89 to show the detailed information available from the switch reports in numerical form. Mousing either a switch icon or a link line brings up a menu that allows the selection of the information to be displayed. For example, at a switch the user has a choice of seeing information pertaining to the switch itself or to all of its trunk groups.

To better support the DCA-Europe demonstrations, a second type of switch icon was added to allow tandem switches to be easily distinguished from end offices. In addition, two sizes of switch icons and two link line thicknesses were provided to show whether the resources being represented in the display are or are not part of the sub-network being monitored by the ACOC in the demonstrations.

A new command was added to allow CCSIM routing tables to be changed by the graphics interface operator in the middle of a simulation. The new routing table must already exist as a

properly formatted net.route file at the time it is to be invoked from the interface.

The facilities for displaying the statistics matrices generated by CCSIM have been enhanced by adding row and column totals. The matrix displays now automatically expand the display space used so that large numbers in any cell of the array can be presented. In earlier versions, the space per cell was fixed, and numbers too large to be displayed were marked with '*'s to show that they were out of range.

The 'snapshot' capability of the interface has been enhanced so that when a snapshot is taken the statistics matrices are saved to a file as well as the switch report data needed to produce the color displays. The saved matrices can be viewed when the snapshot is read back at a later time. In addition, when a snapshot is to be taken, the interface now proposes a unique name for the file which the operator can accept or override, and when a snapshot is to be read back, the interface provides a menu of saved snapshots from which the operator can choose. These features simplify the taking and viewing of snapshots and reduce the likelihood of operator error.

4.0 TRANSMISSION SYSTEM ALARM INTEGRATION STUDIES

This Section is identical to Section 5.0 of the FY89 Annual Report submitted by Lincoln Laboratory to Rome Air Development Center. The topic area of this Section is closely related to the MITEC Expert System which is the main focus of the RADC-sponsored work, hence it was appropriate to include this material in the RADC report. Sponsorship for the work described in this section is provided by DCEC/R220, hence it is appropriate to include the same material in the FY89 Annual Report to DCEC.

4.1 Introduction

A significant part of the FY89 effort in the MITEC project has been to begin to scope and understand the problems of correlating and interpreting transmission system alarms and presenting filtered results to Tech Control and Network Management decision makers. In particular, it is recognized that these decision makers will be able to function more rapidly and effectively if they can obtain immediate information about transmission system problems from TRAMCON and DPAS systems, rather than having to wait while the user community slowly reacts to transmission impairments and produces traffic pattern distortions that allow the managers to recognize and correct the problems. By analogy with the success of the Call-by-Call Simulator (CCSIM) as a portrayal of realistic network behavior to use in developing network management knowledge, it has been conjectured that TRAMCON and DPAS alarm generator systems could make up for the lack of access to real transmission networks for MITEC software developers, as well as the fact that real networks seldom produce interesting alarm patterns. An FY89 task for Lincoln Laboratory has been to define and specify these alarm generation systems, with an eye to beginning their implementation in FY90. Section 4.2 is the statement of required functions for the alarm generators that was written to serve as a guide for a study effort in the remainder of the year, and Appendix D is the result of that study, namely a Software Requirements Specification for the TRAMCON Event Generator (TEG).

4.2 Specification for TRAMCON and DPAS Alarm Generators

4.2.1 Background

Figure 4.1 illustrates 1) a military TRAMCON (TRAnsmission Monitoring and CONtrol) system, with the microwave radio network segment from which it gathers alarm information; 2) a DPAS (Digital Patch and Access System), for which the radio segment provides one or more of the T1 carriers to be switched and cross-connected; 3) a system control software facility, specifically the MITEC (Lincoln Laboratory Machine Intelligent Tech Controller) expert system, receiving and interpreting TRAMCON and DPAS alarm information; and 4) a simulation system

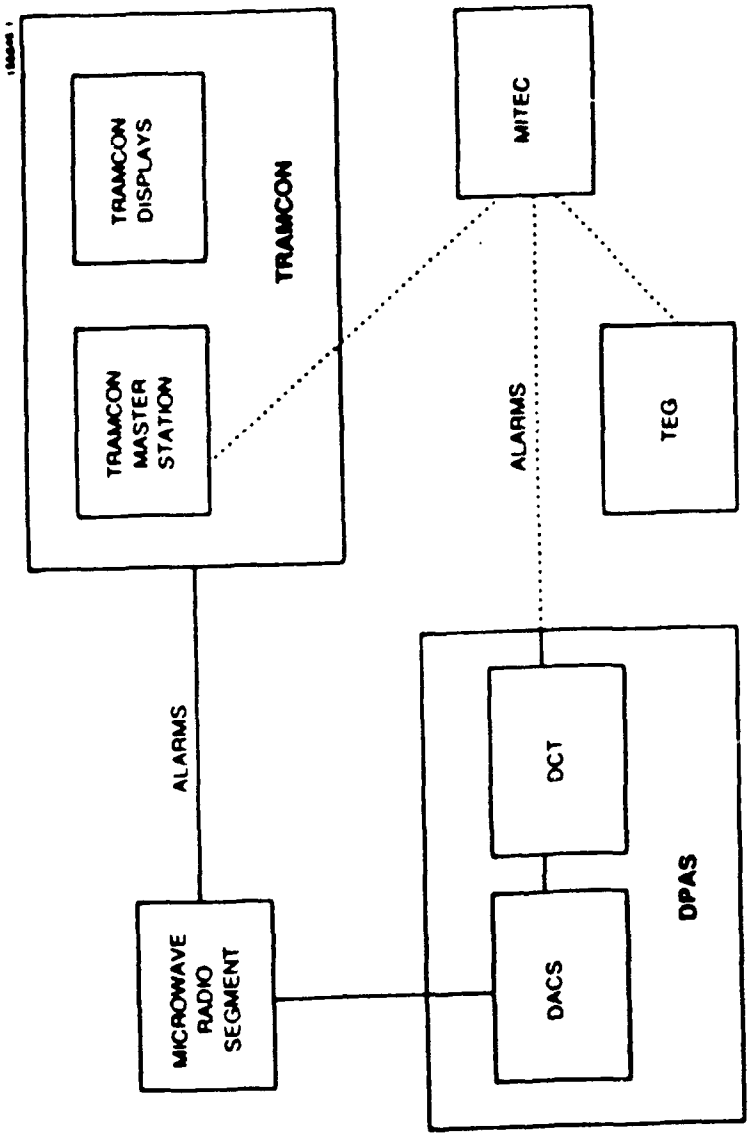


Figure 4.1
Alarm Monitoring Systems

that produces data streams realistically representing TRAMCON and DPAS alarm patterns, in terms of format, content, timing, and relationship to other network faults and phenomena being simulated elsewhere. (An example of such other simulation is the Call-by-Call Simulator of the Defense Switched Network, in which voice traffic behavior is affected by outages of trunk circuits carried by the microwave radio network.)

The purpose of this Specification is to describe the requirements for the TRAMCON and DPAS alarm simulators.

4.2.2 Microwave Radio System Description

A Microwave Radio Segment consists of a string of microwave sites providing voice and data transmission via standard military transmitter/receiver equipment. Typically these sites also have multiplexing equipment which is used to combine lower-speed data streams from the local area into high-speed composite (i.e., T1 and T3) signals for radio transmission. All of this equipment is provided with various built-in fault detection features, with front-panel alarm lights coupled with electrical alarm signals that can be remotely monitored. There are moderately complex relationships among the various possible radio system failures, and the patterns of primary and sympathetic alarms they precipitate in other equipment, locally and at other sites. As one example, loss of transmitter power at an upstream site will cause a data loss alarm to occur at every downstream site.

4.2.3 TRAMCON Operation

There are two motivations for centralized monitoring of all the alarm signals generated by an entire radio segment: 1) some sites are manned only part-time, or not at all, and 2) simultaneous observation of alarm patterns from throughout a segment can lead to more rapid diagnosis of problems than would be possible from the restricted viewpoints of individual operators at local sites. These factors led to the development and deployment of TRAMCON systems, consisting of Datalok 10 alarm logging and command transmission devices at each radio site in a segment, all of which are connected to a TRAMCON Master Station located at a key site. The Datalok devices at all sites are polled on a regular basis by an HP1000 computer which is the main component of the TRAMCON Master Station; the alarm information is stored and summarized by the HP1000, and displayed in either color graphics or tabular form as selected by the operator.

A complete list of alarms monitored by TRAMCON is available elsewhere. Besides a variety of communications equipment alarms, this list includes critical facility information such as smoke detector outputs and emergency generator readiness. A skilled TRAMCON operator can recognize distinctive patterns of these alarms as presented by his display, quickly identifying the root

causes of many different problems. In many cases the radio equipment will have switched automatically to spare modules, and the TRAMCON operator dispatches repair crews to correct specific problems in the on-line modules; in other cases the operator can alleviate problems by sending certain commands to the remote equipment via TRAMCON.

4.2.4 DPAS Description

Whereas TRAMCON is a deployed operational system, the procurement of DPAS systems is just beginning. Over 100 of them will ultimately be deployed in the Defense Communications System. The DPAS consists of an AT&T DACS II (Digital Access and Cross-connect System, Version II) plus a DCT (DACS Control Terminal), together providing the capability to switch and multiplex numerous T1 signals at both the channel and subchannel levels. The DPAS provides its own set of alarm signals which are collected and displayed by the DCT, and follow standard commercial telecommunications practices as to their sources and meanings. Moderately complex relationships exist among microwave radio fault conditions and DACS alarms. An important objective of the work described here is to formulate and analyze these relationships.

4.2.5 Potential MITEC Interactions with TRAMCON and DPAS

The primary function of the Machine Intelligent TEch Controller (MITEC) expert system is to perform fault isolation and service restoral at a Tech Control Facility (TCF) for dedicated military circuits, including faults within the TCF as well as in transmission facilities linking it to other TCFs. In the latter case, human Tech Controllers can be very substantially aided in troubleshooting by having TRAMCON and DPAS alarm information available: they can pinpoint a trouble location more quickly by analyzing alarm data, rather than conducting a sequence of tests to infer the nature and source of the trouble. It is within the purview of MITEC to embody and apply the knowledge that human Tech Controllers use in exploiting TRAMCON and DPAS; the questions are 1) how best to get the TRAMCON and DPAS information into MITEC, and 2) what reasoning and pattern recognition processes to use upon the information.

The operator displays at a TRAMCON master station contain the entire body of alarm information currently available. In principle MITEC could access the same data stream that drives the displays, but the useful information is enmeshed in screen formatting and control codes that are necessary for the color graphics monitor on the HP1000. One would not want to waste MITEC's resources in stripping out all of these unneeded codes; instead, the ideal course of action is ultimately to create a new interface module in the TRAMCON software that implements

efficient computer-to-computer communication of the alarm information in a form that is directly exploitable by MITEC.

The fault alarm vocabulary of the DPAS is known, since it is identical to that of the DACS-II which is widely deployed in the commercial telecommunications environment. For T1 signals which are provided to a DPAS by way of military microwave links, the relationships among TRAMCON and DPAS alarms can be deduced and analyzed for a variety of fault conditions. In the future, when DPAS systems are more widely deployed, simultaneous access to both DPAS and TRAMCON alarm information will be able to speed the diagnosis of various fault types. It is possible during the present ongoing development of the DACS Control Terminal to specify a computer-to-computer communication link to provide future access by MITEC to DPAS alarm data.

4.2.6 TRAMCON and DPAS Alarm Generator Requirements

The TRAMCON and DPAS alarm simulation system shown in Fig. 1 has two primary purposes: to support the development of MITEC software for alarm analysis and fault diagnosis, and to form a part of a broader simulation of Defense Communications System (DCS) operation. To meet these goals it must provide the following functions:

1. Storage of an internal representation of microwave and DPAS network segments, including all the equipment items and their alarm functions;

2. An operator interface which permits selection of any realistic failure event(s) in the networks;

3. A message interface which will permit failure event selection by a remote computer as an alternative to local selection by a human operator;

4. Internal generation of all the primary and sympathetic TRAMCON and DPAS alarms that would result from the selected failure event(s); and

5. Communication of this alarm information to MITEC in a manner consistent with the way TRAMCON and DPAS would communicate the alarms if they were provided with communication links to MITEC.

4.2.7 Development Sequence

It is clear that meeting all aspects of the requirements above would be a lengthy and expensive process. On the other hand, we believe that some modest level of completeness and precision, at a more modest cost in time and manpower, will

satisfy our needs in the near term. We envision addressing this development in several stages, so as to identify and reach that modest level. Since we are breaking new ground in this effort, it does not make sense to try to tightly specify milestones, manpower and completion dates the way one could do for a job that closely resembled past projects.

The first step will be to develop a more complete software design specification. This will include analyzing a variety of information on the TRAMCON system, the radio segment alarms and their relationships, and the DPAS system and alarms. Much of this information and analysis is already available at Lincoln Laboratory, and more will be obtained as necessary from a variety of sources (to be listed separately). This activity will lead to an initial understanding of the rules that govern the creation of alarm patterns. Preliminary ideas on the hardware/software environment for the alarm generation system will be advanced, and a high-level software design will be prepared. It is anticipated that this first step will be done by the end of FY89, with the level of completeness and sophistication to be determined adaptively as the work progresses.

The second step, to be undertaken in FY90, will be to choose an appropriate set of functions from this software design and proceed with initial implementation. Test and experimentation with the resulting system will lead to practical choices for further work. The goal is to have a practical, working TRAMCON/DPAS alarm simulator in hand in FY90.

APPENDIX A NMES FIELD INSTALLATION AND DEMONSTRATION PLAN

A.0 Introduction and Summary

The DCA has requested field demonstrations at DCA-Europe of a Network Management Expert System (NMES) and a DCOSS DSN Network Management Operator Trainer developed by Lincoln Laboratory, and a Learning and Recognition System (LARS) neural net implementation developed by GTE. The purpose of this Plan is to describe the functions of the Lincoln Laboratory equipment, the demonstration scenarios that will be performed with it, and the procedures and requirements for installing, checking out and demonstrating the systems. While GTE will provide a separate plan for LARS, the two plans will be closely coordinated through DCEC and the work will be performed cooperatively at DCA-Europe.

A.1 Functional Description of Demo System

Figure A-1 illustrates in a single diagram the existing DSN network management facilities at DCA-Europe and the new facilities that are to be delivered to the site in July and demonstrated in September, namely the GTE LARS neural net pattern recognition system; the Lincoln Laboratory DCEC/R610-sponsored NMES (Network Management Expert System); and the Lincoln Laboratory DCA/B230-sponsored DCOSS DSN Network Management Operator Trainer. The following paragraphs in this section describe the functions and interconnections of these facilities.

A.1.1 Existing DSN Network Management Facilities

Figure A-2 shows the existing Network Management Support System (NMSS) for DSN Network Management at DCA-Eur, supporting the ACOC staff in the tasks of monitoring DSN status, diagnosing network problems, and taking corrective measures. (The other equipment in the diagram has been concealed with cross-hatching, to avoid confusion in describing the NMSS.) This structure has been fully described in documentation produced by GTE contract support personnel at DCA-Europe. Only a brief summary is given here to set the stage for the demo system descriptions to follow.

At the upper left in Fig. A-2 are Northern Telecom DMS-100 switches installed to date in the European DSN and provided with network management (NM) connectivity to DCA-Europe. The number of such switches is on the order of 12, and slowly increasing, but still a modest fraction of the expected total number of European DSN switches. The NM connectivity is obtained via 1200-bps modems on a dedicated telephone circuit to a Maintenance and Administration Position (MAP) port on each of the DMS-100 switches.

Within the Area Communications Operations Center (ACOC) at DCA-Eur is an IBM PC running the NMSS Data Acquisition Interface

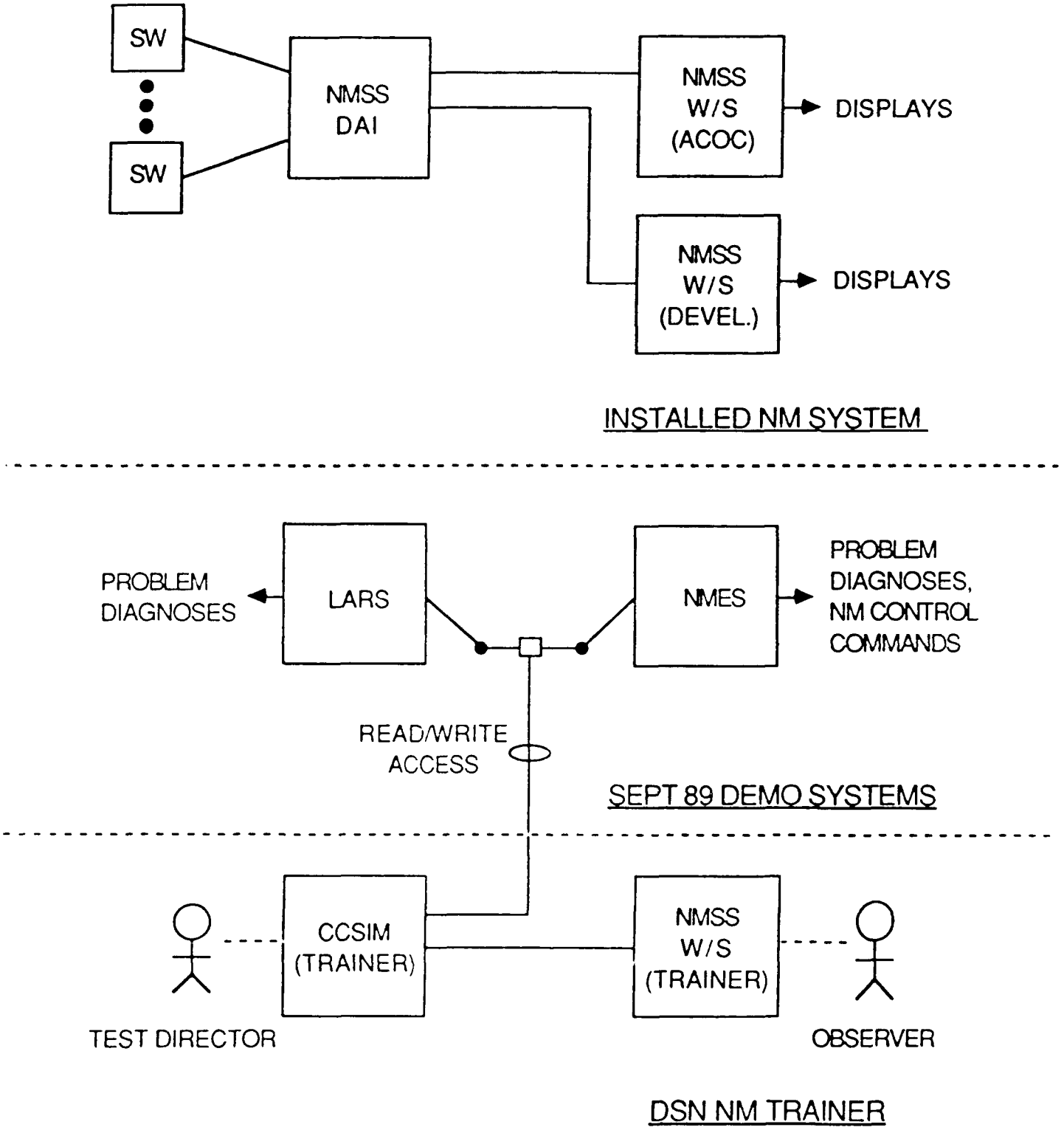
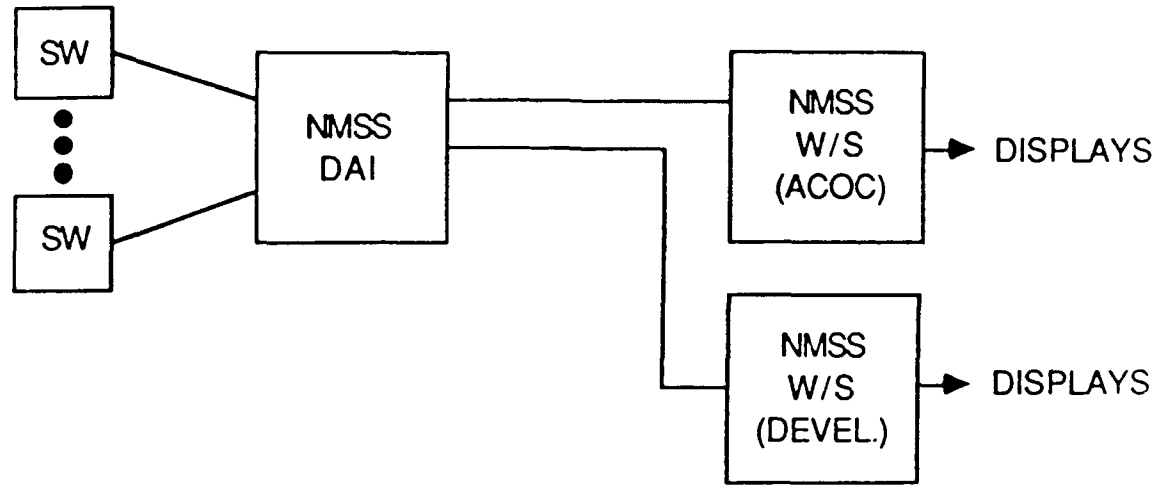
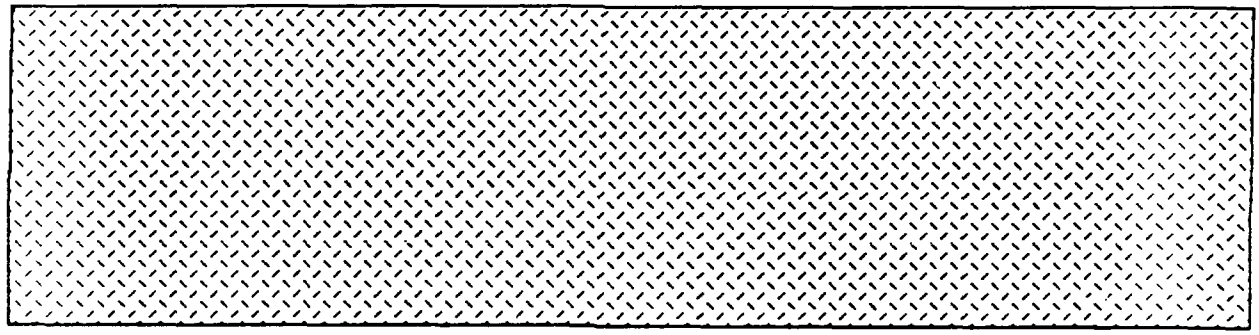


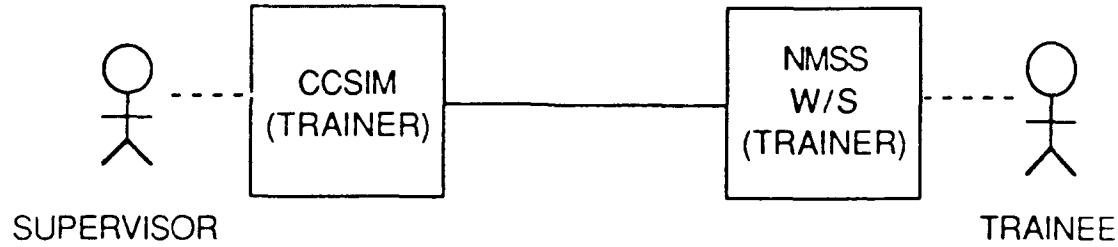
Figure A-1
DCA-Eur NM Facilities, Sept 89



INSTALLED NM SYSTEM



SEPT 89 DEMO SYSTEMS



DSN NM TRAINER

Figure A-2
Normal ACOC NM Operation

(DAI) software system at the top center of Fig. A-2, which is connected through a multi-port smart switch (not shown) to the DMS-100 MAP ports via modems and phone lines. Initially the host computer for the DAI software has been an IBM PC/AT; by September 1989 it will be replaced by a considerably more powerful IBM PS/2 Model 80. A MAP port is normally used by switch operations personnel on site to perform maintenance actions and to make data base changes and the like. The DAI is actually logged in as a user on all of the MAP ports. In current operation, the DAI polls each DMS-100 every 15 minutes, collecting formatted activity reports which it then processes and condenses.

At the upper right in Fig. A-2 are shown two identical NMSS workstations, which are also IBM PC/ATs and will soon be replaced by IBM PS/2 Model 80 computers. One NMSS workstation is used operationally in the ACOC, and the other is kept in the office area of the GTE support contractors, where it supports ongoing NMSS software development. Each workstation has two monitors: one provides color graphics displays of the latest network status updates from the DAI, and the other provides detailed text displays of supporting information. The ACOC staff analyze this data and select NM control commands when appropriate to correct observed network problems. The NMSS has the capability to accept these control commands from the ACOC staff and to transmit them to the DMS-100 switches. Under normal circumstances, ACOC controllers coordinate all control requirements with switch site personnel.

A.1.2 The LARS Neural Net Pattern Recognition System

The central portion of Fig. A-3 shows the September 1989 network management demonstration systems that are to be installed at DCA-Eur. Both are shown connected via a read-only port to the DAI, in the live-data demonstration configuration discussed below. At the left is the LARS (Learning and Recognition System) which is to be developed, installed and tested by GTE. It is a Neural Net implementation whose purpose is to continually scan the same DMS-100 activity report summaries that the DAI sends to the NMSS workstations, and to announce occurrences of any events in the repertoire of DSN problem conditions that it has been trained to recognize. This information is to be presented to the ACOC operator for his consideration in selecting and applying NM control commands. Further information about LARS will be provided by GTE as agreed between them and the Government.

While Fig. A-3 shows the GTE LARS and the Lincoln NMES connected simultaneously to live data, please note that it is not intended that formal demonstrations of the two systems will be given simultaneously in September 1989. As noted above, the two development efforts have been entirely independent to date. Their displays, functions and operator interfaces overlap in some ways

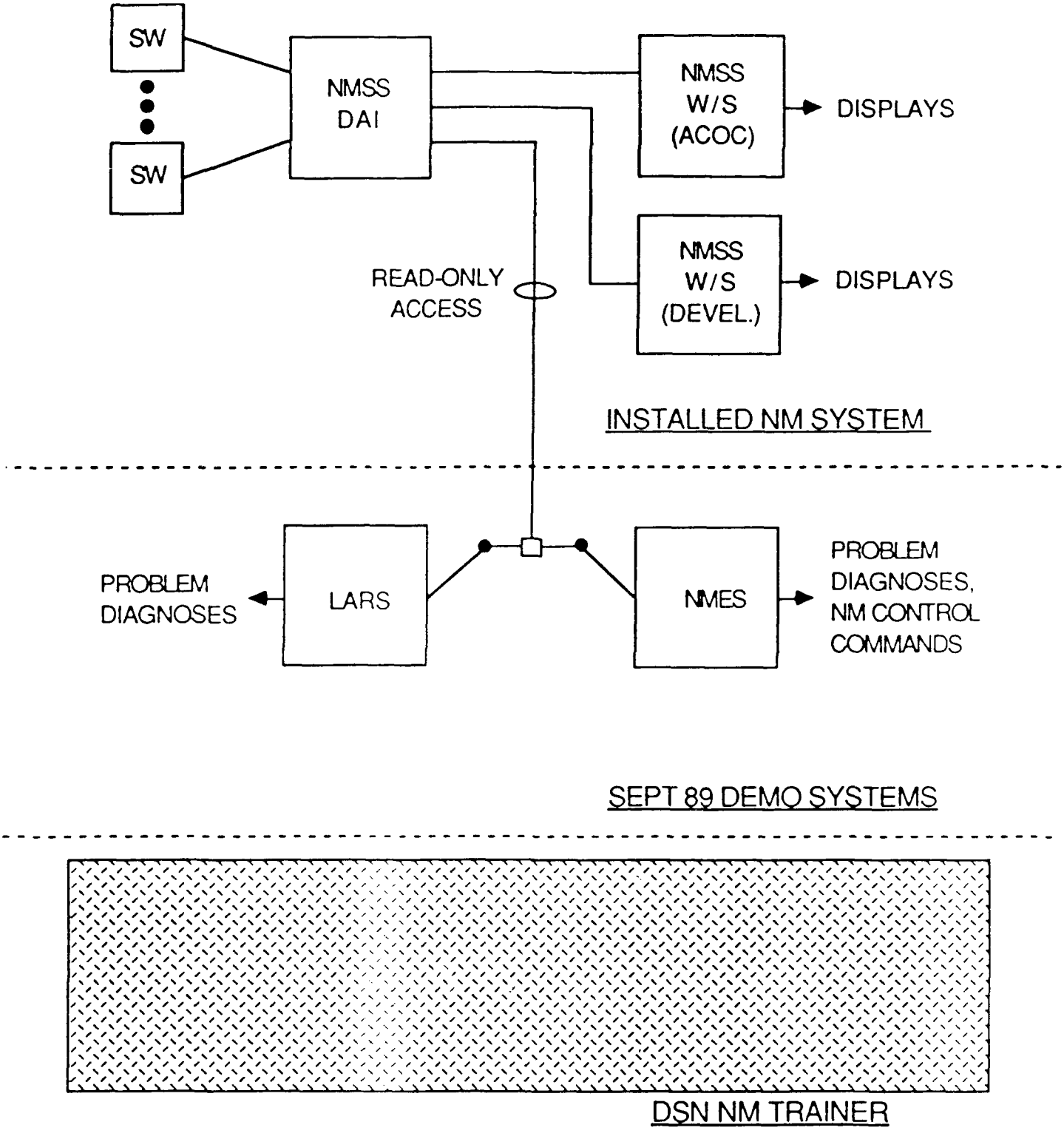


Figure A-3
NMES/LARS Demo (Live Data)

but are very different otherwise; consequently simultaneous demonstrations for visitors could lead to great confusion.

As explained below, DCEC plans to have GTE and Lincoln Laboratory merge the two systems in FY90, and at that time a combined demonstration will be entirely meaningful.

A.1.3 The Network Management Expert System (NMES)

The functions of the Lincoln Laboratory NMES in Fig. A-3 are:

i. Continually scan the DMS-100 report summaries;

ii. Recognize those DSN problem conditions that are included in the NMES Knowledge Base;

iii. Select and parameterize NM control commands as appropriate to combat observed DSN problems, within the capabilities of the NMES Knowledge Base;

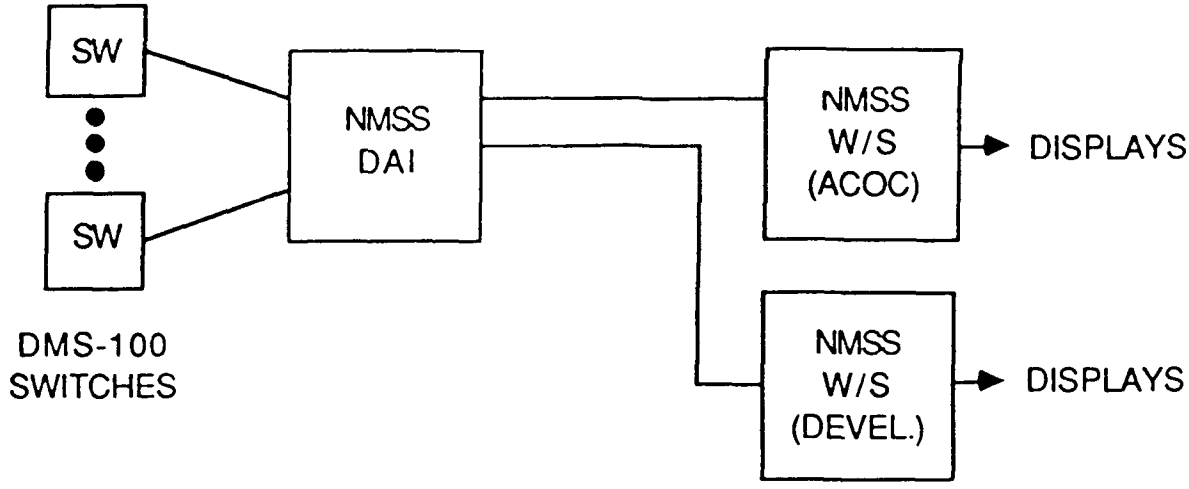
iv. Apply the selected commands to the DMS switches involved (note: for the present this must be done by manual intervention, as explained in Section A.1.1 above);

v. Scan the DMS-100 reports for indications that the problem conditions have gone away, and remove the control commands as soon as possible.

NMES functions i. and ii. above are generically similar to the functions of LARS, and are implemented in software structures called "Monitors" and in some of the rules within the Expert System, which were designed prior to the current involvement among LARS, DCA-Eur and NMES. The September demonstrations are a prelude to a DCA-requested FY90 collaborative effort in which LARS will be interfaced with NMES as a potentially efficient way to perform functions i. and ii.

A.1.4 The DCOSS Operator Trainer

Figure A-4 illustrates the operation of the DSN Network Management Operator Trainer. Independently of the development of neural nets and expert systems, there is an immediate need for ACOC staff to be trained in DSN network management. Until automated systems are fully developed and deployed the NM can only be done by human operators, and they must be prepared to respond effectively to damage and overload scenarios that have never yet occurred in the new and uncompleted DSN. The accumulated past experience of DCA AUTOVON network management operators is generally not applicable, because the NM capabilities of AUTOVON switches are very limited compared to those of the modern DSN equipment.



INSTALLED NM SYSTEM

SEPT 89 DEMO SYSTEMS

DSN NM TRAINER

Figure A-4
ACOC NM Operator Training Mode

The mechanism selected by DCA/B230 to provide the necessary training is network simulation, based upon the existing DSN Call-by-Call Simulator (CCSIM) that has been developed by Lincoln Laboratory under DCEC/R610 sponsorship to support NM knowledge acquisition and NMES development. The Trainer will be delivered to DCA-Eur in the same time frame as the NMES, and will have an important function in the demonstration of NMES, as described below. CCSIM is a large software system running in a Sun 3/260 workstation which simulates all the user and network activity in a theater-wide DSN, including call origination, route selection processing, call blocking events, and preemptions. The CCSIM operator (shown as the Supervisor in Fig. A-4) has a color graphics interface which allows him to monitor the progress of a simulation run, and to inflict network damage and traffic distortion events at will. CCSIM produces a flow of switch activity reports for each switch in the simulated network, identical in format and meaning to the output of a DAI.

The NMSS workstation at the bottom right of Fig. A-4 is to be identical to those at the top of the diagram, and is to be dedicated to operator training. Connected to CCSIM, this workstation will produce displays that are indistinguishable from live data. The Supervisor will set up simulations and problem scenarios as he sees fit, and the Trainee will learn to analyze the displays, recognize problems, and apply control commands using the Trainer NMSS workstation. CCSIM will accept and implement the control commands, and subsequent switch reports will reflect their effects. The Supervisor can remove the network problem conditions at will, and require the Trainee to recognize that fact and respond by removing the control commands.

CCSIM can be loaded with any desired network topology, connectivity and traffic matrices. Besides training operators on the current DSN configuration, then, it can train them on advanced configurations that will exist after further network upgrades.

A.1.5 Use of the Trainer to Demonstrate NMES and LARS

The DMS-100 is a reliable, modern system, and it can be anticipated that live DSN problems in the recognition repertoires of LARS and NMES may occur rarely if at all during the September 1989 demos. While a legitimate function of LARS and NMES is to report that DSN operation is normal, it will be an uninteresting demo if that is all they have occasion to do. Also, it was noted above that the current DCA-Eur NM system accesses only a modest fraction of the DSN switches; this will make it impossible to diagnose problems elsewhere in the net. Conversely, access to switch statistics from throughout the net would make for a much more interesting diagnosis capability.

Figure A-5 illustrates the use of CCSIM in the Trainer as an attractive solution to these problems. As noted above, its outputs are indistinguishable from those of a real DAI on a real network; hence LARS and NMES can select the CCSIM as an alternative to the real DAI during part of the demo period, so that controlled, interesting network problem scenarios can be set up and run at will by the Test Director. Although the September demonstrations of LARS and NMES will be conducted independently of each other, there is nothing to prevent having them connected at the same time to the same stream of CCSIM switch reports as shown in Fig. A-5. Throughout these simulated-data tests, a knowledgeable Observer at the NMSS (Trainer) console can analyze his graphics and alphanumeric displays and assess the correctness of LARS and/or NMES performance. Interest in such tests is substantially enhanced by the fact that NMES can have full read-write access to the MAP ports of the simulated switches in CCSIM; thus NM control actions selected by NMES can in fact be implemented immediately in CCSIM, and their effects can be observed by the demo attendees. Moreover, the Test Director at the CCSIM control console can respond to requests by demo attendees to set up and explore any additional cases they may wish to see.

A.2 Demonstration Scenarios

Although LARS and NMES are shown connected side by side in Figs. A-3 and A-5, their September 1989 demos will be functionally independent of each other, as previously noted. They may run concurrently to whatever degree is found convenient for GTE, Lincoln Laboratory and the DCA; but the FY89 LARS and NMES development and implementation schedule constraints are such that no time is available for the collaborative work that would be needed to achieve a meaningful joint demonstration. The objectives of the LARS demo will be arranged between GTE and the Government, and will not be further discussed here. The following paragraphs in this section refer only to NMES scenarios.

A.2.1 Live Demonstration Goals for NMES

When initial plans were being developed between Lincoln and DCEC/R610 in late FY88 for the DCA-Eur demonstration of NMES, it was agreed that, as a minimum, NMES will:

1. Recognize outages among the DMS-100 switches to which it is connected;

2. Recognize outages of trunks connected to those DMS-100 switches;

3. Recommend NM control actions to compensate for these outages;

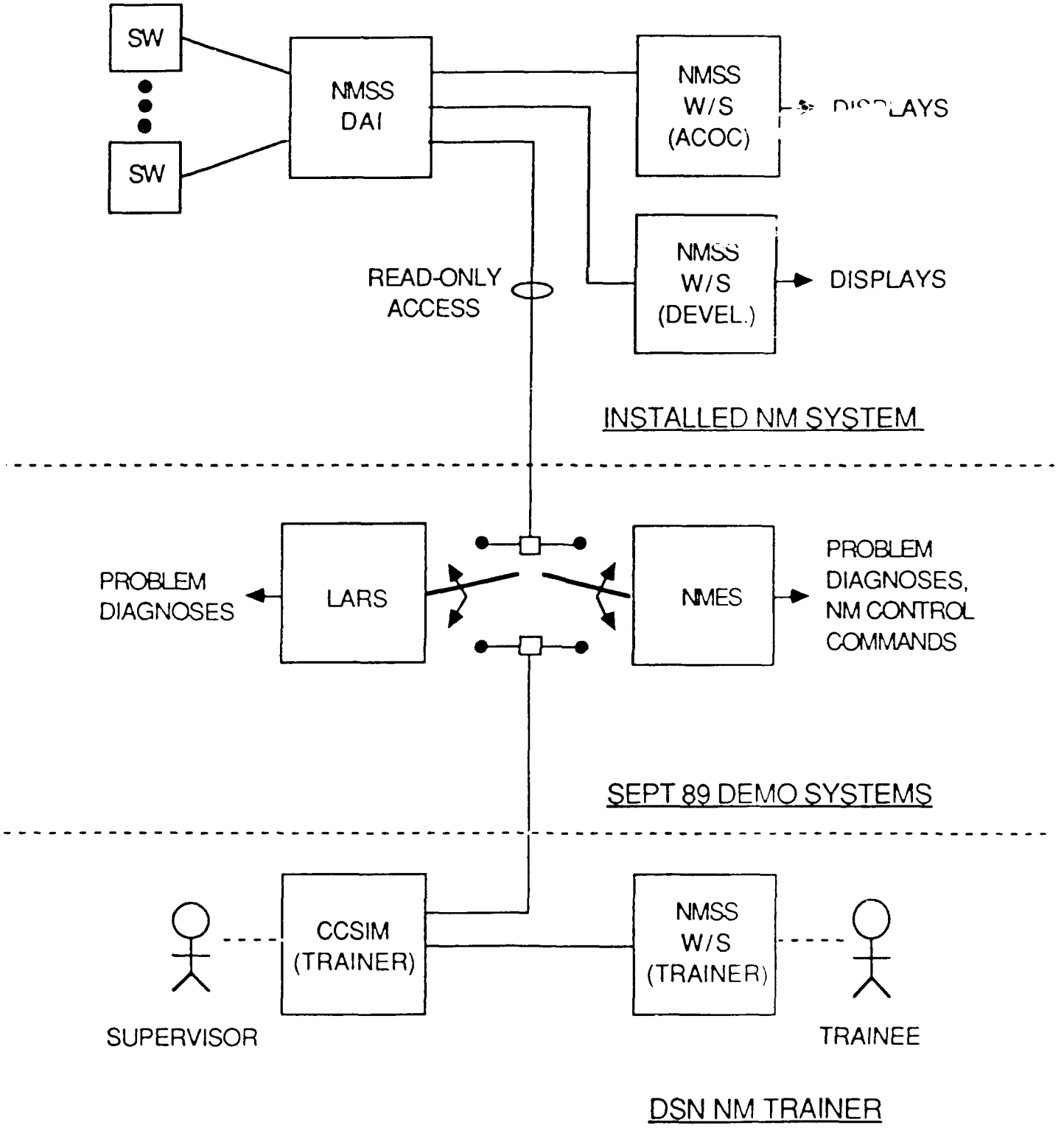


Figure A-1
NMES/LARS Demo (Simulated Data)

4. Recognize return of failed switches or trunks to service; and

5. Recommend removal of controls as appropriate when these facilities are back in service.

Work is progressing in NM knowledge acquisition during FY89, and we expect that recognition of additional problem categories in the real European DSN may be possible during September 1989. Candidates include switch congestion, noisy trunks, and overloaded trunks. Such additions would enrich the demo and every effort will be made to do so, but no additions can be promised at this time. Knowledge in these categories tends to be more complex and subtle than in the switch and trunk outage cases, and reliable extrapolation of our CCSIM experiences to the real DSN will require extended access to examples of recorded real switch statistics reports associated with such problems. Such access may be obtained in time to augment the September demo, but it is not certain.

The following subparagraphs define the expected ground rules of the demonstration of NMES with the real DSN switches. Step-by-step details of the scenarios are not relevant for present planning purposes, but will be provided in the July time frame.

A.2.1.1 Live Demonstration Test Plan

The test plan for live demonstrations of NMES can be summarized very briefly:

i. Connect NMES to the DAI as shown in Fig. A-3;

ii. Observe all NMES actions with respect to switch and trunk outages, as described below; and

iii. Review independent evidence of such outages (switch site reports, ACOC operator reports) to ascertain whether NMES correctly identified and responded to them.

There will be two classes of NMES connection time on the live data from the DAI: 1) routine connection during all available time when NMES is not being used for something else, and 2) scheduled demonstrations for witnesses during periods to be allocated by the DCA in September 1989.

Whenever NMES is in operation it keeps a running log of all actions it takes. In particular, if left connected day and night to the DAI it will make a record of each switch and trunk outage it discovers, as well as each control action recommendation it makes. This record can be examined and compared with records of actual events observed at the switches and the ACOC. If this

identifies instances in which NMES is known to have correctly recognized outages, the result can be exhibited to demo witnesses as an adjunct to scheduled live operation on DAI data (during which, as remarked earlier, it is likely that no events of note will be happening).

A.2.1.2 Switch Outage Event Responses

Each time NMES recognizes that a switch outage has occurred, it will announce that fact. It will then recommend application of CB (Code Block) controls at all European DSN switches for 100% of the calls destined for the failed switch. NMES will also recommend application of the SKIP and REROUTE controls, to provide alternate treatment for all calls that would normally have been routed through the failed switch on the way to some other destination. In the present Engineered Routing treatment in the European DSN the SKIP and REROUTE controls cannot actually be used effectively, but they will be important with the more sophisticated routing that the DCA expects to implement in the future.

A.2.1.3 Switch Restoral Event Responses

Each time NMES recognizes the return to service of a switch previously declared to be out of service, it will announce that fact. It will also recommend removal of all CB, SKIP and REROUTE controls that were suggested at the time of discovery of the outage.

A.2.1.4 Trunk Outage Event Responses

Each time NMES recognizes an outage of a trunk group connected to one of the live DMS-100 switches it is monitoring, it will announce that fact. It will also recommend the application of SKIP and REROUTE controls to provide alternate treatment for all calls that would otherwise have attempted to use the failed trunk group.

A.2.1.5 Trunk Restoral Event Responses

Each time NMES recognizes the return to service of a trunk group previously declared to be out of service, it will announce that fact. It will also recommend the removal of all SKIP and REROUTE controls that were suggested at the time of discovery of the outage.

A.2.2 Simulated Network Demonstration Goals for NMES

In order to provide variety and interest, it is planned that CCSIM will be selected as the input to NMES during scheduled periods in the September demonstrations, as determined by the DCA. The configuration will be as shown in Fig. A-5. All of

Section A.2.1 could be repeated here, but the extra verbiage would serve no purpose; the essence of the test plan is that Lincoln will generate a controlled sequence of representative switch and trunk outage events in CCSIM runs simulating the current European DSN. NMES will recognize problems and recommend control actions as in Section A.2.1, except that the controls will be applied directly to CCSIM by NMES, and the demo witnesses will be able to see them take effect in the subsequent performance of the simulated DSN.

In addition to the switch and trunk outages, it is planned that other interesting network problem events will be induced in CCSIM, and that NMES will recognize and respond to them. The knowledge engineering work to support these additions is still in progress, therefore at the present time it is not possible to state details of what will be demonstrated. These details will be finalized in July, at the time of delivery of the demo hardware systems to DCA-Europe.

To reiterate, it is expected that the September demo will include the set of live network functions agreed upon with DCEC/R610 in late FY88, as detailed in Section A.2.1 above. The demo will also include a simulated-network version of Section A.2.1, based upon CCSIM, in which events of interest can be induced at will for the convenience of the demo witnesses. It is considered likely that more functions will be included in the simulated-network demonstrations; no commitment has been made in this area, but every effort is being made to add value to the demo in this way.

A.3 Installation and Checkout Plan

In order to achieve the September 1989 demonstration capabilities diagrammed in Fig. A-1, GTE and Lincoln need to deliver and install equipment at DCA-Eur beginning in July. This will allow time for thorough integration and checkout of interfaces with the existing NM system at DCA-Eur, and to resolve any problems that are discovered in the course of this process. The paragraphs to follow refer explicitly to only the Lincoln Laboratory actions in this regard, and it is expected that GTE will provide separate documentation detailing their plans relative to the work they will do. It is intended by Lincoln Laboratory, GTE and DCEC that these two sets of plans will be fully coordinated with each other and with DCA-Eur in advance, and that Lincoln Laboratory and GTE will cooperate in expeditiously carrying out the planned activity.

A.3.1. Site Support Needs Prior to Equipment Arrival

Appendix B below contains descriptions of the operating space, prime power and data interconnection needs of the NMES and Trainer equipment. These are very straightforward requirements,

since the equipment consists of off-the-shelf computer hardware, viz., two Sun 3/260 workstations and one IBM PS/2 Model 80. It is anticipated that these needs will be coordinated with DCEC and DCA-Europe, and that the needs can be accommodated satisfactorily prior to arrival of the equipment and the installation team on site, i.e., by 1 July 1989.

A.3.2. Shipping Plans

The equipment will be shipped by Lincoln Laboratory so as to arrive at DCA-Europe during the first week of July 1989. It is anticipated that the site personnel will be able to arrange for delivery of the equipment (still in its shipping containers) to the workspace where prime power and data cables have been made available, and where it will be operated while at DCA-Europe. The two Sun workstations and their monitors will be shipped in four small, strong, reusable containers which the Lincoln installers will send back home; and the IBM PS/2 Model 80 will be shipped in its factory cartons.

A.3.3. Installation/Checkout Team Composition

A team of four Lincoln Laboratory personnel will arrive at DCA-Europe within five working days of the equipment arrival, i.e., during the second week of July. Three of them will be staff members who have had major roles in the development of the NMES and Trainer systems, and will return to Lexington after installation and checkout are complete. Approval will be requested from DCA-Eur for the fourth Lincoln individual to remain at DCA-Eur until completion of the September demos, as noted in Section A.3.5 below.

A.3.4. Installation Procedures

The Lincoln team will unpack the NMES and Trainer equipment, set it up, load the software, and verify that all operation is normal. A baseline requirement is that the equipment operate correctly in isolation, that is, that the NMES and CCSIM operate correctly together exactly as they did before shipment, and similarly that the CCSIM and the Trainer NMSS operate correctly together. "Correct operation" is defined for this purpose as successful execution of a set of test exercises that will be devised and validated at Lexington prior to shipment of the equipment, to the satisfaction of the developers of the systems. (Descriptions of these test exercises will be provided to DCA-Eur site personnel if they so desire.) If any problems are disclosed in this process, such as malfunctions caused by the stresses of shipment, the Lincoln team will perform or secure repair services as necessary.

After correct operation has been demonstrated in isolation, a cable will be connected between the DAI and NMES as indicated

in Fig. A-3. Note that this will be a "read-only" connection at the DAI end, that is, the NMES will be so configured that it receives data from the DAI but does not send any signals back. Operation in this mode will be carefully monitored for an extended period, and NMES responses will be evaluated against evidence obtained by observing an NMSS workstation, until confidence is established that the DAI/NMES interface is correctly transferring DAI outputs to NMES. Throughout this procedure, whenever a switch or trunk outage chances to occur within the purview of the DMS-100 switches reporting to the DAI, the responses of NMES will be evaluated to confirm that it is correctly diagnosing each outage.

It is estimated that the installation and checkout process will require two weeks. The schedule has a built-in margin (namely the month of August) to allow for the resolution of any unexpected time-consuming problem and still meet the September demo schedule. Throughout their time on site the Lincoln personnel will conduct their activities in such a way as to absolutely not interfere with ACOC operations, and to interact with site personnel only to an extent and in a manner that will be agreed upon in advance among DCA-Eur, DCEC, Lincoln Laboratory and GTE.

A.3.5. Post-Installation Activities

At the end of the installation/checkout activity, three of the Lincoln staff will return to Lexington. The fourth person will remain on site (given that approval is obtained from DCA-Eur), and will actively work to insure that the NMES and Trainer equipment is frequently exercised, remains in working condition, is properly looked after in the event of any moves or changes that DCA-Eur finds necessary, and in general is kept in condition to reliably support the September demos. In the course of this work the Lincoln person will keep careful records of any puzzling or erroneous behavior he observes in the NMES or Trainer, and will communicate these facts to Lexington. From time to time the Lexington people may issue software updates to correct such observed problems, or to serve other necessary purposes, and the on-site Lincoln person will install and test these updates and report the results to Lexington. In short, this person will act as the on-site eyes and hands of the Lexington staff. In addition, this person will be continually available to answer questions, provide impromptu demonstrations for interested site personnel, and serve as liaison between the site and Lincoln.

A.4 Physical Requirements

A.4.1. Floor Space

As noted above, the Lincoln demo equipment consists of two Sun 3/260 workstations and an IBM PS/2 Model 80 computer.

Each Sun 3/260 workstation consists of a CPU/Disk Cabinet, plus a user terminal consisting of a color graphics CRT monitor, a keyboard and a mouse. The CPU/Disk cabinet rests on casters on the floor, and occupies a space 18 inches wide by 30 inches deep by 28 inches high. The terminal requires a table or desktop space approximately 30 inches square. While it is possible for the Sun terminal to be separated from its CPU by up to 50 cable feet, for initial purposes at DCA-Eur it is recommended that they be located together for ease in loading of tapes and the like. Assuming that this is done, and allowing room for the operator and his chair, each of the two Sun 3/260 workstations will require a space about 4 feet wide by 5 feet deep.

Lincoln does not yet have the IBM PS/2 Model 80, since it is a new modification to NMSS and the first units have just arrived at DCA -Eur. Lincoln's unit has just been ordered in accordance with precise specifications obtained from DCA-Eur. Meanwhile, the size (and power) requirements of the IBM PS/2 Model 80 can be ascertained at DCA-Eur by querying the staff who are using the new equipment. For the moment we will make a rough estimate that the IBM PS/2 and its operator will need the same space as a Sun, namely 4 by 5 feet.

The net space requirement for the Lincoln NMES and Trainer equipment is approximately 60 square feet, arranged in three 4 by 5 foot pieces. A rough allowance for sitting or standing room for several demo attendees would be another 60 square feet. For example, one could visualize a rectangular space 12 feet wide by 10 feet deep, with the equipment lined up along the longer dimension and the demo witnesses looking over the shoulders of the equipment operators. Alternative layouts could be tailored to suit the physical constraints of the location designated by DCA-Eur for the demonstrations.

For initial planning purposes, subject to modification as necessary by GTE, we could postulate a LARS equipment and operator space requirement similar to that of NMES, viz., 4 by 5 feet. Thus the total space requirement for all the demo and training equipment in the lower two-thirds of Fig. A-1 is four 4 by 5 foot pieces, plus 60 square feet for the demo attendees, or 140 square feet, arranged to fit the designated location.

A.4.2. Power and Data Cable Requirements

Each Sun CPU/Disk cabinet requires prime power input of 5.0 amperes at 115 volts, or 2.5A at 220V, with the voltage selectable by positioning a strap in the power supply. These figures have been verified by actual measurements with a clip-on ammeter at Lincoln Laboratory. The power supplies in the Suns will accept any input frequency between 44 and 66 Hz. The Sun user terminal requires 1.6A at 115V, or 0.8A at 220V, with the voltage selectable by a switch on the back of the monitor. The total power of about 800 watts determines the air conditioning requirement for each Sun workstation.

The power requirements for the IBM PS/2 Model 80 are not known at Lincoln at the moment, but are known to the DCA-Eur users of NMSS as noted above. For planning purposes, we recommend sufficient raw power service to reliably provide 13.2A/115V or 6.6A/220V, plus the needs of an IBM PS/2 Model 80, plus a comfortable margin of 50 percent or so. The total heat load will be about 1600 watts plus the IBM PS/2 power dissipation.

Lincoln Laboratory will supply the data cables and splitter boxes needed for interconnecting the CCSIM, Trainer NMSS, and NMES with each other as shown in Fig. A-1. The one data cable needed between the demo equipment and the installed site facilities is the RS232 cable from the DAI. This cable is functionally identical to the one already installed at DCA-Eur between the DAI and the software development NMSS workstation in the GTE support engineers' area, and should be constructed identically; the only difference is its length, which depends upon the location designated by DCA-Eur for the demo equipment. It is anticipated that arrangements can be made with DCA-Eur to install a suitable cable from the DAI (in the ACOC) to the demo equipment space. If availability of appropriate multi-conductor cable or RS232 connectors at DCA-Eur is limited, Lincoln can ship the required materials; if line drivers are required because of the cable length, Lincoln can ship them with the equipment.

A.5 Summary of Site Support Needs

The purpose of this Appendix is to summarize in one place the anticipated needs for logistic and personnel support by DCA-Eur to enable the successful installation and test of the NMES and the NM Operator Trainer. Note that this Appendix refers only to the Lincoln Laboratory demo equipment; the GTE LARS needs will be provided separately by GTE to DCEC, and thence to DCA-Eur. This Appendix is organized below by time period, i.e., Pre-Delivery (now to 5 July); Installation (5-20 July); Post-Installation (20 July - 5 September); Demo Period (5-20 September); and Post-Demo (20 September on). Note that these dates are approximate, and purely for descriptive purposes; the

actual schedules are expected to be negotiated between DCA-Eur and DCEC.

A.5.1. Logistic Support Needs

i) Pre-Delivery (now to 5 July)

Two kinds of logistic support are needed in this period, with completion by 5 July: designation and preparation of the demo equipment location, and provision of a suitable RS232 cable from the installed DAI in the ACOC to the demo location. This includes provision of clear floor space as described in Section A.4.1; three table tops or desk tops, each at least 30 inches square, to support the three monitors and keyboards; chairs for the three operators; prime power and air conditioning capability as listed in Section A.4.2; and a data cable as described in Section A.4.2. Note that Lincoln Laboratory will supply the materials for the cable, should they be in short supply at DCA-Eur; but we need the help of the site in actually running the cable.

ii) Installation (5-20 July)

The four-person Lincoln installation team will bring along the necessary tools and equipment. It would be a great convenience for them to have the use of a single desk and chair in a designated DCA-Eur office area as a base of operations; other than that, their desk and seating needs will be satisfied in the space designated for the demo equipment, as described in Section A.4.1.

iii) Post-installation (20 July - 5 September)

The Lincoln person to remain at the site after installation (as described in Section A.3.5 above) can continue to use the single desk and chair mentioned in the preceding paragraph as his base of operations.

iv) Demo Period (5-20 September approx.)

Three categories of visitors will be present at DCA-Eur during this period, needing amenities such as badges, escorts, seating space, etc.: 1) a Lincoln demo team, back up to 4 people total; 2) DCEC participants, as determined between DCEC and DCA-Eur; and 3) visitors to see the demos, including DCA personnel of various ranks, as determined between DCEC and DCA-Eur. (In addition there will be the GTE LARS demo team, whose needs will be addressed separately between GTE, DCEC and DCA-Eur.) The logistics needs of the Lincoln team will be the same as in the Installation period discussed above.

v) Post-Demo (20 September on)

As described in Appendix D below, there are great potential advantages in maintaining an on-site engineer for an extended period after the September demonstrations. From a logistics standpoint, this engineer would need the same desk and chair described in paragraphs ii) - iv) above.

A.5.2. Personnel Support Needs

i) Pre-Delivery (now to 5 July)

The primary needs in this period are for technical information exchange required for successful design/development of the demo systems. There was one visit to DCA-Eur by Lincoln personnel in September 1988, one in November 1988, the current one (March 1989), and presumably another in June 1989. In each case the net personnel time cost to DCA-Eur is on the order of 20-40 man-hours, distributed among a number of DCA and GTE support personnel, in which the site personnel describe software and hardware interfaces and functionality for the Lincoln visitors and participate in planning for the installations and demonstrations.

ii) Installation (5-20 July)

In this period the Lincoln installation team will be making sporadic requests to site personnel (DCA as well as GTE support people) for assistance in resolving problems that arise during checkout -- unexpected signals, anomalous behavior, suspected bugs, etc. The estimated total time cost distributed across the site personnel is 10-20 man-hours.

iii) Post-Installation (20 July - 5 September)

Here it is expected that the Lincoln on-site person will not be costing DCA-Eur any support man-hours; rather, he will be answering questions of interest to the site, providing training and impromptu demos, and generally providing useful services.

iv) Demo Period (5-20 September)

A reasonable estimate is one person, full time, throughout this period (not serving Lincoln Laboratory needs specifically, but helping to look after the visitors and support the demos). Some higher-ranking visitors may attend, and may create extra demands upon DCA-Eur management.

v) Post-Demo (20 September on)

Again, it is expected that the on-site person will make a net positive contribution to operations at DCA-Eur, rather than costing any support man-hours.

A.6 Post-Demo Recommendations

It is anticipated that the NMES and Trainer equipment will remain at DCA-Eur upon completion of the September demos, and that there will be a desire on the part of DCA management to exploit them as they evolve through continuing development by Lincoln Laboratory under DCEC sponsorship. Since these systems do not fit within the operational requirements to which present-day ACOC staff are trained, however, there is a distinct possibility that this equipment could remain unused and could even be detrimental to ACOC operation unless provisions are made for continuing support on site. This support should be provided by a capable engineer, and should include:

1. Training in DSN network management skills;

2. Assistance in developing lesson plans and training procedures for use with the DSN Operator Trainer system;

3. Training in operation of NMES;

4. Installation and test of successive NMES and CCSIM software versions;

5. Observation and reporting of problems with either system; and

6. In general, support for DCA-Eur on any matters relating to NMES and the Trainer.

It should be noted that very similar considerations apply to the GTE LARS system, and that the opportunity exists for the same individual to support both the Lincoln and GTE systems.

A.7 NMES Test and Evaluation Plan

A.7.1. Purpose

The purpose of this Appendix is to lay out a Test and Evaluation Plan which begins with the validation activity of the September 1989 proof-of-concept demonstrations at DCA-Europe, and extends through the point at which NMES can be accepted as an operational tool in the ACOC environment. The scope of the main body of this document is primarily the September field demonstrations. The NMES features and capabilities to be demonstrated at that time will be progressively extended and augmented in the ensuing months, as the development of NMES continues.

There is a challenging set of requirements in validating the performance of NMES in the September demonstrations, as noted in Section A.2 above, and these requirements will become steadily more complex as the sophistication of NMES grows.

A.7.2. Key Functions and Features of the NMES

As illustrated in Fig. A-3 and discussed in Section A.1.3, the role of the Network Management Expert System is to assist the DSN Network Management staff in the ACOC in detecting and remedying problems in the network. The specific technical features of the NMES are as follows, and a major requirement of the validation activity is to verify that these features are correctly implemented.

1. Provide for convenient initial loading of descriptions of the topology, connectivity and expected traffic matrices of the network NMES is to observe.

2. Present an interface that is physically, electrically and functionally identical to that of the real NMSS workstation in the ACOC.

3. Interact with the DAI via that interface, exactly as the NMSS interacts with the DAI, to request and accept switch reports. For the present, NM control commands recommended by NMES may be input manually at the switch sites in response to verbal requests from the ACOC. In the future, when NMES recommends a control command the operator will think about it and (if he chooses to do so) authorize direct transmission of the command from NMES to the switches via the DAI.

4. Recognize each instance of a pattern in the switch reports that indicates the existence of one of the network problem types in the current NMES knowledge base. (As explained in Section A.2.1, the minimum capability in the September 1989 demonstration system knowledge base will be recognition of switch and trunk outages. Successive NMES software versions will add new capabilities.)

5. Notify the ACOC NM operator of the existence of each problem condition.

6. Recommend the appropriate NM control commands and parameters to remedy the problem condition.

7. Recognize the cessation of any problem condition previously reported, and notify the operator.

8. Recommend removal of controls as appropriate in view of the problem cessation.

9. Provide a color graphics display for the convenience of the operator in rapidly reviewing the current status of all NMES activity.

10. Maintain a continuous log of all NMES transactions.

11. Provide for interaction with the CCSIM in the DCOSS Operator Trainer as an alternative to live operation with the DAI (see Appendix F for a discussion of the design through which the CCSIM interface is indistinguishable from that of the DAI).

A.7.3. Validation of NMES Performance

NMES is an Expert System, and one of the normal procedures for validating the performance of an Expert System is to comprehensively demonstrate it to the expert from whom the knowledge was elicited to embed in the system. In the present case this is impossible, however, since the practice of Network Management for the DSN is in its infancy; the network is not even fully implemented yet.

A substantial part of the knowledge base of NMES was obtained through experimentation with the Call-by-Call Simulator which forms the heart of the DCOSS Operator Trainer discussed in Appendix F. The validation of NMES is therefore intimately associated with the Trainer, in that it should be demonstrable that NMES can correctly recognize every permitted variation of problem conditions set up by the CCSIM operator (or Test Director, as illustrated in Fig. A-5 and discussed in Section A.1.5 of this document). This is a straightforward if very lengthy process, and (like the CCSIM validation tests discussed in Appendix F) the process will be addressed through selecting and demonstrating a set of representative examples that span the range of cases of interest, since it will be impossible to do every single case.

Much of this performance validation activity will be done at Lincoln Laboratory prior to delivery of NMES versions to DCA-Eur. However, here is an important case where the skill and experience of ACOC personnel can be very helpful. Even without pre-existing expert knowledge in DSN network management, these individuals will be able to critically watch NMES performance under a variety of simulated problem scenarios and look for inconsistencies and unexplained behavior. In fact, whenever feasible it will be a good idea to have NMES running in the background on the Trainer CCSIM outputs during supervised operator training sessions, so that the supervisor can observe whether NMES reacts to problem conditions in the same way that he thinks the trainee should react.

Finally, operation at the ACOC is the ideal opportunity for long-term checkout of NMES performance on live DSN data. As noted in Section A.2.1.1, NMES should be left connected to the live DAI data at all times when it is not being used for some other purpose. The built-in logging feature of NMES will keep track of every network anomaly it identifies and control action it recommends, and these can be compared with the opinions and actions of the on-duty ACOC personnel. These personnel (while they may not have as much experience in DSN network management as they would like) are the best available reservoir of intuition and experience on how the network should be reacting to problems and events.

A.8 DCOSS Operator Trainer Test and Evaluation Plan

A.8.1. Purpose

The purpose of this Appendix is to lay out a Test and Evaluation Plan which begins with the validation activity of the September proof-of-concept demonstrations and extends through the point at which the Trainer is accepted by the ACOC staff community as a useful operational tool. As delivered and installed at DCA-Europe in late 1989, the Trainer will have a set of capabilities that are potentially applicable immediately for training NMSS operators to deal with certain classes of DSN network management problems. These capabilities will be progressively augmented and extended as the ongoing development of the Call-by-Call Simulator continues at Lincoln Laboratory, and successive software versions are sent to DCA-Europe. The question is whether in fact the Trainer accurately represents real situations in the real world at each stage in this development. This Appendix discusses the nature and meaning of this question and describes the techniques for answering it.

A.8.2. Key Functions and Features of the Trainer

As illustrated in Fig. A-4 and discussed in Section A.1.4, the role of the Trainer is to provide a completely lifelike time-varying simulated DSN for the trainee to monitor with an actual NMSS workstation and to control with actual DSN network management commands. The training supervisor's control console allows him to set up any desired damage or overload situations, and he can guide, observe and evaluate the trainee's performance in assessing the problems and applying remedies. Typically the Trainer may run faster than real time (depending on the size and complexity of the network being simulated), which will make for efficient training exercises.

The specific technical features of the Call-by-Call Simulator (CCSIM) as implemented in the Trainer are as follows:

1. Provide for initialization with any desired network topology, connectivity and traffic matrices, within size ranges that comfortably encompass the present and future DSN in a theater.

2. Randomly generate every call in the network, in accordance with arrival rate and call duration distributions matching the DCA's models of the real world, having averages that agree with the input traffic matrices.

3. Perform the route selection processing for each call exactly as it would be done by the corresponding real-world DSN switches along the path from source to destination. Routing procedures may be specified at initialization time to match the present DSN implementation or a selection of more sophisticated versions planned for the future DSN.

4. Take note of all blocking and preemption events occurring among calls competing for network resources.

5. Accumulate statistics for each simulated switch in the network, patterned precisely after the data gathered by real DSN switches.

6. Present an interface to the trainee's NMSS workstation that is physically, electrically and functionally identical to the interface of the real DAI in the ACOC.

7. Provide a stream of periodic switch reports over that interface which are identical in format and meaning with the real report stream provided by the DAI to the NMSS in the ACOC.

8. Provide convenient means for the training supervisor to alter the behavior of the simulated network at any time during a run by

   i. increasing or decreasing any component of traffic;

   ii. damaging or restoring the call handling capability of any switch;

   iii. removing or restoring statistics reporting connectivity to any switch;

   iv. damaging or restoring trunks on any link in the network; or

   v. temporarily freezing simulated time to provide leisure for questions or discussion.

9. Accept DSN network management control commands from the trainee via the NMSS, and put the commands into effect so that they begin to be reflected in subsequent switch reports to the NMSS.

10. Provide a color graphics display for the supervisor which gives him comprehensive knowledge of what is happening to user traffic throughout the network, and what effects the applied controls are really causing.

11. Provide detailed statistics summaries for later analysis by the supervisor.

12. Maintain a continuous log of all supervisor and trainee actions for later review.

A.8.3. Validation of Provable Aspects of the Trainer

The correctness of many aspects of the Trainer can be verified by study and analysis. These include formats of switch reports, as a simple example, and more complex questions such as the exact implementations of routing and preemption procedures and of NM controls within CCSIM.

The development of CCSIM has been marked by a continual succession of exercises to define features and functions that needed to be implemented, in which Lincoln personnel searched reports and literature, consulted with knowledgeable authorities in the DCA and industry, and in a few cases finally made reasonable assumptions to fill in details that were not obtainable from any of these sources. Each of these provable aspects of CCSIM is subjected to exhaustive testing, typically by scrutinizing the code and convincing oneself that the implementation matches the definition.

A strong contributor to validation of provable aspects of CCSIM is DCEC, where a copy of CCSIM has been in place for over a year and is used as a system engineering tool by DCEC personnel. Their applications subject CCSIM to a whole different set of stresses from those examined at Lincoln Laboratory, and in a number of instances this has led to realization that some aspect of a function definition should be changed, or that some feature should be added or removed.

Despite all of this care, it is likely that discrepancies remain which will be noticed by skilled ACOC personnel as the Trainer is used. Any such discoveries will be welcome, and the next delivered version of CCSIM software will correct the problem.

A.8.4. Validation of Simulation Aspects of the Trainer

Computer simulations of physical processes are inherently approximate, and they are all subject to question as to whether their approximations are accurate enough for the simulations to be valid. It is easy to make this statement, and if you are knowledgeable about the process being simulated it is also easy to visualize the meaning of "accurate enough" and "valid". It is exceedingly difficult, however, to create either practical definitions of these terms or constructive procedures for measuring them. Books have been written on the subject, and much could be said about it; but it boils down to exercising the simulation over a wide range of conditions, evaluating the results against all available standards and criteria, accumulating many examples of operation that appears to be correct, and observing that none of the test cases disclosed incorrect operation. Thus validating a simulation is much like establishing a reputation as an honest person: one cannot prove honesty in advance, but must demonstrate it over time by producing numerous observed examples of honest behavior and no counterexamples.

Four types of simulator validation activity are being pursued with CCSIM, of which the last two are particularly applicable at DCA-Europe:

1. Perform as much qualitative and quantitative testing as we can afford;

2. Calibrate it against known and trusted alternative tools;

3. Compare it with all available examples of real-world behavior; and

4. Expose it to the scrutiny of the best available experts on the behavior of the DSN.

A.8.4.1 CCSIM qualitative and quantitative testing

The first type of activity is pursued strongly at Lincoln Laboratory as part of the implementation of each new feature. Exhaustive testing of every case and variation is generally impossible, because it can take several hours' simulation to get each point on each family of curves one would like to check; consequently judgement and intuition are applied in choosing a set of representative experiments spanning the range of interesting cases. The results tend to show statistical variation because of the random nature of CCSIM traffic generation, but care is taken to show that the variation is within expectations.

Qualitative and quantitative testing of CCSIM operation is also performed by DCEC personnel as an adjunct to their system engineering activity. On a number of occasions they have

identified trends or phenomena that did not look right, triggering investigations and some code changes. Work is in progress to define a joint testing methodology between DCEC and Lincoln to improve the efficiency of CCSIM testing.

A.8.4.2 Calibration against known tools

The second type of simulator validation was addressed several months ago, when CCSIM was calibrated against a software system called "Katz" which has been in use for years at DCEC/R700. Katz implements numerical solutions of well-known steady-state telephone network grade-of-service equations, and is used by R700 for engineering AUTOVON and now the DSN. Network performance figures were obtained with Katz for a particular representative set of DSN network and traffic characteristics, and the identical set of inputs was applied to CCSIM. Much care had to be taken to correctly reconcile the statistically-varying outputs of CCSIM with the smoothed outputs of Katz; however, excellent agreement was achieved.

A.8.4.3 Real-world data comparisons.

It must be noted that the network grade-of-service calibrations with Katz have relatively little to do with simulation by the Trainer of the DSN switch statistics reports that are looked at by the NMSS. For this purpose we need the third type of simulator validation mentioned above, namely comparisons between the Trainer and real-world behavior. To this end DCA-Europe has made available a number of NMSS log files illustrative of various network problems, and work is in progress to insure that the Trainer outputs look similar to these log files when the corresponding problems are induced in the Trainer.

Another aspect of comparison between the Trainer and the real world can be accumulated over time while the Trainer is in use at DCA-Europe. Every time an event of note in the real DSN is observed (such as switch or trunk outages now, and more complex events in the future), steps can be taken to reproduce the event as nearly as possible in the Trainer. The results can then be compared with the NMSS log files recorded during the actual event. It must be noted that it will never be possible for the Trainer to precisely replicate the numbers produced by the real DSN switches: the exact numbers depend upon the microscopic details of all the actual calls made in the DSN at the time, and such details are not known at the ACOC. The Trainer will be able to achieve good qualitative agreement by running its internal call generators at the approximate average call intensities of the real traffic (if these can be determined), and the numbers can be expected to agree within normal statistical fluctuations.

A.8.4.4 Exposure to the experts

In the opening paragraph of Section A.8.4 it was noted that it is easy to visualize what "accurate" and "valid" simulations are if you are knowledgeable about the process being simulated. Put another way, the experts in the problem domain will be able to evaluate the simulator by watching it in operation, provided that they watch it through enough examples and variations. To this end it will be very valuable for the Trainer to be installed and used at DCA-Europe. The best mechanism for achieving the benefits of these interactions is frequent application of the Trainer in a variety of training scenarios, with the training supervisor always skeptically watching for behavior that makes sense, and taking careful note of instances that do not match expectations.

APPENDIX B NMES MONITORS AND RULES

This Appendix contains descriptions of the monitors and a sample of the rules used in the version of NMES created just after the demonstration at DCA-Europe in September 1989. This version differed from the one used in the demonstration in that it takes into account some of the knowledge gained during the demonstrations about the affects of damage on network resources (see Section 3.2.7).

The monitors described below are C routines. The rules are written in CLIPS. Each monitor or rule is given an English language name intended to be descriptive of the function of the monitor or rule. That name is followed by the actual name used within the program in parentheses. In the case of the rules, the descriptions paraphrase the actual CLIPS forms. The descriptions omit some details that relate to CLIPS syntax and other internal programming issues in order to simplify the presentation without losing information relative to the functions performed by the rules.

B.1 NMES Monitors

Monitors are C language routines that process switch report data and/or the outputs of other monitors and store the results in a set of C language structures representing the network. Many of them also work to help maintain (in C arrays) lists of switches and trunk groups that have anomalous (non-normal) states. The items on the lists combine the switch or trunk group with an anomaly identifier. When a monitor recognizes an anomaly, it appends an item to the appropriate list. It may also remove an item previously put on the list. There can be more than one item on the list for a particular switch or trunk group, but in such a case, the anomaly will be different.

Monitors generally write messages to a scroll window on the NMES graphics interface so that the user can be made aware that a monitored situation has occurred. That activity is omitted from the descriptions that follow.

The order in which individual monitors are run is important since some use the results generated by others. In the following sections, they are described in the order in which they run in MNES.

B.1.1 Switch Monitors

The monitors in this and the following section are run as each switch report arrives.

Monitor - Note arrival of switch report
(mark_switch_report_received)

Function - Remove any not-reporting state information for the switch. The 'reporting' state is not considered an anomaly and is not kept on the switch state list.

remove any not-reporting state for switch X that might be left on the switch state list from a previous cycle

Monitor - Check for MF receiver overflow
(find_no_mf_receiver_free)
Function - Check the switch report for evidence that attempts to assign a Multi-Frequency (MF) receiver found none free (overflow occurred).

IF the RCVR OM report for switch X is available for this cycle AND the value of the MF receiver overflow peg count is greater than zero
THEN
assert for CLIPS the fact that switch X showed an instance of no-MF-receiver-free for this cycle append a no-MF-receiver-free state to the switch state list
ELSE
remove any no-MF-receiver-free state for switch X which might be left from a previous cycle

Monitor - Check for CCB seizure overflow
(find_ccb_seizure_overflows)
Function - Check the switch report for evidence that attempts to seize a Call Condense Block (CCB) found none free (overflow occurred).

IF the CP OM report for switch X is available for this cycle AND the value of the CCB overflow peg count is greater than zero
THEN
assert for CLIPS the fact that switch X showed an instance of ccb-seizure-overflow for this cycle append a ccb-seizure-overflow state to the switch state list
ELSE
remove any ccb-seizure-overflow state for switch X which might be left from a previous cycle

B.1.2 Trunk Group Monitors

A normal switch report contains OM reports for each of the trunk groups that connect the switch to its neighbors. Each trunk group (tg) report is identified by a group number. Communication problems during switch polling may cause the loss of any or all of the tg reports associated with the poll. The following monitors are run as the switch report is being

processed. They are run for all the tg reports that actually arrive. An additional monitor is run at the end to identify trunk groups for which no report was received.

Monitor - Check for low/decreasing capacity
(find_clli_trunks_down)
Function - Check tg report for evidence that the number of trunks in service is less than the equipped value, and/or is less than the number reported previously.

IF the reported number of trunks in service is less than the reported equipped number
THEN
append a decreased_capacity state for this tg to the tg state list
ELSE
remove any decreased_capacity state for this tg which might be left from a previous cycle
IF the reported number of trunks in service is less than the number reported previously
THEN
assert for CLIPS the fact that a capacity change has occurred on this tg during this cycle

Monitor - Check for 100% overflows with zero usage
(find_hundred_overflow)
Function - Check tg report for the case where the overflow peg count equals the outgoing attempts peg count and the usage value is zero. This situation corresponds to one type of trunk failure.

IF there are outgoing attempts on the trunk group AND the overflows equal the outgoing attempts AND the trunk usage is zero
THEN
assert for CLIPS the fact that a hundred-percent-overflow -zero-usage condition exists on this tg at this report time append an overflows-100-usage-zero state for this tg to the tg state list
ELSE
remove any overflows-100-usage-zero state for this tg which might be left from a previous cycle

Monitor - Check for attempts with zero overflows and zero usage
(find_zero_overflow)
Function - Check tg report for the case where there are outgoing attempts, no overflows, and no usage. This situation can occur if 100% SKIP controls are put on at both ends of the trunk group.

IF there are outgoing attempts on the trunk group AND the overflow peg count is zero AND the trunk usage is zero

```
   THEN
       assert for CLIPS the fact that a
       zero-percent-overflow-zero
       -usage condition exists on this tg at this report time
       append an overflows-zero-usage-zero state for this tg to
       the tg state list
   ELSE
       remove any overflows-zero-usage-zero state for this tg
        which might be left from a previous cycle

Monitor - Check for high Incoming Attempts per Circuit per Hour
 (ICCH)  (find_high_icch)
   Function - Check for cases where the incoming attempts per
   circuit per hour exceed a fixed value (currently 20).  ICCH is
   calculated by dividing the number of incoming attempts by the
   number of trunks in service and then multiplying by the number
   of switch report periods per hour.

   IF there are any trunks in service in this tg
   AND ICCH when calculated exceeds 20
   THEN
       append a high-icch state for this tg to the tg state list
   ELSE
       remove any high-icch state for this tg which might
       be left from a previous cycle

Monitor - Check for no incoming attempts
 (find_no_incoming_attempts)
   Function - Check for evidence of incoming attempts on the tg.

   IF the peg count of incoming attempts equals zero
   THEN
       assert for CLIPS the fact that a no-incoming attempts
       condition exists on this tg at this report time

Monitor - Check for low holding time (find_low_ht_calls)
   Function - Check for evidence of low average holding time for
   calls on the tg.  Average holding time (HT) is calculated by
   dividing the usage by the total number of connections (sum of
   incoming attempts plus outgoing attempts less overflows).
   When the number of connections in a report period is zero,
   NMES arbitrarily sets HT to zero to avoid a problem with
   division by zero.  It would probably be better to set it to a
   large real value unless the usage is also zero.

   IF the calculated HT is less than 12 seconds
   THEN
     assert for CLIPS the fact that a low-holding-time condition
     exists on this tg at this report time
     append a low-holding-time state for this tg to the tg state
     list
   ELSE
```

remove any low-holding-time state for this tg which might be left from a previous cycle

The following monitor is run after the monitors described above have been run for all the trunk groups reported in a switch report.

Monitor - Find trunk groups with no reports (find_cllis_without_reports)
Function - By going through a list of trunk groups that exist at the reporting switch and for which reports are expected, find and mark those missing.

FOR all trunk groups in the C structure representing switch X
DO
IF the time stored in the trunk group representation does not equal the time of the current switch report
THEN
assert for CLIPS the fact that a no-tg-information -received condition exists for this tg at this report time
append a no-clli-report state for this tg to the tg state list
remove all other tg states for this tg which can no longer be believed in the absence of a tg report
ELSE
remove any no-clli-report state for this tg which might be left from a previous cycle

B.1.3 Switch Outage Monitors

Switch outage monitors are run when all switch reports for an NMES cycle have been received and the data they contained has been processed by the switch and trunk group monitors described above. A routine called 'run_interval_monitors' goes through a list of all switches in the network calling each of the following monitor routines for each switch on the list.

Monitor - Find switches with no reports (find-nodes-not -responding)
Function - Identify switches from which no report was received in the current cycle. Update the relevant CLIPS facts and status lists.

IF switch X is a reporting switch AND if no report was received from switch X this cycle
THEN
assert for CLIPS the fact that switch X had no report for this cycle
append a not-responding state to the switch state list
FOR all trunk groups on the list of trunk groups at switch X

```
   DO
    assert for CLIPS the fact that a no-tg-information
    -received condition exists for the tg at this report time
    append a no-clli-report state for this tg to the tg state
    list
    remove all other tg states for the tg which can no longer
    be believed in the absence of a tg report
   ELSE
    remove any not-reporting state for switch X which might be
    left from a previous cycle
```

The following monitors make use of information obtained from reporting neighbors of a switch. The C structure representing each switch contains a list of all neighbors to the switch. The information needed for all of these monitors is calculated on a single pass through the neighbor list which is carried out in a routine called 'find_no_outgoing_attempts'. For each neighbor on the neighbor list, all trunk groups between the neighbor and the switch in question are examined. The functions for all of the following monitors are carried out within that routine, with the consequence that the individual monitors do no have routine names as do the monitors described above.

```
Monitor - Find switches from which neighbors see no outgoing
 attempts
  Function - Check for evidence of outgoing call attempts by
   switch X seen as incoming calls from switch X at its
   neighbors.

  IF one or more neighbor switches to switch X reported AND the
   total incoming attempts from switch X seen by those neighbors
   was not greater than zero
  THEN
   assert for CLIPS the fact that neighbors-see-no-incoming
   -signals from switch X at this time
   append a no-outgoing-attempts state for switch X to the switch
   state list
  ELSE
   remove any no-outgoing-attempts state for this switch which
   might be left from a previous cycle

Monitor - Neighbors see trunk groups with successful connections
   Function - Check for evidence that one or more neighbors is
    achieving successful connections to switch X.

  IF one or more neighbor switches made attempts to access trunk
     groups to switch X
  AND either usage was observed on the accessed group
  OR overflows were observed on the accessed group at less than
    100%
  THEN
    assert for CLIPS the fact that neighbors-see-cllis-with
```

-successful-connections from switch X at this time

B.2 NMES Rule Descriptions

B.2.1 Rules for Switch Outages

The following rules are intended to deal with switch outages. They make use of and change switch outage status which can have the values: unknown, active, inactive, or very inactive (down). When NMES starts running, all switches have unknown outage status. A switch of unknown status is assumed to be functioning properly and therefore not in need of attention from NMES. Its status will remain unknown until symptoms of inactivity are detected. Once having become inactive, a switch's status is tracked by NMES and the duration of its current state is maintained. Rules that change switch status to inactive are run at a higher CLIPS salience so that they will fire before other rules that might be ready to fire on the same cycle. The CLIPS fact database also has a 'down-switch' control status that remembers whether or not controls appropriate to the 'down' status have been applied at other switches.

Rule - Change switch status from active to inactive
(inact-known-node)
Function- Detects switches that were active but that are now displaying signs of being inactive.

IF switch X is marked as active at the current time
AND no report has been received from switch X
AND switch X's neighbors see no incoming calls from switch X
AND switch X's neighbors see no successful calls to switch X
AND attempts were made from the neighbors to switch X
THEN
mark switch X as inactive but preserve its control status

Rule - Change switch status from unknown to inactive
(inact-new-node)
Function - Detects switches that had unknown status but that are now displaying signs of being inactive.

IF switch X status is unknown at the current time
AND no report has been received from switch X
AND switch X's neighbors see no incoming calls from switch X
AND switch X's neighbors see no successful calls to switch X
AND attempts were made from the neighbors to switch X
THEN
mark switch X as inactive and set its control status to no-control

Rule - Report inactive switches (inact-node)
Function - Detects all inactive switches and passes information via C routines to the graphics.

```
   IF switch X is marked inactive
   THEN
       call C routines to show inactive status to user

Rule - Update switch inactivity duration  (still-inact-node)
   Function - Detects switches that have been inactive for more
    than one time period and updates duration of inactivity.

   IF switch X was inactive or very inactive last period
   AND no report has been received from switch X
   AND switch X's neighbors see no incoming calls from switch X
   AND switch X's neighbors do not see any successful calls to
        switch X
   AND attempts were made from the neighbors to switch X
   THEN
       leave switch X marked as inactive or very inactive
       add one report period to duration of inactivity

Rule - Change switch status from inactive to very inactive (down)
 (very-inact-node)
   Function - Detects switches that have been inactive for
    a long enough time to be considered very inactive (down).

   IF switch X was inactive for a specified number of report
       periods
   AND the number of currently reporting neighbors who see no
        incoming calls is greater than 1 (i.e. there is current
        evidence that the switch is inactive)
   THEN
       mark switch X as very inactive(down)
       call C routines to inform user and recommend controls

Rule - Apply controls for down switch  (apply-down-ctrl)
   Function - Detects that a switch is very inactive, that no
     down
     -switch controls have been applied, and that the user has
     confirmed the recommended application of controls.

   IF switch X is marked very inactive
   AND down-switch controls have not been applied
   AND user has confirmed control application
   THEN
       call C routines to apply the down-switch controls
       call C routines to inform user
       mark switch X as very inactive with controls for a down
       switch applied (The controls are applied at other
       switches.)

Rule - Change switch status to active  (act-node)
   Function - Detects switches that were inactive, but now
     display activity.
```

```
   IF switch X is marked inactive or very inactive(down)
   AND switch X's neighbors see incoming calls from switch X
    OR switch X is reporting and a switch report was received)
   THEN
       mark switch X as active
       call C routines to inform user

Rule - Remove down-switch controls  (remove-down-ctrl)
   Function - Detects switches that are active but for which down
    -switch controls have been applied.

   IF switch X is marked active with down-switch controls applied
   AND user has confirmed that controls should be removed
   THEN
       call C routines to remove controls
       mark switch X as active with no down-switch controls
       call C routines to inform user

Rule - Confirm that a switch is still active  (still-act-node)
   Function - Detects that a switch is still active with or
    without the receipt of a report from the switch.

   IF switch X is marked as active
   AND neighbors see incoming calls
    OR a report from switch X has arrived
   THEN
      mark switch X as active
      add one report period to duration of activity
```

APPENDIX C NMSS TRAINER DOCUMENTATION

C.0 INTRODUCTION

This Appendix reproduces the NMSS Trainer documentation that was delivered to the sponsor during the year. Section C.1 is a brief User's Manual telling an operator how to initialize and use the software. Section C.2 is a detailed NMSS Trainer Software Description which is intended to provide all the information necessary for a programmer to carry out software maintenance and upgrade of the Trainer. Section C.2 was published and distributed (to sponsor-approved government agencies only) as a Lincoln Laboratory Project Memorandum.

C.1 NMSS TRAINER USER'S MANUAL

Trainer Overview. The purpose of the Trainer is to harness the network simulation capabilities of CCSIM to drive the NMSS workstation for the purpose of training ACOC personnel. The hardware configuration consists of a Sun Workstation running the CCSIM programs and a PS/2 Model 80 running the NMSS Workstation. The two must be connected by a cable in order to communicate.

Starting the NMSS workstation. In order to train, the NMSS workstation must have the proper hardware configuration and be in working order. The hardware consists of the system unit, a text monitor, a graphics monitor, an external floppy disk drive and a printer. All pieces must be connected and have no obvious problems. To start the PS/2, remove any diskettes from the "a:" drive, turn on all peripherals and then turn on the system unit. Some numbers will flash on the screens as the computer runs self-tests. If everything is ok with the hardware then both monitors will display some text the last line containing an indication of the "C" drive. At the prompt type Trainer to start the NMSS workstation.

Running the Trainer. runsys.trainer is a script file (a type of small program) which is used to invoke the Trainer for a variety of purposes. The script MUST be invoked from a network directory. The script file starts all the necessary programs. To use it, enter one of these commands (without the quotes), where (net) is the network name:

```
"runsys.trainer (net) nmss"       send switch reports only to nmss
"runsys.trainer (net) nmes"       send switch reports only to nmss
"runsys.trainer (net) lars"       send switch reports only to lars
"runsys.trainer (net) nmss nmes" send switch reports to nmss and
                                    nmes
"runsys.trainer (net) lars nmes" send switch reports to lars and
                                    nmes
```

Of the five different invocations of runsys.trainer, the ones including "nmss" require that the NMSS workstation be running and prepared for a training session. Those invocations specifying "lars" requires that LARS be running and prepared for a training session. When "nmes" is specified, NMES will be started on the same SUN as the runsys.trainer.

The only difference between typing "lars" or "nmss" is that the switch report file names produced by CCSIM/Trainer will contain "S" instead of a "T" to indicate to LARS that the switch reports it is receiving are simulated switch reports. (LARS can be driven by a runsys.trainer invocation including either "lars" or "nmss".)

The Serial Cable: Connecting The Sun and PS/2. For a training session, both the Sun and the PS/2 must be running their respective software and they must be connected by a RS232 serial cable link containing one null modem.

The RS232 serial cable is a cable which can connect to many types of computers. When viewed from the end, each connector has either 25 "pins" or 25 holes depending from which end the cable is viewed. Each end of the cable always has the 25 little connectors but there are two types of cable links. One type is round, about 3/8 inches in diameter. The other type looks like a ribbon. Either of two types of serial cables will work.

The PS/2 RS232 serial port is located between the printer connection and the text monitor connection. The Sun has two ports, each located on the upper left back of the computer. They are labeled "Serial Port -A-" and "Serial Port -B-". Either one can be used with the Trainer. The default is the "B" port but the "A" port may be specified when invoking the Trainer.

A "null modem" in the context of this manual is a short RS232 connector. Its purpose is to "cross" some of the 25 wires inside the serial cable which otherwise would connect straight through one for one between the two computers. The Sun to PS/2 connection for the Trainer must contain exactly one null modem.

Maintaining consistency. The (net).clli file, the xref tables and the sample switch reports must be kept consistent. Support of CCSIM and the Trainer as well as NMES is concerned with consistency among several independent systems: DAI/NMSS workstation and CCSIM/Trainer/NMES. Ideally, a new installation of CCSIM/Trainer/NMES will consist not only of the respectful executables and documentation but it will also consist of consistent (net) files, xref tables and sample switch reports.

When a new (net) is defined (currently done exclusively at Lincoln Lab), input is derived from DAI switch reports and NMSS workstation xref tables. The actual DAI and NMSS workstation

quickly outdate the {net} files. Therefore, one solution to the CCSIM/Trainer/NMSS consistency problem is to provide the sample switch reports used by the Trainer and the xref tables used by the NMSS with each new installation of CCSIM/Trainer/NMES software.

Before the Trainer can be run, the {net}.clli file used by CCSIM must be consistent with the xref tables used by the NMSS workstation and both of these must be consistent with the sample switch reports contained in your {net}/sample directory. If there are inconsistencies, then the NMSS workstation will complain by sending "XREF mismatch" messages to the printer.

The following three paragraphs describe each entity in the consistency problem.

xref tables:
There is one xref table for each reporting switch in the NMSS workstation. Basically, each table contains information about all trunk groups connected to the switch.

sample switch reports:
The sample switch reports are mentioned in the previous section.

{net}.clli file:
The {net}.clli file contains trunk group definitions for CCSIM. Each line in the file contains one trunk group record. CCSIM and the NMSS workstation look at trunk groups differently. The main difference is in naming and numbering conventions; therefore, when CCSIM is run as the Trainer, it needs to be able to produce switch reports in the NMSS workstation convention. At the current time, the clli file contains 13 fields of which 5 are needed to map CCSIM's clls to NMSS' trunk groups.

The document "clli.file" in the doc directory under the installed ccsim directory (i.e. /usr/{host}/dsn/ccsim/doc, where {host} is a machine name) defines the CCSIM format for the {net}.clli file. This file also contains information needed by the aforementioned mapping. The fields needed for the mapping are clli src dst cap and grp where clli is the 16 character link name, src is the 3 character source switch name, dst is the 3 character destination switch name, cap is the 3 digit integer Capacity (number of lines in trunk group) and grp is the 3 digit trunk group number.

When producing switch reports for the NMSS workstation, a CCSIM switch report is reformatted into the format accepted by the NMSS. It is at this time when the information from the CCSIM switch report convention is changed into the NMSS workstation convention using the information contained in the {net}.clli file.

The consistency of CCSIM's (net).clli file and the NMSS workstation's xref tables as well as the sample switch reports is essential. Needless to say, all three of these sources of information must use consistent three character switch names.

Besides the switch names, their relationship is as follows:

NMSS xref table	Found on the NMSS PS/2. There is one table for each Switch, one line in the table for each trunk group which includes among other things the trunk group number and its capacity
CCSIM (net).clli file	Found in the (net) directory. There are two entries for each clli (or trunk group), each line includes among other things the trunk group number and the trunk capacity sample switch reports Found in the (net)/sample directory. There is one sample switch report for each reporting switch, for those trunk groups not simulated by CCSIM, the trunk group report is taken from the appropriate sample file containing the trunk group number and capacity

To be insured of consistency:

1. start with a working set of (net) files that were created at the same time as the sample switch reports and the xref tables
2. obtain sample switch reports whose origination was the DAI at the same time as the (net) files were created and the xref tables were last edited copy them to the (net)/sample directory
3. make sure the NMSS workstation uses xref tables consistent with both 1. and 2.

trainer_input.
This is a program to allow input to CCSIM by an interface other than the graphics. It was first used to learn about Unix socket based inter process communication and to test the Trainer code without firing up the full fledged system. It is invoked automatically by the Trainer when "input" is specified as a command line argument. (See section "Running the Trainer").

C.2 NMSS TRAINER SOFTWARE DESCRIPTION

C.2.1 Introduction: About The Software Description Document

The purpose of the Trainer is to harness the network simulation capabilities of CCSIM to drive the NMSS workstation. This involves communicating with the NMSS workstation and making CCSIM look and act like the real world.

Communicating with the workstation is achieved by adhering to the DAI/NMSS Communications protocol. The majority of this communication is the transferring of switch reports. Other communication includes error messages indicating a switch is not responding and control response messages indicating the success or failure of an NMSS control request.

CCSIM produces switch reports every five minutes of simulated time. CCSIM is made to look like the real world by accumulating simulated data for 15 minutes (three switch reports), formatting the data into the standard comma separated data format and sending switch reports to the NMSS workstation. The number of CCSIM switch reports to accumulate is a compile time parameter.

The purpose of this document is to provide programmer level documentation for the Trainer system. This document assumes a basic working knowledge of the IDSIM project. C.2.2, Trainer Function Overview, describes the four parts of the trainer function: DATA INITIALIZATION AREA, OTHER PROCESS INITIALIZATION AREA, CONNECTION TO OTHER PROCESSES AREA, MAIN CONTROL LOOP.

C.2.2 Trainer Function Overview

The entry point to the trainer executable is the main function found in trainer_main.c. The main function

- creates a log file
- checks for proper command line invocation
- calls initialize_csd
- calls trainer

One or more command line arguments constitutes proper invocation, the first of which is taken to be the network name. If trainer was called with a proper invocation, initialize_csd is called and passed the command line arguments. It returns successfully upon reading the network files. At this point, the trainer function is called.

Execution does not return to the main function because the program terminates via the "C" exit call made as a result of the trainer function.

Before reaching the main control loop, the trainer function performs several initialization tasks. The initialization tasks and the main control loop are divided into the following four parts:

DATA INITIALIZATION AREA
initialize global and local variables

OTHER PROCESS INITIALIZATION AREA
fork the commm process (for communication with the NMSS)

CONNECTION TO OTHER PROCESSES AREA
connect to other processes (those forked (such as commm) and CCSIM) (the connections are via UNIX sockets)
MAIN CONTROL LOOP
loop, reading messages from other processes, process the message, respond

The trainer function contains the main control loop for the Trainer. It's purpose is to act as a "go-between" for other processes therefore it is responsible for reading and writing from and to sockets for communication with other processes such as CCSIM and commm (which communicates with the NMSS). A good example of the trainer function's action is receiving switch reports from CCSIM, formatting them, and then sending them to the NMSS. The trainer function is found in the file trainer_funs.c and it calls functions in several other files, including csdfuns.c.

The following section describes the main control loop.

C.2.3 The Trainer Function's Main Control Loop

C.2.3.1 Overview of the Main Control Loop

The main control loop consists of 5 parts.

- update time variables used for running in real time and for querying the NMSS
- query the NMSS
- set up the mask of file descriptors for the select call
- call select
- act on the result of the select call

The trainer function terminates upon receiving a STPMSG (stop message) from some process.

The most important steps in the main control loop are making a select call, reading a message from another process and acting on the message.

The select call is used to determine when file descriptors (connections to other processes) have sent a message to the trainer. It returns either 0 or the number of file descriptors ready to be read.

When select returns 0, there are no file descriptors ready for reading. This situation is used for running in real time. The zero return value means that all programs are in an idle state. They are not sending messages to the trainer function (notably CCSIM). If it is time to pause CCSIM, do so now.

If select returns a value > 0 then other processes have sent messages to the Trainer. The message is read and a switch is done on the source of the message. Possible sources are DAI_ID, NMSS_ID, CCSIM_ID, NMES_ID, INPUT_ID, or, GRPH_ID. The messages passed between processes take the form of a structure defined in msgstruct.h. The type of a message is determined by the mtype field. Valid mtype values are defined in msgcnst.h.

The next section describes the messages received by the trainer and the actions taken.

C.2.3.2 Messages Handled by the Main Control Loop

C.2.3.2.1 Messages from the DAI

The trainer_funs.c "C" file is used in the compilation of the translate executable for translating switch reports received from the DAI before passing them on to the NMES. When the source is DAI_ID, the DAI has sent either a switch report file name or an error 11 message (which indicates that a switch is not responding to a DAI poll request). A file name is handled by reading the switch report file, Translating it into the CCSIM switch report format and sending this formatted switch report to NMES. The error 11 message is Translated into an empty switch report message type (from the CCSIM domain) which is sent to NMES. Some general date and time processing happens with each switch report file name received. The date and time is compared with the last date and time received from the DAI and when the previous and current date or time changes, a date or time message is sent to NMES.

C.2.3.2.2 Messages from the NMSS

When the source is NMSS_ID, the NMSS is responding to a Trainer query by sending a control request. Here, the Trainer makes CCSIM act like the real world by receiving the control request from the NMSS. The Trainer calls handle_NMSS_control which attempts to format the control message into the CCSIM control message format. Upon a successful control application in CCSIM the Trainer sends an affirmative response in the form of a file name to the commm process for the NMSS. This process then

calls comm_send to send the affirmative response file to the NMSS. Upon an unsuccessful control application, the Trainer proceeds in a similar manner to send an unsuccessful control response file to the NMSS.

C.2.3.2.3 Messages from the CCSIM

Overview

When a message has been read with the source CCSIM_ID, the Trainer calls handle_ccsim_msg. The CCSIM message is handled the same way that all other incoming messages are handled, with a switch on the message type. This function handles the following message types:

CNFRM	confirmation	generally an echo of messages received by CCSIM
ASK	ask	a query type message to let processes know that CCSIM has completed a "RUN" message
ESW_RPT	empty switch report	indication that there will be no switch report for this switch this period
SW_RPT	switch report	a switch report for some switch for one period
STIMEMSG	simulator time	current CCSIM simulation time

When trainer_funs.c is used in the compilation of the translate executable, CCSIM messages are sent to NMES. After a message is processed by the switch statement, handle_ccsim_msg sends the CCSIM message to the NMES. All messages are passed directly to NMES except ESW_RPT and SW_RPT. These two messages are processed by this function and the result is sent to NMES.

The CNFRM ASK ESW_RPT and STIMEMSG Messages

The handle_ccsim_msg function is called by the trainer function when it receives a message from CCSIM. When a CNFRM message is received, the Trainer prints the text portion of the message to standard output. The first CNFRM received from CCSIM is the Lincoln Lab/CCSIM version message. After this, CNFRM messages are echoes of commands received by CCSIM from other processes.

The ASK message is sent out by CCSIM when the current RUN period ends. At this point CCSIM expects another RUN command. When the ASK is received by the Trainer, a message is printed to standard out.

The first ASK is a special case. At this time, the Trainer knows that CCSIM is up and running so the Trainer sends sw-report-on commands for all switches in the (net).reporting

file. It also sends a RUN command for a long time period. It does this with the intention that someone is operating CCSIM via the graphics interface. For CCSIM to run, this person must produce RUN commands also.

The ESW_RPT message is used to keep track of a switch when a switch report is not being produced. The function process_switch_report is called specifying an empty switch report. This function returns a switch report structure if it is time (enough switch report periods have elapsed) otherwise it returns NULL. With the switch report structure returned, handle_ccsim_msg sends an ERR11 message to the NMSS and an empty switch report message to NMES.

Another last message handled from CCSIM is the STIMEMSG. This is the current CCSIM simulated time which is sent out every 10 seconds (the current CCSIM default). The value is in ticks (tenths of seconds) which this function interprets in seconds. If the Trainer is running in real time, the handle_ccsim_time function is called. This function basically adjusts time variables and sends a PAUSE to CCSIM if it is running too fast.

The SW_RPT Message: Making Comma Separated Data Format Switch Reports

This section describes how the Trainer produces comma separated switch reports from CCSIM switch reports.

The actions taken upon receiving a SW_RPT are the same as those taken upon receiving an ESW_RPT. It is the result that is different.

The SW_RPT message indicates that CCSIM has produced data for a switch for one CCSIM period. The process_switch_report function is called passing the CCSIM switch report. This function returns a comma separated data switch report structure if it is time (enough CCSIM switch report periods have elapsed) otherwise it returns NULL. With the switch report structure returned, handle_ccsim_msg prints the switch report to a switch report file. It then takes the appropriate action of sending the switch report name to the NMSS commm process. For the case of the translate executable, the switch report structure is translated back to CCSIM format and the translated switch report is sent to NMES. This concludes the high level description of what happens when the trainer receives a switch report from CCSIM.

The remainder of this section walks through the function hierarchy beginning with the process_switch_report function.

A prerequisite that should be noted is that the Trainer accumulates data from CCSIM for several CCSIM switch report

periods before sending a comma separated data switch report to the NMSS. CCSIM produces switch reports every five minutes of simulated time; thus, the Trainer must accumulate three of these CCSIM period switch reports to make one (15 minute) NMSS switch report. The number of CCSIM switch reports to accumulate before making a comma separated data switch report is defined by NUMBER_SIM_SR_TO_ACCUMULATE in "csd.h".

CCSIM uses a different notation from the NMSS to refer to switches and trunk groups. It gets all the information it needs from a set of network files. The Trainer reads the net.clli, the net.node and the net.sw files among others. It uses the information from these files to map from a CCSIM switch report to a comma separated data switch report.

Upon receiving a switch report from CCSIM, the Trainer calls the function process_switch_report. This section walks through the hierarchy of function calls rooted at this function.

This hierarchy as depicted by "cflow" is:

```
process_switch_report: struct*(), <csdfuns.c 336>
      printf: 2
      insert_switch_in_accumulation_ring: void(), <csdfuns.c
  1095>
         printf: 2
         calloc: <>
         copy_switch_report_to_accumulation_ring: void(),
<csdfuns.c 687>
         printf: 2
      accumulate_sim_switch_report: struct*(), <csdfuns.c 558>
         printf: 2
         copy_switch_report_to_accumulation_ring: 18
         formulate_csd_report_from_accumulation: struct*(),
<csdfuns.c 861>
            printf: 2
            calloc: 17
            make_csd_date_string_and_file_name: void(),
<csdfuns.c 1647>
            strcpy: <>
            get_long_time: <>
            get_struct_time_from_secs: <>
            bcopy: <>
            minus_five_minutes: int(), <csdfuns.c 1566>
               log: 8
               sprintf: <>
               log: 8
               calculate_quarter: int(), <csdfuns.c 1615>
               log: 8
               get_ccsim_switch_from_number: 3
            bcopy: 30
            log: 8
```

```
            get_ccsim_switch_from_number: 3
            get_capacity_from_clli: <>
            get_trunk_group_from_clli: <>
            free: <>
          add_switch_report_to_accumulation_ring: void(),
<csdfuns.c 749>
            printf: 2
```

process_switch_report first picks out the CCSIM switch number from the switch report structure. It then accesses the accumulation_ring variable. This is a global static pointer to a ring of structures used to accumulate CCSIM switch reports. The ring contains one accumulation structure for each unique switch number received from CCSIM. Process_switch_report searches the accumulation_ring for an accumulation structure whose switch_node_number matches that of the CCSIM switch report. If it finds such a structure then the function accumulate_sim_switch_report is called. If there is no accumulation structure with a switch_node_number matching the CCSIM switch report then the function insert_switch_in_accumulation_ring is called.

See the file csd.h for the exact contents of an accumulation structure. The most important fields of the structure are

- switch number
- start and end time of the accumulated switch report data
- count of accumulated CCSIM switch reports
- pointers to the next and previous accumulation structures
- switch_report_data array

The single most important field is switch_report_data. This is an array of shorts used to accumulate the short data from a CCSIM switch report.

The global, static pointer accumulation_ring points to one of the accumulation structures (it is NULL at start-up time). When process_switch_report searches the ring for a structure whose switch number matches that of the newly received CCSIM switch report it simply moves through the ring via the "next" field. Process_switch_report only calls the two functions mentioned above. Of them, insert_switch_in_accumulation_ring simply callocs some space for an accumulation structure and fills in the switch_report_count, switch_node_number, and start_time, among the fields. It then adjusts the pointers of the new accumulation structure so that it becomes part of the accumulation_ring. insert_switch_in_accumulation_ring also calls copy_switch_report_to_accumulation_ring to fill in the switch_report_data field.

Note that the two functions process_switch_report and insert_switch_in_accumulation_ring are the only two functions which change the value of accumulation_ring. Although, other functions access the fields of the structure pointed to by accumulation_ring.

The second function called by process_switch_report is accumulate_sim_switch_report. It is called after process_switch_report has positioned accumulation_ring so that it is pointing to the accumulation structure for the desired switch. accumulate_sim_switch_report then does a three-way switch, whose case depends on how many CCSIM switch reports have been accumulated for this particular switch.

If the sim_switch_report_count is 0 for this switch then this CCSIM switch report is the first of the number to accumulate. In this case the sim_switch_report_count is set to 1, the switch number is set and the CCSIM switch report data is copied into the accumulation structure with a call to copy_switch_report_to_accumulation_ring possibly overwriting old data. A NULL csd_report is returned because not enough data has been accumulated to formulate a complete comma separated data report.

If the sim_switch_report_count is one less than the number needed to formulate a complete comma separated data switch report, then the addition of this current CCSIM switch report is enough data to formulate a comma separated data switch report.

For this case, first the sim_switch_report_count is set to 0 to be prepared for the next accumulation. Then the add_switch_report_to_accumulation_ring function is called and then formulate_csd_report_from_accumulation is called, whose value is returned.

For the case where the sim_switch_report_count is greater than 0 but one less than the number of switch reports needed to formulate a comma separated data switch report, the sim_switch_report_count is incremented and the CCSIM switch report data is accumulated with a call to add_switch_report_to_accumulation_ring. This case of the switch then returns a NULL csd_report because not enough data has been accumulated to formulate a complete comma separated data report.

The main objective of all the function calls in accumulate_sim_switch_report is to get prepared for formulate_csd_report_from_accumulation. This function is called, as described above, when enough data has been accumulated from CCSIM for one switch to formulate a comma separated data switch report.

The first thing this function does is to produce the comma separated data switch report file name and date string from the start time and end time fields of the accumulation structure. The file name and date strings are then copied into the csd_report structure.

At this time, if we don't have a valid switch report then we got here because we have gotten NUMBER_SIM_SR_TO_ACCUMULATE CCSIM switch reports (some of which were empty switch reports) to formulate a comma separated data report. For this case simply return the csd_report. The file_name field contains a valid file name so that the Trainer function can pick out the quarter and hour for formation of an empty switch report.

If we have a valid switch report accumulation, then we have accumulated NUMBER_SIM_SR_TO_ACCUMULATE valid CCSIM switch reports, so start filling in the comma separated data report structure with the start time and end time. These times are copied into the fn_start_time and fn_end_time rather than the hl_start_time and hl_end_time. The fn_ stands for "file name" and the hl_ stands for "header line". This is in reference to a comma separated data switch report because the time encoded in the file name is GMT where as the time from the header line of the switch report is the switch's local time. Use of the header line time was superseded by the file name time.

Next, formulate_csd_report_from_accumulation goes through the short data a short at a time. Recall that this short data is simply an accumulation of the short data from a CCSIM switch report. Some of the information has already been processed and some of the shorts are unused, so not every short is accessed.

The short data is accessed via a pointer to a short, switch_report_short_data. The value of this variable is initialized upon entry to the function with a pointer to the short data field of the accumulated structure. The pointer is incremented with each access to a short. (By reading the CCSIM document describing a CCSIM switch report, you can follow along short for short. The "csdfuns.c" code also describes the information short for short as it is accessed.)

From the header of the short data, the CCSIM switch number is accessed. Then, about the half of the short data corresponding to the CCSIM switch report header is skipped. The second half contains data pertaining to the CP, RCVR, RADR and DTSR one-liner reports. This data is read and filled into the report_data field of the CSD_REPORT structure. At the same time, constants are filled in as well as provisioning fields.

After the data for the five one-liner reports has been filled in, the constant 6 is filled in for the "TRK" report. At this point, switch_report_short_data is pointing to the trunk

data. From the first five shorts, get the number of "trunks in service" and the "clli number". The "clli number" will be used to access the trunk group number and number of trunks equipped which was read from the {net}.clli file at program initialization.

A comma separated data report does not distinguish among the 5 CCSIM call priority levels; therefore, take the total of the five shorts. This is done for all the statistics in each clli report. Each of these totals is filled into the comma separated data switch report structure. (Again, the CCSIM document which describes a CCSIM switch report documents the clli part of the switch report short for short. The "csdfuns.c" code also describes the information short for short as it is accessed.)

The comma separated data switch report structure is now completely filled in, so return a pointer to it. (Many functions in the chain of calls started from process_switch_report return a pointer.) The return from process_switch_report returns the pointer to the caller who will utilized the structure being pointed to.

Besides the csdstub debugging/testing program, process_switch_report is only called twice, once for a valid switch report and once for an empty switch report. These calls are within the trainer function; thus, it is the function which must do something with the comma separated switch report structure.

C.2.3.2.4 Messages from the NMES

Again, for the case of the translate executable, when the source is NMES_ID, the Trainer passes along the message from NMES to CCSIM as long as CCSIM is running. Consequently, the handle_ccsim_msg function passes along most CCSIM messages directly to NMES. The Trainer filters out switch reports and empty switch reports because it Translates this information and then sends it to NMES.

C.2.3.2.5 Messages from the Trainer Input Program

When the source is INPUT_ID, the trainer_input program has sent a text message to the Trainer. The trainer_input program enables input to CCSIM via CMDFMT messages. These are character string messages that CCSIM interprets as if it were found in the {net}.cmds file. The main purpose of running this process is to allow the Trainer user a means to enter simple commands to CCSIM without having to run with dsntool. The use of this program is not recommended because the person who types the commands must not make a mistake. The recommended means of input to CCSIM is via the graphics interface which is mouse driven and is designed to accept correct input only before sending a command to CCSIM.

C.2.3.2.6 Messages from the CCSIM Graphics Interface

The last source is GRPH_ID. The two messages from the graphics are RUN_REAL_ON and RUN_REAL_OFF. Which stand for turning run real time on or off respectively. Run real time is implemented in the Trainer by sending PAUSE and CONT messages to CCSIM to regulate its running speed.

C.2.4 "C" Structures Walk-Through of Switch Report Formation

With reference to "C" structures defined in "msgstruct.h" and "csd.h", the following paragraph describes the transitions from one structure to the next.

After enough SWRPT or ESWRPT structures received from CCSIM have been accumulated in an ACCUMULATOR_SWITCH_REPORT structure (one in the accumulation_ring) formulate_csd_report_from_accumulation is called. It formulates a CSD_SWITCH_REPORT structure from the ACCUMULATOR_SWITCH_REPORT structure. A pointer to the CSD_SWITCH_REPORT structure is returned up the chain of calls to the Trainer function. That function either prints the CSD_SWITCH_REPORT structure to a file for transfer to the NMSS workstation or it Translates the CSD_SWITCH_REPORT structure into a SWRPT structure for writing to the NMES.

C.2.5 Using the Switch Report Output Functions

The Trainer/Translator concept of comma separated data switch reports is file based. The reason for this is that the DAI/NMSS protocol is file based. In the Trainer scenario, before a comma separated data switch report is transferred to the NMSS, it must be in a file whose name denotes important information about the contents. When using canned data to drive the NMES, note that the source of the comma separated data switch reports is the DAI. The main source of DAI switch reports is Paul Gesswein who sends tapes to Lincoln Lab. The tape contains file based switch reports.

There are functions for reading and writing both CCSIM switch reports and comma separated data switch reports. The functions which read fill in a "C" structure with the information found in a file and the writing functions create a file from the information in the same "C" structures. There are also functions for converting between "C" structures. (See "csd.h" for the structure definitions.)

All these functions work well with each other. For example, when the trainer is used to drive the NMES, CCSIM switch reports are accumulated and then formulated into comma separated data switch reports which are then translated back into CCSIM format switch reports.

The functions which read and write files for the SWRPT structure are mainly used with the csdstub program for developmental purposes. The file format is the same as CCSIM's output for one switch.

The functions which read and write files for the CSD_SWITCH_REPORT structure are used routinely in the Trainer/Translator. This file format is that of a comma separated data switch report.

Both of the output functions take a pointer to a structure as the second argument. The first argument defines the file name for output and calls for different actions in the two functions. The following table lists the three different arguments and the output file name for the two functions:

argument	print_CCSIM_switch_report	print_csd_switch_report
"" (empty string)	"CCSIM.sw.for." + 3 letter switch name file name	the standard comma separated data switch report
"stdout"	outputs to stdout	outputs to stdout
any other string	the string is interpreted as the file name	string is interpreted as the file name

The print_csd_switch_report_w_extra_tgs function takes the same actions as print_csd_switch_report with the additional feature of adding trunk groups from comma separated data switch reports found in a sub-directory.

The input functions, read_csd_sr and read_CCSIM_switch_report must be called with the literal file name.

C.2.6 List of Files

HEADER FILES (found in ...trainer/src)

comm.h	declarations for NMSS/DAI protocol
csd.h	header for Trainer/Translate related files
env.h	contains extern declarations for global variables
global.h	global useful constants
msgcnst.h	my new version of CCSIM header file
msgstruct.h	symbolic links to CCSIM header file
nmssfuns.h	
trainer.h	non-extern declarations for the env.h variables defines for organizing process communication

C FILES (found in ...trainer/src)

canned.c	main module for running with canned data (fo. NMES)
comm.c	NMSS/DAI protocol functions
comm_main.c	main module for accessing comm.c functions
controls.c	
csdfuns.c	Trainer/Translate functions
csdstub.c	stub program used to test csdfuns functions
dev_controls.c	functions for control of /dev/ttya and /dev/ttyb
env.c	processing of Trainer/Translate runtime parameters
inval.c	gets values from the (net).inval file
log.c	functions for logging and file manipulation
myio.c	io functions (e.g. yes_no_p, get_int...)
mystrings.c	string functions
network_funs.c	functions related to getting (net) info
nmssfuns.c	mainly processing of NMSS workstation control requests
time.c	unit date/time calls
trainer.c	acts as in-between for CCSIM and other programs
trainer_funs.c	Trainer/Translate high level functions
trainer_main.c	"main" access to the function Trainer
translate_main.c	"main" access to the function Trainer

SCRIPT FILES (written for Trainer/Translate and found in the installed bin directory /usr/(host)/dsn/bin)

runsys.trainer	starts up necessary programs for Trainer
runsys.translate	starts up necessary programs for Translate
runtrainer	starts up Trainer and checks for timeout
runtranslate	starts up Translate and checks for timeout

SOCKET BASED COMMUNICATIONS (functions not written by Mike Walsh, copies found in ...trainer/src)

get_socket.c	basically does a unix "socket" and then a "bind"
makesocket.c	basically calls get_socket and then a "connect"
servsetup.c	does a "socket", "bind", "listen" and "accept"
socket_comm.c	function for reading from a socket

MISCELLANEOUS FILES (found in /home/athena/walsh/trainer/doc/old or ...trainer/misc)

```
error-solution    errors/solution encountered while running the
                  Trainer
affirmative_control_response
                  file appended to an NMSS control file and
                  sent to NMSS
negative_control_response
                  file appended to an NMSS control file and
                  sent to NMSS
```

DOCUMENTATION FILES (found in ...trainer/doc)

```
canned.programmer.doc
canned.user.doc
commm.programmer.doc
commm.user.doc
csdstub.programmer.doc
csdstub.user.doc
trainer.programmer.doc
trainer.user.doc
translate.programmer.doc
translate.user.doc
```

NETWORK FILES

These are those network files accessed by the Trainer:

```
(net).clli        for clli information
(net).inval       for srsync
(net).node        for node information
(net).reporting   This file specifies which switches should be
                  reporting to the NMSS workstation so
                  that the Trainer can send a "sw-report-on" to
                  CCSIM and receive switch reports for the
                  desired switches.
(net).sw          This file contains the provisioning data
                  which is basically constant data in the
                  non-trunk data portion of a comma separated
                  data switch report.
(net).tconfig     "trainer config" specifies options for
                  running the Trainer as well as some debugging
                  flags (use of this file is being phased out)
```

C.2.7 Requirements for Running the Trainer

C.2.7.1 Introduction

Running the trainer is similar to running CCSIM. In a manner of thought, running the trainer is a superset of running CCSIM because the Trainer is the addition to CCSIM of the ability to communicate with the NMSS workstation.

Therefore, in order to run the Trainer, one should first know how to run CCSIM. Once a good level of proficiency is obtained with running CCSIM, one can then easily run the trainer.

C.2.7.2 Creating the Necessary Directories

Refer to the CCSIM users manual for instructions for using setupsim. The result of running setupsim is the copying all the network files specified in the section "list of files" into your network directory, referred to in this document as {net}.

Setupsim provides you with all the network files that are necessary in order to run CCSIM and the trainer. Setupsim also creates a directory named "sample" which is only used for the trainer.

The {net}/sample directory must contain sample switch reports from the DAI. The sample reports are needed to supply trunk group data not simulated by CCSIM so that the trainer can produce switch reports with full sets of trunk group data for the NMSS workstation. If the add_csd_trunk_groups variable is false then the Trainer will not look for these sample reports. In this case the trainer produces switch reports with incomplete trunk group data.

C.2.7.3 Customizing the Network Files for the Trainer

C.2.7.3.1 The Inval File

The inval file value srsync=0 must be present in the file {net}.inval. This value instructs CCSIM to produce switch reports for time intervals beginning at the same time for all switches.

C.2.7.3.2 The Reporting File

The {net}.reporting file should contain the three letter switch abbreviations for all switches which are polled by the DAI for the NMSS workstation.

C.2.8 Tips on Modifying and Debugging the Trainer Code

C.2.8.1 Using Dbxtool

The select returns an error status of -1 while debugging the Trainer using dbxtool, closing the window and then re-opening the window. This has no effect on the debugging session because the trainer main loop returns to the select upon the next cycle.

Under the same circumstances, the servsetup call returns a -1 instead of hanging, waiting for some other process to connect. This has a negative effect on the debugging session because the servsetup call fails to connect to some other process.

C.2.8.2 Debugging Forks

dbxtool (dbx) does not agree with forks. (Or, I can't make it debug forks.) So, when modifying the Trainer/Translate code, comment out the segments of code which contain a fork and which are in the line of execution. Open a window and start the program by hand (or run the program in dbxtool.) Run the Trainer or Translate in dbxtool and you are all set for debugging.

C.2.8.3 The Csdstub Program

csdstub's purpose is to allow access to Trainer/Translate functions without having to fire up CCSIM and other programs. csdstub can read in either a CCSIM switch report or a csd switch report and then call any functions you desire. csdstub has no forks and hence can be easily debugged in dbxtool.

C.2.8.4 The UNIX PS and NETSTAT Commands

When running Trainer, check machine by doing ps and netstat. These commands will yield information about processes using the ports used by the IDSIM programs. During Trainer/Translator development, many times, processes were left running (connected to a port). The next attempted run would fail to connect.

A problem has arisen twice which I could not duplicate purposefully. While two processes were trying to connect, the first process connected at the designated port; but, the second process did not. Somehow, a port was waiting to be connect to even though I could not associate any process with that port. The port status could be seen with a "netstat -a". The problem was resolved by rebooting. A comparison of process status before and after the reboot did not yield any information.

C.2.8.5 System Status of the Serial Ports

C.2.8.5.1 The Sun Serial Ports with Regard to the Trainer

Either port "a" or "b" can be used by specifying "commm_port=a" or "commm_por t=b" on the command line. If no port is specified, the default is commm_port=b.

The serial port communications code has proven to be very reliable when all is well in Unix. Three things to check in the Unix domain which have an effect on the serial ports are:

C.2.8.5.1.1 Read/Write Permission

The user running the program which used the serial port must have read/ write permission to the files /dev/ttya and /dev/ttyb. NOTE: do an "ls -l /dev/tty[ab]" to see if you have read/write permission. An example desirable output would be:

```
0 crw-rw-rw-  1 root      12,   0 Jun 19 16:25 /dev/ttya
0 crw-rw-rw-  1 root      12,   1 Oct 31 12:22 /dev/ttyb
```

C.2.8.5.1.2 Daemons

The ports must be free of daemons. Do an "ls" as shown in 1. to see who is the owner. If the owner is root then the /dev file has no daemons. If the owner is daemon then there is probably a process running in the background which has control over the port. A good example of this is a process which uses the port for printing. Somehow free the port of the daemon.

C.2.8.5.1.3 Login Ports

Do a "cat /etc/ttys | more" to see information about tty devices. There should be an entry for both ttya and ttyb near the top of the list. A 1 in the first column indicates that the tty is login-able. A 0 indicates that it is not. For the Trainer to use the serial ports, the port must be a non-login port. The first three lines of an example desirable output would be:

```
12console
02ttya
02ttyb
```

C.2.9 Special Function Calls

There are system calls in trainer_funs.c for creating shelltools after a fork. There are system calls in log.c for several file utilities. There is a system call in csdfuns.c for doing an "ls" of the <net>/sample directory. The "ls" is in the

form of a wildcard looking for switch reports for a particular source switch.

C.2.10 Trainer Input Program

When trainer_input is true, the Trainer or Translate program will fork the trainer_input process. It allows the user to type CCSIM commands by hand into a window. All strings except "quit" are passed directly to CCSIM as a CMDFMT. The main purpose of trainer_input is to allow detailed control of a simulation without using the graphics. (I have not used it since early coding; but, it is a useful tool for running the Trainer and entering commands to CCSIM dynamically.)

C.2.11 Notes on Trainer Emulation of the DAI

The date string in the switch reports: " 1:" - regards software version in switches. Don't worry about this for the Trainer. The NMSS doesn't seem to care. csd is an abbreviation for "comma separated data".

APPENDIX D NEW CCSIM INPUT FILES

D.1 The CLLI File

The CLLI file (<net>.clli) contains all information needed by CCSIM about the Trunk Groups (TGs) involved in a simulation. It is an ASCII file, each line of which is one TG record. When two or more TGs connect a pair of switches, the order in which the TG records appear in the file determines the order in which the TGs are searched when a call is routed in CCSIM.

Each TG record (file line) must have 13 fixed fields and may have an additional optional field. The meanings of the fields are associated with their positions on the line (there are no field identifiers). Six of the fields are not currently used by CCSIM but are specified in anticipation of possible future extensions of the simulation capabilities. There must be at least one blank space between each field and no blank lines in the file.

The fields in the TG record are as follows:

1. Name.
   The clli name of the trunk group to which the record applies. The clli name is that by which the group is known to the source switch. In a real network clli names need only be unique at each switch, but CCSIM requires that they be globally unique. To guarantee uniqueness, CCSIM uses the convention that the clli name is a concatenation of the three-character name of the source switch and an arbitrary string that must be unique among all the trunk groups at the source switch. The name may be as many as 16 characters in length overall. When a real network is to be simulated, the arbitrary string can be set to match the real clli used in the network. When an abstract network is to be simulated and no real cllis are available, a utility called 'link-to-clli' can be used to create a clli file with arbitrary names that match all the CCSIM requirements.

2. Source Switch.
   The 3-character name by which the source switch is known to CCSIM. The value of this field must match a switch name found in the net.node file associated with the simulation.

3. Destination Switch.
   The 3-character name by which the switch at the distant end of the TG is known to CCSIM. The value of this field must match a switch name found in the net.node file associated with the simulation.

4. Delay Type.
   A field used by CCSIM to determine signaling delay on the TG. Acceptable values are TERR for terrestrial and SAT for satellite groups.

5. Capacity.
   The number of trunks in the TG.

6. Other Clli.
   The clli name by which the TG is known to the destination switch. There must be a TG record in the file with a 'Name' field that matches this field.

7. Group Number.
   The trunk group number that identifies the group in switch reports. This number is not checked by CCSIM but must agree with the real group number if the switch reports from CCSIM are to match those from the real switch.

8. Type.
   A type field used in graphics displays of TG properties. Acceptable values are VF, PCM, and VFDG. If CCSIM were to route data calls differently from voice calls, PCM and VFDG trunks would be given preference for data.

9. Signaling.
   Tells CCSIM the type of signaling used for the TG. Acceptable values are CCS for Common Channel Signaling and IB for In- Band signaling.

10. Transmission.
    A type field used in graphics displays of TG properties. Acceptable values are TROPO, MWAVE, CABLE, and SATEL. Currently CCSIM makes no use of this field.

11. Supplier.
    Another type field used in graphics displays but not currently by CCSIM. Acceptable values are LEASE and MIL.

12. Directionality.
    Acceptable values are '2W' for normal two-way groups, '1O' for one-way outgoing groups, and '1I' for one-way incoming groups. Conceptually, CCSIM should route calls only on '2W' and '1O' groups, but the field is not currently used by CCSIM, and affects only the graphics displays. To achieve a simulation with one-way trunk groups, an experimenter must use a suitable combination of routing tables and controls to directionalize trunk groups.

13. Function.
    The function of the TG in the network. Acceptable values are 'IS' for inter-switch trunks, 'PBX', and 'TEST'. Conceptually, CCSIM should deal only with 'IS' trunks, since they are the only type simulated, but the field is currently ignored by CCSIM and affects only the graphics displays.

14. Change Search Order.
    This is an optional field that can be used to split traffic between two trunk groups connecting a pair of switches. It is looked at by CCSIM only on the first of the two groups to be found in the clli file. It specifies the probability that the second group found will be searched first for a free trunk when a call is to be routed. Acceptable values are integers between 0 and 100. This field allows an experimenter to spread the traffic between the groups to approximate the behavior of a real network which routes by trunk group lists rather than the next- switch lists used in CCSIM.

D.2 The Switch File

The Switch (<net>.sw) file contains provisioning and other information needed to generate detailed switch reports for the NTI DMS switches in the European DSN. It is a column-formatted ASCII file with seven characters per column. There are 12 columns, each containing a left- justified four-decimal-digit field. The meanings of the fields with comments as to their use in CCSIM and/or the TRAINER are as follows:

1. The number of CP letters (message blocks for call processing) available in the switch. Used for the CP switch report line.

2. The number of wakeup blocks available. Also used for the CP line.

3. The number of call processes available. Another CP line item.

4. The number of Call Condense Blocks (CCBs) available. This value is used by CCSIM to calculate CCB seizures, usage, and overflows for the CP line.

5. The number of MF receivers available for interswitch signaling. CCSIM uses this value to calculate MF receiver usage and overflows for the RCVR switch report line.

6. The number of Digitone receivers available for local subscriber and PBX signaling. CCSIM uses this value to calculate Digitone receiver usage and overflows for the RCVR line.

7. The number of MF test calls per hour. Used as a field in the RADR report line and to calculate the number of test calls actually carried out during the report interval. This value appears in the RADR line and affects values in the RCVR line.

8. The lower delay threshold for MF receiver tests. Used for the RADR report line.

9. The upper delay threshold for MF receiver tests. Used for the RADR report line.

10. The number of Digitone test calls per hour. Used for the RADR report line and to calculate the number of test calls carried out during the report interval.

11. The lower delay threshold for Digitone receiver tests. Used for the RADR report line.

12. The upper delay threshold for Digitone receiver tests. Used for the RADR report line.

APPENDIX E CCSIM NETWORK MANAGEMENT CONTROLS

Three types of controls are implemented in CCSIM. They are listed in the following table. Pre-route controls are applied at a switch before an attempt is made to route a call out of the switch. Pre-hunt controls are applied before an attempt is made to find a free trunk in the trunk group to which the control is applied. Post-hunt controls are applied after the call has overflowed the trunk group. Post-route controls are applied after a call has failed to find a route out of the switch. Within each group the controls are listed in the order in which they are applied. The term 'DMS' in the right-hand column indicates that the control is implemented in accordance with Northern Telecom Practices for DMS switches. The term '490L' after the DRZ control indicates that this is the directionalize control as implemented in the AUTOVON 490L switches. The term 'EXP' after the CAN-EOC-OVF indicates a control introduced for experimental purposes that does not correspond to a control to be found in any real switch. A blank entry in the third column indicates that the control is implemented according to our interpretation of the DCA generic switch specification.

Pre-route Controls:		
1. Code Block	(CB)	DMS
2. Call Gap	(GAP)	
3. Alternate Route Cancel (B)	(ARC-B)	
Pre-hunt Controls:		
1. Alternate Route Cancel (A)	(ARC-A)	
2. Directional Reservation of Equipment	(DRE)	DMS
3. Directionalization	(DRZ)	490L
4. Cancel To (Percent)	(CANT-%)	DMS
5. Cancel To (Rate)	(CANT-RATE)	
6. Skip	(SK)	DMS
Post-hunt Control:		
1. Cancel From	(CANF)	DMS
Post-route Control:		
1. Cancel End-of-Chain Overflows	(CAN-EOC-OVF)	EXP

CCSIM does not have different command for applying and removing NM controls as do real switches. Instead, it uses the same command both to apply and remove a control. Removal occurs when a particular parameter (often a percent) is set to a particular value (zero for the percent parameter). In the following descriptions of the individual NM controls, the percent-equal-to-zero-for-removal convention is assumed for all controls having a percent parameter. For other controls, the parameter values for removal are specified explicitly.

Controls that cancel calls have parameters specifying the kind of announcement message to which a canceled call should be connected. There are three such announcements. In CCSIM, specifying Emergency Announcement 1 (EA1) or Emergency Announcement 2 (EA2) will cause an affected call to be counted as failed and not to be retried. The No Circuit Available (NCA) announcement causes the call to be handled just as it would have been if it had blocked due to the lack of a trunk out of the switch at which the control was applied. In that case, CCSIM will retry the call subject to the retry parameters applicable to the simulation run.

CCSIM does not yet have tables to translate between codes (telephone numbers) and switch (node) names. Consequently, controls such as Code Block (CB) use destination node names instead of codes to specify the calls that are to be blocked. In this respect, such controls differ in syntax from those in real switches, but the effect on traffic is the same as would occur if all office codes at a switch were specified in a real network application of a code control.

In the following descriptions the word 'ALL' is permissible as a value for some parameters. When used, its affect is the same as would be achieved by issuing a sequence of controls, one for each of the allowable values of the parameter. For example, it allows a CB control to be applied at all switches with a single command. There is no corresponding capability in a real telephone network.

Pre-Route Controls:

CB - Code Block

The CB control is put on at a switch and applies to originating calls only. It blocks a specified percentage of the traffic to a destination switch from entering the network. When the control is applied at less than 100%, only routine calls are affected. At 100%, CB blocks calls of all precedences. Blocked calls are handled according to the announcement type specified in the control.

In a real network, the CB control would apply to codes and could be used to block calls to individual telephone numbers. In CCSIM it can be used only for all the codes identified with a particular switch.

Usage: CB node1 node2 percent ann
node1 is the three-letter node name for the switch at which the control is to be applied (or ALL)
node2 is the three-letter node name for the destination switch to which calls are to be blocked (or ALL)
percent is the percentage (0-100) of calls to block
ann is the announcement type (NCA, EA1 or EA2)

Example: CB ALL UXB 50 EA2
Cancels 50% of the routine calls from all nodes to Uxbridge. The canceled calls will not be retried.

GAP - Call Gap

The GAP control is put on at a switch and applies to originating calls only. It determines the rate at which traffic to a particular destination switch is allowed to enter the network. After an attempt to route a call to the specified destination has been allowed by the GAP control, subsequent calls to that destination are blocked for a period of time designated as the "gap interval." After the expiration of the gap interval, the next call to that destination will be allowed to attempt to find a route. The gap interval is chosen from the interval set 0 (no-control), 0.10, 0.25, 0.50, 1, 2, 5, 10, 15, 30, 60, 120, 300, 600 seconds, and infinity. An infinite interval prohibits all attempts. Blocked calls are handled according to the announcement type specified in the control. At intervals 0 through 600, only routine calls are affected. At the infinite interval, calls of all precedences are affected.

In a real network, the GAP control would apply to codes and could be used to gap calls to individual telephone numbers. In CCSIM it can be used only for all the codes identified with a particular switch.

Usage: GAP node1 node2 index ann
node1 is the three-letter node name for the switch at which the control is to be applied (or ALL)
node2 is the three-letter node name for the destination switch to which calls are to be gapped
index is a pointer into the following table of gap intervals (1 removes the control)
ann is the announcement type (NCA, EA1 or EA2)

Index (Seconds/Call)	Gap Interval	Calls per Minute
1	0	All
2	0.1	600
3	0.2	240
4	0.50	120
5	1	60
6	2	30
7	5	12
8	10	6
9	15	4
10	30	2
11	60	1
12	120	1/2
13	300	1/5

14	600	1/10
15	Infinity	None

Example: GAP ALL UXB 11 NCA
Allows only one call per minute to enter the network from each of the other switches. Blocked calls are allowed to retry.

ARC-B - Alternate Route Cancellation (Type B)

When the ARC-B control is put on at a switch, it applies to both tandem calls and originating calls. All calls to the specified final destination are allowed to use only the direct route out of the switch. A call for which a free or preemptible trunk cannot be found on the primary route will be blocked. The control can be specified to apply to either routine-only or all-precedence traffic.

Usage: ARC-B node1 node2 precedence
node1 is the three-letter node name for the switch at which the control is to be applied (or ALL)
node2 is a three letter node name for the final destination switch for the call (or ALL)
precedence is either R (routine-only), AP (all-precedences) or NONE (remove the control)

Example: ARC-B TJS UXB R
Deny alternate routes for routine traffic between Torrejon and Uxbridge

Pre-Hunt Controls:

ARC-A - Alternate Route Cancellation (Type A)

The ARC-A control is applied to a link, i.e., one or more trunk groups between a pair of switches. It causes traffic which would normally use the link as an alternate route to skip to the next route, if any, in the routing table. Its effect is to give preference to traffic that would use the link as a direct route. It can be applied to affect routine-only or all-precedence traffic. Calls affected by these controls are not counted as attempts on the link.

Usage: ARC-A node1 node2 precedence
node1 is the three-letter node name for the switch at which the control is to be applied (or ALL)
node2 is a three letter node name of the switch at the remote end of the link to which the control is to be applied
precedence is either R (routine-only), AP (all-precedences) or NONE (remove the control)

Example: ARC-A TJS UXB AP
Allow only direct routed traffic to access the link between Torrejon and Uxbridge at Torrejon

DRE - Directional Reservation of Equipment

The DRE control is applied at a switch to a trunk group. It gives priority to incoming traffic by reserving a number of idle trunks in the group. When the number of idle trunks is equal to or less than the number of reserved trunks, all traffic (direct- and alternate-routed) is skip-routed. Calls of all precedences are affected by the DRE control. Calls affected by these controls are not counted as attempts on the link.

Usage: DRE node1 clli reserve
node1 is the three-letter node name for the switch at which the control is to be applied
clli is the clli name of the trunk group to which the control is to be applied
reserve is the number of trunks to be reserved

Example: DRE UXB UXBTJS084 1
UXB must have more than one free trunk in the group 'UXBTJS084' before it can use that group for routing a call of any precedence via Torrejon.

DRZ - Directionalization

The DRZ control is applied at a switch to a trunk group. It places a limit on the number of trunks which may be used for outgoing calls on the trunk group. The limit ranges from 1 to the maximum capacity of the trunk group. Setting the limit to zero removes the control. Traffic (direct and alternate-routed) which would exceed the limit is skip-routed. Calls of all precedences are affected by the DRZ control.

Calls affected by these controls are not counted as attempts on the link.

Usage: DRZ node1 clli limit
node1 is the three-letter node name for the switch at which the control is to be applied
clli is the clli name of the trunk group to which the control is to be applied
limit is the number of trunks that may be used by outgoing calls

Example: DRZ DVN DVNTJS080 3
Donnersburg may use no more than 3 trunks in the group 'DVNTJS080' for outgoing calls to be routed via Torrejon.

CANT-DIRECT-%/CANT-ALTER-% - Cancel To (Percent)

These controls are applied to a trunk group. They cancel a percentage of the routine traffic offered to the group. CANT-DIRECT-% cancels only direct routed routine traffic. CANT-ALTER-% cancels only alternate routed routine traffic. Calls affected by these controls are *not counted as attempts on the link.* Canceled calls are handled according to the announcement type specified in the control.

Usage: CANT-DIRECT-% node1 clli percent ann
or CANT-ALTER-% node1 clli percent ann
node1 is the three-letter node name for the switch at which the control is to be applied
clli is the clli name for the trunk group to which the control is to be applied
percent is the percentage (0-100) of calls to cancel
ann is the announcement type (NCA, EA1 or EA2)

Example: CANT-DIRECT-% UXB UXBTJS084 80 EA1
Cancels 80% of the routine direct-routed calls offered to the trunk group 'UXBTJS084' from Uxbridge to Torrejon. The canceled calls will not be retried.

CANT-DIRECT-RATE/CANT-ALTER-RATE - Cancel To (Rate)

These controls are applied to a trunk group. They control the rate at which calls are allowed to access the group. After one call is allowed access to the group, subsequent calls that otherwise attempt to use the group are canceled until a period of time (the 'gap interval') has elapsed. The first call to arrive after the gap interval will escape cancellation and be allowed to access the group. The gap interval is chosen from the interval set 0 (no-control), 0.10, 0.25, 0.50, 1, 2, 5, 10, 15, 30, 60, 120, 300, 600 seconds and infinity. *An infinite interval cancels* all calls attempting to access the group. CANT-DIRECT-RATE affects only direct routed traffic. CANT-ALTER-RATE affects only alternate routed traffic. When the interval is less than infinity, only routine calls are canceled. An infinite interval cancels calls of all precedences.

Usage: CANT-DIRECT-RATE node1 clli index
or CANT-ALTER-RATE node1 clli index
node1 is the three-letter node name for the switch at which the control is to be applied
clli is the clli name of the trunk group to which the control is to be applied

index is a pointer into the following table of gap intervals (1 removes the control
ann is the announcement type (NCA, EA1 or EA2)

Index (Seconds/Call)	Gap Interval	Calls per Minute
1	0	All
2	0.1	600
3	0.2	240
4	0.50	120
5	1	60
6	2	30
7	5	12
8	10	6
9	15	4
10	30	2
11	60	1
12	120	1/2
13	300	1/5
14	600	1/10
15	Infinity	None

Example: CANT-DIRECT-RATE TJS TJSUXB084 10 EA2
Allows at most 2 direct-routed routine calls per minute to search trunk group 'TJSUXB084' for a trunk to Uxbridge from Torrejon. All other direct- routed, routine traffic that would otherwise search the trunk group will be canceled and not retried.

SK-DIRECT/SK-ALTER - Skip

The SKIP control is applied to a trunk group. It skip-routes traffic to the next trunk group in the routing chain. SK- DIRECT affects only direct routed calls. SK-ALTER affects only alternate routed calls. Calls of all precedences are affected by the SKIP control. Traffic that is skip-routed is not counted in the statistics as attempts on the link.

Usage: SK-DIRECT node1 clli percent
or SK-ALTER node1 clli percent
node1 is the three-letter node name for the switch at which the control is to be applied
clli is the clli name for the trunk group to which the control is to be applied
percent is the percentage (0-100) of calls to be skip-routed

Example: SK-DIRECT UXB UXBTJS084 70
Skip-routes 70 percent of the direct-routed calls that attempt to use trunk group 'UXBTJS084' from Uxbridge to Torrejon.

Post-Hunt Controls:

CANF-DAR/CANF-AR - Cancel From

The CANF control is applied to a trunk group. It cancels a specified percentage of the traffic overflowing from the group and prevents it from continuing to the next group in the routing chain. Canceled calls are handled according to their announcement type. CANF-DAR cancels 100% of any alternate routed calls and the specified percentage of direct routed calls. CANF-AR cancels only the specified percentage of alternate routed calls. Only routine calls are affected by this control. Calls affected by the CANF control are counted as attempts on the link.

Usage: CANF-DAR node1 clli percent ann
or CANF-AR node1 clli percent ann
node1 is the three-letter node name for the switch at which the control is to be applied
clli is the clli name of the trunk group to which the control is to be applied
percent is the percentage (0-100) of direct-routed (CANF-DAR) or alternate-routed (CANF-AR) calls to cancel
ann is the announcement type (NCA, EA1 or EA2)

Example: CANF-DAR UXB UXBTJS084 10 EA2
Cancels 100% of the alternate-routed and 10% of the direct-routed traffic that overflows trunk group 'UXBTJS084' from Uxbridge to Torrejon. The canceled calls will not be retried.

Post-Route Controls:

CAN-EOC-OVF - Cancel End-of-Chain Overflows

The CAN-EOC-OVF control is applied at a switch. It cancels a percentage of routine calls that overflow the last trunk group in the routing chain for a specified destination switch. The canceled calls are not retried. They are counted as attempts and overflows for all trunk groups in the chain, and they are counted in a special statistic associated with the control. This control was added to CCSIM for experimental purposes. It is not found in real switches.

Usage: CAN-EOC-OVF node1 node2 percent
node1 is the three-letter node name for the switch at which the control is to be applied (or ALL)
node2 is the three-letter node name for the destination switch for which overflowing calls are to be canceled (or ALL)
percent is the percentage (0-100) of calls to cancel

Example: CAN-EOC-OVF TJS ALL 50
Cancels 50% of the routine calls at Torrejon that fail to find a route out of the switch, no matter what their destination. The canceled calls will not be retried.

APPENDIX F CCSIM USER'S MANUAL DESCRIPTION

One of the FY89 deliverable items in the Statement of Work was a user's manual for the Call-by-Call Simulator. This document was written, and copies have been distributed to the sponsor and placed at locations where CCSIM is in use, specifically including DCEC/DRFB and DCA-Europe. Additional copies will be provided upon request.

The CCSIM User's Manual is not included as an Appendix to this report, because it is quite large (about 170 pages). The following is a list of the major section headings of the document, to provide an idea of its contents as an aid to the reader in determining whether he wants to see the full manual.

1. Background
2. CCSIM Controls Descriptions
3. How to Execute the Simulator
4. Tutorial in CCSIM Operation
5. Graphics Interface Description
6. Batch Operation
7. CCSIM Parameters
8. Preparation of Input Files
9. CCSIM Outputs
10. Troubleshooting
11. Installation
12. Maintenance

Appendix A Error Messages
Appendix B Support Programs for CCSIM
Appendix C Limitations and Restrictions

APPENDIX G TRAMCON EVENT GENERATOR SPECIFICATION

G.1. SCOPE

G.1.1 Identification

This Software Requirements Specification establishes the requirements for the Transmission Monitoring and Control (TRAMCON) Event Generator (TEG).

G.2 Purpose

G.2.1 TRAMCON Mission

The TRAMCON alarm monitoring and reporting system collects and reports alarms associated with the Digital European Backbone (DEB). The DEB is a network of microwave relay sites that carries T1 trunks for the Defense Communications System (DCS) in Europe. Each TRAMCON system consists of one or more (normally one) TRAMCON Master (TM) stations and a number of remote units. The TM stations collect and process the data from the remote units which are connected directly to monitoring and control points on the transmission equipment. The remote units can be located at arbitrary distance from the central computer and communicate with the central computer via direct connection or remotely via modems.

The primary functions of TRAMCON are to remotely collect equipment status and performance data and to allow the remote operation of relay switches associated with the equipment. TRAMCON monitoring includes transmission alarms as well as facility alarms (e.g. power failure, intrusion, fire) and status. The TRAMCON systems also process the status and performance data in various ways to assist TRAMCON operators in assimilating and acting on the data. This processing typically includes the following:

a. Alarm processing, such as comparison of collected data to alarm threshold values.

b. Alarm correlation, i.e., the presentation to the operator of information on the status of related pieces of equipment.

G.1.2.2 DPAS Mission

The Digital Patch and Access System (DPAS) provides for the control, monitoring, and patching of T1 trunk circuits (not only those carried by the DEB, but also other media). DPAS is implemented using the AT&T Digital Access and Cross-Connect System (DACS II) and has been projected to include a Network Control System (NCS) as well. DPAS and the DACS II equipment

also monitor and report alarms associated with their components and the T1 trunk signals.

G.1.2.3 MITEC Mission

The Machine Intelligent Technical Controller (MITEC) will assist the Technical Controller (TC) in troubleshooting telecommunications circuits in a Technical Control Facility (TCF). MITEC will reduce TC manpower requirements and improve TC effectiveness by reducing the time to troubleshoot a circuit, automating currently manual quality assurance and recordkeeping activities, and refining the accuracy and consistency of troubleshooting and quality assurance.

G.1.2.4 Intelligent Alarm Filtering

The occurrence of a failure in the equipment monitored by TRAMCON and DPAS typically causes not only a primary alarm but also a number of sympathetic alarms (alarms that do not themselves indicate a fault but are consequences of the primary fault). When the number of these sympathetic alarms is large, it takes substantial skill and patience on the part of the human operator to identify the primary alarm.

Two systems under development at Lincoln Laboratory, MITEC and the Network Management Expert System (NMES), are intended to perform the task of identifying the fault that underlies a communications system failure and institute corrective action with a minimum of effort on the part of the human operator. MITEC addresses the diagnosis and rerouting of point-to-point user and trunk circuits; NMES addresses the diagnosis of switched networks and the application of network management controls. Both systems would profit from the introduction of TRAMCON and DPAS alarm data as a source of diagnostic information.

G.1.2.5 Role of the TRAMCON Event Generator (TEG)

In order to develop adequate techniques for the exploitation of alarm data in MITEC and NMES, a system is needed that can supply such data in response to a wide range of possible faults. TEG will be a simulator which produces the entire constellation of primary and sympathetic alarms associated with a fault and supplies these data to MITEC and NMES. The knowledge needed to define the relationships between faults and alarms already exists and has to some extent been recorded by USAF, Army, and DCS personnel.

The TRAMCON and DPAS alarm simulation system shown in Figure 1 has two primary purposes: to support the development of MITEC software for alarm analysis and fault diagnosis, and to form a part of a broader simulation of Defense Communications System

(DCS) operation. To meet these goals it must provide the following functions:

a. Storage of an internal representation of microwave and DPAS network segments, including all the equipment items and their alarm functions;

b. An operator interface which permits selection of any failure event(s) in the networks which are recognizable by TRAMCON and/or DPAS;

c. A message interface which will permit failure event selection by a remote computer as an alternative to local selection by a human operator;

d. Internal generation of all the primary and sympathetic TRAMCON and DPAS alarms that would result from the selected failure event(s); and

e. Communication of this alarm information to MITEC in a manner consistent with the way TRAMCON and DPAS would communicate the alarms if they were provided with communication links.

The first increment of TEG will address TRAMCON events. The final delivery of TEG will address both TRAMCON and DPAS events.

G.2. APPLICABLE DOCUMENTS

G.2.1 Government Documents

ASISM 25-50-1, Information Management, DIGITAL SYSTEMS OPERATIONS MANUAL (DSOM) for DEB IIA, Support /Maintenance for DEB IIA Headquarters, U.S. Army Information Systems Command, Fort Huachuca, Arizona, September 1985.

CLIPS Reference Manual, Version 4.3 of CLIPS, Artificial Intelligence Section, Lyndon B. Johnson Space Center, June 1989.

Computer System Operator's Manual for the Transmission Monitor and Control System (TRAMCON), Version 1.8, Command and Control Systems Office, CCSO Information Systems Division, Tinker AFB, Oklahoma, 25 February 1988.

DEB Equipment Troubleshooting Procedures for Equipment Alarms, Det 12, 1945th C.G. Feldberg RRL, Germany Technical Control Operations, 29 June 1989.

DPAS and TRAMCON Interoperability Study, AT&T/Harris DCS System Integration Team, Arlington, Virginia, 05 November 1988.

Giarratano, Joseph C., Ph.D., CLIPS User's Guide, Version 4.3 of CLIPS, Lyndon B. Johnson Space Center, June 1989.

J4AMF/ASF/AST304X0 -033, PDS Code 9DC, Technical Training, TRAMCON-DATALOK 10, FTD 936, Rhein-Main AB, Germany.

Master TRAMCON Alarm Listing for TRAMCON Alarms, Det 12, 1945th C.G. Feldberg RRL, Germany Technical Control Operations, 29 June 1989.

Prime Item Development Specification for the Transmission Monitor and Control System (TRAMCON), CCSO/COI, Tinker AFB, Oklahoma, 05 February 1988.

TRAMCON Alarm Description for TRAMCON Alarms, Det 12, 1945th C.G. Feldberg RRL, Germany Technical Control Operations, 29 June 1989.

TRAMCON On-Line Software Reference Manual, June 1988.

TRAMCON Phase I Baseline, 17 June 1986.

Troubleshooting the D.E.B. Digital Equipment thru TRAMCON Alarms, Det 12, 1945th C.G. Feldberg RRL, Germany Technical Control Operations, 29 June 1989.

G.2.2 Non-Government Documents

365-301-002, AT&T, DACS II Reference Manual, AT&T, May 1987.

365-301-603, AT&T, DACS II Input/Output Message Reference Manual, AT&T, May 1987.

365-301-610, AT&T, DACS II Generic 2 Input/Output Message Manual, AT&T, Dec 1988.

Booch, Grady, Software Engineering with Ada, Second Edition, Benjamin/Cummings Publishing Company, Inc., Menlo Park, California, 1986.

G.3. REQUIREMENTS

G.3.1 Programming Requirements

The following subparagraphs establish the requirements for programming TEG. These subparagraphs establish the language and language processors to be used as well as guidelines to be followed in programming.

TEG is intended to be a knowledge-based system. That is, TEG will represent in declarative style the knowledge of equipment, circuit connections, fault trees, alarms, and polling needed to simulate TRAMCON alarms (Figure 2). This knowledge will be represented in the form of assertions that declare that certain facts and relationships exist, as opposed to the form of

procedures that embody these facts and relationships in their execution.

TEG will employ a relational view of this encoded knowledge, rather than a more general object-oriented view (Figure 3). The relational view provides a means of representing data that is more directly useful in production rules than an object-oriented view would be; also, the relational representation would be more easily stored in and retrieved from a relational database management system (RDBMS), should this become necessary in the future.

TEG will be a rule-based system. That is, TEG will represent the procedures needed to manipulate data and produce alarms as "IF...THEN" rules, also known as production rules. However, unlike many rule-based systems, the particular facts and relationships known to TEG will be represented as assertions, while the rules will operate generally over the asserted data (Figure 4).

TEG will employ both data-driven ('forward chaining') and goal-directed ('backward chaining') formalisms. The majority of TEG execution will be data-driven. The goal-directed formalism will be employed to limit search and computation when this is necessary (Figure 5). Procedural programming will be used to implement external interfaces and to perform housekeeping.

G.3.1.1 Programming Languages

The various functions of TEG shall be programmed in languages suited to their requirements. Requirements and recommendations for each of these languages are given in this paragraph. Use of alternative languages will be permitted upon submission of adequate justification and approval of the Lincoln Laboratory Program Manager.

G.3.1.1.1 Rule-Based Language

All of the TEG simulation shall be programmed in a rule-based language. This language shall fulfill the following requirements:

a. Facts. There shall be a mechanism to assert and retract facts. There shall be a static class of facts that, once read in, remain present. There shall be a dynamic class of facts that may be freely asserted and retracted. There shall be a means to represent facts that is semantically equivalent to a relation.

b. Rules. There shall be a mechanism for the data-driven execution of rules. The rule language shall include a means for selection and binding of data used in rule execution.

The rule language shall include a means to execute both procedures coded in the rule language and procedures coded in an external language. Note: If data-driven execution is provided, there is no need for an additional goal-directed execution mechanism. As discussed in 3.1.3 below, it is practical to emulate goal-directed execution under a data-driven mechanism.

c. Embedding. There shall be a means to embed the rule-based language in a program written in a conventional language and a means to call functions written in a conventional language from the rule-based language.

d. Efficiency. The rule-based language shall employ an algorithm to limit the amount of search needed to select data. This algorithm shall be at least as effective as the widely used Rete algorithm.

e. Portability. Programs written in the rule-based language shall be portable without modification of source code, so long as the target computer provides adequate resources. The minimum scope of portability shall encompass the IBM PC/AT (PC-DOS operating system), the Sun 3 (Berkeley Unix), and the AT&T 3B2 (System V Unix).

f. Multitasking. The rule-based language shall be capable of being executed in a multitasking environment.

The rule-based language for TEG should be CLIPS. CLIPS is deemed to meet all of the above requirements. Alternative rule-based languages may be used subject to justification and approval.

Conventional Language

Those portions of TEG that are not practical to program in a rule-based language shall be programmed in a conventional high-order language (HOL). The following components of TEG are deemed suitable for programming in a conventional HOL:

a. System interface. This component includes procedures to execute and terminate execution of TEG, to interface to system resources required for TEG operation, and to perform housekeeping required for TEG operation.

b. Man-machine interface. This component includes procedures to format and present displays to TEG user and to accept and validate inputs from the user.

c. Database interface. If an RDBMS is used to store and retrieve facts, then the interface between the rule-based language and the RDBMS may be written in a conventional HOL.

d. Multitasking. The HOL shall be capable of being executed in a multitasking environment.

Additional components of TEG may be programmed in a conventional HOL, subject to justification and approval.

The conventional HOL for TEG should be Ada. For the initial implementation of TEG only, the C language may be used without additional justification or approval.

It is believed that assembly language will not be needed to accomplish any function of TEG. Use of assembly language would compromise portability of TEG. Assembly language shall not be used for any programming without justification and approval of the Program Manager. Justification for any use of assembly language shall show why it is impractical to accomplish the programming objective without the use of assembly language.

Database Language

Should it prove necessary to store and retrieve facts from mass storage (as opposed to maintaining all facts in main memory), an RDBMS shall be used to implement the Database function. The RDBMS shall be capable of executing the following functions:

a. Projection. The RDBMS shall be capable of retrieving a table containing a set of selected attributes (a projection). It shall be possible to retrieve a projection that includes data from more than one table (a join). It shall be possible to specify commonly used projections (views).

b. Restriction. The RDBMS shall be capable of retrieving only those data that satisfy a given predicate (a restriction). The RDBMS shall support compound predicates and shall recognize, at a minimum, the following operators: the logical operators AND, OR, and NOT; the numerical relation operators <, <=, =, ¦=, >=, and >; the arithmetic operators +, -, *, /, and mod; and the relation operators = and ¦= for strings.

c. Transaction processing. The RDBMS shall be capable of executing a sequence of operations as a transaction. It shall be capable of updating the database on transaction completion ('commit') and of restoring the database to the pre-transaction state on transaction abort ('rollback'). It shall be capable of establishing checkpoints and of restoring the database to the checkpointed state. An archive/restore mechanism is an acceptable means of implementing the checkpointing requirement.

d. Interface. The RDBMS shall provide a means of interfacing to the selected conventional HOL. This interface should consist of RDBMS functions callable from the conventional HOL.

The RDBMS shall use the language Structured Query Language (SQL). The RDBMS should be one of the following: ORACLE or UNIFY. Another RDBMS may be substituted, subject to justification and approval.

G.3.1.2 Language Processors

Language processors chosen for TEG shall conform to the requirements of 3.1.1 and its subparagraphs above. The language processors shown in Table 1 are recommended; these are deemed to satisfy the requirements of 3.1.1. Alternative language processors may be substituted, subject to justification and approval.

TEG shall be programmed to compile and operate correctly under the language processor versions current at the time of delivery.

Table 1
Language Processors Suitable for TEG

Computer/Operating System				
Target	IBM PC/PC-DOS	Macintosh	Sun 3/Unix	AT&T 3B2/UNIX
Rule Lang Conv. HOL	CLIPS/PC	CLIPS/ ac	CLIPS/Unix	CLIPS/Unix
Ada	TBD Ada	Ada Vanatage	Verdix Ada Ready Systems RTAda	Verdix Ada
C	Microsoft C Turbo C	THINK C	Unix C	Unix C
RDBMS	ORACLE XDB-SQL	ORACLE	UNIFY ORACLE	UNIFY

G.3.1.3 Programming Standards

The programming standards herein established are intended to assure that TEG will conform to the programming model discussed in 3.1 and that TEG will meet its objectives for understandability, maintainability, expandability, reusability, and portability. The following programming standards apply to the programming of TEG:

a. Rule programming style.

i. Programming in the rule language shall employ forward chaining to the extent practical, backward chaining when necessary as an alternative to forward chaining, and procedural code as a last resort.

ii. Use of control facts should be minimized. Generally, control facts that represent goals and subgoals in a backward chaining formalism will be permitted, while other types of control fact should be carefully justified.

iii. Use of salience should be minimized.

iv. If the rule-based language supports deterministic order of rule firing based on the sequence in which the rules are loaded, rule loading order may be used to control execution. This method of controlling execution order should only be used as an alternative to either salience or adding one or more additional rules whose sole purpose is to control the order of execution.

Where rule loading order is critical, the concerned rules must be contiguous within the same file; furthermore, a comment shall be provided for each of the concerned rules stating the required sequencing relationship among these rules.

v. If CLIPS is used for the rule language, the guidelines to programming style in the CLIPS User's Guide should be followed.

b. Conventional HOL programming style. Programming in the conventional HOL shall be in accordance with the commercial practice commonly known as 'structured programming'. Software Engineering with Ada should be used as a guide.

c. System-dependent code. The writing of system-dependent code is discouraged unless it is impractical to accomplish a desired function in a system-independent fashion.

i. If it is necessary to write code that is dependent on particular characteristics of a language processor, operating system, host computer, or peripheral equipment, then this code shall be identified and isolated. Information hiding shall be used to restrict the scope of impact of system-dependent code.

ii. If conditional compilation is available in the language processor(s) used, then this shall be used to ensure that the system-dependent code is compiled only for those systems that require it. In this situation, it is possible (and permissible) that several versions of code specific to particular systems will exist and that only one version will be compiled.

iii. When alternative versions of system-dependent code exist, these shall be grouped together.

iv. It shall be acceptable to use the Unix termcap facility as a mechanism for avoidance of system-dependent programming.

d. Control of recompilation. Recompilation of those software components that require it (because dependent on a changed component) shall be accomplished automatically (as in Ada or Turbo C) or through a makefile (as in Unix C).

e. Self-modifying code. No self-modifying code shall exist in TEG. In the rule-based language, rules shall be considered to be code; facts shall not.

f. Table names. Duplication of table names (e.g. deffacts blocks in CLIPS) shall only be allowable between tables when

the intention is to allow one of the tables to override the other.

g. Modularity. Each functional group of rules, data or procedures shall be contained in a separate file. There shall be a header section for file. The format of these header sections shall be consistent for all such files of the same type, e.g., all rule file headers shall conform to the same style.

h. Readability.

   i. All variable names shall be meaningful.

   ii. Each rule and procedure shall have a comment explaining its function. Any restrictions, or assumptions shall be explained.

   iii. If two or more rules are similar, a comment should be provided with each of these similar rules which distinguishes the purpose of each rule.

   iv. In the case of CLIPS, the wildcard (? or $?) should be used, rather than variable names, to represent items which are don't care in rule patterns. The reason for this is to allow the reader to readily identify the fields that are pertinent to the matching process.

i. Performance.

   i. Any unnecessary clauses inserted for the future or 'just in case' shall be commented out until the need is identified.

   ii. The use of multifield variables shall be minimized. This type of construct shall be permitted only when the number of items cannot be predicted in advance or failure to use this construct will result in the production of additional rules.

   iii. Facts should be cleaned up as soon as they are no longer appropriate.

   iv. Statements whose sole purpose is to test variable values against constants should be avoided if the test(s) can be accomplished within the pattern matching statements (e.g. use of "~" or "&:" construct in CLIPS).

j. Maintainability.

   i. Automatic symbol generation facilities (e.g. the gensym function in CLIPS) shall be used, at most, sparingly.

G.3.2 Design Requirements

G.3.2.1 Sizing and Timing Requirements

The performance of TEG shall be validated against the following timing criteria while operating, on a 3B2 600 or equivalent machine:

a. A response to a request from MITEC shall be ready for transmission to MITEC within forty (40) seconds of receipt from MITEC.

b. Each packet of information shall be transmitted to MITEC within one (1) second of receipt of acknowledgement of the previous packet.

c. A response to a request from the operator shall be presented to the operator within two (2) seconds of receipt.

d. Program start and re-initialization shall be completed within one (1) minute.

TEG shall be capable of running on a 3B2 600 or equivalent machine equipped with a minimum of eight (8) megabytes of main memory. The disk storage requirement currently estimated is five (5) megabytes.

G.3.2.2 Design Standards

Software which is to be implemented using the high-order language shall be developed using structured methodology. Critical design decisions shall be validated by the use of prototyping prior to initiation of software coding activities. Prototyping shall be used to validate the man-machine interface and the feasibility of attaining system sizing and timing requirements.

G.3.2.3 Design Constraints

Fault propagation shall be performed with the use of a forward-chaining rule-based design.

G.3.3 Interface Requirements

G.3.3.1 Interface Relationships

This paragraph discusses the interface relationships between TEG and its interfaces. Figure 6 shows the TEG context diagram. It depicts the major data flows between TEG and its interfaces.

G.3.3.2 Interface Identification

This paragraph specifies the proper identification of each interface:

a. MITEC Interface

b. User Interface

G.3.3.3 Detailed Interface Requirements

TEG Interface to MITEC CSCI

MITEC is an ongoing government-sponsored Air Force project at M.I.T. Lincoln Laboratory to research and develop intelligent computer-controlled support to the area of technical control of government communication facilities. A basic charter of the project is to develop techniques which minimize the human involvement in technical control and maximize the analysis and decision-making by computer software. As such, it is necessary for the computer to obtain directly, i.e., without human involvement, status and alarm information forum communication devices; to insure commands to the devices to effect changes in status, connectivity, etc.; and to obtain measurements of signal strength and quality.

This Section attempts to provide specifications for such computer-to-device communication using ASCII characters over an RS-232 line. It is concerned with the 'syntax' (form) of the communication and does not address the 'semantics' (content) from the point of view of MITEC.

Command - Response Paradigm

Communication between MITEC and TEG shall consist of the following sequence:

a. MITEC issues a 'command' to TEG to supply information. The contents of legal commands shall be specified by TEG and will be generated by MITEC software.

b. TEG responds to the command with exactly one 'response'.

This command-response sequence shall continue as long as MITEC issues commands.

Communication Character Set

All commands from MITEC and responses from TEG shall be within the set of printable ASCII characters, i.e., characters whose values are between octal 41 and 176 (inclusive) plus: tab (11), line-feed (12), carriage-return (15), and space (40).

Characters intended for formatting a terminal display shall not be included in responses to MITEC.

XON-XOFF

TEG shall be capable of accepting any XON-XOFF flow control requests issued by MITEC.

Prompts

All responses from TEG to MITEC shall conclude with a unique string of one or more characters that does not appear in any other context. This string will be referred to as a 'prompt'.

Communication Path Initialization

It shall be possible to initialize the physical characteristics of the communication path (baud-rate, parity, etc.) between TEG and MITEC once during installation of TEG. It shall not be necessary to re-initialize such characteristics after a power off-on sequence, crash, reboot, etc.

Establishing Known Device State

TEG shall be capable of accepting a synchronization string from MITEC. This synchronization string will consist of a finite string of characters. The following types of synchronization strings shall be supported:

a. re-initialize TEG

b. suspend simulation until receipt of next command from MITEC

It shall not be necessary for MITEC to pace itself (e.g. by inserting delays) or to watch characters coming back from TEG to determine if the state has been reached.

Types of Responses

Every response from TEG to MITEC shall be one of the following:

a. Positive confirmation of receipt of the command.

b. Positive confirmation of receipt of the command plus the requested information.

c. A message indicating the unacceptability of any command from MITEC. Unacceptable commands shall be defined as including invalid commands, valid commands which are not valid in the current context, and commands containing transmission errors.

Asynchronous Output from TEG

There shall be no asynchronous output from TEG to MITEC; that is, output not in response to a command from MITEC.

Response After Action

In those cases where a command from MITEC requests TEG to perform an action or change a state, the response shall be produced after the action has been completed.

Large Responses

Large responses shall be partitionable and a mechanism shall be provided for handling such large response. A large response is one in excess of 2048 bytes or 24 lines, including the 'prompt'. TEG shall be capable of accepting requests for a selected portion of the output from MITEC.

a. The initial increment of TEG shall support a 'more' mechanism in which an unambiguous and easily determinable indication is included in the response to show that the response is incomplete and that more information is available. TEG shall provide a command which MITEC may use for obtaining the next portion of the output.

b. If returning alarms by category becomes useful in the future, the ability for MITEC to request a selected category of alarms (e.g., major equipment alarms) will be added.

Subcommands

It shall be possible for MITEC to issue any command together with all of its parameters without entering a 'subcommand' mode in which each parameter is individually prompted for. TEG shall be capable of receiving any command from MITEC with all of its parameters at full input speed (subject to XON/XOFF flow control).

Echo

Future increments of TEG may support a command for turning on or off the echo of input characters that appear in commands from MITEC. The default mode for this characteristic shall be 'echo off'.

No Password Protection

Entry of a password shall not be required in order to execute TEG.

TEG User Interface

The first increment of TEG shall support a user-friendly menu-driven (keyboard) interface. Future increments of TEG may additionally support a scripting interface that allows for batch processing.

The keyboard user interface shall not require the user to learn formatting and syntax rules. For the first increment of TEG, the User Interface to TEG may be a set of menus driven by the rule-based language. If the first phase is developed using CLIPS, the user interface inputs shall include a batch file for the TRAMCON segment that is modelled in the TEG database. This batch file shall include commands for clearing memory, loading all rules and static facts associated with the TRAMCON segment, and performing the commands that cause a re-initialization of the rule-based environment (e.g., CLIPS reset).

Menus that allow the operator to select from a set of various functions will provide a consistent set of options with a visual cue to allow the operator to differentiate between available and unavailable options based on the current state of the simulator.

Selection of any function which requires selection of additional criteria will result in the presentation of sub-menus allowing the user to either select from the set of applicable values or respond to a prompt with a brief textual response. For a particular function, these menus and prompts will be presented in a consistent sequence. When a particular sub-menu or prompt is not applicable based on previous selections, that sub-menu or prompt will not be presented.

All user responses shall be validated. Selection of an unavailable option shall be considered an invalid user response. An invalid user response shall never cause the simulation to terminate prematurely or otherwise behave in an abnormal fashion. Because the required user responses shall be brief and simple, and the set of valid user responses and valid formats shall always be clearly stated on the menu or prompt, a user shall

always be given an indefinite amount of attempts to re-try after entering an invalid response.

An escape mechanism will be provided to allow the user to cancel a function selection prior to completing selection from all sub-menus associated with that function. When this escape mechanism is invoked, control will be returned to the menu from which the function was originally selected.

The user interface associated with future increments of TEG shall provide a means for selection of TRAMCON segments, and the loading and saving of scenario files.

TEG Interface to MITEC Network HWCI

TEG will be connected to MITEC using an RS-232 connection.

G.3.4 Functional and Performance Requirements

TEG shall simulate a single TRAMCON segment at a time.

G.3.4.1 Simulation Input Function

The Simulation Input function shall be responsible for receiving and pre-processing all inputs to TEG.

Simulation Input Function Inputs

Inputs to the Simulation Input function shall include the following:

a. Keyboard inputs from the operator

b. TEG knowledge base

c. Commands from scenario files

d. Messages from external programs and/or devices

Simulation Input Function Processing

The keyboard interface shall consist of a hierarchy of menus displayed appropriately for obtaining direction from the operator. Where appropriate, text, rather than a menu selection, is the expected response from the user.

The main simulation functional menu shall include, at a minimum, the following options:

a. modify polling sequence

b. change automatic switchover state

c. change operational side

d. insert a fault

e. remove a fault

f. poll

g. run simulation

h. exit simulation

Each of the above options, except for the exit simulation option, shall allow the user to specify an associated time. Time will be accepted from the user in hh:mm:ss format. A default time shall also be provided by TEG. An example of default time is the last time entered by the user.

Selection of the modify polling sequence option shall cause the current polling sequence to be displayed and allow the user to specify the exact order of polling of the TRAMCON sites within the selected TRAMCON segment. The user shall have the capability to specify any permutation of these sites. Any site(s) may be omitted from the polling sequence. For example, if the available sites are BST, HST, and RAG, any of the following sequences may be specified:

BST, HST, RAG	BST, RAG, HST	HST, BST, RAG	HST, RAG, BST
RAG, HST, BST	RAG, BST, HST	BST, HST	BST, RAG
HST, RAG	HST, BST	RAG, HST	RAG, BST
RAG HST	BST	none	

After a complete valid user response is entered, it will be forwarded to the applicable function(s) for further processing.

Selection of the change auto switchover state option shall cause a menu of sites to be displayed. After a valid user response is entered, a menu of equipment located at the selected site that has more than one redundant side will be displayed. After a valid user response is entered, a menu of valid switchover states will be displayed. After a valid user response is entered, the database shall be updated to reflect the new auto switchover state.

Selection of the change operational side option shall cause a menu of sites to be displayed. After a valid user response is entered, a menu of equipment located at the selected site that has more than one redundant side will be displayed. After a valid user response is entered, a menu of all sides applicable to this piece of equipment will be displayed. After a valid user response is entered, a primary fault event will be created for processing by the Event Generation function (3.4.2).

Selection of the insert fault option shall cause a menu of sites to be displayed. After a valid user response is entered, a menu of equipment located at the selected site will be displayed. After a valid user response is entered, a menu of primary faults applicable to the selected equipment will be displayed. After a valid user response is entered, a menu of ports applicable to the selected equipment will be displayed. After a valid user response is entered, a menu of sides will be displayed if the selected equipment has redundant sides. After a valid user response is entered, the complete request will be forwarded to the applicable function(s) for further processing.

Selection of the remove fault option shall allow the user to select a fault for removal from the set of all faults that were directly inserted by the operator and have not yet been removed by the operator. The user shall be required to provide a repair time.

Selection of the poll option shall allow the user to specify one of the following types of polling:

a. Poll once at an absolute simulation time

b. Poll once at a relative simulation time

c. Poll n times starting at an absolute or relative simulation time at a specified or default polling frequency

d. Poll indefinitely starting at an absolute or relative simulation time at a specified or default polling frequency

e. Stop indefinite polling at an absolute or relative simulation time

Selection of the run simulation option shall cause the simulation to run for the specified period of simulation time.

Selection of the exit simulation option shall cause return of control to the operating system.

Selection of any of the above options, except exit simulation, shall cause the simulation menu to be redisplayed after completion of execution of the selected option.

In addition to the above options, the following functions will also be supported from the keyboard interface:

a. select TRAMCON segment

b. load scenario file

c. save scenario file

Selection of the select TRAMCON segment option will cause a menu of segments available within the TEG Knowledge Base to be displayed. After a valid user response is entered, the current segment will be set to the selected segment.

Selection of the load scenario file option will cause a list of all available scenario files to be presented to the user. The user will be capable of selecting a file (or no files) from this list. If a file is selected, the dynamic facts within that file will be loaded.

Selection of the save scenario file option will cause the save state to be toggled. The initial save state will be off. When the state is toggled on, the user will be prompted to specify a file name for the newly created scenario file. All selections made by the user between the time that the state is toggled on and the time that the state is toggled off will be saved within that file.

Once all required user inputs have been collected for a given option, the assembled inputs shall be validated to ensure that the entire request is valid. Valid requests shall then be forwarded to the appropriate function(s) for further processing.

The scripting interface will support the full functionality provided by the keyboard interface.

Messages from MITEC shall be validated and processed as if they had been entered by the user; except, in the case of an error in a message from MITEC, the simulator shall output an error message to MITEC and prepare to receive another message. The simulator shall not output menus to MITEC.

Simulation Input Function Outputs

The outputs of the Simulation Input function shall include:

a. menus

b. prompts

c. validated simulation requests

d. error messages to MITEC

e. message acknowledgements to MITEC

G.3.4.2 Event Generation Function

TEG shall support the following types of events:

a. equipment state transitions, resulting from primary faults or sympathetic faults

b. primary faults, resulting only from the insert fault option (3.4.1.2)

c. sympathetic faults, resulting from primary or other sympathetic faults

d. alarms, resulting from primary faults, sympathetic faults, or other alarms

Each possible alarm and sympathetic fault will have exactly one of the following causes:

a. the alarm/sympathetic fault can be caused by a single alarm/fault

b. the alarm/sympathetic fault can be caused by the presence of one or more of a set of causing alarms/faults (i.e., an OR condition)

c. the alarm/sympathetic fault can be caused by a combination of faults/alarms, all of which must exist for the alarm/sympathetic fault to exist (i.e., an AND condition)

Removal of an alarm/fault shall result in the following:

a. removal of all alarms/sympathetic faults caused by this alarm/fault alone

b. removal of all alarms/sympathetic faults which can be caused by this alarm/fault or other alarms/faults (OR condition), when no other causing alarms/faults exist

c. removal of all alarms/sympathetic faults which can only be caused by this alarm/fault in conjunction with other alarms/faults (AND condition), regardless of whether or not any such other alarms/faults exist

If the operational side of the applicable device is faulted and the non-operational side is not faulted, modification of the switchover status enabling automatic switchover results in switchover of the device to the other side.

If the non-operational side of the applicable device is not faulted, a manual switchover request results in switchover of the device to the other side; otherwise, manual switchover requests are ignored.

Either automatic or manual switchover of a device results in removal of any alarms/faults uniquely associated with the previous operational side.

Event Generation Function Inputs

The inputs to the Event Generation function shall include operator requests to add or remove events as well as the Event, Causality, Connectivity, Equipment, Equipment Status, Port, and Side tables. The fault constellation is also an input to the Event Generation function.

Event Generation Function Processing

This function shall produce and retract sympathetic faults, alarms, and equipment status. A sympathetic fault is a fault that is caused by a primary fault (a fault input by the user through the Simulation Input function, 3.4.1) or another sympathetic fault. When a fault (or occasionally a logical combination of faults) that is monitored at the Datalok device or TRAMCON occurs, an alarm results. Equipment status consists of various indicators that are monitored but are not considered alarms because they may arise in normal operation; equipment status may be input by the operator or change in response to the occurrence of faults.

Fault Propagation

The Event Generation function shall produce the entire constellation of sympathetic faults for one or more given primary faults. The process by which sympathetic faults are produced is called fault propagation. The Event Generation function shall carry out fault propagation according to the causality relationships in the CAUSES table (3.4.5.3.2.4.2). Fault causality relationships exist within equipment (CAUSES SAME in the CAUSES table) and between items of equipment (CAUSES SENDER, DISTANT, or LINK-END). The Event Generation function shall propagate faults for all of these causality relationships:

a. Fault propagation within a device (SAME relationships) shall occur whether or not the device (or the side of the fault in a device with redundancy) is on-line. Fault propagation between devices shall occur only between on-line devices (or between on-line sides in devices with redundancy). On-line status shall be determined from the Equipment Status (EQSTATUS, 3.4.5.3.1.4) table.

b. Fault propagation from sender to receiver (SENDER relationships) shall occur when there is a direct connection from the sender to the receiver. The

presence of a direct connection shall be determined from the Connection (CONN, 3.4.5.3.2.1) table.

c. Fault propagation to the distant end (DISTANT relationships) shall occur when there is a connection (which may be direct or indirect) from a sender to its distant-end counterpart receiver. Connection tracing (3.4.2.2.2) shall be used to determine the distant end equipment.

d. Fault propagation to the opposite link end (LINK-END relationships) shall occur when the sending equipment and receiving equipment are the link ends of a circuit. The opposite link-end shall be determined from the Circuit (CKT, 3.4.5.3.2.3.1) table.

Fault propagation within an item of equipment is complex and allows for many combinations. The Event Generation function shall propagate faults for all of the following classes of causality relationship:

a. Within a port.

b. From a near port to the far port.

c. From the far port to all near ports.

d. To all ports on the equipment.

The Event Generation function shall support OR-causality of faults. In OR-causality, a fault shall occur when at least one of the causes of the fault has occurred.

The Event Generation function shall time-stamp faults. The time of a fault shall be the time at which the cause of the fault occurred. If the fault has more than one cause, the time shall be the time of the first cause to occur.

Connection Tracing

When there is a fault on a device that causes a fault on its distant-end counterpart, connection tracing shall be used to determine the distant-end counterpart device. Connection tracing shall proceed from the location of the fault in the direction of signal travel until the corresponding port on a device of the same class has been reached. The direction of signal travel shall be assumed as follows:

a. For a fault originating on a near port, first to the far port of the same device and thence along the connection of the far port.

b. For a fault originating on a far port, along the connection of the far port.

The corresponding port on the distant-end device shall be determined as follows:

a. For a fault originating on a near port, any near port (not necessarily the same near port) of the first device of the same class encountered.

b. For a fault originating on a far port, the far port of the first device of the same class encountered.

Alarm Generation

Alarms shall be generated in the same manner as fault propagation: when a primary fault, sympathetic fault, or other alarm that is the cause of an alarm occurs, the alarm shall be generated. Alarm generation is simpler than fault propagation in that the alarm is always raised on the equipment on which the fault exists (that is, all alarm causality relationships are of the form CAUSES SAME).

The Event Generation function shall support OR-causality and AND-causality for alarms.

a. OR-caused alarms shall be generated when one or more of the causes of the alarm exist.

b. AND-caused alarms shall be generated when all of the causes of the alarm exist.

Alarms shall be time-stamped with the time of occurrence of the event or events that caused the alarm. In the case of OR-caused alarms, the time of the first-occurring cause shall be the time of the alarm. In the case of AND-caused alarms, the time of the last-occurring cause shall be the time of the alarm.

Event Removal

The Event Generation function shall remove events (both faults and alarms) for which the cause of the event has been removed (whether due to switchover (3.4.2.2.5) or due to removal by the user through the Simulation Input function (3.4.1)). Event removal shall be immediate when the cause of the event is removed.

a. For faults and OR-caused alarms with more than one cause, the event shall be removed when all of the causes have been removed.

b. For AND-caused alarms, the event shall be removed when any of the causes has been removed.

In order to perform event removal, the Event Generation function shall maintain, for every caused event, the primary key of every cause of that event. (The primary key is the fault (or alarm) symbol and the device, port, and side on which the fault or alarm occurred.)

Switchover

Certain equipment types known to TEG employ redundancy. These types may be identified by the presence of more than one entry in the Side table (3.4.5.3.1.3). In these types of equipment, only one redundant side is on-line at any time. Since only the on-line side is in communication with other equipment, fault propagation between equipment occurs only between the on-line sides. Therefore, switchover between sides will affect fault propagation, alarm generation, and event retraction. The Event Generation function shall execute automatic switchover and shall account for switchover in the constellation of generated events as follows:

a. When there is a fault on the on-line side of a device with redundancy and automatic switchover is enabled for that device, the Event Generation function shall:

   i. Make the current side of the device off-line.

   ii. Make the previously off-line side on-line.

   iii. Disable automatic switchover on that device until reenabled by the user.

b. When there is a switchover (whether automatic or manual), the Event Generation function shall:

   i. Remove all faults that had propagated from other devices to the previously on-line side and propagate these to the new on-line side.

   ii. Remove all faults that had propagated from the previously on-line side to other devices.

   iii. Propagate all previously existing and newly propagated faults from the new on-line side within the device and to other devices.

Event Generation Function Outputs

The outputs of the Event Generation Function shall include the fault constellation and the alarm constellation.

G.3.4.3 Polling Simulation Function

The Polling Simulation Function shall simulate TRAMCON polling. TRAMCON polls one site and waits for the response, which takes approximately six (6) seconds, before polling the next site. If a response to a poll is not received within a ten-second time-out period, TRAMCON times out and continues polling the next site in the sequence.

Polling Simulation Function Inputs

The inputs to the Polling Simulation function shall include information from the Simulation Input function reflecting polling sequence and poll selections made by the user and faults and alarms output by the Event Generation function. This function shall also use information included in the Equipment, Connection, and Causality tables as inputs.

Polling Simulation Function Processing

The Polling Simulation Function shall simulate polling of each site included in the specified polling sequence. Polling shall occur in the same sequence as specified in the polling sequence. The starting time, number of times each site is polled, and polling interval shall be as specified in the polling selection. The poll selections made by the user are described as overall poll requests. This function shall interpret each overall poll request and convert it to individual specific poll requests. For example, an overall poll request specifying that each site is to be polled four times will be converted to four specific poll requests.

Changes in polling sequence or frequency shall take effect at the specified time unless polling of a site is in progress at that time, in which case the changes will take effect immediately following completion of polling at that site.

The poll time shall be incremented by six seconds each time another site is polled. This function shall detect a no response situation based on faults specified in the TRAMCON equipment. When a no response situation is detected, the poll time shall be incremented by ten seconds when the next site is polled.

This function shall examine the alarm constellation and shall select those alarms which would be received in response to an actual TRAMCON poll. Selection criteria for output shall include:

a. the fault responsible for the alarm must be in existence at the time the poll is made (i.e., must have already been inserted and not yet repaired).

b. the alarm must either be a Datalok 10 alarm that is transmitted to TRAMCON or an alarm derived by the TRAMCON system.

c. for Datalok 10 alarms, the TRAMCON equipment connecting the site at which the appropriate Datalok 10 device (i.e., the Datalok 10 which detects the fault that causes the alarm) is located to the site at which the TRAMCON Master is located must not contain any faults that would result in inability of either the poll request to be received at the Datalok 10 or the poll response to be received at the Master.

This function shall determine, for each poll handshake, whether or not the poll communication can be successful.

Polling Simulation Function Outputs

The outputs of the Polling Simulation function shall include a list of all TRAMCON alarms generated that correspond to a particular poll, in the form of poll responses. The output for each fault shall include complete information identifying the fault (e.g., equipment ID, port, side), the time at which the alarm was detected by the poll, and a textual description of the alarm.

G.3.4.4 Output Control Function

Output Control Function Inputs

The inputs to the Output Control function shall include valid requests from external programs (e.g., MITEC) and poll responses provided by the Polling Simulation function.

Output Control Function Processing

The Output Control function shall interpret requests, assemble the poll responses into the requested report, and output the requested report in accordance with the protocol specified in 3.3.3.1.

Output Control Function Outputs

The outputs of the Output Control function shall include a list of all TRAMCON faults and alarms generated from the most recent poll in a format that conforms to the TEG to MITEC CSCI interface requirements (paragraph 3.3.3.1). Each of these outputs shall be one of the following types, depending upon the request received from MITEC:

a. Summary - 1 line / polled site

b. Detailed - 1 line/alarm at site specified in MITEC request

Examples of summary and detailed reports are shown in Figures 7 and 8. Future increments of TEG will support output of DPAS alarms.

G.3.4.5 Database Function

The TRAMCON Event Generator Knowledge Base defines the data tables used to store the representations of equipment, interconnections, circuits, faults, alarms, and site locations known to TEG.

The Knowledge Base defines both tables that are specific for each unique TRAMCON segment as well as tables that are applicable to all TRAMCON segments. These two sets will be known as the segment-specific knowledge base and the non-segment-specific knowledge base, respectively. Each separate segment-specific knowledge base and the non-segment-specific knowledge base shall be contained in separately loadable files. The first increment of TEG shall model the DEB IIA TRAMCON segment. Future increments of TEG shall support the capability to model any TRAMCON segment.

Database Function Inputs

For the first phase of TEG development, inputs to the database function shall be made through a text editor. Database function inputs for future phases of TEG are TBD.

Database Function Processing

Database processing in the first increment of TEG shall provide the functions to enter, edit, and delete individual facts and blocks of facts. If the rule language requires facts to be "loaded" or "compiled" the database function shall provide the requisite processing. Database function processing for future phases of TEG is TBD.

Database Function Outputs

The outputs of the Database Function shall include all tables that constitute the TEG Knowledge Base. The TEG Knowledge Base shall include the tables used to store the representations of equipment, interconnections, circuits, faults, alarms, and site locations known to TEG.

Equipment Representation

The following tables shall be used to define the representation of equipment known to TEG, aside from their

interconnections: the Equipment, Port, Side, and Equipment Status tables. Additional tables required to define the attributes of a equipment may be specified at a later time.

Equipment Table

The Equipment table shall define the static attributes of each item of equipment known to TEG. The Equipment table shall be a part of the segment-specific knowledge base. Equipment shall include multiplexers, encryption equipment, and radios; this list may be extended as needed in the future. Equipment shall not include the transmission medium, e.g. wires, patch panels, or radio links. Equipment table records may not be retracted by other functions of TEG. The Equipment table shall consist of one record for each item of equipment; each record will contain the fields defined in the subparagraphs below.

Equipment Table ID

This field will contain the symbol EQUIP. It shall identify all Equipment table records and distinguish these from all other types of record.

Equipment Class

This field shall contain a symbol that identifies the class of equipment represented by the record, e.g. FCC-99 or FRC-171.

Equipment ID

This field shall contain a symbol that identifies the item of equipment represented by the record. This symbol will be the primary key of the Equipment table; therefore, it will uniquely identify the item of equipment. In the simulation of a TRAMCON segment for which the equipment nomenclature is known, the symbol shall be the actual name of the item of equipment; otherwise, it shall be a meaningful and unique name.

Equipment Location

This field shall contain a symbol that identifies the facility at which the equipment is located. The set of symbols used in this field will be the set of three-letter facility IDs (e.g. DON for Donnersberg) used in the DEB.

Equipment Pretty Name

This field shall contain a string that describes the item of equipment. This string will be for use by display and report software that needs a suitable print name for the item.

Port Table

The Port table shall define the port symbols that are used for each class of equipment. The Port table shall be a part of the non-segment-specific knowledge base. There shall be at least one Port table record for each class of equipment, but most classes will have two or more records. Port table records may not be retracted by other functions of TEG. Each record of the Port table will consist of the fields defined in the subparagraphs below. Note that the primary key of the Port table consists of both the Equipment Class and the Port Symbol fields.

Port Table ID

This field will contain the symbol PORT. It shall identify all Port table records and distinguish these from all other types of record.

Equipment Class

This field shall contain a symbol that identifies the class of equipment that possesses the port defined by this record; for example, FRC-171. The set of symbols used in this field shall be the same as the set of symbols used in the Equipment Class field of the Equipment table (3.4.5.3.1.1.2).

Port Symbol

This field shall contain a symbol that is the name of a port on the subject class of equipment. The symbol FAR is distinguished: there shall be at most one 'far' port record for each class of equipment. Any symbol other than FAR is the name of a 'near' port. For example, the FRC-171 radio has the ports MBS-1, MBS-2, SCBS, and FAR.

Equipment that is always a data source or sink to the TRAMCON-monitored system shall have just one Port table record: the 'far' port record. The FCC-98 and CY-104A are examples of such equipment. Other equipment classes shall have two or more Port table records.

The numbering of 'near' ports shall follow the Tech Control practice for each class of equipment. When this is not known, the 'near' ports will be numbered from 1 through N, where N is the total number of 'near' ports. For non-multiplex equipment, such as the KG-81, the symbol for the 'near' port may be NEAR. The symbol NEAR shall not be used as a port symbol for multiplex equipment.

The convention for the distinction between 'near' and 'far' ports is that the 'near' ports are those conceptually 'closer' to

the end user or the abstract 'center' of a through facility; the 'far' ports are those 'closer' to the transmission medium between facilities. The demultiplexed ports of a multiplexer are 'near'; the multiplexed port is 'far'. The unencrypted ('clear' or 'black') port of a KG-81 is 'near'; the encrypted ('red') port is 'far'. This distinction may not survive for matrix equipment such as DACS II and may need to be revisited when it becomes necessary to model such equipment.

Side Table

The Side table shall define the side symbols and automatic switchover attributes that are applicable for each class of equipment. The Side table shall be a part of the non-segment-specific knowledge base. There shall be at least one Side table record for each class of equipment, but some classes will have two records. Side table records may not be retracted by other functions of TEG. Each record of the Side table will consist of the fields defined in the subparagraphs below. Note that the primary key of the Side table consists of both the Equipment Class and the Side Symbol fields.

Side Table ID

This field will contain the symbol SIDE. It shall identify all Side table records and distinguish these from all other types of record.

Equipment Class

This field shall contain a symbol that identifies the class of equipment that possesses the side defined by this record; for example, FRC-171.

Side Symbol

This field shall contain a symbol that is the name of a side on the subject class of equipment. The set of legal symbols is: Only, A, B. Only will be used to represent classes of equipment that have only one side (i.e., no manufactured redundancy). For example, the FCC-98 second level multiplexer has Only one side. A and B will be used to represent classes of equipment possessing manufactured redundancy. For example, the FRC-171 radio has the sides A and B.

Switchover Attributes

This field shall contain a symbol that indicates the automatic switchover attribute for the subject class of equipment. The set of legal symbols is: ALWAYS, TOGGLE, UNKNOWN, NEVER. ALWAYS is used to indicate that automatic switchover is always enabled for the subject class of equipment.

TOGGLE is used to indicate that the switchover state may be toggled, via operator input, between automatic and manual switchover. UNKNOWN is used to indicate that the switchover attribute is unknown. NEVER is used to indicate that switchover never occurs. This symbol should be used with all equipment that only has one side.

Equipment Status Table

The Equipment Status table shall define the modifiable attributes of each item of equipment known to TEG. Equipment shall include all items contained in the Equipment table. The Equipment Status table shall be a part of the segment-specific knowledge base. The Equipment Status table shall consist of one record for each item of equipment. Equipment Status records may be retracted by other TEG functions. Each record will contain the fields defined in the subparagraphs below.

Equipment Status Table ID

This field will contain the symbol EQSTATUS. It shall identify all Equipment Status table records and distinguish these from all other types of record.

Equipment ID

This field shall contain a symbol that identifies the item of equipment represented by the record. This symbol will be the primary key of the Equipment Status table; therefore, it will uniquely identify the item of equipment. The symbols shall match those used in the Equipment ID field of the Equipment table.

Operational Side

This field shall contain a symbol that identifies the side on which the equipment specified in the Equipment ID field is currently operating. The symbols shall match those used in the Side Symbol field of the Side table. Until contradictory information is provided, the initial operational side for redundant devices will be assigned the symbol A.

Automatic Switchover State

This field shall contain a symbol that indicates whether or not automatic switchover is currently enabled for the device specified in the Equipment ID field. The set of symbols will be: AS-ENAB, AS-DISAB, to represent auto-switchover enabled and auto-switchover disabled, respectively. Until contradictory information is provided, the initial automatic switchover state for all devices will be assigned the symbol AS-ENAB.

Connectivity Representation

The following tables shall be used to define the interconnections of equipment known to TEG: the Connection table and the Link table. Additional tables required to define attributes of connectivity may be specified at a later time.

Connection Table

The Connection table shall define the interconnections between items of equipment; it shall also provide a 'hook' for the future representation of patch panels, patches, and matrix switching. The Connection table shall be a part of the segment-specific knowledge base. There shall be at most two (and usually one) Connection table records for each pair of connected ports known to TEG; the Connection table record shall denote that a connection (whether in one or both directions) exists between these ports. This implies that for a duplex connection there will be one Connection Table record, and both the sender and receiver fields will denote ports capable both of sending and of receiving data. Two Connection table records may be used to denote a duplex connection in which the connection paths actually differ, if this should be necessary.

Each record of the Connection table will consist of the fields defined in the subparagraphs below. Note that the primary key of the Connection table consists of all of these fields: Connection Status, Sending Equipment ID, and Sending Port; the combination Connection Status, Receiving Equipment ID, and Receiving Port is also a candidate key and may be used as such.

The Connection table shall also identify ports that are either not connected or are connected to equipment not modeled in TEG. The Equipment ID symbols described under 3.4.5.3.2.1.4 shall be used for this purpose. Note that the use of one of these symbols as the Sending Equipment ID specifies the connection status of the corresponding Receiving Equipment ID and Port, while the use of one of these symbols as the Receiving Equipment ID specifies the connection status of the corresponding Sending Equipment ID and Port. It would be meaningless for both the Sending and Receiving Equipment ID fields to contain one of these symbols.

Connection Table ID

This field will contain the symbol CONN. It shall identify all Connection table records and distinguish these from all other types of records.

Connection Status

This field shall contain a symbol to represent the status of the connection. The keywords for this field are NOMINAL and ACTUAL. A connection with 'nominal' connection status shall be the connection as specified in the TSO, multiplex plan, or other appropriate source of information as to the intended status of the connection in normal operation. A connection with 'actual' connection status shall be the connection as established by a patch, cross-connect, or other variation from the normal connection. The initial increment of TEG shall not be required to support the connection status of ACTUAL.

The concept of 'nominal' vs. 'actual' connection status serves to distinguish the original state of the network from a state that currently exists due to the presence of a patch. While patches may be used as temporary workarounds in operation, they are generally removed as soon as the failed equipment has been repaired and is placed back in service. Therefore it is necessary to retain the nominal state of the network in order to restore this state following the removal of a patch.

In database search, software that is concerned with the network as it is presently constituted (for example, in fault propagation) should first attempt to bind an 'actual' connection for a given sender or receiver; if this attempt fails, it should then attempt to bind a 'nominal' connection.

Duplexity

This field shall contain a symbol that denotes whether the connection is one-way (simplex) or two-way (duplex). The symbols SIMPLEX and DUPLEX will be used to represent simplex and duplex connections, respectively. In the case of a 'duplex' connection, the Sending Equipment ID and Sending Port fields may refer to either of the ports participating in the connection: the assignment of 'sender' and 'receiver' may be arbitrary. Consequently, software that is searching for the connection of a given port should be capable of selecting the connection record whether the given port is the 'sender' or the 'receiver'.

Sending Equipment ID

This field shall contain the Equipment ID symbol of the equipment participating as sender in the connection. The Equipment ID symbol will be either a unique Equipment ID as defined in 3.4.5.3.1.1.1 or any of the following keywords: SPARE, UNUSED, FAILED, or UNKNOWN. The initial increment of TEG shall not be required to support the symbols SPARE, FAILED, or UNKNOWN. The interpretation of these keywords is as follows:

The use of SPARE as a Sending Equipment ID will identify a receiver that is designated for use as a spare in patching.

The use of UNUSED as a Sending Equipment ID will identify a receiver that is not used for any purpose (and presumably could be used as a spare). The distinction between SPARE and UNUSED is thought to be useful because the sparing and service restoration algorithms of MITEC make use of this distinction.

The use of FAILED as a Sending Equipment ID will identify a receiver that is not used because it has failed and has been removed from service. This is only possible for Connection records of ACTUAL status.

The use of UNKNOWN as a Sending Equipment ID will identify a receiver that is connected to equipment not represented in this database.

Sending Port

This field shall contain the Port symbol of the port participating as sender in the connection. The Port symbol shall be as defined above. When the Sending Equipment ID is any of the keywords SPARE, UNUSED, FAILED, or UNKNOWN, the Sending Port symbol will be the same as the Receiving Port symbol.

Receiving Equipment ID

This field shall contain the Equipment ID symbol of the equipment participating as receiver in the connection. The Equipment ID symbol shall be as defined above or may be any of the following keywords: SPARE, UNUSED, FAILED, or UNKNOWN. The interpretation of these keywords is as defined above except these symbols will here identify the receiver.

Receiving Port

This field shall contain the Port symbol of the port participating as receiver in the connection. The Port symbol shall be as defined above. When the Receiving Equipment ID is any of the keywords SPARE, UNUSED, FAILED, or UNKNOWN, the Receiving Port symbol will be the same as the Sending Port symbol.

Connection Medium

This field shall identify the type of connection that exists between the sending and receiving ports. The set of connection types that shall be supported by the initial increment of TEG shall be WIRE, MICRO, and NONE. The set of connection types that will be considered for support in future increments of TEG includes JACK, PATCH, TEST, TROPO, CABLE, SATCOM, and MATRIX.

The matter of defining connection medium types is not settled, and considerable change in this area may be expected.

WIRE will be used to represent a hard-wired connection between the sending and receiving ports. In the initial increment of TEG, which will not support patching, this type shall subsume the types JACK, PATCH, and TEST.

MICRO will be used to represent a microwave radio connection between the sending and receiving ports.

NONE will be used to represent a port that is not connected to anything. Some ports that are actually disconnected or connected through a medium unknown to TEG may use this symbol.

JACK will be used to represent a connection made through a patch panel jack. JACK connections shall be restricted to those made through a set of jacks in their normal (i.e. closed) configuration; connections that involve an open jack shall be of type 'patch'.

PATCH will be used to represent a connection made through a patch panel using a patch cord or plug.

TEST will be used to represent a connection made through a test and access point.

TROPO will be used to represent a troposcatter radio connection between the sending and receiving ports.

CABLE will be used to represent a fiber-optic or wire land line or submarine cable connection between the sending and receiving ports.

SATCOM will be used to represent a communications satellite connection between the sending and receiving ports.
MATRIX will be used to represent a connection through a matrix switch (e.g. a DACS II) between the sending and receiving ports.

Connection ID

This field shall be a symbol that together with the Connection Medium symbol identifies the individual connection between the sender and receiver. The set of legal values for this symbol depends on the value of the Connection Medium symbol. With the exception of NONE, the Connection Medium and Connection ID fields, taken together, correspond to the primary key of the Link table.

For a Connection Medium value of NONE, the Connection ID will also be NONE.

For a Connection Medium value of WIRE, the Connection ID may be NONE, or it may be some symbol identifying the particular wire. If it is desired to simulate a fault involving the wire itself, the Connection ID should not be NONE. If not NONE, the symbol will correspond to a record in the Link table for that wire.

For a Connection Medium value of MICRO, the Connection ID will be a symbol identifying the particular microwave link. This symbol shall correspond to a record in the Link table for that microwave link.

Connection ID symbols for other values of Connection Medium are TBD.

Link Table

The Link table shall provide a 'hook' for simulating failures on transmission links and (in later increments of TEG) for implementing patches, cross-connects, and tests. The Link table shall be a part of the segment-specific knowledge base. Link table records may not be retracted by other TEG functions. This table will consist of the following fields:

Link Table ID

This field will contain the symbol LINK. It shall identify all Link table records and distinguish these from all other types of record.

Link Class

This field shall contain a Link Class symbol. This will be any of the symbols defined for Connection Medium (3.4.5.3.2.1.8), except NONE.

Link ID

This field shall contain a Link ID symbol. This shall correspond to a symbol used in a Connection ID field (3.4.5.3.2.1.9).

Link Pretty Name

This field shall contain a string that describes the specified link. Software that needs a print name for the link will use this field.

Circuit Representation

The following tables shall be used to define the representation of circuits known to the TEG: the Circuit table.

Circuit Table

The Circuit table shall define each circuit known to TEG. The Circuit table shall be a part of the segment-specific knowledge base. Since the connectivity of all equipment is fully specified by the Equipment, Port, Connection, and Link tables, the only additional information needed to define the circuit is the end points of the circuit. For future extensions of TEG, it may be necessary to define additional characteristics of the circuit that will be stored in this table. Circuit table records may not be retracted by other TEG functions. The Circuit table will consist of the following fields:

Circuit Table ID

This field will contain the symbol CKT. It shall identify all Circuit table records and distinguish these from all other types of record.

Circuit ID

This field shall contain a symbol that is the unique identifier of the circuit. This field will be the primary key of the Circuit table. For most circuits known to TEG, this will be the CCSD used in Tech Control.

Origin Equipment ID

This field shall contain a symbol that is the unique identifier of the equipment where the circuit originates. This symbol shall be an Equipment ID and correspond to a record in the Equipment table. (For a unidirectional, or simplex, circuit, the origin is the sending end. For a bidirectional, or duplex, circuit, the choice of origin is arbitrary; the preferred choice is the end that appears at the top or left-hand side of a multiplex plan or other appropriate source.) For trunk circuits, which are the only kind to be implemented in the initial increment of TEG, the Origin Equipment ID should be a multiplexer, the 'far' port of which is the origin of the trunk.

Origin Port

This field shall contain a symbol that identifies the port of the equipment where the circuit originates. This symbol shall be a Port symbol valid for the class of the origin equipment. For trunk circuits, the Origin Port should normally be FAR, specifying the 'far' port of the originating multiplexer.

Destination Equipment ID

This field shall contain a symbol that is the unique identifier of the equipment where the circuit terminates. This

symbol shall be an Equipment ID and correspond to a record in the Equipment table. For trunk circuits, which are the only kind to be implemented in the initial increment of TEG, the Destination Equipment ID should be a multiplexer, the 'far' port of which is the destination of the trunk.

Destination Port

This field shall contain a symbol that identifies the port of the equipment where the circuit terminates. This symbol shall be a Port symbol valid for the class of the destination equipment. For trunk circuits, the Destination Port should normally be FAR, specifying the 'far' port of the destination multiplexer.

Circuit Pretty Name

This field shall contain a string that provides a description of the circuit. Software that needs a print name for the circuit should use this field.

Fault Representation

The following tables shall be used to define the representation of faults known to TEG: the Event table and the Causality table. Additional tables required to define the attributes of faults may be specified at a later time.

Event Table

The Event table shall define the attributes of each possible status, failure and alarm recognizable by TEG. The Event table shall be a part of the non-segment-specific knowledge base. Failures shall include both primary and secondary (also known as sympathetic) failures. Alarms shall include all alarms detectable by TEG resulting from primary and secondary failures. Event table records may not be retracted by other TEG functions. The Equipment Class and Event ID fields, taken together, correspond to the primary key of the Event table. The Event table shall consist of one record for each type of failure; the Event table shall consist of the following fields:

Event Table ID

This field will contain the symbol EVENT. It shall identify all Event table records and distinguish these from all other types of record.

Equipment Class

This field will contain a symbol that identifies the class of equipment associated with the event represented by the record.

The set of symbols used in this field shall be the same as the set of symbols used in the Equipment Class field of the Equipment Table.

Event ID

This field shall contain a symbol that identifies the fault or alarm represented by the record, e.g. RX-FAILURE.

Event Source

This field shall contain a symbol that describes the source of the event. The set of symbols used in this field will be: PRIMARY, SECONDARY, DATALOK, TRAMCON, and DPAS. PRIMARY will be used to represent an actual failure. SECONDARY will be used to represent a failure resulting from and depending strictly upon a PRIMARY failure. DATALOK will be used to represent an alarm reportable by a Datalok-10 device. TRAMCON will be used to represent alarms derived by TRAMCON based on combinations of DATALOK alarms. DPAS will be used to represent alarms reportable by the DPAS.

Event Pretty String

This combination of fields will consist of zero or more String Component-Substitution ID pairs, followed by a Completing String. The string resulting from processing these fields will result in a single string that will be used for reporting an event to interfacing units. The resulting string will also be for used by display and report software that needs a suitable print name for the item.

String Component-Substitution ID pairs

This pair of fields will consist of a string portion followed by a substitution ID portion. There may be zero or more Event Pretty String Component-Substitution ID pairs.

String Component

This field will contain a portion of a string that describes an event, including the aspect related to the corresponding substitution ID field. It will include any characters necessary for formatting.

Substitution ID

This field will contain a generic field name which will be used for the purpose of substituting specific names in the event string component. The set of symbols used in this field will be: ID, PORT, SIDE. ID will be used to represent a substitution for actual equipment ID in the event string component. PORT will be

used to represent a substitution for actual port symbol in the event string component. SIDE will be used to represent a substitution for the actual side of the equipment associated with the event in the event string component.

Completing String

This field will contain the remaining portion of a string that describes an event. It will include characters necessary for formatting event time and any required overall string formatting characters.

Causality Table

The Causality table shall express the causal relations between faults and alarms. The Causality table shall be a part of the non-segment-specific knowledge base. All events whose source is not primary shall be represented in one or more records of the Causality table. Causality table records may not be retracted by other TEG functions. The primary key of the Causality table consists of all of these fields: Causing Event ID, Resulting Equipment Class, Resulting Event ID. The Causality table will consist of the following fields:

Causality Table ID

This field will contain the symbol CAUSES. It shall identify all Cause table records and distinguish these from all other types of record.

Equipment Relationship

This field shall contain a symbol that identifies the relationship between the equipment causing the fault and the equipment in which the sympathetic fault is caused. The set of symbols used in this field will be: SAME, DISTANT, SENDER, and LINK-END.

SAME will be used to refer to the same instance of equipment as the equipment causing the fault, i.e. equipment having the same equipment ID.

DISTANT will be used to refer to a piece of equipment of the same class that is connected through the far port of the equipment causing the fault. Other pieces of equipment may reside between the two pieces, but no piece of equipment residing between the device causing the fault and the distant device may be of the same class as these devices.

SENDER will be used to refer to a piece of equipment directly connected to the equipment that causes the fault and is a sender to that equipment (either as the left-hand-side of a

SIMPLEX CONN relationship or as either side of a DUPLEX CONN relationship (3.4.5.3.2)). There is no requirement that the two pieces of equipment be collocated at the same site; only that a CONN relationship exist.

LINK-END will be used to refer to a piece of equipment residing at one of the ends of the connection path of which the equipment causing the fault is a part.

Causing Port Symbol

This field shall contain a symbol that identifies the port associated with the causing event on the equipment causing the fault or alarm. The set of symbols used in this field will be: ANY, NEAR, FAR.

ANY will be interpreted to mean that a fault of the type contained in the Causing Event field on any port of the equipment causing the fault is capable of causing the sympathetic fault or alarm.

NEAR will be interpreted to mean that a fault of the type contained in the Causing Event field on one of the near ports of the equipment causing the fault is capable of causing the sympathetic fault or alarm.

FAR will be interpreted to mean that a fault of the type contained in Causing Event on the far port of the equipment causing the fault is capable of causing the sympathetic fault or alarm.

Causing Event ID

This field shall contain a symbol that identifies the causing fault or alarm. The set of symbols used in this field shall be a subset of the set of symbols used in the Event ID field of the Event Table.

Resulting Equipment Class

This field shall contain a symbol that identifies the type of device in which the sympathetic fault or alarm is resulting. The set of symbols used in this field shall be the set of symbols used in the Equipment Class field of the Equipment Table.

Resulting Port Symbol

This field shall contain a symbol that identifies the port on the equipment in which the sympathetic fault or alarm is resulting. The set of symbols used in this field will be: ALL, NEAR, FAR.

ALL will be interpreted to mean that a fault of the type contained in the Causing Event field is capable of causing the Resulting Event in all ports of the resulting device.

NEAR will be interpreted to mean that a fault of the type contained in the Causing Event field on one of the near ports of the equipment causing the fault is capable of causing the Resulting Event in the corresponding near port of the resulting device. (Note: corresponding near ports have the same port identifier, e.g., MBS-2). FAR will be interpreted to mean that a fault of the type contained in Causing Event is capable of causing the Resulting Event in the far port of the resulting device.

Resulting Side

This field shall contain a symbol that identifies the side of the equipment in which the sympathetic fault or alarm is resulting. The set of symbols used in this field will be: ALL-SIDES, OPER-SIDE.

ALL-SIDES will be interpreted to mean that a fault of the type contained in the Causing Event field is capable of causing the Resulting Event on all sides of the resulting device.

OPER-SIDE will be interpreted to mean that a fault of the type contained in the Causing Event field is capable of causing the Resulting Event in the operational side of the resulting device.

Resulting Event ID

This field shall contain a symbol that identifies the resulting fault or alarm. The set of symbols used in this field shall be a subset of the set of symbols used in the Event ID field of the Event Table.

Segment Representation

The following tables shall be used to define the attributes of a segment known to TEG: the Site table. Additional tables required to define the attributes of a segment may be specified at a later time.

Site Table

The Site table shall define each site located in a particular TRAMCON segment known to TEG. The Site table shall be a part of the segment-specific knowledge base. Site table records may not be retracted by other TEG functions. The Site table will consist of the following fields:

Site Table ID

This field will contain the symbol SITE. It shall identify all Site table records and distinguish these from all other types of records.

Site Symbol

This field shall contain a symbol that identifies the facility at which the equipment is located. The set of symbols used in this field will be the set of three-letter facility IDs (e.g. DON for Donnersberg) used in the DEB. This symbol shall be the primary key of the Site table.

Site Pretty String

This field shall contain a string that describes the site location. The string should contain both the full name of the area as it is known to the DEB community, e.g., "Donnersberg", "Reese-Augsburg", etc. as well as the site symbol. This string shall be used for reporting an event to interfacing units. This string will also be for used by display and report software that needs a suitable print name for the item. The site pretty string will also contain any characters required by display and report software for formatting. An example of the site pretty string is:

"Donnersberg (DON)%n"

G.3.5 Adaptation Requirements

This section describes data that can be modified to change the scope of TEG operation within the prescribed limits.

G.3.5.1 System Environment

Adaptation of TEG to different host computers shall be supported by use of a device-independent rule-based language and a device-independent high order language.

G.3.5.2 System Parameters

This section is not applicable to this specification.

G.3.5.3 System Capacities

At a minimum, storage for the following values in the data base for each TRAMCON segment modelled shall be supported:

a. twenty-one (21) sites

b. 500 devices

c. 750 connections

G.3.6 Quality Factors

This section defines the quality factors that are applicable to TEG and describes how the applicable quality factors will be applied to TEG.

G.3.6.1 Correctness Requirements

The correctness quality factor is the extent to which the system satisfies the requirements defined in this Software Requirements Specification.

The correctness of TEG will be assured by frequent walkthroughs during the design process and by conducting rigorous testing.

G.3.6.2 Reliability Requirements

The reliability quality factor is the extent to which the system is expected to consistently perform its intended function.

The reliability of TEG will be assured by conducting rigorous testing.

G.3.6.3 Efficiency Requirements

The efficiency quality factor is a measure of the efficient use of the computing resource and memory by the system.

The rule-based language algorithm will provide reasonable efficiency. Programming standards for the rule-based language will enhance the efficiency of the search algorithm. Computing and memory usage risk areas will be prototyped early in system development and the results of the prototyping will be factored into the resulting implementation.

G.3.6.4 Integrity Requirements

This requirement is not applicable to this specification.

G.3.6.5 Usability Requirements

The usability quality factor is a measure of the effort required to learn and operate the system.

The usability of TEG will be ensured by designing the man-machine interface to be a user friendly and menu driven system. All messages to the user shall be self-explanatory.

G.3.6.6 Maintainability Requirements

The maintainability quality factor is a measure of the effort required to locate and fix errors in the software.

The maintainability of TEG will be assured by adherence to the programming standards called out herein. In particular, the use of well structured HOL code, well formed rules, and extensive commenting will minimize the number of latent errors in the system and minimize the effort required to locate and fix the errors.

If the rule-based language used is compatible with an appropriate semantic checker, a semantic checker shall be used for style checking and generating cross-references. If CLIPS is used, the Cross Reference, Style, and Verification (CRSV) utility will be used for these purposes.

G.3.6.7 Testability Requirements

The testability quality factor is a measure of the effort required to qualify that the software performs its intended functions. All TEG software requirements shall be testable. The testability of TEG shall be assured by the use of a well structured man-machine interface. Testability shall be further enhanced through the use of an ASCII-only interface with MITEC and use of standard transmission protocols.

G.3.6.8 Flexibility Requirements

The flexibility quality factor is a measure of the effort required to enhance the operational software.

Use of a data-driven rule-based language is an important element in providing for flexibility of the TEG software. Because of this feature, substantial changes may be made to the system by introducing changes into the data base, rather than modifying source code. Use of structured, modular coding within the high order language and well-formed rules in the rule-based language further enhance TEG's flexibility.

Future increments of TEG may include a graphical user interface. To the maximum extent possible, code developed for this purpose to support MITEC will be reused.

G.3.6.9 Portability Requirements

The portability quality factor is a measure of the effort required to transfer the software from one hardware configuration and/or system environment to another.

The portability of TEG will be maximized by the use of portable rule-based and high order language and the avoidance of system-dependent features. When system-dependent features are required, those areas of the code will be isolated and identified with potential portability impacts.

G.3.6.10 Reusability Requirements

The reusability quality factor is a measure of the effort required to use the software in other applications.

Source code and data developed for use in TEG may be usable for the MITEC alarm filtering function and/or a future TRAMCON trainer.

G.3.6.11 Interoperability Requirements

The interoperability quality factor is a measure of the effectiveness of the system's interface with other systems.

TEG will be designed to communicate with MITEC in a manner consistent with the way TRAMCON and DPAS would communicate with MITEC if such communication links were provided.

G.4. QUALIFICATION REQUIREMENTS

G.4.1 General Qualification Requirements

The purpose of these qualification tests is to verify that TEG fulfills the requirements of this SRS. In order to assure the timely completion of system integration and acceptance test, qualification tests are begun during software development. In testing TEG, the evaluation of the following quality factors shall predominate:

a. Usability. Can the prospective user community of TEG operate the program and interpret its results without difficulty?

b. Correctness. Are the fault and alarm events generated by TEG the same as those that occur in TRAMCON: do the results of TEG correspond to known TRAMCON results, or are TEG results believable to experienced TRAMCON operators?

G.4.1.1 Qualification Approach

G.4.1.2 Qualification Phases

Qualification testing of TEG will be accomplished in two phases: Computer Program Test and Evaluation (CPT&E) and CSCI Acceptance Test (CSAT). The purposes of these two phases differ

in that CPT&E is intended to verify primarily the correct operation of code modules and internal interfaces while CSAT is intended to verify the quality factors of usability and correctness discussed in 4.1 above.

Computer Program Test and Evaluation (CPT&E)

The first phase of TEG qualification testing will be CPT&E. CPT&E will be conducted by the TEG programming staff according to internally developed procedures. CPT&E will normally be used to verify internal and developmental requirements, such as the correct operation of individual code modules and internal interfaces, software integration, and observance of design and programming standards. CPT&E will be conducted concurrently with programming and continue until TEG is complete and ready for CSAT. When CPT&E is used to verify a requirement stated in this SRS, the procedures and test results will be recorded and made available for inspection at CSAT.

CSCI Acceptance Test (CSAT)

The second and final phase of TEG qualification testing will be CSAT. CSAT will be conducted by the TEG programming staff, with witnesses from Lincoln Laboratory and the TRAMCON user community. Certain tests that demonstrate usability of TEG functions will be conducted by Lincoln Laboratory personnel and TRAMCON users. Test results that require expert evaluation will be evaluated by TRAMCON users, assisted by the TEG programming staff.

Problems detected during CSAT will be recorded and tracked until corrected or otherwise closed. The Lincoln Laboratory Program Manager shall be the final authority on the disposition of problem reports.

G.4.1.3 Qualification Methods

The following qualification methods shall be used to test TEG: inspection, analysis, demonstration, and test.

Inspection

Inspection is a form of testing in which a requirement is verified by reading off evidence that the requirement has been implemented. Inspection is most often used to verify that design and coding standards have been followed, or that required items are in fact present. For example, it may be verified by inspection that TEG stores the data needed to produce each of its required displays.

Analysis

Analysis is a form of testing that differs from inspection in that the evidence verifying a requirement cannot simply be read off, but must be argued or deduced. Analysis is commonly used to verify requirements that are otherwise impractical to test; for example, to predict the performance of software with a database much larger than that constructed for the test.

Demonstration

Demonstration is a form of testing in which a requirement is verified by executing the system and reading off some result that is evidence that the requirement has been met; for example, that TEG produces a display that contains the expected data. While demonstration is in general the preferred form for testing, it is not practical to use for all kinds of requirements.

Test

Test is a form of testing in which a requirement is verified by executing the system and analyzing the results in order to determine whether the requirement has been met. In this sense, test is a combination of demonstration and analysis. Certain kinds of requirements are susceptible to test but not to demonstration; for example, the requirement that TEG generate the correct set of alarms corresponding to a particular inserted fault, or that the alarm messages are meaningful to a TRAMCON operator.

G.6.GLOSSARY

AB	Air Base
AFB	Air Force Base
ASCII	American Standard Code for Information Interchange
AT&T	American Telephone & Telegraph
CCSD	Command/Control Service Designator
CCSO	Command and Control Systems Office
CLIPS	C Language Inference Programming System
CPT&E	Computer Program Test and Evaluation
CRSV	Cross Reference, Style, and Verification
CSAT	CSCI Acceptance Test
CSCI	Computer Software Configuration Item
DACS	Digital Access and Cross-Connect System
DCS	Defense Communications System
DEB	Digital European Backbone
DPAS	Digital Patch and Access System
DSOM	Digital Systems Operations Manual
HOL	High-Order Language
HWCI	Hardware Configuration Item
IBM	International Business Machines
ID	Identification

MBS	Mission Bit Stream
M.I.T.	Massachusetts Institute of Technology
MITEC	Machine Intelligent Technical Controller
NCS	Network Control System
NMES	Network Management Expert System
PC/AT	Personal Computer/Advanced Technology
PC-DOS	Personal Computer Disk Operating System
RDBMS	Relational Data Base Management System
RS-232	EIA Recommended Standard 232, specifies electrical characteristics of serial connections.
SCBS	Service Channel Bit Stream
SQL	Structured Query Language
SRS	Software Requirements Specification
TBD	To Be Determined
TEG	TRAMCON Event Generator
TM	TRAMCON Master
TRAMCON	Transmission Monitor and Control
TSO	Telecommunications Service Order
XON-XOFF	Transmission On - Transmission Off
USAF	United States Air Force

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	30 September 1989	Annual Report, 1 October 1988 – 30 September 1989

4. TITLE AND SUBTITLE

Knowledge-Based System Analysis and Control Defense Switched Network Task Areas

5. FUNDING NUMBERS

C — F19628-90-C-0002
PE — 62702F

6. AUTHOR(S)

Harold M. Heggestad

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Lincoln Laboratory, MIT
P.O. Box 73
Lexington, MA 02173-9108

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

DCA Engineering Group
1860 Wiehle Avenue
Reston, VA 22090-5500

10. SPONSORING MONITORING AGENCY REPORT NUMBER

ESD-TR-90-141

11. SUPPLEMENTARY NOTES

None

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

The primary thrust of the FY89 program was installation and demonstration of the Network Management Expert System (NMES) at DCA Headquarters in Stuttgart, Germany. Two major efforts were involved: completion of improvements and extensions to NMES itself and performance of numerous tasks in matching the hardware and software to the DCA-Europe environment so that it could be meaningfully operated there. Additional components of the FY89 program were implementation of an Operator Trainer to develop operator skills in the use of the current manual "Network Management Support System," as well as execution of a set of enhancements to, and investigations with, the Call-by-Call Simulation (CCSIM).

The FY89 SOW also called for substantial effort to begin to scope and understand the problems of correlating and interpreting DCS transmission system alarms and presenting filtered results to Tech Control and Network Management decision makers. By analogy with the success of the CCSIM as a portrayal of realistic network behavior to use in developing network management knowledge, it has been conjectured that TRAMCON and DPAS alarm pattern simulation systems could make up for the lack of access to real transmission networks for MITEC software developers, as well as the fact that real networks seldom produce interesting alarm patterns. In FY89 a specification was developed for a TRAMCON alarm generator for this purpose, with an eye to implementing it in FY90.

14. SUBJECT TERMS			15. NUMBER OF PAGES
expert systems	system control	Defense Switched Network (DSN)	170
communications control	network management	communication network simulation	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unclassified