BNL-52605
Formal Report
ISPO-411

The "Mailbox" Computer System for the IAEA Verification Experiment on HEU Downblending at the Portsmouth Gaseous Diffusion Plant

July 31, 2000

UNITED STATES MEMBER STATE SUPPORT PROGRAM TO IAEA SAFEGUARDS

DEPARTMENT OF ENERGY
DEPARTMENT OF STATE
NUCLEAR REGULATORY COMMISSION
DEPARTMENT OF DEFENSE

Prepared by

Arnold L. Aronson and David M. Gordon

Brookhaven National Laboratory
Upton, New York 11973

OFFICIAL FILE COPY

BNL-52605
Formal Report
ISPO-411

The "Mailbox" Computer System for the IAEA Verification Experiment on HEU Downblending at the Portsmouth Gaseous Diffusion Plant

July 31, 2000

Prepared by

Arnold L. Aronson and David M. Gordon

Brookhaven National Laboratory
Upton, New York 11973

This work was performed under the auspices of the U.S. Department of Energy under Contract No. DE-ACO2-98CH10886.

Disclaimer

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency, contractor, or subcontractor thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency, contractor or subcontractor thereof.

Abstract

In April 1996, the United States (US) added the Portsmouth Gaseous Diffusion Plant to the list of facilities eligible for the application of International Atomic Energy Agency (IAEA) safeguards. At that time, the US proposed that the IAEA carry out a "Verification Experiment" at the plant with respect to the downblending of about 13 metric tons of highly enriched uranium (HEU) in the form of uranium hexafluoride (UF_6). During the period December 1997 through July 1998, the IAEA carried out the requested Verification Experiment. The verification approach used for this experiment included, among other measures, the entry of process-operational data by the facility operator on a near-real-time basis into a "Mailbox" computer located within a tamper-indicating enclosure sealed by the IAEA. These data included information on the weights and the connection/disconnection times for the cylinders containing the HEU UF_6 being downblended. The facility operator had also agreed to keep the cylinders connected to the feed stations for agreed lengths of time (the declared "residence times") which depended on cylinder size and contained mass of UF_6. The IAEA inspectors then performed short-notice random inspections at the plant to verify the data entered by the facility operator. This report describes the hardware and software for the "Mailbox" computer system and the role that the "Mailbox" system plays in short-notice random inspections. In addition, it provides a listing of the software developed for this application.

(THIS PAGE INTENTIONALLY LEFT BLANK)

Table of Contents

Abstract iii

List of Appendices vi

Acknowledgments vii

1. Introduction 1

2. Description of the HEU Downblending Process 1

3. Verification Approach for the Verification Experiment 2

   3.1 Design Information Verification and Re-verification 3
   3.2 Operator Declarations of Cylinder Feed Operations via Computerized "Mailbox" 3
   3.3 Load-Cell-Based Weighing System for HEU Feed Cylinders 3
   3.4 Video Surveillance of the HEU Feed Stations 4
   3.5 Verification of HEU Feed Cylinder Weights 4
   3.6 Non-Destructive Assay (NDA) Measurements for HEU Feed Cylinders 4
   3.7 Application of Seals to Process Valves and Flanges 5
   3.8 NDA Measurements for HEU and LEU Piping 5
   3.9 Short-Notice Random Inspections (SNRI) 6
   3.10 Role of the "Mailbox" Computer System in the Verification Experiment 6

4. "Mailbox" Computer System Design Requirements 7

   4.1 Background 7
   4.2 Design Requirements for the "Mailbox" System for the Verification Experiment 7

5. "Mailbox" Computer System - Hardware 8

   5.1 Operator's Data-Preparation Computer 8
   5.2 The "Mailbox" Computer 9
   5.3 IAEA Inspector's Data-Analysis Computer 9

6. Overview of "Mailbox" Computer System Operations 12

7. "Mailbox" Computer System - Software 15

   7.1 Overview of the Software 15
   7.2 Plant Operator's Data-Preparation Computer 17
   7.3 The "Mailbox" Computer 20
      7.3.1 Software Used for Data Entry 20
      7.3.2 Software Used for Data Downloading 20
   7.4 The IAEA's Data-Analysis Laptop Computer 21
      7.4.1 The Pre-processing Software 21
      7.4.2 The Report Generation Software 24

8. Experience with the "Mailbox" Computer System 25

References 27

List of Appendices

A. The IAEA SP-1 Support Programme Task Proposal..... A-1

B. The Task Statement for the "Mailbox" Computer System.....B-1

E. First Line Manager's Instructions for the Plant Data-Preparation Computer.....C-1

E. First Line Manager's Instructions for the "Mailbox" Computer..... D-1

E. IAEA Inspector's Instructions for the "Mailbox" Computer..... E-1

E. Inspector's Instructions for Generating Reports Using the Data-Analysis Laptop Computer.....F-1

E. The Plant Operator (First Line Manager) Data-Entry Program..... G-1

E. The FLM Supervisor's Password Program.....H-1

E. FLM Supervisor's Instructions for the Password Program.....I-1

E. The AUTOEXEC.BAT File for the "Mailbox" Computer..... J-1

E. "Mailbox" Software to Collect the Plant Operator's Declaration.....K-1

E. Batch File to Download the Plant Operator's Declaration from the "Mailbox" Computer to the IAEA Representative's Floppy Diskette.....L-1

E. The RESET Program to Reset the "Mailbox"..... M-1

E. Programs to Incorporate the Plant Operator's Declarations of Purity (Uranium Weight Percent) and Assay (U-235 Weight Percent).....N-1

E. Programs to Incorporate the Plant Operator's Declarations of Minimum Residence Times for UF_6 Cylinders Attached at Feed Stations.....O-1

E. The Pre-processor Programs.....P-1

E. The Report Generator Programs.....Q-1

E. Sample Reports.....R-1

Acknowledgments

Many persons and organizations contributed to the successful completion of the IAEA Verification Experiment on HEU Downblending at the Portsmouth GDP. However, particular thanks and appreciation go to the following people:

- O. Lynn Calvert, Thomas N. Bonner, and Jeff Castle at the Portsmouth Gaseous Diffusion Plant, whose dedication, help and strong personal support made the "Mailbox" computer system, the short-notice random inspections, and the Verification Experiment a success;

- David A. Shisler, whose organization prepared the nuclear material accountability data, and Karl Shucker, who was extremely helpful in providing data that resolved discrepancies in data entry for the "Mailbox";

- Neville Whiting, Prachaks Jinamornphongs, Tamas Biro, and Eugene Chilkine of the IAEA inspection team, whose hard work and patience were essential to the success of the Verification Experiment;

- Willi Theis (Section Head, IAEA Division of Safeguards Concepts and Planning), who was instrumental in planning and gaining approval for undertaking the Verification Experiment;

- Mano Subudhi of Brookhaven National Laboratory, who was extremely helpful in reviewing the "Mailbox" data, resolving data-entry discrepancies, and assisting with the Experiment; and

- Amy B. Whitworth and Ronald C. Cherry of the Department of Energy International Safeguards Division, who provided overall guidance and support for the project.

(THIS PAGE INTENTIONALLY LEFT BLANK)

1. Introduction

In April 1996, the United States (US) added the Portsmouth Gaseous Diffusion Plant (GDP) to the list of facilities eligible for the application of International Atomic Energy Agency (IAEA) safeguards. At that time, in the event that the IAEA chose not to select the plant for the application of traditional safeguards, the US proposed that the IAEA carry out a "Verification Experiment" at the plant with respect to the downblending of 13.2 metric tons of highly enriched uranium (HEU, with enrichment ≥20% U-235) in the form of uranium hexafluoride (UF_6). This material is part of the 226 metric tons of fissile material that President Clinton has declared to be excess to US national-security needs and which will be permanently withdrawn from the US nuclear stockpile. However, because of the nature of the downblending process and equipment, it was not feasible to perform a nuclear-material balance around the blending point where the HEU and the low-enriched uranium (LEU, with enrichment <20% U-235) are mixed to produce an LEU product with slightly greater enrichment than the LEU blendstock. Thus, the focus of the Verification Experiment was on technical measures to verify the HEU downblending operation.

The US and the IAEA met during April, August, and September of 1997 for technical exchanges; on the US side, the participants included US Government personnel (Department of Energy, Department of State, Nuclear Regulatory Commission, and Arms Control and Disarmament Agency), staff from several of the Department of Energy's national laboratories, the United States Enrichment Corporation (USEC, the operator of the Portsmouth GDP), and Lockheed Martin Utility Services, Inc. (LMUS, the operating contractor for the plant). Following these meetings, the IAEA committed to performing the Verification Experiment based on the agreed verification measures and procedures. The Verification Experiment began on December 1, 1997, and the IAEA carried out its design information verification activities during December 1-17, 1997. At the commencement of the Verification Experiment, about 3.7 metric tons of the original 13.2 metric tons remained available for verification. The HEU downblending activity was completed on July 17, 1998, and the IAEA performed a close-out inspection during October 5-14, 1998.

2. Description of the HEU Downblending Process

Highly enriched uranium in the form of UF_6 was blended down to LEU in the uranium-enrichment gaseous-diffusion cascade at the Portsmouth GDP. This material had a variety of enrichments ranging from about 20 - 98% U-235 (with an average enrichment of about 70% U-235 for uranium subject to the Verification Experiment) and was contained in 5-, 8-, and 12-inch-diameter cylinders. The HEU UF_6 was fed at nine cylinder stations located in a room on the first floor of Process Bldg. X-326; each station included a process-control load-cell-based weighing system (LCBWS) for real-time measurement of cylinder weights as the contents of the cylinders were being transferred to the cascade blending point. The solid HEU UF_6 in the cylinders was converted to gaseous UF_6 by sublimation at temperatures slightly elevated above room temperature. In principle, the HEU UF_6 gas could be transported by either of two piping headers to any of several cells of operating gaseous-diffusion equipment located on the second

floor of this process building. In actual practice, the same header was used to carry the gas to the same cell during the entire course of the Verification Experiment. The length of this piping header was about 240 m. Almost all of the HEU feed piping from the cylinder feed stations to the blending point was contained within heated enclosures typically maintained at temperatures of about 70-85 °C; the remainder was heat-traced and surrounded with insulating jackets. While some of the piping was readily visible, that portion contained within heated enclosures was not visible; as such, it was not traceable without removing the enclosure panels.

For his own security purposes, the plant operator prevented the flow of HEU UF_6 to any other location by closing, locking, and sealing all valves for piping that branched off from the selected header and which could provide a flow path to other locations. The closed, locked, and sealed valves along the piping headers from the feed stations to the selected blending point which defined and limited the HEU UF_6 flow to one and only one destination -- the selected blending point -- were called "boundary" valves. "In-line" valves along the piping were open valves that permitted the flow of HEU UF_6 to the selected blending point.

The blending of the HEU UF_6 feed stock with the LEU UF_6 blend stock took place within the gaseous-diffusion equipment comprising the LEU enrichment cascade. Specifically, the blending point (or blending "T") was the juncture of the 3-inch-diameter pipe (carrying the HEU UF_6) with the 20-inch-diameter pipe between the Stage 12 diffuser and the Stage 12 compressor of Cell 3 within Unit 25-7, which is in the upper portion of the cascade. The HEU mass-flow rate in the 3-inch pipe was in the range from about 1 to 80 kg U per day, while the LEU mass-flow rate in the 20-inch pipe was typically about 39000 kg U per day. The LEU mass-flow rates were calculated quantities; there is no instrumentation available which can directly measure the flow of the LEU UF_6 either upstream or downstream of the blending point. Thus, a small amount of HEU UF_6 carried in a small pipe was mixed with a very large amount of LEU UF_6 carried in a very large pipe; the U-235 enrichment of the LEU stream was increased by only about 1% (relative). After leaving the compressor, the blended LEU UF_6 entered an 8-inch-diameter pipe, and after passing through a cooler, the blended LEU entered the first stage of the next higher on-stream cell in the cascade.

3. Verification Approach for the Verification Experiment

The purpose of the IAEA's verification measures was to verify the quantity of HEU UF_6 that was fed to the cascade and blended down to LEU UF_6. In order to accomplish this, the IAEA needed to verify (1) the amount of HEU UF_6 removed from the feed cylinders, and (2) the fact that the HEU UF_6 removed from the cylinders entered the blending point and was blended down to LEU UF_6. It was recognized that the verification purpose and the associated verification measures did not constitute a material balance for the operation. The verification measures to accomplish this purpose are outlined below. Of these, the measures in Sections 3.2 - 3.6 were intended primarily to verify the amount of HEU UF_6 removed from the feed cylinders, while the measures in Sections 3.1, 3.7, and 3.8 were intended primarily to verify that HEU UF_6 removed from cylinders entered

the blending point and was blended down to LEU UF_6. The verification measure in Section 3.9, the short-notice random inspections (SNRIs), served both purposes.

3.1 Design Information Verification and Re-verification

In November 1997, the Portsmouth GDP provided the IAEA with comprehensive design information concerning the HEU downblending equipment and process. The design information verification and re-verification measures that could be performed by the IAEA inspectors included: visual checking of the HEU feed piping at five randomly selected locations during each inspection for comparison with the declared piping drawings and for detection of undeclared piping; performing a pressure check and a vacuum check to confirm that all "boundary" valves had been placed in the closed position; performing a flow check to confirm the existence of a flow path from the feed stations through the blending point; and performing a volume measurement for comparison with the engineering calculated volume of the entire HEU feed piping system.

3.2 Operator Declarations of Cylinder Feed Operations via Computerized "Mailbox"

The facility operator entered process-operation data on a near-real-time basis into a "Mailbox" computer, which was located within a tamper-indicating enclosure sealed by the IAEA. The data to be entered included the HEU feed cylinder identification number, HEU feed station number, type of event (cylinder connection, cylinder disconnection, or error correction), date and time of the event, cylinder adjusted gross weight, cylinder adjusted tare weight, and concise note for explanations. The facility operator also provided accountability data for each feed cylinder subject to the Verification Experiment, including accountability gross weight, accountability tare weight, UF_6 purity (grams of uranium per gram of UF_6), and U-235 enrichment (grams of U-235 per gram of uranium). The facility operator also agreed to keep the cylinders connected to the feed stations for agreed lengths of time (the declared "residence times") which depended on cylinder size and contained mass of UF_6.

3.3 Load-Cell-Based Weighing System for HEU Feed Cylinders

HEU UF_6 cylinders could be fed at any of nine feed stations. Each of these stations included an operator-installed load-cell-based platform scale for process control. The electrical outputs from these nine platform scales were connected by cable to nine corresponding computer-compatible readout units, each of which contained additional electronics for data storage and data authentication; these readout units had tamper-indicating enclosures sealed by the IAEA. The nine readout units were connected by cable to an IAEA computer contained within a tamper-indicating enclosure sealed by the IAEA. The IAEA computer collected weight data every ten seconds and stored weight data every 30 minutes from each of the nine readout units. Events such as rapid changes in scale readings were logged and also used to trigger the video-surveillance cameras

viewing the nine feed stations. Stored data could be downloaded and printed out, plotted, or manipulated with standard data-base programs. The LCBWS permitted the IAEA to correlate real-time weight data, obtained as the HEU UF_6 feed cylinders were emptied, with the "Mailbox" declarations and video-surveillance system results.

3.4 Video Surveillance of the HEU Feed Stations

The nine HEU feed stations were under continuous observation by two separate video-surveillance cameras equipped with wide-angle lenses; each camera viewed all nine feed stations, with diminished resolution for those cylinders farthest from the camera location. The video-surveillance system was used to detect attempts to falsify the weight values recorded by the LCBWS and to detect removal of HEU UF_6 from feed cylinders through piping attached to the second cylinder valve. Each of the video cameras, along with additional electronics for data storage and data authentication, was contained within its own tamper-indicating enclosure sealed by the IAEA. The cameras were connected by cable to an IAEA computer contained within a tamper-indicating enclosure sealed by the IAEA. The video cameras were triggered to record images in any of three ways: motion detection based on detection of scene changes within the field of view; rapid changes in the weight observed by any of the nine load-cell-based weighing systems; or elapsed time interval. The video surveillance system and the load-cell-based weighing systems provided a mutually supporting set of data for confirming the operator's "Mailbox" declarations.

3.5 Verification of HEU Feed Cylinder Weights

Whenever the IAEA inspectors arrived for an inspection, they could request that the facility operator weigh any cylinder in the feed area on the facility's confirmatory scale. Any four of the nine cylinders attached at the feed stations could be selected. The IAEA could authenticate this confirmatory scale using its own standard weights. These verified cylinder weights could be used to check the functioning and accuracy of the LCBWS scales.

3.6 Non-Destructive Assay (NDA) Measurements for HEU Feed Cylinders

Whenever the IAEA inspectors arrived for an inspection, they could perform gamma-ray NDA measurements of U-235 enrichment for the UF_6 in all cylinders in the feed area. The purpose of these measurements was to verify the operator's "Mailbox" enrichment declarations. These measurements were made using a portable high-resolution gamma-ray detector and were based on the standard enrichment-meter principle, with corrections for the attenuation of the 186-keV gamma rays by the intervening materials. In general, these measurements were made *in situ* through the cylinder-supporting sleeve of the transport dolly, with the cylinder remaining attached at the feed station. However, the IAEA could request that up to four of the nine cylinders be

detached from their feed stations and removed from their dollies for measurements made directly through the cylinder wall.

3.7 Application of Seals to Process Valves and Flanges

The IAEA installed seals on all 30 closed "boundary" valves along the piping from the HEU feed stations to all the potential blending points; these valves defined and limited the HEU UF_6 flow path to one and only one destination - the declared blending-point. In addition, the IAEA sealed a number of valve bodies and piping flanges, which might otherwise have been used to divert the flow of HEU UF_6 to locations other than the blending point. Each of the "boundary" valves was sealed with two seals - an AIMS Fiber-Optic Seal (AFOS) and a standard IAEA E-type metal seal and seal wire. The AFOS seals were part of a radio-frequency-based remote-monitoring Authenticated Item Monitoring System (AIMS) which included the fiber-optic seals and redundant data-collection systems (repeaters, receivers, and Receiver Processing Units) to collect, store, and read out the seal events. Each Receiver Processing Unit monitored the seal-activity data for all the installed seals; these data could be downloaded to a computer for detailed analysis.

3.8 NDA Measurements for HEU and LEU Piping

Whenever the IAEA inspectors arrived for an inspection, they were in principle able to perform NDA measurements of the enrichment for the UF_6 in the 3-inch-diameter piping carrying HEU UF_6 to the blending point and the 8-inch-diameter piping carrying blended LEU UF_6 from the blending point. The purpose of these measurements was to confirm that HEU UF_6 was actually arriving at the blending point and that LEU UF_6 was actually available for blending with the HEU UF_6. The NDA method used in both cases was the Cascade Header Enrichment Monitor (CHEM) technique used by the IAEA at other enrichment facilities. The measurements were performed on an intermittent basis (i.e., during SNRIs), not on a continuous unattended basis. Between inspections, the equipment was stored in tamper-indicating enclosures sealed by the IAEA.

However, these NDA measurements met with only limited success, since the high-purity germanium detector used for the measurements was not able to withstand the high temperature environment (85 ^{o}C) when clamped onto the hot piping for extended periods. As an alternative measure, the IAEA was able to obtain samples of UF_6 from the LEU piping near the blending point. These samples were shipped to IAEA Headquarters in Vienna, Austria, where they were measured using a mass spectrometer to determine their U-235 enrichment. These measurements confirmed the operator's declarations concerning the enrichment of the LEU UF_6 at the blending point.

3.9 Short-Notice Random Inspections (SNRI)

The IAEA performed short-notice random inspections to confirm the facility operator's "Mailbox" declarations (including confirming the actual presence of HEU cylinders that had been declared to be present, performing NDA measurements of enrichment for HEU feed cylinders, and confirming cylinder weights); to re-verify the design information; to confirm the status of the applied seals; to perform gamma-ray NDA measurements of the enrichment of UF_6 in HEU and LEU piping near the blending point; to review and correlate the LCBWS data, the video-surveillance data, and the "Mailbox" data; and to review operating records and accounting ledgers maintained by the facility operator. The IAEA inspectors coming for SNRIs were given access to the plant at any time including holidays, weekends, and off-normal working-day hours as well as normal working hours. It was anticipated that inspector access would take about one hour during normal working hours (7:30 AM to 4 PM local time, Monday through Friday) and up to about three hours during non-working hours.

3.10 Role of the "Mailbox" Computer System in the Verification Experiment

A major feature of the Verification Experiment was the implementation of a Short-Notice Random Inspection (SNRI) approach. Three conditions must be met for this approach to applicable: (i) all items in the population of HEU UF_6 cylinders must be available for selection for verification; (ii) the plant operator must declare to the IAEA values for the nuclear-material content of HEU UF_6 cylinders before knowing which ones will be verified; and (iii) the operator must not be able to alter item identity or content after learning that an item has been chosen for verification and before the verification actually occurs. The "Mailbox" was a dedicated computer in a tamper-indicating enclosure with keyboard access only by IAEA inspectors; the "Mailbox" computer system was used to fulfill condition (ii). The "Mailbox" declarations were unalterable operator statements of accountancy values and operational data against which IAEA inspectors could compare the results of verification measurements. Rapid access to the cylinders during short-notice random inspections, coupled with the load-cell-based weighing systems and video surveillance of the cylinders attached at the feed stations, were used to fulfill condition (iii).

Each month, the facility operator (through the US Government) made official declarations, on a cylinder-by-cylinder basis, of the amounts of highly enriched uranium and U-235 that had been downblended during the previous month. The IAEA verified these official facility/US declarations by comparing them to the near-real-time declarations deposited in the "Mailbox" computer and the results of the verification measurements performed on the cylinders that were present at the time of the short-notice random inspections. These measurements included NDA measurements of the enrichment for the UF_6 contained in the cylinders and confirmatory weighings of individual cylinders. In addition, the IAEA could confirm the amount of UF_6 removed from each of the cylinders by examining the data from the load-cell-based weighing systems, which provided a time-stamped and real-time record of the cylinder-plus-dolly weight as a function of time. Finally, the IAEA could confirm the connection and disconnection times of the

cylinders by reviewing the time-stamped images from the video surveillance system that viewed the cylinders at the feed stations. The "Mailbox", the load-cell-based weighing system, and the video surveillance system provided a powerful and mutually supporting set of data from which to draw verification conclusions concerning the amounts of HEU UF_6 removed from the cylinders.

4. "Mailbox" Computer System Design Requirements

4.1 Background

Short-notice random inspections for verification of inventory changes were first analyzed theoretically by Gordon and Sanborn (Ref. 1) based on a suggestion by Brenner (Ref. 2). Their analysis was directed toward IAEA verification of inventory changes for feed, product, and tails at gas centrifuge enrichment plants as part of the Hexapartite Safeguards Project (Ref. 3). According to this approach, the plant operator declares the contents of nuclear material items before knowing if an inspection will occur to verify them. Additionally, items about which declarations are made should remain available for verification for an agreed-upon length of time (the "residence time"). If these conditions are met, then a statistical inference can be made from the verification results for items verified during SNRIs to the entire population, even if inspectors were not present when the items were received or processed.

During 1993, the IAEA conducted a six-month field test of the feasibility of such SNRIs at the Westinghouse Electric Corporation Commercial Nuclear Fuel Division LEU fuel-fabrication plant located at Columbia, South Carolina, USA (Fishbone, et al., Ref. 4). Westinghouse personnel made daily declarations about both feed and product items (uranium hexafluoride cylinders and finished fuel assemblies), using a custom-designed computer "Mailbox". IAEA safeguards inspectors conducted eight SNRIs to verify these declarations; the inspectors arrived unannounced at the plant, usually after travel from the IAEA regional office in Canada. The field test demonstrated the feasibility and practicality of key elements of the SNRI approach for a large LEU fuel fabrication plant, particularly the "Mailbox" computer system and the SNRIs themselves.

4.2 Design Requirements for the "Mailbox" System for the Verification Experiment

During the course of the technical exchanges in August and September of 1997, the IAEA specified the main design requirements for the "Mailbox" computer system to be used for the Verification Experiment on HEU Downblending at Portsmouth. These were:

- the "Mailbox" computer system hardware and software should be as similar as possible to that used for the Westinghouse field test. The software should use as much of the existing Westinghouse "Mailbox" software as possible, with changes only as needed to take account of the differences in the items being declared. This requirement was necessary because of the need to begin the Verification Experiment as soon as possible in order to be able to verify the maximum possible amount of the remaining HEU.

- the plant operator would enter the cylinder operations data into the "Mailbox" computer on a near-real-time basis, rather than on a once-per-day basis. That is, the data for a cylinder connection or disconnection should be entered as soon as feasible after that connection or disconnection actually took place. This was necessary so that the data in the "Mailbox" would be current regardless of when the IAEA inspectors arrived for a short-notice random inspection. Since cylinder-handling operations took place on a 24-hour per day, 7-day per week basis, this meant that the data would have to be entered by Production Operations personnel rather than by Nuclear Material Control and Accountability personnel. Thus, additional data-preparation software and procedures had to be developed and written for use by the Operations personnel. In addition, a data-preparation computer was provided to the plant.

- some data would be declared by the plant operator before the start of the Verification Experiment. This would eliminate the need for the Production Operations staff to enter these data, as well as eliminate data-entry errors for these quantities. These data included the agreed cylinder residence times (which depended on the cylinder size and contained mass of UF_6) and accountability data for each feed cylinder subject to the Verification Experiment (including the U-235 enrichment and UF_6 purity). Software had to be written to incorporate these data into the database files so that the quantities of uranium and U-235 for each cylinder could be calculated by the computer from the declared weights.

- the IAEA specified the type of reports that it desired to generate from the "Mailbox" data. Four types of reports were requested; these are described in Section 7.4.2.

In order to obtain "Mailbox" computer hardware and software that would meet the needs of the Verification Experiment, the IAEA requested the US Government to provide them through the US Program of Technical Assistance to IAEA Safeguards. The IAEA SP-1 Support Programme Task Proposal is provided as Appendix A, while the Brookhaven National Laboratory task statement to meet this SP-1 Proposal is provided as Appendix B.

5. "Mailbox" Computer System - Hardware

The "Mailbox" computer system consisted of three computers - the operator's data-preparation computer, the "Mailbox" computer itself, and the IAEA inspector's data-analysis computer. These are described in turn below.

5.1 Operator's Data-Preparation Computer

The operator's data-preparation computer was a Gateway E Series E3000-1166 desktop computer. This computer had an Intel 166-MHz Pentium processor with 16 MB of RAM, a 1.6-GB hard drive, two 3.5-inch 1.44-MB floppy diskette drives, a standard 104-key keyboard, and a

15-inch monitor. Since the operating system was MS-DOS, no mouse was used.

5.2 "Mailbox" Computer

The "Mailbox" computer was a Gateway E Series E3000-1166 desktop computer. This computer had an Intel 166-MHz Pentium processor with 16 MB of RAM, a 1.6-GB hard drive, two 3.5-inch 1.44-MB floppy diskette drives and a 15-inch monitor. Since the operating system was MS-DOS, no mouse was used. The keyboard was stored near the computer until needed by the IAEA inspectors to take control of the "Mailbox" computer and download the data stored within it.

The "Mailbox" computer was contained within a tamper-indicating enclosure (TIE) designed and built by Sandia National Laboratories - Albuquerque (SNLA). SNLA staff had removed the computer's external case in order to obtain a closer and better fit between the computer and the TIE. The enclosure was constructed of three pieces of anodized aluminum. Two of these pieces surrounded the computer and were sealed together with a standard IAEA E-type metal seal and seal wire. The third piece covered the port for the keyboard; this third piece was sealed to the other two pieces with a standard IAEA E-type metal seal and seal wire. When the computer was sealed within the TIE, only the "B" floppy-diskette drive was accessible; the "A" drive was located behind the TIE. The power cable for both the "Mailbox" computer and its monitor were connected to a single, switched power strip with surge protection. To initiate the operation of the "Mailbox" computer, the facility operator or the IAEA inspectors needed only to turn on the power switch.

During the Verification Experiment, the operator's data-preparation computer and the "Mailbox" computer were located side-by-side on a table in a plant Area Control Room near the location where the HEU UF_6 cylinders were being fed to the cascade. Figure 5.2.1 shows the data-preparation computer and the "Mailbox" computer installed in the Area Control Room. The "Mailbox" computer, located on the right in the photograph, was enclosed within its tamper-indicating enclosure. Figure 5.2.2 shows a side view of the "Mailbox" computer within its tamper-indicating enclosure.

5.3 IAEA Inspector's Data-Analysis Computer

The IAEA inspector's data-analysis computer was a Toshiba Satellite 430 CDS Satellite Pro laptop computer with a 120-MHz Pentium processor, 32 MB of RAM, a 1.3-GB hard drive, a 10X CD-ROM drive, and a 3.5-inch 1.44-MB floppy diskette drive. The installed operating system was Microsoft Windows 95 but the analysis programs were run from the MS-DOS operating system.

Figure 5.2.1

The data-preparation computer (on the floor, left) and the "Mailbox" computer in its gold-colored tamper-indicating enclosure (on the desk, right) installed in the area control room

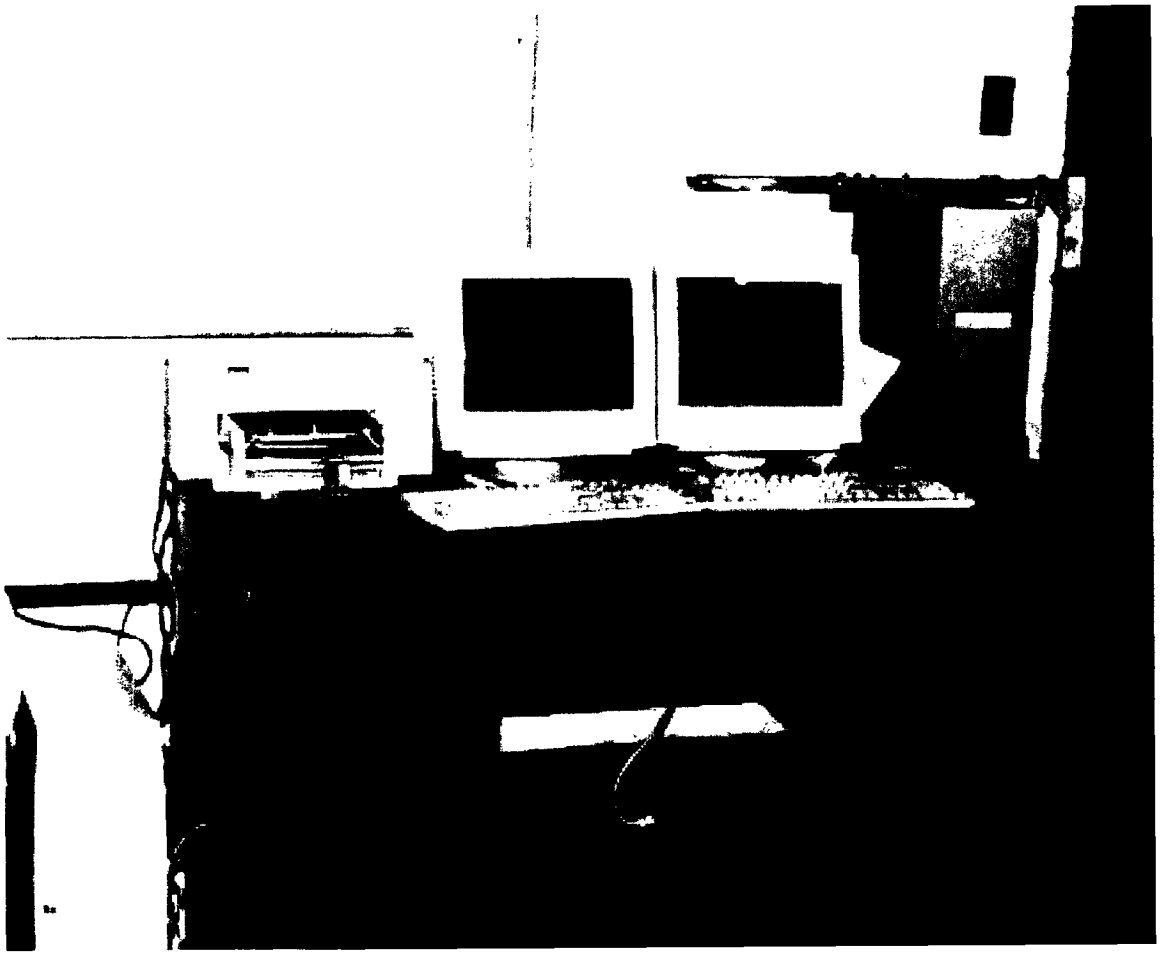
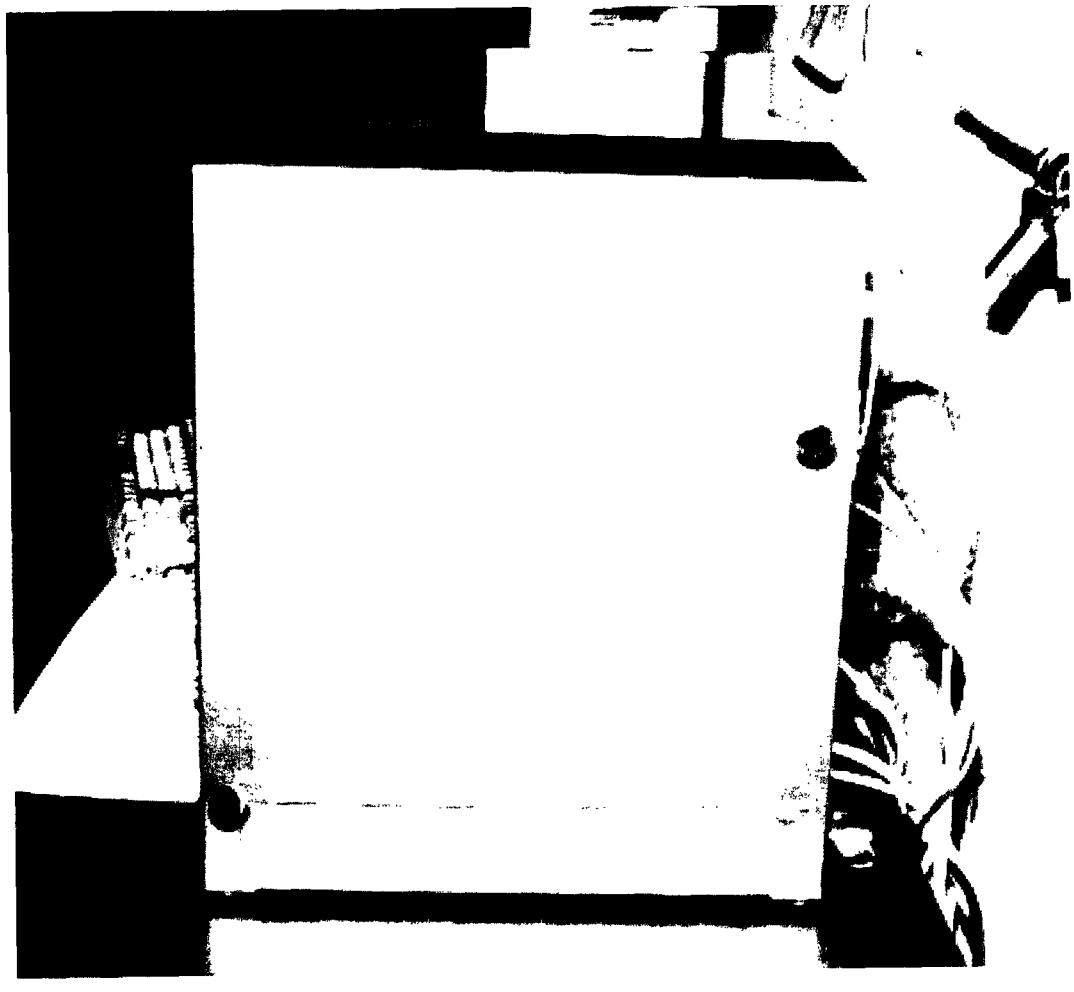


Figure 5.2.2

Side view of the "Mailbox" computer in its gold-colored tamper-indicating enclosure, showing the installed IAEA E-type seals and seal wires



6. Overview of "Mailbox" Computer System Operations

The data for the "Mailbox" computer system used in connection with the IAEA Short-Notice Random Inspections at the Portsmouth Gaseous Diffusion Plant started with data declarations made by the facility operator. The Portsmouth GDP operating staff connected, emptied, and disconnected the HEU UF_6 cylinders whose contents were to be downblended according to the need for production of LEU UF_6 and the cascade operating conditions. Each time that a Portsmouth GDP Operations staff member connected or disconnected an HEU UF_6 cylinder at one of the nine feed stations, he or she filled out a data sheet that provided the cylinder identification number, its gross and tare weights, the time and date of its connection or disconnection, and the feed station ("Position") at which the connection or disconnection took place. An example of this data sheet is provided as Figure 6.1. After filling out the form, the staff member sent it by facsimile machine to the local Area Control Room, where it was received by a member of the Production Operations supervisory staff known as the First Line Manager (FLM). The FLM then prepared the declaration for this cylinder, or set of cylinders, using a dedicated computer and a special-purpose data-input program; the computer used for this purpose was the data-preparation computer described in Section 5.1 above. The instructions for this data-entry procedure are provided in Appendix C. The FLM preparing the declaration turned on the computer (which started the program), supplied a password to allow execution, and then entered his name and badge number. The FLM was then prompted to enter the following information:

- cylinder identification number (a 6-character string)
- HEU feed station number (a 2-character string)
- type of event (c = connection, d = disconnection, e = error correction)
- event date (a 6-character string, in the format mmddyy)
- event time (a 4-character string in the format of a 24-hour clock)
- cylinder adjusted gross weight (in grams)
- cylinder adjusted tare weight (in grams)
- a concise note, if appropriate (up to four 80-character lines)

If the event type was "e" (that is, a correction for a previous incorrect entry), then the FLM was prompted to enter the date and time of the event being corrected, since that was the only way to identify an event uniquely (i.e., two events could not take place at exactly the same time). If the data entry that was being corrected was the type of event (e.g., "c" or "d"), then the FLM entered the corrected event type.

After the data for an event were entered, the FLM saw them displayed on the screen and had an opportunity to revise them. The revised entries were then displayed until the FLM indicated by an appropriate response to the prompts that no further revisions needed to be made. At that time, the information was entered into a database file in the computer. The FLM was prompted for any additional events to be entered during the current session at the computer. When the FLM indicated that there were no additional events to be entered, the database file was copied from the computer to a 3.5-inch floppy diskette, and the database file on the computer was erased.

IAEA Mailbox Data

DATE 5-4-98

Position	Cylinder	Time	Disconnect	Connect	Ops Adj Gross Wgt	Ops Adj Tare Wgt
1						
2						
3						
5						
6						
7	05-0293	0912	✓		28560	27971
8						
9						
10	08-0466	0916	✓		52617	51033

Data Entered on Mailbox_____
FLM Initials

Figure 6.1
Operator's Data Sheet for Cylinder Operations

The FLM then entered the declaration data on the floppy diskette into the "Mailbox" computer; this was the computer described in Section 5.2 above. The instructions for this procedure are provided in Appendix D. According to this procedure, the FLM turned on the "Mailbox" computer and was asked by the computer "Are you an inspector?" Since the keyboard was disconnected and the keyboard port was covered by the tamper-indicating enclosure, the FLM was unable to answer this query. In the absence of an affirmative response after a short period of time, the computer recognized that the plant operator wished to enter declaration data. The FLM inserted the floppy diskette into the diskette drive of the "Mailbox" computer (the "B" drive, which was the only drive accessible from outside the tamper-indicating enclosure). The "Mailbox" computer read the data, stored it in the appropriate files, and then instructed the FLM to turn off the "Mailbox" computer. The "Mailbox" computer accumulated the data declarations entered by the plant staff.

When the IAEA inspectors arrived at the plant for a short-notice random inspection, they were able to download the set of data declarations made by the plant operating staff. The instructions for this procedure are provided in Appendix E. The IAEA inspectors checked their seals on the tamper-indicating enclosure surrounding the "Mailbox" computer, and then removed the small section of the TIE which covered the keyboard port. They connected the keyboard to the port and then turned on the "Mailbox" computer, using the switch on the electrical power strip. The computer asked, "Are you an inspector?" and the inspectors responded by typing a "Y" (for "Yes") on the keyboard. The inspectors now had control of the computer and could download the data from it, as well as perform other operations. The inspectors inserted a floppy diskette into the diskette drive of the "Mailbox" computer (again, the "B" drive) and downloaded the operator's data declarations. After downloading these declarations onto the diskette, the inspectors removed the diskette from the "Mailbox" computer and then inserted the diskette with the operator's declarations into their data-analysis laptop computer (described in Section 5.3 above). The inspectors ran an analysis program to determine which HEU feed cylinders should be present and at which feed stations they should be located, based on the operator's declarations of cylinder connections and disconnections and upon the agreed cylinder residence times. The instructions for running this program and generating the relevant report (Report Type I) are provided in Appendix F. After determining this information, the inspectors then "reset" the "Mailbox" computer, disconnected the keyboard, reinstalled the tamper-indicating enclosure over the keyboard port, and reinstalled the IAEA seal and seal wire.

When the inspectors returned to their work area in the Administration Building, they could generate any of four types of reports based on the operator's declaration data. The instructions for generating these reports are provided in Appendix F. These reports include the following:

- Report Type I - provided a listing of cylinders that should be present at the feed stations either because their residence time had not been exceeded or because there had been no disconnection event recorded for them

- Report Type II - provided detailed reports of the accumulated amount of material removed from the feed cylinders over a specified period
- Report Type III - provided a compact printed table of the amount of material removed from all cylinders over a specified time period
- Report type IV - provided a report of actual residence times for cylinders compared with their declared residence times

7. "Mailbox" Computer System - Software

7.1 Overview of the Software

As noted in Section 6, the First Line Manager (FLM) used the data-preparation computer and a special purpose data-input program to prepare the declarations regarding cylinder connection and disconnection operations. After the FLM completed the data entry for a cylinder connection or disconnection, these data were entered into the database file named INVENTOR.DBF, located in directory c: \SNRI\DBF within the data-preparation computer. When the FLM indicated that there were no additional events to be entered, the INVENTOR.DBF file was copied from the directory on the data-preparation computer to a floppy diskette, and the file on the computer was erased (initialized, or "zapped"). The FLM then inserted the floppy diskette into the "Mailbox" computer.

The software to accept the plant operator's declaration diskette into the "Mailbox" computer and allow the IAEA inspectors to download the declarations and reset the "Mailbox" computer to receive future declarations was adapted with only a few changes from software written by the IAEA for short-notice random inspections at the Westinghouse Electric Corporation Nuclear Fuel Division fuel-fabrication plant (Ref. 4).

The "Mailbox" computer also contained a file named INVENTOR.DBF, located in directory c:\SNRI\DBF. In the "Mailbox" computer, this file was initialized ("zapped") at the beginning of the Verification Experiment, and was re-initialized whenever the "Mailbox" was "reset" by the IAEA inspectors during an inspection. Between resets, the several floppy diskette entries were accumulated in the INVENTOR.DBF file within the "Mailbox". When the IAEA inspector downloaded the data from the "Mailbox", the current INVENTOR.DBF file was copied to the inspector's floppy diskette, but was not altered. When the inspector was satisfied with the data and then reset the "Mailbox" computer, the current INVENTOR.DBF file was copied to a dated backup file in directory c:\snri\backup, and the INVENTOR.DBF file was initialized (zapped). These backup files are dated by having file names assigned as "YYMMDD.DBF", the date of the reset. The backup files could be copied to floppy diskettes by inserting a floppy diskette in the "B" drive and typing

```
copy c:\snri\backup\*.*b:\
```

The IAEA inspectors then inserted the floppy diskette with the downloaded operational data into the laptop computer and ran the pre-processor program (named PREPRO). The pre-processor

program was run once per inspection. The downloaded declaration data on the floppy diskette was read into the computer and stored in file named c:\SNRI\DBF. The pre-processor program then performed a number of operations on this file:

- it eliminated duplicate records that would be present if the FLM had typed a record more than once, or had entered the same floppy diskette into the "Mailbox" computer more than once;
- it filled in the accountability data (weight % uranium and weight % U-235) for each relevant cylinder from a database file previously prepared from declarations provided by the facility operator;
- it computed the cylinder net weight by subtracting the cylinder adjusted tare weight from the cylinder adjusted gross weight;
- it assigned a minimum declared residence time from a database file prepared from a table of residence times as a function of cylinder size and net weight;
- it then added the completed INVENTOR.DBF file, minus error-correction records, to a database file named HISTORY.DBF in directory c:\SNRI\HISTORY. This file was initialized at the beginning of the Verification Experiment, and was incremented by each entry of a diskette with an INVENTOR.DBF file read by the pre-processor program.

The pre-processor program then checked for and eliminated duplicate records in case a diskette downloaded from the "Mailbox" had been read into the IAEA computer more than once, or in case a "download" had been performed more than once during an inspection. The pre-processor then made the error corrections on the HISTORY.DBF file called for by the "e"-type records on the INVENTOR.DBF file. The HISTORY.DBF file was the one used to generate the various reports that could be displayed on the screen and/or printed.

After the pre-processor finished the operations described above, the IAEA inspector could request the reports mentioned in Section 6 above; these reports could cover any desired period or periods during the entire previous history of the Verification Experiment. The report generation program was run as many times as desired.

All the programs used in connection with this activity were run under the DOS 6 operating system. The program to collect the declaration was written in C, and that part was not recompiled from the program used at the Westinghouse plant. The remainder of the coding was written in CLIPPER5 (Ref. 5). Most of the CLIPPER compilations were done using the CLIPPER RMAKE utility, but some were done using a DOS batch file. Loading of compiled code was done using BLINKER2 (Ref. 6), and used additional subprograms from the IAEA, Department of Safeguards, Division of Safeguards Information Treatment (SGIT) library (Ref. 7).

7.2 Plant Operator's Data-Preparation Computer

The facility First Line Managers (FLMs) prepared declarations of cylinder connections and disconnections using the plant operator's data-preparation computer and a program written specifically for that purpose. The software was written in CLIPPER5 and prompted the FLM entering the data to provide the required information. After each declaration, the FLM preparing the declaration could review and change what had been entered. After the preparer had indicated that no further review was desired, a printed copy of the declaration was generated. As many declarations as desired could be made during each session at the computer. When no further declarations were to be made in the current session, a floppy diskette was generated from the data in the structure of the database file INVENTOR.DBF.

The fields in the database file INVENTOR.DBF and their meaning are given in Table I. Not all of the fields were supplied through the plant operator's data-preparation computer. The fields supplied by the FLM are indicated on the second page of Table I. The remaining fields were filled in from other sources, and are discussed in later sections of this report.

The source code for the plant operator's data-preparation program, input instructions for the CLIPPER RMAKE utility, input for the BLINKER2 loader, and the batch file to execute the data entry program are all listed in Appendix G. Instructions for the plant First Line Managers, who prepare the declarations, are given in Appendix C.

The data-entry program had a password requirement to prevent unauthorized persons from making entries into the "Mailbox" computer. The Supervisor for the First Line Managers could change the password. The program to change the password was supplied to the Supervisor on a floppy diskette. When the Supervisor executed the batch file to change the password, the diskette was read from the "A" drive on the data-preparation computer. The Supervisor was prompted on how to change the password. After the program was finished, the program was deleted from the hard disk. The password was contained in the database file PASS.DBF that remained on the hard disk. The source code for the password program, input for the CLIPPER5 RMAKE utility, input for the BLINKER2 loader, the batch file to execute the password program, and the database file PASS.DBF are all listed in Appendix H. Instructions for the use of the Supervisor's password program are given in Appendix I.

Table I

Structure of the Database File INVENTOR.DBF

Field Number	Field Name	Type	Width	Description
1	ITEMID	Character	6	Cylinder Number
2	TRANS_TYPE	Character	1	Type of event (connection, disconnection, or error correction)
3	PURITY	Numeric	7	Uranium weight percent
4	ASSAY	Numeric	7	Enrichment weight percent
5	LOCATION	Character	2	Feed station number
6	GROSSWT	Numeric	15	Cylinder adjusted gross weight (grams)
7	TAREWT	Numeric	15	Cylinder adjusted tare weight (grams)
8	NETWT	Numeric	15	Cylinder adjusted net weight (grams)
9	CDATE	Date	8	Date of event being corrected, if error correction
10	CTIME	Character	8	Time of event being corrected, if error correction
11	EDATE	Date	8	Event date
12	ETIME	Character	8	Event time
13	SIZE	Character	2	Cylinder size (diameter in inches)
14	CYCLE	Numeric	2	Ordinal number of cylinder connection
15	RESID	Numeric	3	Declared minimum residence time (hours)
16	NOTES1	Character	80	Concise note line 1
17	NOTES2	Character	80	Concise note line 2
18	NOTES3	Character	80	Concise note line 3
19	NOTES4	Character	80	Concise note line 4
20	CORR_TYPE	Character	1	Corrected type of event being corrected, if changed
21	CLOCK_DATE	Date	8	"Mailbox" input date
22	CLOCK_TIME	Character	8	"Mailbox" input time

Table I (continued)

Structure of the Database File INVENTOR.DBF

Field Number	Field Name	Source of Data
1	ITEMID	Entered by First Line Manager
2	TRANS_TYPE	Entered by First Line Manager
3	PURITY	Supplied during preprocessing from plant operator's prior declaration
4	ASSAY	Supplied during preprocessing from plant operator's prior declaration
5	LOCATION	Entered by First Line Manager
6	GROSSWT	Entered by First Line Manager
7	TAREWT	Entered by First Line Manager
8	NETWT	Computed during preprocessing
9	CDATE	Entered by First Line Manager
10	CTIME	Entered by First Line Manager
11	EDATE	Entered by First Line Manager
12	ETIME	Entered by First Line Manager
13	SIZE	Taken from cylinder identification during preprocessing
14	CYCLE	Computed during preprocessing
15	RESID	Supplied during preprocessing based on cylinder size and gross weight, according to plant operator's prior declaration
16	NOTES1	Entered by First Line Manager
17	NOTES2	Entered by First Line Manager
18	NOTES3	Entered by First Line Manager
19	NOTES4	Entered by First Line Manager
20	CORR_TYPE	Entered by First Line Manager
21	CLOCK_DATE	Automatically entered by "Mailbox" computer
22	CLOCK_TIME	Automatically entered by "Mailbox" computer

7.3 The "Mailbox" Computer

7.3.1 Software Used for Data Entry

The "Mailbox" computer was a dedicated computer in a tamper-indicating enclosure. It had a special-purpose AUTOEXEC.BAT file that started when the computer was turned on. When the computer booted up, the screen prompts asked the question: "Are you an inspector?" When the IAEA inspectors were not present, the "Mailbox" computer did not have a keyboard attached. If a "y" (for yes) was not entered within 5 seconds, the default was "n" (for no) and the plant operator was instructed to enter the data diskette prepared on the data-preparation computer (see Section 7.1) into the "B" floppy diskette drive of the "Mailbox" computer. The diskette was read and the operator was instructed to remove the diskette and turn off the "Mailbox" computer. The AUTOEXEC.BAT file that resided on the "Mailbox" computer is listed in Appendix J, and the plant operator's (FLM's) instructions for entering data into the "Mailbox" computer are given in Appendix D.

The "Mailbox" computer software to collect the plant operator's declaration from the floppy diskette is listed in Appendix K. It includes five "C"-language programs that were not changed from those used during the field test of short-notice random inspections at the Westinghouse fuel fabrication plant (Reference 4) and were not re-compiled. In addition, there are the following files:

- a CLIPPER5 program (named APPEND.PRG) to append the data from the current floppy diskette to the database file accumulated since the last visit by the IAEA inspectors
- a header file used by APPEND.PRG and some of the C-language programs
- instructions to compile APPEND.PRG
- input to the BLINKER2 loader for APPEND.PRG.

The APPEND.PRG was changed slightly from the one used at the Westinghouse plant, and was re-compiled. It is called from the "C" language routine COLLECT.C.

In the "Mailbox" computer, the file INVENTOR.DBF was in directory c:.dbf. The executable program collect.exe was in directory c:.exe. The executable program append.exe was in directory c:.exe.

7.3.2 Software Used for Data Downloading

On arrival at the facility, the IAEA inspectors opened the tamper-indicating enclosure and connected the keyboard to the "Mailbox" computer. They then started the computer. In response to the question "Are you an inspector?" they responded by entering the letter "y". Then they inserted a blank floppy diskette into the "B" drive of the "Mailbox" computer and typed

"download". The declarations accumulated in the "Mailbox" computer in the database file INVENTOR.DBF since the last inspection were copied to the floppy diskette for processing in the IAEA inspector's data-analysis laptop computer. The batch file to download the plant operator's declaration from the "Mailbox" computer to the IAEA inspector's floppy diskette is listed in Appendix L. In the "Mailbox", it was loaded at c:\download. The instructions for the IAEA inspectors to download the contents of the database file INVENTOR.DBF are given in Appendix E.

The IAEA inspectors entered the floppy diskette into their laptop computer and performed the pre-processing of the data. Then they generated a screen report of the cylinders that should be present at the feed stations. If there did not appear to be any problem, they reset the "Mailbox" computer to receive new declarations, disconnected the keyboard, and closed and sealed the tamper-indicating enclosure.

The source code for the program to reset the "Mailbox" computer (named RESET.PRG), instructions to compile RESET.PRG, and instructions to load the program and generate the executable file RESET.EXE are given in Appendix M. The executable code was loaded at c: reset.exe. The instructions for the IAEA inspectors to reset the "Mailbox" computer are given in Appendix E.

7.4 The IAEA's Data-Analysis Laptop Computer

7.4.1 The Pre-processing Software

When the IAEA inspectors entered the floppy diskette with the data that had been downloaded from the "Mailbox" computer into their laptop computer, the first procedure to be executed was the pre-processor. This was invoked by typing "prepro". Prompts told the IAEA inspector to insert the diskette, and to remove it when the pre-processing was finished. During the pre-processing, any indication of an abnormal condition in the data appeared on the screen. The pre-processing step was executed once per inspection, and merged the current declaration with information that the plant operator had previously supplied, in order to fill in some of the required information. Other information was generated from the information already present (e.g., the net weight was the gross weight minus the tare weight). Error-correction-type declarations were implemented. The completed current declaration was then added to the previous declarations accumulated in the file HISTORY.DBF. This file had the same structure as the INVENTOR.DBF file.

Data for two fields of the file INVENTOR.DBF were taken from declarations supplied by the plant operator in a different form. At the beginning of the Verification Experiment, the plant operator provided a floppy diskette that contained the plant's declaration of the cylinder identification number, the purity (uranium weight percent) and the assay (U-235 weight percent) for the UF_6 contained in each cylinder to be included in the Verification Experiment. A CLIPPER5

program was written to read this diskette and convert the information into a database file that would be read by the pre-processor program. The source code for this program (named DECLARE1.PRG), the instructions to the CLIPPER RMAKE utility to compile it, the instructions to the BLINKER2 loader to load it and create an executable module (named DECL1MN.EXE), and the batch file to execute that module (named ACCOUNT.BAT) are all listed in Appendix N. An intermediate database file used in the processing (named ACCOUNT.DBF), and the final database file read by the pre-processor (named DECLARE.DBF) are also listed in Appendix N.

The plant operator's declaration of agreed minimum residence time is given in Table II. For each of the three sizes of cylinders involved in the Verification Experiment, the agreed minimum residence time was a function of the cylinder adjusted net weight ("UF_6"). This minimum residence time was the minimum time that the cylinder would remain connected at a feed station. Exceptions for unusual conditions (e.g., a plugged cylinder valve that caused a cylinder to be removed from the feed station before expiration of the agreed minimum residence time) were explained by the operator in the concise notes.

A program (named RES.PRG) was written to create three database files, one for each cylinder size, to contain the information in Table II. These database files (named N5RES.DBF, N8RES.DBF and N12RES.DBF) were then read by the pre-processor and the minimum residence time for each cylinder, based on its size and net weight, was inserted in the INVENTOR.DBF file. The source code listing for RES.PRG, instructions to the CLIPPER RMAKE utility to compile it, instructions to the BLINKER2 loader to load it and create an executable module (named RESMAIN.EXE), and the batch file to execute that module are all listed in Appendix O, along with the (identical) structures of N5RES.DBF, N8RES.DBF, and N12RES.DBF.

These database files were created before the start of the Verification Experiment. The pre-processor module had a main program (named PREPRO.PRG) that controlled the flow of information, and seven sub-programs for specific functions. These programs are listed in Appendix P. Program COPYINV.PRG copied the database file from the floppy diskette downloaded from the file INVENTOR.DBF on the "Mailbox" computer into a database file with the same name and file structure on the inspector's laptop computer. Program ELIMDUPS.PRG examined INVENTOR.DBF and deleted duplicate records (identical except for the "Mailbox" input date and time). Program FILLUP.PRG added the purity and assay from file DECLARE.DBF. It computed the cylinder adjusted net weight by subtracting the cylinder adjusted tare weight from the cylinder adjusted gross weight and extracted the cylinder size from the cylinder identification number, since two of the characters in the identification number indicated the cylinder size. It then read the minimum residence time from file N5RES.DBF, N8RES.DBF or N12RES.DBF. Program PHYINV.PRG added INVENTOR.DBF to HISTORY.DBF.

Table II

Agreed Minimum Residence Times for Cylinders To Be Connected at Feed Stations

Five-inch-diameter UF_6 Cylinders

If UF_6 is					>	7.5	kg,	then residence time	=	66	hours
If UF_6 is	≤	7.5	kg	and	>	3	kg,	then residence time	=	42	hours
If UF_6 is	≤	3	kg	and	>	0	kg,	then residence time	=	18	hours

Eight-inch-diameter UF_6 Cylinders

If UF_6 is					>	62	kg,	then residence time	=	138	hours
If UF_6 is	≤	62	kg	and	>	33	kg,	then residence time	=	114	hours
If UF_6 is	≤	33	kg	and	>	18	kg,	then residence time	=	90	hours
If UF_6 is	≤	18	kg	and	>	9	kg,	then residence time	=	66	hours
If UF_6 is	≤	9	kg	and	>	3	kg,	then residence time	=	42	hours
If UF_6 is	≤	3	kg	and	>	0	kg,	then residence time	=	18	hours

Twelve-inch-diameter UF_6 Cylinders

If UF_6 is					>	149	kg,	then residence time	=	186	hours
If UF_6 is	≤	149	kg	and	>	110	kg,	then residence time	=	162	hours
If UF_6 is	≤	110	kg	and	>	74	kg,	then residence time	=	138	hours
If UF_6 is	≤	74	kg	and	>	52	kg,	then residence time	=	114	hours
If UF_6 is	≤	52	kg	and	>	33	kg,	then residence time	=	90	hours
If UF_6 is	≤	33	kg	and	>	15	kg,	then residence time	=	66	hours
If UF_6 is	≤	15	kg	and	>	6	kg,	then residence time	=	42	hours
If UF_6 is	≤	6	kg	and	>	0	kg,	then residence time	=	18	hours

Note: the symbol UF_6 means the net weight of the UF_6 cylinder

7.4.2 The Report Generation Software

There were several reports that could be generated from the HISTORY.DBF file on the IAEA data-analysis laptop computer. This program, or any portion of it, could be run as many times as desired during an inspection. The instructions for generating reports on the data-analysis laptop computer are provided in Appendix F. The listings of the programs are provided in Appendix Q. Examples of the reports that are available are provided in Appendix R.

7.4.2.1 Report on Cylinders Attached at Feed Stations

At the time of an inspection, the IAEA inspector could generate a report which identified the cylinders that should be attached at the feed stations because their minimum residence time had not expired, or because no disconnect event had been recorded for them. In order to generate this report, the IAEA inspector entered the date and time for the start of the current inspection. The program then looked at the HISTORY.DBF file and examined each entry of a cylinder connection. Based on the date and time of the cylinder connection, and the declared minimum residence time for that cylinder, the program calculated the expiration date and time of the minimum residence time and compared this date and time with the date and time for the start of the current inspection. If the minimum residence time had not expired, the program displayed the information about the cylinder on the screen. An example report of this type is provided in Appendix R, item (2).

The program then searched the HISTORY.DBF file to seek a disconnect event for all cylinders for which a connect event had been recorded. The program then displayed on the screen the identification number of all cylinders for which a connect event had been recorded without a corresponding disconnect event prior to the start of the current inspection date and time. If any cylinder had been connected and disconnected one or more times, and finally connected but not disconnected before the date the current inspection began, the program generated a report for that cylinder based on the last connection. An example report of this type is provided in Appendix R, item (3).

If a printer was attached to the portable computer at the time that the report was generated, then reports for all such cylinders could be printed, as well as displayed on the screen.

7.4.2.2 Report on Amounts of Uranium and U-235 Removed During a Specified Interval

The computer could generate detailed reports of the accumulated amount of uranium and U-235 removed from cylinders over a specified time period. The IAEA inspector entered the starting date and time, and the ending date and time, for the period of interest. This period could be from the start of the Verification Experiment to the start of the current inspection, or any interval in between. The program then examined the HISTORY.DBF file for connect/disconnect pairs for each cylinder. Where such pairs were found, the difference between the cylinder adjusted gross

weight at the time of connection, and the cylinder adjusted gross weight at the time of disconnection was the amount of material removed. The computer could display this report on the screen, and could also print it. An example report of this type is provided in Appendix R, item (4).

7.4.2.3 Summary Report on Amounts of Uranium and U-235 Removed

The computer could generate reports, in the form of a compact table, of the amount of uranium and U-235 removed from all cylinders over a specified period. This report was only available as a printed report, and not on the screen. It contained a summary of the information from the reports described in Section 7.3.2.2. An example report of this type is provided in Appendix R, item (5).

7.4.2.4 Report on Actual Residence Times of Cylinders

The computer could generate reports of the actual residence time of a cylinder compared with its declared minimum residence time for normal feed operations. This type of report showed, for each time that a cylinder was connected and disconnected, the actual residence time (in hours) between the connect and the disconnect. It also showed the declared minimum residence time inferred from the cylinder size and adjusted net weight. The program then subtracted the declared minimum residence time from the actual residence time, and displayed this time difference. A negative residence time difference meant that the cylinder had been connected for less than the minimum declared residence time. The report also showed the difference between the time when the event was reported as having taken place, and the time when the declaration was entered into the "Mailbox" computer. A negative number implied that the declaration had been entered into the "Mailbox" before the event actually took place. Any such occurrence would need to be resolved by the IAEA inspectors in consultation with the plant operator. An example report of this type is provided in Appendix R, item (6).

8. Experience with the "Mailbox" Computer System

The "Mailbox" computer system operated successfully during the course of the Verification Experiment, which took place from December 1, 1997 through July 17, 1998. During this time, the IAEA conducted a design information verification visit followed by two scheduled inspections and five short-notice random inspections. In addition, the IAEA performed a final "close-out" inspection during the first two weeks of October 1998. There were no computer equipment failures until the very end of the experiment, when the failure of the operator's data-entry computer prevented the operator from declaring the connections and disconnections for the last two cylinders involved in the experiment. The "Mailbox" data supported the official declarations of HEU downblending reported by the plant operator and the US Government, and the "Mailbox" data were corroborated by the load-cell-based weighing system and video-surveillance data.

However, there were numerous data-entry errors that required resolution. Since the IAEA inspectors could arrive at any time on a short-notice random basis, and since cylinder feed operations take place around the clock, the data for cylinder operations (e.g., cylinder identification number, connection/disconnection date and time, and weights) had to be entered on a near-real-time, 24-hour-per-day basis. This requirement meant that Production Operations staff, rather than the nuclear-materials-accountability staff, entered the data. These data-entry errors would probably have been avoided if the data had been entered by the accountability staff, but then the data entry could not have been done on a near-real-time basis and hence would not have met the timeliness requirement set by the IAEA.

The operator maintained an extensive set of operating records and logs that facilitated the resolution of the data-entry errors. Except for one cylinder, all data-entry errors and discrepancies were resolved by the IAEA and the plant operator.

Based on the verification approach and the results of the verification activities, including the "Mailbox" declarations, the IAEA was able to conclude, with a high degree of confidence, that 3.7 metric tons of HEU had been blended down to LEU during the course of the Verification Experiment. This confirmed the US Government's declaration for the quantity of HEU downblended during this time.

References

1. D.M. Gordon and J.B. Sanborn, "An Approach to IAEA Material-Balance Verification with Intermittent Inspections at the Portsmouth Gas Centrifuge Enrichment Plant", Brookhaven National Laboratory Report BNL 51782, Brookhaven National Laboratory, Upton, New York USA, 1984.

2. L.M. Brenner, 1983 private communication, op.cit. Gordon and Sanborn 1984.

3. The Hexapartite Safeguards Project (HSP), "Safeguards Approach for Gas Centrifuge Enrichment Plants," Nuclear Materials Management, Vol. XII, No. 4, p. 30, 1983.

4. L.G. Fishbone, G. Moussalli, G. Naegele, P. Ikonomou, M. Hosoya, P. Scott, J. Fager, C. Sanders, D. Colwell, R. Ervin and C. J. Joyner, "Field Test of Short-Notice Random Inspections for Inventory-Change Verification at a Low-Enriched-Uranium Fuel-Fabrication Plant", IAEA Document STR-302/ISPO-371, Vols. 1, 2, and 3, International Atomic Energy Agency, Vienna, Austria, April 1995.

5. CLIPPER 5.01, The Nantucket Corporation, 1990.

6. BLINKER2, Assembler Software Manufacturers, Inc. ("Blincinc"), 1992.

5. SGIT PC Systems Development Handbook, R. Ellifritt, G. Hood, and K. Javorszky, International Atomic Energy Agency, Vienna, Austria, 1990.

(THIS PAGE INTENTIONALLY LEFT BLANK)

Appendix A

The IAEA Form SP-1 for the "Mailbox" Task

SP-1
TASK PROPOSAL PART

1. Task Proposal ID: 97/PST-001 **Date received in SPA:** 97-07-28

2. Task Title: Revision of "Mailbox" Software for Use During Verification Experiment at Portsmouth GDP

3. Task Category: D

R&D/IS Need and its Priority:

D4.1 New IAEA Verification Missions Important

4. Safeguards Problem Identification:

4.1 What is needed, why and when:

As part of its effort to reduce its nuclear-weapon-material stockpile, the U.S. is currently blending down 13 MTU of HEU UF6 to LEU UF6 in the cascade equipment at the Portsmouth Gaseous Diffusion Plant (GDP). This material is part of the 200 MT of fissile material that President Clinton has declared to be excess to U.S. national-security needs and which will be permanently withdrawn from the U.S. nuclear stockpile. In the blending process, a relatively small flow of HEU UF6 is blended with the relatively large flow of LEU UF6 in a cascade stage to form a blended LEU UF6 product with slightly increased enrichment.

The U.S. has proposed that the IAEA engage in a "verification experiment" with this material to assess the extent to which the IAEA can verify the U.S. declared quantities of HEU being blended down to LEU. A major feature of this verification experiment is the implementation of a Short Notice Random Inspection (SNRI) approach. Three conditions must be met for this approach to be applicable: (1) all items in the population must be available for selection for verification; (2) the plant operator must declare to the IAEA values for the nuclear-material content of items before knowing which items will be verified; and (3) the operator must not alter item identity or content after learning that an item is chosen for verification and before the verification actually occurs. The "Mailbox " concept is used to fulfill condition (2). "Mailbox" declarations are unalterable operator statements of accountancy values and operational data against which IAEA inspectors can compare the results of verification measurements. In a field test of the SNRI approach at the Westinghouse Electric Corporation Commercial Nuclear Fuel Division Fabrication Facility located in Columbia, South Carolina, USA, these operator declarations were made using a tamper-proof IAEA computer at the plant site to which the plant operator directly inserted information.

Software for the computer "Mailbox" was developed by the IAEA for use during the field test of SNRIs at the Westinghouse fuel fabrication facility. This field test was successfully completed in 1993. The software developed for that purpose must be revised for use during the HEU Downblending Verification Experiment at Portsmouth for two reasons: (1) the nuclear material strata and operating arrangements for the HEU downblending operations at Portsmouth are different from those at the Westinghouse fuel-fabrication facility; (2) several of the improvements suggested in the final report (STR-302/ISPO-371) for the Westinghouse field test should be incorporated in updated software.

The U.S. continues to blend down HEU UF6 at the Portsmouth GDP, and anticipates that blending operations there will be completed in August 1998. Accordingly, it is important that the Verification Experiment commence as soon as possible so that the IAEA can gain its desired experience in application of SNRIs (including "Mailbox" declarations by the plant operator) and design re-verification at selected locations on a short-notice random basis. This task needs to begin immediately so that the "Mailbox" software will be available by the time the other Verification-Experiment preparations have been completed and the Design Information Verification for the HEU downblending equipment at the Portsmouth GDP has been performed.

4.2 How will the task results be used and by whom?

Within the IAEA, the results of this task will be used by the Division of Operations B (SGOB1) and the Division of Concepts and Planning (SGCP) as a key verification measure during the course of the Verification Experiment for HEU Downblending at the Portsmouth Gaseous Diffusion Plant. More generally, the results of this task will be used by the IAEA to provide assurance, with a defined level of confidence, to the international community of NPT States that the U.S. has, in fact, complied with its commitment to reduce its stockpile of weapons-grade HEU by the declared quantity.

4.2 Consequences if task is not performed:

The use of a "Mailbox" for the operator declarations is essential to the SNRI approach. If a "Mailbox" is not available, then the SNRIs cannot be performed and as a consequence the HEU Downblending Verification Experiment cannot be performed. If this happens, the IAEA will miss an important opportunity to strengthen its public acceptance as a credible independent verification authority for arms-control agreements.

5. IAEA Proposed Work Outline

5.1 Major task stages with timing:

1.	Establish the functional requirements for the revised software	8/30/97
2.	Carry out the agreed revisions to the software	10/31/97
3.	Test the revised software	11/15/97
4.	Provide the revised software to the IAEA	11/30/97
5.	Document the revised software in a report	11/30/97

5.2 Estimated task completion date: 1997/11

5.3 Estimated duration in months: 3

6. Estimated Costs:

6.1 To the IAEA to conduct the task:

Person-months:	0	**Person-trips:**	0
Other costs in US$:	$0.00	**Total costs in US$:**	$0.00

6.2 To the Support Programme (SP) to conduct the task:

Person-months:	3	**Person-trips:**	2
Other costs in US$:	$5,000.00	**Total costs in US$:**	$65,000.00

Remarks: The cost estimate is only to indicate the expected order of magnitude.

7. Divisions Involved:

7.1 Requestor Division / Section / Person:

SGCP / PST / Theis

7.2 End User Division(s) / Section(s): SGCP / PSS,PST
SGOB / OB1

7.3 Support Division(s) / Section(s): SGCP / PST

8. Related SP Tasks:

SP(s) to which the task is proposed: USA

Is this a joint task for the SPs? No

(THIS PAGE INTENTIONALLY LEFT BLANK)

Appendix B

The Task Statement for the "Mailbox" Computer System

TASK AREA D - Information Processing and Evaluation

97/PST-001

Subtask: Revision of "Mailbox" Software for Use during the IAEA Verification Experiment at Portsmouth GDP

Contractor: BNL/SSN

Problem: The IAEA Verification Experiment on downblending of HEU at the Portsmouth GDP will make use of the Short Notice Random Inspection (SNRI) approach. The use of a "Mailbox" computer is essential to the SNRI approach, since the data involved is classified by the USA as Confidential National Security Information and cannot be transmitted offsite by facsimile. However, the current "Mailbox" software is not usable in its current form and must be revised to make it compatible with Portsmouth GDP nuclear material strata and operating arrangements and procedures.

Objective: Revise the "Mailbox" software to make it compatible with the nuclear material strata and operating arrangements for the HEU downblending operations at the Portsmouth Gaseous Diffusion Plant, and incorporate several of the software improvements suggested in the final report (STR-302/ISPO-371) from the SNRI field test at the Westinghouse fuel fabrication facility.

Activities:

a. Establish the functional requirements for the revised software through discussions with the IAEA and the facility, including desired improvements selected from the potential improvements identified in STR-302/ISPO-371.

b. Purchase the "Mailbox" computer

c. Complete the agreed revisions to the software

d. Test the revised software on the "Mailbox" computer

e. Provide revised software to the IAEA

f. Provide training to the inspectors who will participate in the Verification Experiment for HEU downblending at Portsmouth GDP

g. Document the revised software in a report

Schedule and Milestones:

1. Start task - September 15, 1997
2. Complete (a) - September 22, 1997
3. Complete (b) - October 7, 1997
4. Complete (c) - November 7, 1997
5. Complete (d) - November 15, 1997
6. Complete (e) - November 30, 1997
7. Complete (f) - November 30, 1997
8. Complete (g) - December 15, 1997

Funding:

1997 $65,000

Appendix C

First Line Manager's Instructions For the Plant Data-Preparation Computer

1. At the DOS prompt (C:\>), type in the word "plant" and then press [ENTER]. If you enter an incorrect password, start over.

2. Follow the instructions on the screen.

3. Please enter event times uniquely; that is, no two events should take place at exactly the same time!

   Example: Cylinder at feed position No. 1 connected at time 1405
   Cylinder at feed position No. 2 connected at time 1406

4. Save the printed record of the data entry!

5. Then follow the instructions for the "Mailbox" computer

NOTE: Cylinder identification and event type should be entered in lower case. If they are not, the program will change upper case to lower case!

(THIS PAGE INTENTIONALLY LEFT BLANK)

Appendix D

First Line Manager's Instructions for "Mailbox" Computer

The "Mailbox" computer is contained in a tamper-indicating enclosure. Plant access to the computer is limited to the power cable and the "B" drive. When the machine is turned "ON," the system will start up automatically.

All system functions are automated. When the system prompts "Turn OFF the machine," the computer must be turned OFF to restart the data-collection procedure.

Operating Instructions:

1. Turn "ON" the computer (Switch on the power strip)

2. Do not wait for a screen prompt to insert the data diskette (insert it at the question, "Are you an inspector?")

3. The system will verify the data file name and structure

   If correct, data will be appended to an existing file

   If incorrect, note the error message, correct the problem, and if necessary, repeat the procedure

4. Wait for the message, "Turn this machine off."

5. Remove diskette

6. Turn "OFF" the computer

When there is an error:

The error-message prompt will be followed by a pause. After three prompts, the message "Turn the machine OFF" will be displayed. The machine must be turned OFF. Correct the data diskette and try again later.

Possible file-handling-function error messages:

Error: drive not ready reading drive B:
Diskette is not in drive B:

Error: file not found on drive B:
The DBF file name on the diskette does not match the target DBF file name.
Put in the correct diskette.

Error: B: DBF contains extra fields
DBF file structure does not match the target file.
Put in the correct diskette.

(THIS PAGE INTENTIONALLY LEFT BLANK)

Appendix E

IAEA Inspector's Instructions for the "Mailbox" Computer

STEP 1: Download the Inventory Change Transactions from the "Mailbox" Computer

Overview:
Open the tamper-indicating enclosure and connect the keyboard allowing for full control of the computer. When the computer is turned on, the program for collection of operator's data begins automatically. You can prevent the continuation of this program by responding to the question, "Are you an Inspector?" Respond by typing "Y" to gain control of the system. If you fail to answer before the time elapses, the data-collection program will continue. To gain control of the system, you must restart the computer.

Two procedures must be completed on the "Mailbox" computer. At the beginning of the inspection, download the inventory change transactions for processing. Then reset and re-secure the "Mailbox" computer within its tamper-indicating enclosure.

Operating Instructions:

1. Open the tamper-indicating enclosure permitting access to the computer, and connect the keyboard.

2. Turn the machine "ON."

3. When the question, "Are you an Inspector?" is displayed, type "Y" *within 10 seconds*. Enter new date/time or press [ENTER] to accept the current date/time.

   Note:
   If you fail to respond with a "Y" within ten seconds, the plant's data collection program will execute. Wait until the system prompts "Turn this machine off." Turn the machine off, and begin again.

4. Insert a formatted diskette in Drive B: (the lower drive).

5. At the DOS prompt, (i.e., when you see "C:\>"), type "DOWNLOAD" and press [ENTER]. The screen should look as follows:

   Wed 11-18 15:00 c:\DOWNLOAD
   Insert a diskette into drive B:
   Press any key to continue....
   Any files on this diskette will be overwritten.
   Copying C: \SNRI\DBF\INVENTOR.DBF to B:\
   1 file(s) copied
   Wed. 11-18 15:00 C: \

5. When the DOS prompt is displayed, remove the diskette from drive B by pushing the button just below the slot. Label it with the date of the inspection. The diskette is the input for the Short-Notice Random Inspection (SNRI) pre-processor and will also serve for archive purposes.

5. Turn the machine "OFF". Do not disconnect the keyboard and do not secure the computer inside the tamper-indicating enclosure at this time.

STEP 2: Identify Cylinders That Should be Present at Feed Stations in Bldg. X-326

1. Turn "ON" the portable computer. Go into DOS.

2. At the DOS prompt "C:\>", type "prepro". This is a pre-processing procedure which is performed only once per inspection.

3. At the screen prompt, insert the diskette that has just been removed from the "B" drive of the "Mailbox" computer into the floppy disk drive of the portable computer. Press [ENTER].

4. At the next screen prompt, remove the diskette from the floppy disk drive. Follow the agreed-upon procedure for handling of the diskette.

5. At the DOS prompt "C:\>", type "reports". This will start the report generation program. Several reports are available, with output to the screen and/or printer. Respond to the prompts.

6. At the appropriate prompt, type "y" to ask for reports of cylinders which should be connected because their declared residence time has not been exceeded, or because no disconnect has been recorded. Do NOT ask for a printed copy at this time because no printer is attached. By hand, note the identification of the cylinders that should be attached, and any other information that is required immediately. Press [ENTER] to advance from screen to screen.

7. Do not ask for other reports at this time. [See STEP 4 for the other reports.]

8. If there is no problem with pre-processing of the diskette or obtaining the reports, turn "OFF" the portable computer.

9. Proceed to the instructions for resetting the "Mailbox" computer.

10. If there are problems with pre-processing of the diskette or the report generation, try downloading the inventory-change transaction data from the "Mailbox" computer using a different diskette.

STEP 3: Reset the "Mailbox" Computer

This procedure should not be executed until after the inventory-change data has been successfully read into the inspector's portable computer. This procedure copies the plant data to a dated backup file and erases the original inventory-change transaction data. The dated backup files are in the "Mailbox" computer in directory c:\snri\backup. To copy these files to a floppy disk, insert the formatted disk into the B drive at the DOS prompt following completion of item 2 on this page, and type
"copy c:\snri\backup*.*b:\".

Operating Instructions:

1. Turn the machine "ON."

2. When the question, "Are you an Inspector?" is displayed, type "Y" *within 10 seconds.*

Note: If you fail to respond with a "Y" within the ten seconds, the plant's data collection program will execute. Wait until the system prompt, "Turn this machine off", is displayed before restarting the computer.

3. At the DOS prompt C:\>, type "RESET" and press [ENTER].
   The screen should look as follows:

   C:\>RESET

   WARNING: All data in the collected inventory database will be removed.
   Are you sure you know what you are doing?

At this point you should re-confirm that the inventory change data has been successfully read into the inspector's portable computer. If you are sure that this is the case, continue.

4. Type "Y" and press [ENTER].

   The following messages will be displayed:

   num, Records copied to BACKUP.DBF
   INVENTOR.DBF Zapped
   Turn this machine "OFF," disconnect the keyboard, and secure the computer inside the tamper-indicating enclosure.

5. Turn the machine "OFF." Disconnect the keyboard. Secure the computer inside the tamper-indicating enclosure, and seal the enclosure.

Possible Error Messages:

If the data set has already been backed up and zapped, you will overwrite your backup with an empty file. If you get one of the following messages, STOP and make certain you know what you are doing.

WARNING: The collected inventory change transaction database contains no data. Are you sure you know what you are doing?

This message tells you that you have already completed this action and you should not repeat it. If you do repeat it, you will overwrite the file with blank data.

WARNING: A dated backup file already exists for the date and will be overwritten. Are you sure you know what you are doing?

This message tells you that you have already completed this action and you should not repeat it. If you do repeat it, you will overwrite the file with blank data.

(THIS PAGE INTENTIONALLY LEFT BLANK)

Appendix F

Inspector's Instructions for Generating Reports Using the Data-Analysis Laptop Computer

Step 4:
(Note: Steps 1-3 are provided in Appendix E)

1. Turn "ON" the portable computer. Go into DOS.

2. At the DOS prompt "C:\>", type "reports".

3. Follow instructions on the screen for the various types of reports that can be generated, either on the screen or on the printer.

4. Turn "OFF" the portable computer.

5. Report generation may be repeated as many times per inspection as desired.

Note: Four types of reports are available. The screen prompts for each of the types is given below.

Report Type I

1. The computer will ask 'Do you want to have reports for cylinders that should be attached to feed stations because their residence time has not been exceeded and reports for cylinders that should be attached to feed stations because there is no disconnect recorded? (y/n)'

   If the answer to this question is yes, type "y" and if the answer to this question is no, then type "n"

2. The computer will ask 'Date on which this inspection started (format is ddmmyy)'

   Enter the date on which this inspection started in the format ddmmyy, where dd is the two-digit day, mm is the two-digit month, and yy is the two-digit year

3. The computer will ask 'Time at which this inspection started (format is military time = xxxx)'

   Enter the time at which this inspection started in the format xxxx, where the time is given in military time according to the 24-hour clock

4. The computer will ask 'Do you want printed reports in addition to the screen reports? (y/n)'

   If the answer to this question is yes, type "y" and if the answer to this question is no, then type "n"

Report Type II

1. The computer will ask 'Do you want to have detailed reports of the accumulated amount of material removed from cylinders over a specified period? (y/n)'

   If the answer to this question is yes, type "y" and if the answer to this question is no, then type "n"

2. The computer will ask 'Starting date of period (format is ddmmyy), (000000 means Dec. 17, 1997)'

   Enter the starting date of the desired period in the format ddmmyy, where dd is the two-digit day, mm is the two-digit month, and yy is the two-digit year

3. The computer will ask 'Starting time of period (format is military time = xxxx)'

   Enter the starting time of the desired period in the format xxxx, where the time is given in military time according to the 24-hour clock

4. The computer will ask 'ending date of period (format is ddmmyy)'

   Enter the ending date of the desired period in the format ddmmyy, where dd is the two-digit day, mm is the two-digit month, and yy is the two-digit year

5. The computer will ask 'ending time of period (format is military time = xxxx)'

   Enter the ending time of the desired period in the format xxxx, where the time is given in military time according to the 24-hour clock

6. The computer will ask 'Do you want printed reports in addition to the screen reports? (y/n)'

   If the answer to this question is yes, type "y" and if the answer to this question is no, then type "n"

Report Type III

1. The computer will ask 'Do you want to have a compact printed table of the material removed from all cylinders over a specified time period? (y/n)'

   If the answer to this question is yes, type "y" and if the answer to this question is no, then type "n"

2. The computer will ask 'Starting date of period (format is ddmmyy), (000000 means Dec. 17, 1997)'

   Enter the starting date of the desired period in the format ddmmyy, where dd is the two-digit day, mm is the two-digit month, and yy is the two-digit year

3. The computer will ask 'Starting time of period (format is military time = xxxx)'

   Enter the starting time of the desired period in the format xxxx, where the time is given in military time according to the 24-hour clock

4. The computer will ask 'Ending date of period (format is ddmmyy)'

   Enter the ending date of the desired period in the format ddmmyy, where dd is the two-digit day, mm is the two-digit month, and yy is the two-digit year

5. The computer will ask 'Ending time of period (format is military time = xxxx)'

   Enter the ending time of the desired period in the format xxxx, where the time is given in military time according to the 24-hour clock

Report Type IV

1. The computer will ask 'Do you want to have reports of actual residence time of cylinders compared with declared residence time for normal refeed? (y/n)'

   If the answer to this question is yes, type "y" and if the answer to this question is no, then type "n"

2. The computer will ask 'Do you want printed reports in addition to the screen reports? (y/n)'

   If the answer to this question is yes, type "y" and if the answer to this question is no, then type "n"

If the reported connection declaration time difference is a negative number, this means that the actual time of data entry (as determined by the computer internal clock) precedes the declared time for that event. This discrepancy should be resolved with the plant operator. If the residence time difference is a negative number, this means that the actual residence time was less than the agreed, declared residence time. This discrepancy should be resolved with the plant operator.

(THIS PAGE INTENTIONALLY LEFT BLANK)

Appendix G

The Plant Operator's (First Line Manager's) Data-Entry Program

1) CLIPPER5 source code for the PLANT program:
file c:\snri\src\operator\plant.prg

2) Listing of input data to the CLIPPER RMAKE Utility:
file c:\snri\src\operator\plant.rmk

3) Listing of input to the BLINKER2 loader:
file c:\snri\src\operator\plant.lnk

4) Batch file to execute the PLANT program:
file c:\plant.bat

Appendix G

(1) CLIPPER5 source code for the PLANT program
file c:\snri\src\operator\plant.prg

```
*=========================================================================
* Module (File) Name :    operator.prg
*          Programmer :   Arnold L. Aronson, Brookhaven National Laboratory
*                Date :   10/07/97
*
* Description :     Program for Short Notice Random Inspection System
*                   Operator's Data Preparation Program
*
* Modification History -
*    By:                                    Date:
*    Description:
*
*=========================================================================
Function oper( bw )                                    /* Black & White */
external reading
local yesno1,yesno2,yesno3,yesno4
public data1:=space(6)
public data2:=space(2)
public data3:=space(1)
public data4:=space(6)
public data5:=space(4)
public data6:=0
public data7:=0
public data8:=space(80)
public data9:=space(80)
public data10:=space(80)
public data11:=space(80)
public data12:=space(6)
public data13:=space(4)
public data14:=space(1)

public opername:=space(20)
public badge:=space(8)

set date american
set confirm on
cls
yesno1 = ' '
@ 1,1 say "Portsmouth Gaseous Diffusion Plant"
@ 2,1 say "Plant Data Preparation Program"
@ 3,1 say dtoc(date())+' '+time()
@ 4,1 say "Do you wish to continue (y/n) ?"  get yesno1
read
if (yesno1#'y'.and.yesno1#'Y')
@ 5,1 say yesno1
quit
end
opasswd = '         '
@ 6,1 say 'Enter Password' get opasswd
read
cls
         select 0
         use c:\snri\dbf\pass.dbf alias pass exclusive
     if(opasswd#password)
     @ 7,1 say 'Invalid password'
     quit
     end
     close pass
```

```
opername = '                        '
@ 1,1 say "Enter First Line Manager's Name" get opername
badge = '          '
@ 2,1 say 'Enter badge number    ' get badge
read

            select 0
            use c:\snri\dbf\inventor.dbf alias inventor exclusive
cls
            zap
            yesno2 = ' '
            cls
            while (yesno2#'n'.and.yesno2#'N')
            skip 1
            do reading
cls
     append blank
     inventor->itemid:= data1
//    @ 1,1 say itemid
     inventor->location:=data2
//    @ 2,1 say location
     inventor->trans_type:=data3
     datax = data4
     yy = dodatea(datax)
     set date british
     inventor->edate:=ctod(yy)
     yesno = ' '
//    @ 3,1 say data4 + ' ' + datax + ' ' + yy + ' ' + dtoc(edate) get yesno
//    read
     timex = data5
     inventor->etime:=dotime(timex)
     inventor->grosswt:=data6
     inventor->tarewt:=data7
     inventor->notes1:=data8
     inventor->notes2:=data9
     inventor->notes3:=data10
     inventor->notes4:=data11
        if(trans_type='e')
     datax = data12
     yy = dodatea(datax)
     inventor->cdate:=ctod(yy)
     set date american
     timex = data13
     inventor->ctime:=dotime(timex)
     inventor->corr_type:=data14
          endif

        cls
        yesno2 = ' '
        @1,1 say 'Do you wish to enter additional records (y/n)?' get yesno2
        read
        end
        close
        cls

set device to printer
for i:=26 to 60
@ i,1 say ' '
next
set device to screen


      msgbox( , , ;
            {"Please insert a floppy disk into the A drive.", ;
```

```
            "Then press ENTER",;
            "Any existing file will be overwritten.", ;
            ""},;
            1, , , , , , )

if !diskready("A")
     msgbox(, , {"No diskette"}, 1, , , , , , )
endif

copy file c:\snri\dbf\inventory.dbf to a:\inventor.dbf
cls
if file("a:\inventor.dbf")
            msgbox( , , ;
                  {"Please remove floppy disk from the A drive.", ;
                  "Then press ENTER",;
                  "Immediately enter floppy disk in Mailbox Computer"},;
                  1, , , , , , )
else
@ 1,1 say 'File not on floppy disk in the A drive'
endif
Return NIL
function dodatea(datex)
//   american input
     m = substr(datax,1,2)
     d = substr(datax,3,2)
     y = substr(datax,5,2)
     dd = stuff('   /   /   ',1,2,d)
     mm = stuff(dd,4,2,m)
     yy = stuff(mm,7,2,y)
return yy
function dodateb(datex)
//   american input
     m = substr(datax,1,2)
     d = substr(datax,3,2)
     y = substr(datax,5,2)
     dd = stuff('   /   /   ',4,2,d)
     mm = stuff(dd,1,2,m)
     yy = stuff(mm,7,2,y)
return yy
function dotime(timex)
     a = substr(timex,1,2)
     b = substr(timex,3,2)
     d = stuff('   :   :00',1,2,a)
     e = stuff(d,4,2,b)
return e
function reading
local term,yesno3
set confirm on
set date american
data1 = '        '     // itemid
data2 = '   '          // location
data3 = ' '            // trans_type
data4 = '        '     // cdate
data5 = '      '       // ctime
data6 = 0.0            // grosswt
data7 = 0.0            // tarewt
data8 =  '                                                        '   // notes1

data9 =  '                                                        '   // notes2

data10 = '                                                        '   // notes3

data11 = '                                                        '   // notes4
```

```
data12 = '        '    // edate
data13 = '      '      // etime
data14 = ' '           // corr_type
term  :='repeat'
while  term=='repeat'
cls
@ 1,1  say  "Please  enter  the  cylinder  number  (format  is  xxyyyy)."
@ 2,1  get  data1
read
data1  =  lower(data1)
cls
@ 1,1  say  "Please  enter  the  HEU  Refeed  Position  number  (format  is  xx)."
@ 2,1  get  data2
read
if  substr(data2,2,1)  =  ' '
a  = substr(data2,1,1)
b  = stuff(data2,1,1,"")
c  = stuff(b,2,1,a)
data2  =  c
endif
cls
@ 1,1 say  "Please  enter  the  type  of  event    ( 'c'=connection  to  pigtail,"
@ 2,1 say  "                                        'd'=disconnection  from  pigtail,"
@ 3,1 say  "                                        'e'=error  correction)."
@ 4,1 get  data3
read
data3  =  lower(data3)
cls
@ 1,1 say  "Please  enter  the  date  of  the  event  (format  is  mmddyy)."
@ 2,1 get  data4
read
cls
@ 1,1 say  "Please  enter  the  time  of  the  event;(format  is  military  time
           =xxxx)."
@ 2,1 get  data5
read
cls
@ 1,1 say  "Please  enter  the  cylinder  weight  in  grams  as  measured  on  Scale
           C13  or  Scale  E05."
@ 2,1 say  "For  cylinders  being  connected  this  is  the  initial"
@ 3,1 say  "'Operations  Adjusted  Gross  In'."
@ 4,1 say  "For  cylinders  being  disconnected  this  is  the  final"
@ 5,1 say  "'Operations  Adjusted  Gross  Out'."
@ 6,1 get  data6
read
cls
@ 1,1 say  "Please  enter  the  'Adjusted  Tare'  weight  in  grams  for  the
           cylinder."
@ 2,1  get  data7
read
cls
@ 1,1 say  "Please  enter  a  concise  note  if  desired  (maximum  of  4  lines)."
@ 2,1 get  data8
@ 3,1 get  data9
@ 4,1 get  data10
@ 5,1 get  data11
read
if  (data3  =  'e')
cls
@ 1,1 say  "If  this  is  an  error  correction,  please  enter  the  date"
@ 2,1 say  "of  the  event  being  corrected  (format  is  mmddyy)."
@ 3,1 get  data12
read
@ 5,1 say  "If  this  is  an  error  correction,  please  enter  the  time"
```

```
@ 6,1 say "of the event being corrected (format is military time =xxxx)."
@ 7,1 get data13
read
@ 9,1 say "If this is an error correction and the data being corrected is"
@ 10,1 say "the event type (c [connection] or d [disconnection]), "
@ 11,1 say "please enter the correct event type."
@ 12,1 get data14
read
data14 = lower(data14)
endif
cls
@ 1,1 say 'current data for this record'
@ 3,1 say 'cylinder = ' + data1
@ 4,1 say 'HEU Refeed Position number = ' + data2
@ 5,1 say 'Type of event = ' + data3
datax = data4
yy = dodateb(datax)
@ 6,1 say 'Event Date = ' + data4+' '+'(month/day/year= '+yy+')'
timex = data5
zz = dotime(timex)
@ 7,1 say 'Event Time = ' + data5+' '+'('+zz+')'
@ 8,1 say 'Cylinder Adjusted Gross Weight = ' + str(data6,15,3)
@ 9,1 say 'Cylinder Adjusted Tare Weight  = ' + str(data7,15,3)
@ 10,1 say 'Concise Note: '
@ 11,1 say data8
@ 12,1 say data9
@ 13,1 say data10
@ 14,1 say data11
if (data3='e')
datax = data12
yy = dodateb(datax)
@ 15,1 say 'Date of event being corrected          = '+data12+' '+
           '(month/day/year= '+yy+')'
timex = data13
zz = dotime(timex)
@ 16,1 say 'Time of event being corrected          = '+data13+' '+'('+zz+')'
@ 17,1 say 'Type of corrected event (if changed) = '+data14
endif
yesno3 = ' '
@ 19,1 say 'Do you wish to re-type this record (y/n) ' get yesno3
read
if (yesno3='n'.or.yesno3='N')
term := 'stop'
set device to printer
@ 1,1 say 'Portsmouth Gaseous Diffusion Plant'
@ 2,1 say "Plant Data Preparation Program"
@ 3,1 say dtoc(date())+' '+time()
@ 4,1 say 'First Line Manager:'+' '+opername+' '+'Badge Number:'+' '+badge

@ 6,1 say 'cylinder = ' + data1
@ 7,1 say 'HEU Refeed Position number = ' + data2
@ 8,1 say 'Type of event = ' + data3
datax = data4
yy = dodateb(datax)
@ 9,1 say 'Event Date = '+data4+' '+' (month/day/year= '+yy+')'
timex = data5
zz = dotime(timex)
@ 10,1 say 'Event Time = '+data5+' '+'('+zz+')'
@ 11,1 say 'Cylinder Adjusted Gross Weight = ' + str(data6,15,3)
@ 12,1 say 'Cylinder Adjusted Tare Weight  = ' + str(data7,15,3)
@ 13,1 say 'Concise Note: '
@ 14,1 say data8
@ 15,1 say data9
@ 16,1 say data10
```

```
@ 17,1 say data11
if  (data3='e')
datax = data12
yy = dodateb(datax)
@ 18,1 say 'Date of event being corrected          = '+data12+' '+
           '(month/day/year=  '+yy+')'
timex = data13
zz = dotime(timex)
@ 19,1 say 'Time of event being corrected          = '+data13+' '+'('+zz+')'
@ 20,1 say 'Type of corrected event (if changed) = ' + data14
else
@ 18,1 say ' '
@ 19,1 say ' '
@ 20,1 say ' '
endif
//  for i :=21 to 59
 i:=21 to 25
 @ i,1 say ' '
 next
set device to screen
endif
cls
end
return NIL
```

Appendix G

2) Listing of input data to the CLIPPER RMAKE Utility:
file c:\snri\src\operator\plant.rmk

```
obj1  :=  \snri\obj\plant.obj

// Inference Rules to Compile obj's from prg's
.prg.obj:
  c:\clipper5\bin\clipper  $**  /a/b/m/n/w/o\SNRI\obj\$*
.c.obj:
  cl /c /AL /Dalt /Z1 /Fpa /Gs $*.c
  copy $*.obj ..\..\obj

// Dependency list for prg modules
SNRI\obj\plant.obj:  \SNRI\SRC\operator\plant.prg

// Dependency list for exe module
\SNRI\exe\opermain.exe:  $(obj1)
 c:\blinker2\blinker  @\snri\src\operator\plant.lnk
```

3) Listing of input to the BLINKER2 loader:
file c:\snri\src\operator\plant.lnk

```
NOBELL
FILE  \snri\obj\plant
OUTPUT  plant
LIBRARY c:\lib\v51\sgit51
LIBRARY c:\clipper5\lib\clipper
LIBRARY c:\clipper5\lib\extend
LIBRARY c:\clipper5\lib\terminal
LIBRARY c:\clipper5\lib\dbfntx
LIBRARY c:\clipper5\lib\nt250
```

4) Batch file to execute the PLANT program:
file c:\plant.bat

```
@echo off
plant %1
```

Appendix H

The FLM Supervisor's Password Program

1) CLIPPER5 source code for the PASSWORD program:
file c:\snri\src\operator\pass.prg

2) Listing of input data to the CLIPPER RMAKE utility:
file c:\snri\src\operator\pass.rmk

3) Listing of input to the BLINKER2 loader:
file c:\snri\src\operator\pass.lnk

4) Batch file to execute the PASSWORD program:
file c:\pass.bat

5) Structure of the database file PASS.DBF:
file c:\snri\dbf\pass.dbf

Appendix H

1) CLIPPER5 source code for the PASSWORD program:
file c:\snri\src\operator\pass.prg

```
*==========================================================================
* Module (File) Name  : pass.prg
*          Programmer  : Arnold L. Aronson
*                Date  : 10/07/97
*
* Description  : Main driver program for Short Notice Random Inspection System
*                Operator's Data Preparation
*
* Modification History -
*    By:                                    Date:
*    Description:
*
*==========================================================================
Function passwd( bw )                                    /* Black & White */
set confirm  on
        select 0
        use c:\snri\dbf\pass.dbf alias pass exclusive
cls
@ 7,1 say 'Current password = '+ password
yesno3 = 'n'
@ 8,1 say 'Do you want to change the password ?'
@ 9,1 get yesno3
read
     if(yesno3='y')
     npasswd1 = '          '
     @ 10,1 say 'Enter new password' get npasswd1
     read
     cls
     npasswd2 = '          '
     @ 11,1 say 'Enter new password again' get npasswd2
     read
     cls
         if(npasswd1=npasswd2)
         @ 12,1 say 'New password accepted'
         pass->password:=npasswd2
         else
         @ 12,1 say 'Problem entering new password. Start again.'
         quit
         endif
     endif
     close
Return NIL
```

2) Listing of input data to the CLIPPER RMAKE utility:
file c:\snri\src\operator\pass.rmk

```
obj1  :=  \snri\obj\pass.obj

// Inference Rules to Compile obj's from prg's
.prg.obj:
  c:\clippper5\bin\clipper  $**  /a/b/m/n/w/o\SNRI\obj\$*
.c.obj:
  cl /c /AL /Dalt /Z1 /Fpa /Gs $*.c
  copy $*.obj ..\..\obj

// Dependency list for prg modules
SNRI\obj\pass.obj:  \SNRI\SRC\operator\pass.prg

// Dependency list for exe module
\SNRI\exe\passmain.exe:  $(obj1)
   c:\blinker2\blinker  @\snri\src\operator\pass.lnk
```

3) Listing of input to the BLINKER2 loader:
file c: \snri\src\operator\pass.lnk

```
NOBELL
FILE  \snri\obj\pass
OUTPUT  passmain
LIBRARY c:\lib\v51\sgit51
LIBRARY c:\clipper5\lib\clipper
LIBRARY c:\clipper5\lib\extend
LIBRARY c:\clipper5\lib\terminal
LIBRARY c:\clipper5\lib\dbfntx
LIBRARY c:\clipper5\lib\nt250
```

4) Batch file to execute the PASSWORD program:
file c:\pass.bat

```
@echo off
c:
copy a:\passmain.exe
passmain %1
erase passmain.exe
```

5) Structure of the database file PASS.DBF:
file c:\snri\dbf\pass.dbf

```
Structure for database : C:PASS.dbf
Number of data records :        1
Date of last update    : 10/14/97
Field   Field Name   Type        Width     Dec
    1   PASSWORD     Character       8
** TOTAL **                         9
```

Appendix I

FLM Supervisor's Instructions for the Password Program

To see the current password for the plant data-entry program and/or change the password:

1. At the DOS prompt (C:\>), put the floppy diskette labeled "supervisor" into the "A" drive.
2. Type in the word "Pass" and press [ENTER]. Be patient!
3. Follow the instructions on the screen.
4. At the DOS prompt (C:\>), remove the floppy diskette from the "A" drive.

(THIS PAGE INTENTIONALLY LEFT BLANK)

Appendix J

The AUTOEXEC.BAT File for the "Mailbox" Computer

```
@echo off

rem Copy COMMAND.COM to the Virtual Disk
copy command.com d:\ > nul
if exist d:\command.com set comspec=d:\command.com

cls
REM *  The answer to the following question will default to "n" after five
REM *  seconds if no answer is given.  Since non-inspectors will not have
REM *  a keyboard attached to answer with, they will effectively be forced
REM *  to accept "No" as their answer.
C:\snri\be ask "Are you an Inspector? " ny DEFAULT=n TIMEOUT=10 Bright Cyan
If errorlevel 2 goto inspector
echo n I guess not.
C:\snri\sleep 5
REM *  If the data file on the diskette is named something other than
REM *  "INVENTOR.DBF", change the following line to read
REM *  "c:\snri\exe\collect <database.dbf>"
REM *  For example:
REM *  c:\snri\exe\collect invent.dbf
c:\collect
REM *  The following line will never actually be executed, since the collect
REM *  program intentionally hangs the system instead of exiting.
goto end

REM *  The rest of the autoexec bat is only executed if the user typed "y"
REM *  when asked the above question.
:inspector

rem Load Pushdir
rem lh c:\snri\pushdir
c:\snri\pushdir
c:\snri\popdir

rem Load Doskey
rem lh c:\dos\doskey.com
c:\dos\doskey.com
c:\dos\doskey ls=ls -CF $1 $2 $3 $4 $5 $6 $7 $8 $9
c:\dos\doskey ll=ls -1F $1 $2 $3 $4 $5 $6 $7 $8 $9

rem Set Environment

rem Tell all smart programs where to put temporary files
REM *set tmp=c:\tmp
set temp=c:\tmp
if exist %TMP%\*.tmp del %TMP%\*.tmp

rem Done setting up the environment

rem Display Version and Current Date and Time
ver
REM * tm /1
date
time

:end
```

(THIS PAGE INTENTIONALLY LEFT BLANK)

Appendix K

"Mailbox" Software to Collect the Plant Operator's Declaration

1) C-language source code programs
file c:\snri\src\collect\collect.c
file c:\snri\src\collect\coma.c
file c:\snri\src\collect\dbf_util.c
file c:\snri\src\collect\fstruct.c
file c:\snri\src\collect\harderr.c

2) Header file snri.h:
file c:\snri\src\collect\snri.h

3) Source code for the CLIPPER5 program APPEND.PRG:
file c:\snri\src\append\appends.prg

4) Instructions to compile APPEND.PRG:
file c:\snri\src\append\appends.bat

5 Input to the BLINKER2 loader for APPEND.PRG:
file c:\snri\src\append\appends.lnk

Appendix K

1) C-language source code programs
file c:\snri\src\collect\collect.c
file c:\snri\src\collect\coma.c
file c:\snri\src\collect\dbf_util.c
file c:\snri\src\collect\fstruct.c
file c:\snri\src\collect\harderr.c

```
/*
 * COLLECT
 *
 * This is the main function of the SNRI module that collects the data
 * from the operator on a daily basis.  It is run on a machine that is
 * contained in a sealed box, giving the operator access only to the
 * power switch and the B: drive.  This program is executed from the
 * AUTOEXEC.BAT file located on the C: drive.  Since the operator will
 * have no access to the keyboard, all errors, including hardware errors
 * have to be dealt with automatically.
 *
 */

#include  <stdlib.h>
#include  <stdio.h>
#include  <conio.h>
#include  <dos.h>
#include  "snri.h"

void main(int argc, char **argv)
{
      char  filename[16];
      FILE  *fp;
      int  n_filenotfound_error = 0, n_structure_error = 0;
      int  file_ok;

      /*install our hardware error handler on the hardware problem interrupt*/
      _harderr(handler);
      clrscr();

      /* define the filename from which we should be appending data */
      if (argc == 2) {
              sprintf(filename, "%s\\%s", DRIVE, argv[1]);
      } else {
              sprintf(filename, "%s\\%s", DRIVE, FILENAME);
      }

      /*display the message telling the operator to insert the data diskette*/
      /*in drive B: and give him a chance to do it                          */
      printf(INSERT_MSG, DRIVE);
      sleep(WAIT_TIME);

      /* loop until the file has been located */
      /* and the structure has been verified  */
      file_ok = FALSE;
      while (!file_ok) {

            /* attempt to open the file on the B: drive */
            /* if this produces a hardware error (i.e., no diskette), DOS */
            /* will automatically call our hardware error handler         */
            while ((fp = fopen(filename, "rb")) == NULL) {
```

```
                /* if the file is not present (this is NOT a hardware    */
                /* error), display an error message, pause, and retry     */
                printf("Error: file not found on drive %s\n", DRIVE);
                if (++n_filenotfound_error < MAXERRORS) {
                      printf("Retry in %d seconds\n", WAIT_TIME);
                      sleep(WAIT_TIME);
                } else {
                      /* unless this is the third such error, in which   */
                      /* case it's time to give up                       */
                      coma();
                }
          }

          /* once the file has been successfully opened, verify the      */
          /* structure against the master file to which we're appending */
          if (!fstruct(fp, filename)) {
                /* if the structure is in error, display an error        */
                /* message, pause, and retry                             */
                if (++n_structure_error < MAXERRORS) {
                      printf("Retry in %d seconds\n", WAIT_TIME);
                      sleep(WAIT_TIME);
                      fclose(fp);
                } else {
                      /* again, unless this is the third such error, in */
                      /* which case, die                                */
                      coma();
                }
          }     else {
                /* if the structure was verified, we should get out of   */
                /* this loop                                             */
                file_ok = TRUE;
          }

      }

      /* close the file from which we're appending */
      fclose(fp);

      /* attempt to open the program which actually appends the data, just */
      /* to make sure it's there                                          */
      if ((fp = fopen(APPEND_COMMAND, "rb")) == NULL) {
            /* if it's not, there no sense in retrying */
            printf("Error: %s not found\n", APPEND_COMMAND);
      } else {
            /* if it is, close it to avoid problems of having it open twice*/
            fclose(fp);
            /* ... and execute it */
            system(APPEND_COMMAND);
      }

      /* Now that we're done, it's time to go into a deep, deep sleep */
      coma();
}
```

```
/*
 * coma()
 *
 * This function prints the final message from the program, and then
 * goes into a deep, deep sleep.
 *
 */

#include <stdio.h>
#include <conio.h>

void coma(void)
{
	cputs("\r\nTurn this machine off.");
	#ifdef DEVELOPMENT
		exit(1);
	#else
		while(1)
			sleep(10);
	#endif
}
```

```
/*
 * This is a collection of various functions which are all useful
 * for reading the header of a dbase file
 *
 */

#include <stdio.h>
#include "snri.h"


/* read_header(hd, fp) - read enough data from fp to fill hd
 *
 * If too little data is read, or an error occurs while reading,
 * return FALSE, otherwise return TRUE.
 */
int read_header(header *hd, FILE *fp)
{
	int nread;

	nread = fread((char *) hd, sizeof(char), sizeof(header), fp);
	if (nread != sizeof(header) || ferror(fp))
		return FALSE;
	else {
#ifdef DEBUG
		dump_header(hd);
#endif
		return TRUE;
	}
}


/* read_fields(f, n, fp) - read enough data from fp to fill n copies of f
 *
 * If too little data is read, or an error occurs while reading,
 * return FALSE, otherwise return TRUE.
 */
int read_fields(field *f, int n, FILE *fp)
{
	int nread;

	nread = fread((char *) f, sizeof(char), (sizeof(field) * n), fp);
	if (nread != (sizeof(field) * n) || ferror(fp))
		return FALSE;
	else {
#ifdef DEBUG
		dump_fields(f, n);
#endif
		return TRUE;
	}
}


/* check_header(hd) - verify that this does represent the header of
 *                    a dbase file
 */
int check_header(header *hd)
{
	if (hd->last_update.year < 0 || hd->last_update.year > 99) {
		return FALSE;
	} else if (hd->last_update.month < 1 || hd->last_update.month > 12) {
		return FALSE;
	} else if (hd->last_update.day < 1 || hd->last_update.day > 31) {
		return FALSE;
```

```
        } else if (hd->no_of_records < 0L) {
                return FALSE;
        } else if (hd->header_length < 0) {
                return FALSE;
        } else if (hd->record_length < 0) {
                return FALSE;
        }

        return TRUE;
}


#ifdef DEBUG
/* dump_header(hd) - debugging function to print the header info */
void dump_header(header *hd)
{
        int i;
        char *t;

        t = (char *) hd;
        for (i = 0; i < sizeof(header); i++, t++) {
                fprintf(stderr, "%02X ", *t);
                if (((i+1) % 4) == 0)
                        fprintf(stderr, " ");
                if (((i+1) % 20) == 0)
                        fprintf(stderr, "\n");
        }

        fprintf(stderr, "\n\
dBASE Header info:\n\
\tLast Update: %d.%d.%d\n\
\tNumber of Records: %ld\n\
\tHeader Size: %d\n\
\tRecord Length: %d\n",
                        hd->last_update.year,hd->last_update.month,hd->last_update.
                          day,
                        hd->no_of_records,
                        hd->header_length,
                        hd->record_length);
}


/* dump_fields(f, n) - debugging function to print the field info */
void dump_fields(field *f, int n)
{
        int i;

        fprintf(stderr, "Field List:\n");
        for (i = 0; i < n; i++) {
                fprintf(stderr, "\t%-15s\t%c\t%2d\t%2d\n",
                                f[i].name, f[i].type, f[i].length, f[i].decimals);
        }
}
#endif
```

```
/*
 * fstruct(fp_source,  sourcename)
 *
 * Compare the structure of the source file with that of the target file.
 * Return TRUE if they match and FALSE if they don't.
 *
 * "Matching" is defined as all of the fields in the source file
 * appearing in the target file, with the same data types, lengths,
 * and decimal lengths.  The target file can have extra fields which
 * are ignored.
 * The source file is already open and is pointed to by fp_source. The
 * target file has to be opened (and closed) by this function.
 */

#include  <stdlib.h>
#include  <stdio.h>
#include  <string.h>
#include  "snri.h"

int  fstruct(FILE  *fp_source,  char  *sourcename)
{
      FILE  *fp_target;
      header  hd_source,  hd_target;
      field  *fd_source,  *fd_target;
      int  nfields_source,  nfields_target,  i,  j;

      /*  read the header information from the source file */
      if  (!read_header(&hd_source,  fp_source))  {
            printf("Error:  unable  to  read  header  information  from  %s\n",
               sourcename);
            return  FALSE;
      }

      /*  check the validity of the source file's header */
      if  (!check_header(&hd_source))  {
            printf("Error:  %s  is  not  a  dbase  file\n",  sourcename);
            return  FALSE;
      }

      /*  read the field information from the source file */
      nfields_source  =
                  (hd_source.header_length  -  sizeof(header)  -  1)  /
                     sizeof(field);
      fd_source  =  (field  *)  calloc(nfields_source,  sizeof(field));
      if  (!read_fields(fd_source,  nfields_source,  fp_source))  {
            printf("Error:  unable  to  read  field  information  from  %s\n",
               sourcename);
            return  FALSE;
      }

      /*  open the target file */
      if  ((fp_target  =  fopen(TARGET_DBF,  "rb"))  ==  NULL)  {
            printf("Error:  unable  to  open  target  dbf  file  %s\n",  TARGET_DBF);
            return  FALSE;
      }

      /*  read the header information from the target file */
      if  (!read_header(&hd_target,  fp_target))  {
            printf("Error:  unable  to  read  header  information  from  %s\n",
               TARGET_DBF);
            fclose(fp_target);
            return  FALSE;
      }
```

```
    /* check the validity of the target file's header */
    if (!check_header(&hd_target)) {
          printf("Error: %s is not a dbase file\n", TARGET_DBF);
          fclose(fp_target);
          return FALSE;
    }

    /* read the field information from the target file */
    nfields_target =
                (hd_target.header_length - sizeof(header) - 1) /
                   sizeof(field);
    fd_target = (field *) calloc(nfields_target, sizeof(field));
    if (!read_fields(fd_target, nfields_target, fp_target)) {
          printf("Error: unable to read field information from %s\n",
             TARGET_DBF);
          fclose(fp_target);
          return FALSE;
    }

    /* if the source file has more fields than the target file, */
    /* it automatically doesn't match.                          */
    if (nfields_source > nfields_target) {
          printf("Error: %s contains extra fields\n", sourcename);
          fclose(fp_target);
          return FALSE;
    }

    /* for each field in the source file ... */
    for (i = 0; i < nfields_source; i++) {
          #ifdef DEBUG
          printf("%s\t%c\t%d\t%d\n", fd_source[i].name, fd_source[i].type,
                      fd_source[i].length, fd_source[i].decimals);
          #endif
          /* ... check all of the fields in the target file for a match */
          for (j = 0; j < nfields_target; j++) {
                #ifdef DEBUG
                printf("\t%s\t%c\t%d\t%d\n", fd_target[j].name,
                   fd_target[j].type,
                         fd_target[j].length, fd_target[j].decimals);
                #endif
                /* if a match is found ... */
                if ((strcmp(fd_source[i].name, fd_target[j].name) == 0) &&
                         (fd_source[i].type == fd_target[j].type) &&
                         (fd_source[i].length == fd_target[j].length) &&
                         (fd_source[i].decimals == fd_target[j].decimals)){
                      /* ... don't check the rest of the target fields */
                      break;
                }
          }
          /* if we exited the inner loop because we ran through all the */
          /* fields ... */
          if (j == nfields_target) {
                /* ... there must have been no match */
                printf("Error: %s contains extra fields\n", sourcename);
                fclose(fp_target);
                return FALSE;
          }
    }
    /* if we passed all the checks, the files must match */
    fclose(fp_target);
    return TRUE;
}
```

```
/* This program traps disk errors and prompts the user for action. */
/* Try running it with no disk in drive A to invoke its functions. */

#include  <stdio.h>
#include  <ctype.h>
#include  <dos.h>
#include  <fcntl.h>
#include  "snri.h"

int  buf[500];

/* Define the error messages for trapping disk problems. */
static char *err_msg[] = {
      "write protect",     "unknown unit",
      "drive not ready",   "unknown command",
      "data error (CRC)",  "bad request",
      "seek error",        "unknown media type",
      "sector not found",  "printer out of paper",
      "write fault",       "read fault",
      "general failure",   "reserved",
      "reserved",          "invalid disk change"
} ;

/* Define the array to hold the number of errors of each type encountered */
static int nerror[] = {
      0, 0,
      0, 0,
      0, 0,
      0, 0,
      0, 0,
      0, 0,
      0, 0,
      0, 0
} ;


/* function to print messages on the screen */
static void mesg(char *s)
{
      while (*s)
            bdos(2, *s++, 0);
}


/* Pragma warn -par reduces warnings which occur due to the nonuse of the*/
/* parameter devhdr */
#pragma warn -par

/* The hardware error handler */
void far handler(unsigned deverr, unsigned errval, unsigned *devhdr)
{
      static char msg[80];
      int drive, errorno;
      long i;

      /* If this is not a disk error then another device is having trouble */
      if (deverr & 0x8000) {
            mesg("Device error"); /* report the error */
            /* return to the program directly, requesting abort */
            _hardretn(5);                /* 5 = DOS "access denied" error */
      }
      drive = deverr & 0x00FF; /* otherwise it was a disk error */
      errorno = errval & 0x00FF;
```

```
        /* report which error it was */
        sprintf(msg, "Error:%s on drive %c:\r\n", err_msg[errorno],'A' + drive);
        mesg(msg);

        /* increment the error count */
        nerror[errorno]++;
        /* if the error count is excessive, go to sleep and never wake up */
        if (nerror[errorno] >= MAXERRORS) {
              coma();
        }

        /* Tell the user we will try again after a pause */
        sprintf(msg, "Retry in %d seconds\r\n", WAIT_TIME);
        mesg(msg);
        /* Prompt again to insert the correct diskette */
        sprintf(msg, INSERT_MSG, DRIVE);
        mesg(msg);
        /* And pause.  We can't call the sleep() function here because of */
        /* conflicts with DOS, so we have to fake it                      */
        for (i = 0; i < (WAIT_TIME * 2000000); i++) ;   /* sleep(WAIT_TIME) */

        /* force a retry */
        _hardresume(_HARDERR_RETRY);

}
#pragma warn +par
```

Appendix K

2) Header file snri.h
file c:\snri\src\collect\snri.h

```
#define DRIVE "B:"
#define FILENAME "INVENTOR.DBF"
#define PHYINV "PHYINV.DBF"

#ifdef CLIPPER
      #define TARGET_DBF "C:\SNRI\DBF\INVENTOR.DBF"
      #define PHYINV_DBF "C:\SNRI\DBF\PHYINV.DBF"
      #define BACKUP_DIR "C:\SNRI\BACKUP"
#else
      #define TARGET_DBF "C:\\SNRI\\DBF\\INVENTOR.DBF"
      #define APPEND_COMMAND "C:\\SNRI\\EXE\\APPEND.EXE"
      #define INSERT_MSG "Insert the data diskette into drive %s\r\n"
      #define MAXERRORS 3
      #define WAIT_TIME 5

      #define TRUE 1
      #define FALSE 0

      typedef struct {
              char year;
              char month;
              char day;
      } date;

      typedef struct {
              char memo_entry;
              date last_update;
              long no_of_records;
              int header_length;
              int record_length;
              char reserved [20];
      } header;

      typedef struct {
              char name[11];
              char type;
              int offset;
              int spare;
              char length;
              char decimals;
              char reserved[14];
      } field;

      /* function prototypes */
      void far handler(unsigned, unsigned, unsigned *);
      void coma(void);
      int read_header(header *, FILE *);
      int read_fields(field *, int, FILE *);
      int check_header(header *);
      int fstruct(FILE *, char *);
      #ifdef DEBUG
      void dump_header(header *);
      void dump_fields(field *, int);
      #endif
#endif
```

3) Source code for the CLIPPER5 program APPEND.PRG:
file c:\snri\src\append\appends.prg

```
/*
 * APPEND.PRG
 *
 * This program appends the data from the new data file located on the
 * diskette.  The calling program has already performed all of the error
 * checking, so none of that is necessary here.
 */

#define CLIPPER

#include "snri.h"

local target_dbf, source_dbf, nrecs

set date american

target_dbf = TARGET_DBF
source_dbf = DRIVE + "\" + FILENAME

use (target_dbf)
copy structure to tmp
close

use tmp
append from (source_dbf)
replace all clock_date with date()
replace all clock_time with time()
close

use (target_dbf)
nrecs = reccount()
append from tmp
? reccount() - nrecs, "Records appended.", reccount(), "Records collected to
   date."
close

erase tmp.dbf

return   nil
```

4) Instructions to compile APPEND.PRG:
file c:\snri\src\append\appends.bat

```
c:\clipper5\bin\clipper  appends.prg
c:\blinker2\blinker  @\snri\src\append\appends.lnk
```

5) Input to the BLINKER2 loader for APPEND.PRG:
file c:\snri\src\append\appends.lnk

```
NOBELL
FILE  \snri\src\append\appends
OUTPUT  append
LIBRARY c:\lib\v51\sgit51
LIBRARY c:\clipper5\lib\clipper
LIBRARY c:\clipper5\lib\extend
LIBRARY c:\clipper5\lib\terminal
LIBRARY c:\clipper5\lib\dbfntx
LIBRARY c:\clipper5\lib\nt250
```

(THIS PAGE INTENTIONALLY LEFT BLANK)

Appendix L

Batch File to Download the Plant Operator's Declaration from the "Mailbox" Computer to the IAEA Representative's Floppy Diskette

file c:\snri\src\download\download.bat

```
@echo off

REM
REM DOWNLOAD.BAT
REM
REM This is the Data Removal module of the SNRI system.  All it does is
REM copy the INVENTOR.DBF file from the C:\SNRI\DBF directory to a diskette
REM in the B: drive.
REM

set TARGET_DRIVE=B:
set SOURCE_DRIVE=C:
set DIRECTORY=\SNRI\DBF
set FILE=INVENTOR.DBF

echo Insert a diskette into drive %TARGET_DRIVE%
pause
echo Copying %SOURCE_DRIVE%%DIRECTORY%\%FILE% to %TARGET_DRIVE%\
copy %SOURCE_DRIVE%%DIRECTORY%\%FILE% %TARGET_DRIVE%\
```

(THIS PAGE INTENTIONALLY LEFT BLANK)

Appendix M

The RESET Program to Reset the "Mailbox"

1) CLIPPER5 source code for program RESET.PRG:
file c:\snri\src\reset\reset.prg

2) Instructions to compile RESET.PRG:
file c:\snri\src\reset\reset.bat

3) Input to the BLINKER2 loader for RESET.PRG:
file c:\snri\src\reset\reset.lnk

Appendix M

1) CLIPPER5 source code for program RESET.PRG:
file c:\snri\src\reset\reset.prg

```
/*
 * RESET.PRG
 *
 * This is the process to be run by the Inspector before leaving the
 * facility to secure the sealed computer for continued use by the
 * Operator.  It copies the Inventory data file to a dated backup
 * file, and then zaps the Inventory file.
 */

#define CLIPPER

#include "..\collect\snri.h"

local source_dbf, backup_dbf, nrecs, c, today

set date ansi

? "WARNING: All data in the Collected Inventory database will be removed."
? "Are you sure you know what you're doing? "

c = inkey(0)

if (c != asc("y") .and. c != asc("Y"))
      return
endif

/* the source for purposes of backing up, is the target of the append */
source_dbf = TARGET_DBF

today = dtoc(date())
backup_dbf = BACKUP_DIR + "\" + substr(today, 1, 2) + substr(today, 4, 2) +
               substr(today, 7, 2) + ".DBF"

use (source_dbf)
nrecs = reccount()

if (nrecs == 0)
      ?
      ? "WARNING: The Collected Inventory database contains no data."
      ? "Are you sure you know what you're doing? "

      c = inkey(0)

      if (c != asc("y") .and. c != asc("Y"))
            return
      endif
endif

if (file((backup_dbf)))
      ?
      ? "WARNING: A dated backup file already exists"
      ? "           for this date and will be overwritten."
      ? "Are you sure you know what you're doing? "

      c = inkey(0)
```

```
      if (c != asc("y") .and. c != asc("Y"))
            return
      endif
endif

copy to (backup_dbf)
?
? nrecs, "Records copied to " + backup_dbf
zap
?
? source_dbf + " Zapped."
?
? "Turn this machine off, disconnect the keyboard, and secure "
? "the computer inside the tamper-indicating enclosure."
```

Appendix M

2) Instructions to compile RESET.PRG:
file c:\snri\src\reset\reset.bat

```
c:\clipper5\bin\clipper  reset.prg
c:\blinker2\blinker  @\snri\src\reset\reset.lnk
```

3) Input to the BLINKER2 loader for RESET.PRG:
file c:\snri\src\reset\reset.lnk

```
NOBELL
FILE  \snri\src\reset\reset
OUTPUT  reset
LIBRARY c:\lib\v51\sgit51
LIBRARY c:\clipper5\lib\clipper
LIBRARY c:\clipper5\lib\extend
LIBRARY c:\clipper5\lib\terminal
LIBRARY c:\clipper5\lib\dbfntx
LIBRARY c:\clipper5\lib\nt250
```

Appendix N

Programs to Incorporate the Plant Operator's Declarations
of
Purity (Uranium Weight Percent) and Assay (U-235 Weight Percent)

1) CLIPPER5 source code for program DECLARE1.PRG:
file c:\snri\src\prepro\declare1.prg

2) Instructions to the CLIPPER5 RMAKE utility to compile DECLARE1.PRG:
file c:\snri\src\prepro\decl1.rmk

3) Input to the BLINKER2 loader for DECLARE1.PRG:
file c:\snri\src\prepro\decl1.lnk

4) The batch file to execute DECL1MN.EXE:
file c:\snri\src\prepro\account.bat

5) Structure of the database file ACCOUNT.DBF:
file c:\snri\dbf\account.dbf

6) Structure of the database file DECLARE.DBF:
file c:\snri\dbf\declare.dbf

Appendix N

1) CLIPPER5 source code for program DECLARE1.PRG
file c:\snri\src\prepro\declare1.prg

```
*==========================================================================
* Module (File) Name :  declare1.prg
*          Programmer :  Arnold L. Aronson
*                Date :  10/07/97
*
* Description: Auxiliary program for Short Notice Random Inspection System
*              Prepare account.dbf file from operator's accountability data
*              Declare.dbf is extracted from account.dbf.
*              The declare.dbf file contains 3 fields:cylinder, nat, and
*              enrich.
*              These are taken from fields: Container ID, Purity, and Assay.
*
* Modification History -
*    By:                                  Date:
*    Description:
*
*==========================================================================
Function decl1( bw )                                    /* Black & White */
field cylo, matcode, unit, grosswt, tarewt, netwt, nato, enricho, gramsu
field grams235
field cylinder, nat, enrich
local dummy,nrec, total_records, pr, yesno
local m, data1, data2, data3, data4, cyloz

pr = 'y'
cls
          use c:\snri\dbf\declare.dbf   new alias declare exclusive
          zap
          use c:\snri\dbf\account.dbf new alias account exclusive
          zap
          msgbox( , , ;
                {"Please insert the diskette with the", ;
                "accountability data into the A drive.",;
                "Then press ENTER", ;
                ""},;
                1, , , , , , )

if !diskready("A")
     msgbox(, , {"No diskette"}, 1, , , , , , )
     return NIL
endif

          append from a:\iaea.dta sdf
          go top
          m = 0
          do while .not. eof()
          if cylo # '         '
          m = m + 1
          cyloz = substr(cylo,3,8)
          data1 = nato/100000
          data2 = enricho/1000000
          if m < 20
          @ m,1 say str(m)+' '+cyloz + ' ' + str(nato,5,0) + ' ' +
                str(enricho,6,0) + ' ' + str(data1,7,5) + ' ' + str(data2,8,6)
          endif
          select declare
          append blank
```

```
            declare->cylinder:=cyloz
            declare->nat:=data1
            declare->enrich:=data2
            if pr='y'
            set device to printer
            data3 = nat*100.0
            data4 = enrich*100.0
            @ m,1 say cylinder+'       '+str(data3,6,2)+'     '+str(data4,6,2)
            set device to screen
            endif
            endif
            select account
            skip 1
            enddo

            select declare
            go 25
            nrec =  recno()
            dummy = cylinder
            cls
            @ 1,1 say 'cylinder'+' '+cylinder+' '+dummy + '   ' + str(nrec)
            go top
            index on cylinder to c:\snri\dbf\cylinder
            close declare
            select account
            zap
            close account
            msgbox( , , ;
                  {"Please remove floppy disk from the A drive.", ;
                  "Then press ENTER",;
                  ""},;
                  1, , , , , , )

use c:\snri\dbf\declare index c:\snri\dbf\cylinder new alias declare
        exclusive
go top
nrec = reccount()
@ 2,1 say 'total records = '+str(nrec)
if pr = 'y'
  set device to printer
  @ m+1,1 say 'total records = '+str(nrec)
  set device to screen
endif

data3=nat*100.0
data4= enrich*100.0
@ 3,1 say 'cylinder'+' '+cylinder+' '+str(nrec)+' '+str(data3,6,2)+ ' '
          +str(data4,6,2)
seek dummy
data3=nat*100.0
data4= enrich*100.0
@ 4,1 say 'cylinder'+' '+cylinder+' '+str(nrec)+' '+str(data3,6,2)+ ' '
          +str(data4,6,2)
go bottom
data3=nat*100.0
data4= enrich*100.0
@ 5,1 say 'cylinder'+' '+cylinder+' '+str(nrec)+' '+str(data3,6,2)+ ' '
          +str(data4,6,2)
yesno = ' '
@ 5,1 get  yesno
Return NIL
```

2) Instructions to the CLIPPER5 RMAKE utility to compile DECLARE1.PRG
file c:\snri\src\prepro\decl1.rmk

```
obj1  :=  \snri\obj\declare1.obj

// Inference Rules to Compile obj's from prg's
.prg.obj:
  c:\clippper5\bin\clipper  $**  /a/b/m/n/w/o\SNRI\obj\$*
.c.obj:
  cl /c /AL /Dalt /Z1 /Fpa /Gs $*.c
  copy $*.obj ..\..\obj

// Dependency decl for prg modules
SNRI\obj\declare1.obj:  \SNRI\SRC\prepro\declare1.prg

// Dependency decl for exe module
\SNRI\exe\decl1mn.exe:  $(obj1)
  c:\blinker2\blinker  @\snri\src\prepro\decl1.lnk
```

3) Input to the BLINKER2 loader for DECLARE1.PRG
file c:\snri\src\prepro\decl1.lnk

```
NOBELL
FILE  \snri\obj\declare1
OUTPUT  decl1mn
LIBRARY c:\lib\v51\sgit51
LIBRARY c:\clipper5\lib\clipper
LIBRARY c:\clipper5\lib\extend
LIBRARY c:\clipper5\lib\terminal
LIBRARY c:\clipper5\lib\dbfntx
LIBRARY c:\clipper5\lib\nt250
```

4) The batch file to execute DECL1MN.EXE
file c:\snri\src\prepro\account.bat

```
@echo off
cd  \SNRI\src\prepro
decl1mn  %1
```

5) Structure of the database file ACCOUNT.DBF:
file c:\snri\dbf\account.dbf

```
Structure for database : C:ACCOUNT.dbf
Number of data records :        0
Date of last update    : 12/17/97
Field   Field Name   Type         Width     Dec
    1   CYCLO        Character        8
    2   NATO         Numeric          5
    3   ENRICHO      Numeric          6
** TOTAL **                         20
```

6) Structure of the database file DECLARE.DBF:
file c:\snri\dbf\declare.dbf

```
Structure for database : C:DECLARE.dbf
Number of data records :       11
Date of last update    : 01/30/98
Field   Field Name   Type         Width     Dec
    1   CYLINDER     Character        6
    2   NAT          Numeric          7       5
    3   ENRICH       Numeric          8       6
** TOTAL **                         22
```

(THIS PAGE INTENTIONALLY LEFT BLANK)

Appendix O

Programs to Incorporate the Plant Operator's Declarations
of
Minimum Residence Times for UF_6 Cylinders Attached at Feed Stations

1) CLIPPER5 source code for program RES.PRG:
file c:\snri\src\prepro\res.prg

2) Instructions to the CLIPPER5 RMAKE utility to compile RES.PRG:
file c:\snri\src\prepro\res.rmk

3) Input to the BLINKER2 loader for RES.PRG:
file c:\snri\src\prepro\res.lnk

4) The batch file to execute RESMAIN.EXE:
file c:\snri\src\prepro\res.bat

5) Structure of the Database File N5RES.DBF:
file c:\snri\dbf\n5res.dbf

6) Structure of the Database File N8RES.DBF:
file c:\snri\dbf\n8res.dbf

7) Structure of the Database File N12RES.DBF:
file c:\snri\dbf\n12res.dbf

Appendix O

1) CLIPPER5 source code for program RES.PRG
file c:\snri\src\prepro\res.prg

```
*==========================================================================
* Module (File) Name :  res.prg
*          Programmer :  Arnold L. Aronson
*                Date :  10/07/97
*
* Description : Main driver program for Short Notice Random Inspection System
*               Read and print residence time data
*
* Modification History -
*    By:                                  Date:
*    Description:
*
*==========================================================================
Function resid                                    /* Black & White */
field uf6, res
local  olduf6,which,i,yesno
set confirm on
cls

which = ' 5'
@ 2,1 say 'Minimum Residence Times for Cylinders'
@ 3,1 say 'cylinder size = '+ which +   ' inches'
select 0
use c:\snri\dbf\n5res.dbf alias n5res exclusive
i = 4
do while !eof()
i = i + 1
if(i=5)
@ i,1 say 'if uf6 is' + '                    > ' + str(uf6) + ' grams '+ ' ' +
          'residence time = '+ str(res)+ ' hours'
elseif(i>5)
@ i,1 say 'if uf6 is <= ' + str(olduf6) + ' and > '+ str(uf6) + ' grams '+
          ' ' + 'residence time = '+ str(res) + ' hours'
endif
olduf6 = uf6
skip
enddo
close
yesno = ' '
@ 20,1 say 'Do you wish to continue (y/n) ?'  get yesno
read

cls
which = ' 8'
@ 2,1 say 'Minimum Residence Times for Cylinders'
@ 3,1 say 'cylinder size = '+ which + ' inches'
select 0
use c:\snri\dbf\n8res.dbf alias n8res exclusive
i = 4
do while !eof()
i = i + 1
if(i=5)
@ i,1 say 'if uf6 is' + '                    > ' + str(uf6) + ' grams '+ ' ' +
          'residence time = '+ str(res) + ' hours'
elseif(i>5)
@ i,1 say 'if uf6 is <= ' + str(olduf6) + ' and > '+ str(uf6) + ' grams '+
          ' ' + 'residence time = '+ str(res) + ' hours'
```

```
endif
olduf6 = uf6
skip
enddo
close
yesno = ' '
@ 20,1 say 'Do you wish to continue (y/n) ?' get yesno
read
cls
which = '12'
@ 2,1 say 'Minimum Residence Times for Cylinders'
@ 3,1 say 'cylinder size = '+ which  +  ' inches'
select 0
use c:\snri\dbf\n12res.dbf alias n12res exclusive
i = 4
do while !eof()
i = i + 1
if(i=5)
@ i,1 say 'if uf6 is' + '                    > ' + str(uf6) + ' grams '+ ' ' +
          'residence time = '+ str(res) + ' hours'
elseif(i>5)
@ i,1 say 'if uf6 is <= ' + str(olduf6) + ' and > '+ str(uf6) + ' grams '+
          ' ' + 'residence time = '+ str(res) + ' hours'
endif
olduf6 = uf6
skip
enddo
close
yesno = ' '
@ 20,1 say "Do You Wish To Print This File (y/n) ? " get yesno
read
IF(yesno='y')
set device to printer

which = ' 5'
@ 2,1 say 'Minimum Residence Times for Cylinders'
@ 3,1 say 'cylinder size = '+ which  + ' inches'
select 0
use c:\snri\dbf\n5res.dbf alias n5res exclusive
i = 4
do while !eof()
i = i + 1
if(i=5)
@ i,1 say 'if uf6 is' + '                    > ' + str(uf6) + ' grams '+ ' ' +
          'residence time = '+ str(res) + ' hours'
else
@ i,1 say 'if uf6 is <= ' + str(olduf6) + ' and > '+ str(uf6) + ' grams '+
          ' ' + 'residence time = '+ str(res) + ' hours'
endif
olduf6 = uf6
skip
enddo
close

which = ' 8'
@ i+3,1 say 'Minimum Residence Times for Cylinders'
@ i+4,1 say 'cylinder size = '+ which  + ' inches'
select 0
use c:\snri\dbf\n8res.dbf alias n8res exclusive
i = i + 5
ip6 = i+1
do while !eof()
i = i + 1
if(i=ip6)
```

```
@ i,1 say 'if uf6 is' + '                    > ' + str(uf6) + ' grams '+ ' ' +
          'residence time = '+ str(res) + ' hours'
else
@ i,1 say 'if uf6 is <= ' + str(olduf6) + ' and > '+ str(uf6) + ' grams '+
          ' ' + 'residence time = '+ str(res) + ' hours'
endif
olduf6 = uf6
skip
enddo
close

which = '12'
@ i+3,1 say 'Minimum Residence Times for Cylinders'
@ i+4,1 say 'cylinder size = '+ which + ' inches'
select 0
use c:\snri\dbf\n12res.dbf alias n12res exclusive
i = i+5
ip6 = i+1
do while !eof()
i = i + 1
if(i=ip6)
@ i,1 say 'if uf6 is' + '                    > ' + str(uf6) + ' grams '+ ' ' +
          'residence time = '+ str(res) + ' hours'
else
@ i,1 say 'if uf6 is <= ' + str(olduf6) + ' and > '+ str(uf6) + ' grams '+
          ' ' + 'residence time = '+ str(res) + ' hours'
endif
olduf6 = uf6
skip
enddo
close
SET DEVICE TO SCREEN
ENDIF
cls
Return NIL
```

2) Instructions to the CLIPPER5 RMAKE utility to compile RES.PRG:
file c:\snri\src\prepro\res.rmk

```
obj1 := \snri\obj\res.obj

// Inference Rules to Compile obj's from prg's
.prg.obj:
  c:\clippper5\bin\clipper $** /a/b/m/n/w/o\SNRI\obj\$*
.c.obj:
  cl /c /AL /Dalt /Z1 /Fpa /Gs $*.c
  copy $*.obj ..\..\obj

// Dependency list for prg modules
SNRI\obj\res.obj: \SNRI\SRC\prepro\res.prg

// Dependency list for exe module
\SNRI\exe\resmain.exe: $(obj1)
  c:\blinker2\blinker @\snri\src\prepro\res.lnk
```

3) Input to the BLINKER2 loader for RES.PRG:
file c:\snri\src\prepro\res.lnk

```
NOBELL
FILE \snri\obj\res
OUTPUT resmain
LIBRARY c:\lib\v51\sgit51
LIBRARY c:\clipper5\lib\clipper
LIBRARY c:\clipper5\lib\extend
LIBRARY c:\clipper5\lib\terminal
LIBRARY c:\clipper5\lib\dbfntx
LIBRARY c:\clipper5\lib\nt250
```

4) The batch file to execute RESMAIN.EXE:
file c:\snri\src\prepro\res.bat

```
@echo off
c:
cd \SNRI\src\prepro
resmain %1
```

5) Structure of the Database File N5RES.DBF:
file c:\snri\dbf\n5res.dbf

```
Structure for database : C:N5RES.dbf
Number of data records :        3
Date of last update    : 10/23/97
Field   Field Name   Type         Width      Dec
    1   UF6          Numeric          8        1
    2   RES          Numeric          5        1
** TOTAL **                         14
```

6) Structure of the Database File N8RES.DBF:
file c:\snri\dbf\n8res.dbf

```
Structure for database : C:N8RES.dbf
Number of data records :        6
Date of last update    : 10/23/97
Field   Field Name   Type         Width      Dec
    1   UF6          Numeric          8        1
    2   RES          Numeric          5        1
** TOTAL **                         14
```

7) Structure of the Database File N12RES.DBF:
file c:\snri\dbf\n12res.dbf

```
Structure for database : C:N12RES.dbf
Number of data records :        8
Date of last update    : 10/23/97
Field   Field Name   Type         Width      Dec
    1   UF6          Numeric          8        1
    2   RES          Numeric          5        1
** TOTAL **                         14
```

Appendix P

The Pre-Processor Programs

1) CLIPPER5 source code for the PREPRO main program
file c:\snri\src\prepro\prepro.prg

2) CLIPPER5 source code for the COPYINV program
file c:\snri\src\prepro\copyinv.prg

3) CLIPPER5 source code for the ELIMDUPS program
file c:\snri\src\prepro\elimdups.prg

4) CLIPPER5 source code for the FILLUP program
file c:\snri\src\prepro\fillup.prg

5) CLIPPER5 source code for the PHYINV program
file c:\snri\src\prepro\phyinv.prg

6) CLIPPER5 source code for the CORRECT program
file c:\snri\src\prepro\correct.prg

7) CLIPPER5 source code for the LISTCYL program
file c:\snri\src\prepro\listcyls.prg

8) CLIPPER5 source code for the CYCLO program
file c:\snri\src\prepro\cyclos.prg

9) Listing of input data to the CLIPPER RMAKE utility
file c:\snri\src\prepro\prepro.rmk

10) Listing of input to the BLINKER2 loader
file c:\snri\src\prepro\prepro.lnk

11) Batch file to execute the PREPRO program
file c:\prepro.bat

12) Structure of the database file INVENTOR.DBF
file c:\snri\dbf\inventor.dbf

13) Structure of the database file CYL.DBF
file c:\snri\dbf\cyl.dbf

14) Structure of the database file CYL2.DBF
file c:\snri\dbf\cyl2.dbf

Appendix P

1) CLIPPER5 source code for the PREPRO main program
file c:\snri\src\prepro\prepro.prg

```
*==============================================================================
* Module (File) Name : prepro.prg
*          Programmer : Arnold L. Aronson
*                Date : 10/07/97
*
* Description : Main driver program for Short Notice Random Inspection System
*               Inspector's Preprocessor
*
* Modification History -
*    By:                                    Date:
*    Description:
*
*==============================================================================
Function PREPR1( bw )                                        /* Black & White */
      local yesno,yesno1,yesno2
      public history

      set date british
      set confirm on
      cls

yesno = ' '
cls
yesno2 = ' '
@ 1,1 say "Portsmouth Gaseous Diffusion Plant"
@ 2,1 say "Inspector's Data Preprocessing Program"
@ 3,1 say dtoc(date())+'  '+time()
@ 4,1 say "Do you wish to continue ?"  get yesno2
read
if(yesno2#'y'.and.yesno2#'Y')
quit
end
cls
 use c:\snri\dbf\history.dbf new alias history exclusive
                    if CopyInventory("IC")

                       ElimDupes("IC")

                       do fillups

                       do UpdatePhyInv


                       ElimDupes("HI")

                       do correct

                       do listcyl

                       do cyclo

cls
@ 3,1 say 'Please remove the diskette from the "A" drive.'
@ 4,1 say ' '
@ 5,1 say ' '


                      endif
```

```
close  history

Return  NIL

function  restim(date2,time2,date1,time1)
local  date3,date4,a,aa,b,bb,c,cc,t1,t2,t3,elapse

      DATE3  =  DATE2-DATE1
      DATE4  =  DATE3*24
      @ 2,1 SAY DTOC(DATE2)+'  '+time2+'  '+DTOC(DATE1)+  '  '  +time1+'  '+
                STR(DATE3,6,1)+  '  '  +  STR(DATE4,6,1)
      A  =  SUBSTR(TIME2,1,2)
      AA  =  VAL(A)
      b  =  substr(time2,4,2)
      bb  =  val(b)
      c  =  substr(time2,7,2)
      cc  =  val(c)
      t2  =  aa+bb/60+cc/3600
      @ 3,1 say time2+'  '+a+'  '+str(aa,6,1)+'  '+b+'  '+str(bb,6,1)+'  '+c+'  '+
                str(cc,6,1)+'  '+str(t2,12,3)

      A  =  SUBSTR(time1,1,2)
      AA  =  VAL(A)
      b  =  substr(time1,4,2)
      bb  =  val(b)
      c  =  substr(time1,7,2)
      t1  =  aa+bb/60+cc/3600
      @ 4,1 say time1+'  '+a+'  '+str(aa,6,1)+'  '+b+'  '+str(bb,6,1)+'  '+c+'  '+
                str(cc,6,1)+'  '+str(t1,12,3)
      t3  =  abs(t2-t1)
      if(t2>=t1)
      elapse  =  date4+t3
      else
      elapse  =  date4-t3
      endif
      @ 5,1 say  str(t3,12,3)
      @ 6,1 say 'elapsed  time  (hours)  =  '  +  str(elapse,12,3)
      return  elapse
```

2) CLIPPER5 source code for the COPYINV program
file c:\snri\src\prepro\copyinv.prg

```
/*
 * CopyInv.prg
 *
 * This function copies the INVENTOR.DBF file from the A: drive to the
 * \SNRI\DBF directory, doing all of the necessary prompting and checking
 * along the way.
 */

#define CLIPPER

#include "idd.ch"
#include "..\collect\snri.h"

#define SOURCE_DRIVE "A:"

Function CopyInventory(which)

      local msg1, msg2, msg3, filename, choice, target_dbf

      if which == "IC"
            filename := SOURCE_DRIVE + "\" + FILENAME
            target_dbf := TARGET_DBF
            msg1 := "Insert the Inventory Change diskette into drive " + ;
                        SOURCE_DRIVE
            choice := msgbox( , , {msg1}, 2, {"Continue", "Quit"}, , , , ,)
      end

      if choice = "Quit" .or. choice = "ESC"
            Return FALSE
      end

      choice := "Retry"
      while choice == "Retry"
            if !file(filename)
                  if file(target_dbf)
                        choice := msgbox( , , {filename + " not found"}, 3, ;
                                    {" Abort ", " Retry ", " Ignore "}, ,
                                          SysColour(ERRORBOX), , , , )
                  else
                        choice := msgbox( , , {filename + " not found"}, 2, ;
                                    {" Abort ", " Retry "}, ,
                                          SysColour(ERRORBOX), , , , )
                  end
                  if choice == "Abort"
                        Return FALSE
                  elseif choice == "Ignore"
                        Return TRUE
                  end
            else
                  exit
            end
      end

      copy file (filename) to (target_dbf)

Return TRUE
```

Appendix P

3) CLIPPER5 source code for the ELIMDUPS program
file c:\snri\src\prepro\elimdups.prg

```
/ *
  * ElimDups.prg
  */
/ *
  * ElimDupes()
  *
  * This function processes the INVENTOR.DBF, or the HISTORY.DBF
  * database file, depending on the argument, and eliminates all records
  * which are found to be duplicates.  The duplicate check is based on all
  * fields in the database EXCEPT cycle and the clock_date and clock_time.
  */

Function  ElimDupes(which)

field   itemid, trans_type, purity, assay, ;
                location, grosswt, tarewt, netwt,   ;
                cdate, ctime, edate, etime, size,   cycle, resid, ;
                notes1, notes2, notes3, notes4, corr_type

local item1, item2,item3, item4, item5, item6, item7, item8, item9
local item10, item11, item12, item13, item14, item15, item16

      local old_work_area, compare, title, total_records, deleted_records
      local yesno, nrec1, nrec2

      set date british

// yesno = ' '
// @ 3,1 say 'enter elimdups' +' '+which get yesno
// read
      /* open the file we're processing */
      if which = "IC"
            use c:\snri\dbf\INVENTOR.dbf new alias inventor exclusive

//   cls
//   yesno = ' '
//   @ 1,1 say 'open inventor.dbf' get yesno
//   read
       elseif which = "HI"
              select history


//   cls
//   yesno = ' '
//   @ 1,1 say 'select history.dbf' get yesno
//   read
      endif
      total_records := reccount()

//   @ 2,1 say 'total_records' +' '+str(total_records) get yesno
//   read
//          /* index the file on almost all fields */
      index on  itemid +   ;
      trans_type + ;
      str(purity) + str(assay) +    ;
      location + str(grosswt) + str(tarewt) + str(netwt) + ;
      dtoc(cdate) + ctime + dtoc(edate)   + etime + ;
      size +   str(resid) +   notes1+notes2+notes3+notes4+;
```

```
      corr_type ;
      to tmp
//   @ 3,1 say 'step 1' get yesno
//   read
      /* make sure we're at the top of the file */
      go top
      /* keep going until we hit the end of the file */
        do while !eof()
             /* save the value of all of the fields for the current record */
             nrec1 = recno()
      compare := itemid  + ;
      trans_type + ;
      str(purity) + str(assay) +    ;
      location + str(grosswt) + str(tarewt) + str(netwt) + ;
      dtoc(cdate) + ctime + dtoc(edate)   + etime + ;
      size + str(resid) + notes1+notes2+notes3+notes4+ ;
      corr_type

item1 :=itemid
item2 :=trans_type
item3 :=purity
item4 :=assay
item5 :=location
item6 :=grosswt
item7 :=tarewt
item8 :=netwt
item9 :=cdate
item10 :=ctime
item11 :=edate
item12 :=etime
item13 :=size
item14 :=cycle
item15 :=resid
item16 :=corr_type

// cls
// @ 4,1 say 'step 2 record ' +str(nrec1) get yesno
//   read
            /* skip to the next record */
            skip 1
            nrec2 = recno()
// @ 5,1 say 'step 2.5 record ' +' '+str(nrec2) get yesno
//   read
//   cls
// @ 1,1 say item1 + ' '+itemid
// @ 2,1 say item2+' '+trans_type
// @ 3,1 say str(item3) + ' ' + str(purity)
// @ 4,1 say str(item4)+' '+str(assay)
// @ 5,1 say item5+' '+location
// @ 6,1 say str(item6)+' '+str(grosswt)
// @ 7,1 say str(item7)+' '+str(tarewt)
// @ 8,1 say str(item8)+' '+str(netwt)
// @ 9,1 say dtoc(item9)+' '+dtoc(cdate)
// @ 10,1 say item10+' '+ctime
// @ 11,1 say dtoc(item11)+' '+dtoc(edate)
// @ 12,1 say item12+' '+etime
// @ 13,1 say item13+' '+size
// @ 14,1 say str(item14)+' '+str(cycle)
// @ 15,1 say str(item15)+' '+str(resid)
// @ 16,1 say item16+' '+corr_type
// @ 17,1 say str(nrec1) + ' ' + str(nrec2)
// @ 18,1 get yesno
//   read
            /* if the next record has the same values ... */
```

```
      if compare =  itemid  + ;
      trans_type + ;
      str(purity) + str(assay) +    ;
      location + str(grosswt) + str(tarewt) + str(netwt) + ;
      dtoc(cdate) + ctime + dtoc(edate)  + etime + ;
      size +  str(resid) +  notes1+notes2+notes3+notes4+ ;
      corr_type

// @ 19,1 say 'step 3 delete record ' +str(nrec2) get yesno
//   read
        /* ... delete it and all subsequent records that also have the same */
        /*   values */
      delete while compare =  itemid  + ;
      trans_type + ;
      str(purity) + str(assay) +    ;
      location + str(grosswt) + str(tarewt) + str(netwt) + ;
      dtoc(cdate) + ctime + dtoc(edate)  + etime + ;
      size +  str(resid) + notes1 + notes2 + notes3 + notes4 + ;
      corr_type
//   @ 20,1 say 'step 4 record ' +str(nrec2) +  ' deleted 'get yesno
//   read
            endif

//   @ 21,1 say 'step 5' get yesno
//   read
      enddo

//   @ 22,1 say 'step 6' get yesno
//   read
      /* physically remove the deleted records from the file */
      pack
      deleted_records := total_records - reccount()

      #ifdef DEBUG
            msgbox( , , { ;
                        ltrim(str(deleted_records)) + " of " + ;
                        ltrim(str(total_records)) + " Records Deleted", ;
                        }, 1, , , , , , )
      #endif


      /* close whatever file we opened */
      if which = "IC"
      close inventor
      endif
      /* delete the temporary index file */
      close index
      delete file tmp.ntx


Return NIL
```

Appendix P

4) CLIPPER5 source code for the FILLUP program
file c:\snri\src\prepro\fillup.prg

```
*==========================================================================
* Module (File) Name :  fillup.prg
*          Programmer :  Arnold L. Aronson, Brookhaven National Laboratory
*                Date :  0/07/97
*
* Description :   Module for preprocessing in
*                 Short Notice Random Inspection Software
*                 Fills missing fields in inventor.dbf
*
* Modification History -
*    By:                                  Date:
*    Description:
*
*==========================================================================
Function fillups
field itemid, purity, assay, grosswt, tarewt, netwt, size, resid
field uf6, res, trans_type
field cylinder, nat, enrich
local wt,big1,big2,howlong,match,uranium,u235
local wtnet,iditem, yesno, nrec, nrec1
set date british
set confirm on
yesno = ' '
use c:\snri\dbf\inventor.dbf new alias inventor exclusive

do while .not. eof()
cls
nrec1 = recno()
iditem=upper(itemid)
// get net weight
   wt = grosswt - tarewt
   inventor->netwt:= wt
//  @ 1,1 say 'itemid' + ' ' + str(nrec1)+' '+iditem + ' ' + 'netweight' +
              ' ' + str(netwt)  get yesno
//  read
   wtnet = netwt

// get purity and assay

use c:\snri\dbf\declare.dbf index c:\snri\dbf\cylinder new alias declare
       exclusive
  seek iditem
  if found()
  nrec = recno()
  uranium = nat
  u235 = enrich
//  @ 7,1 say 'accountability data found for '+ ' ' + iditem+' '+str(nrec)
  close declare
  select inventor
  go nrec1
  inventor->purity:=uranium
  inventor->assay:=u235
//  @ 8,1 say 'purity, assay' + ' '+ str(purity) + ' ' + str(assay) get yesno
//  read
  else
  @ 8,1 say 'no accountability data for '+ ' ' + iditem   get yesno
  read
  close declare
```

```
   select  inventor
   go  nrec1
   endif

// get  cylinder  size

    big1  =  substr(upper(itemid),1,2)
    if(big1='05'.or.big1='5N'.or.big1='AL'.or.big1='ST'.or.big1='N5')
    big2  =  '05'
    elseif  (big1='08')
    big2  =  '08'
    elseif  (big1='12')
    big2  =  '12'
    else
    big2  =  'ng'
    endif

    inventor->size:=big2
//   @ 2,1  say  'size'+  '  '  +  size  +  '  '  +  big1  +  '  '  +  big2
   if  (size  =  'ng')
   @ 3,1  say  'Incorrect  cylinder  size  for  residence  time  data'  +  '  '  +  itemid
@ 4,1  get  yesno
read
   skip  1
   loop
   endif
// get  residence  time
         if(size='05')
         use  c:\snri\dbf\n5res.dbf  new  alias  nres  exclusive
         elseif(size='08')
         use  c:\snri\dbf\n8res.dbf  new  alias  nres  exclusive
         else
         use  c:\snri\dbf\n12res.dbf  new  alias  nres  exclusive
         endif
      do  while  .not.  eof()
         if(wtnet  >  uf6)
         howlong  =  res
         exit
         else
         skip  1
         loop
         endif
      enddo
      if  eof()
      howlong  =  0.0
      endif
   close  nres
   select  inventor
   go  nrec1
   inventor->resid:=howlong
//   @ 15,1  say  'residence  time'  +  '  '  +str(resid)    get  yesno
// read
skip  1
loop
enddo
close  inventor

return  NIL
```

Appendix P

5) CLIPPER5 source code for the PHYINV program
file c:\snri\src\prepro\phyinv.prg

```
/*
 * PhyInv.prg
 *
 */

/*
 * UpdatePhyInv()
 *
 * This function appends the INVENTOR.DBF file to the HISTORY.DBF file,
 * thereby combining the Inventory Changes since the last inspection with
 * the database from the last inspection.  It then sorts
 * the resultant file.
 */

field itemid, trans_type, clock_date, clock_time

Function UpdatePhyInv

     local old_work_area, rectotal
     set date british

     /* get the number of records to be appended */
     use c:\snri\dbf\inventor.dbf new alias inventor exclusive
     rectotal = reccount()
cls

     close inventor


     /* open the HISTORY.DBF file */
select history
go top

     /* append the INVENTOR.DBF file to it */
     append from c:\snri\dbf\inventor.dbf for trans_type # 'e'

     #ifdef DEBUG
        msgbox( , , {ltrim(str(rectotal)) + " Records Appended", ;
                     ltrim(str(reccount())) +" Total Records"},1, , , , , , )
     #endif



     #ifdef DEBUG
        msgbox( , ,{ltrim(str(reccount())) +" Records Sorted"}, 1, , , , , , )
     #endif



Return NIL
```

Appendix P

6) CLIPPER5 source code for the CORRECT program
file c:\snri\src\prepro\correct.prg

```
function  correct

field itemid, trans_type, purity, assay, location, grosswt, tarewt, netwt
field cdate, ctime, edate, etime, size, cycle, resid, notes1, notes2
field notes3, notes4, corr_type,clock_date, clock_time
local data1, data2, data3, data4, data5, data6, data7, data8, data9
local data10, data11, data12, data13, data14, data15, data16, data17
local data18, data19, data20, data21, data22, nrecord, pr
set date british


use c:\snri\dbf\inventor.dbf new alias inventor exclusive
pr = 'n'

   do while .not. eof()
      if trans_type # 'e'
      skip 1
      else
      data1 = itemid
      data2 = trans_type
      data3 = purity
      data4 = assay
      data5 = location
      data6 = grosswt
      data7 = tarewt
      data8 = netwt
      data9 = cdate
      data10 = ctime
      data11 = edate
      data12 = etime
      data13 = size
      data14 = cycle
      data15 = resid
      data16 = notes1
      data17 = notes2
      data18 = notes3
      data19 = notes4
      data20 = corr_type
      data21 = clock_date
      data22 = clock_time
      nrecord :=recno() + 1
select history
go top
         do while .not. eof()
           if edate = data9 .and. etime =  data10
           if pr='y'
//            set device to printer
            @1,1 say 'error correction record       record being corrected'
            @2,1 say '======================        ======================='
            @4,1 say data1 + '            ' + itemid
            if data20 = ' '
            data20 = trans_type
            endif
            @5,1 say data20 + '            ' + trans_type
            @6,1 say str(data3) + '            ' + str(purity)
            @7,1 say str(data4) + '            ' + str(assay)
            @8,1 say data5 + '            ' + location
            @9,1 say str(data6) + '            ' + str(grosswt)
```

```
          @10,1 say str(data7) + '              ' + str(tarewt)
          @11,1 say str(data8) + '              ' + str(netwt)
          @12,1 say dtoc(data9) + '              ' + dtoc(cdate)
          @13,1 say data10 + '              ' + ctime
          @14,1 say dtoc(data11) + '              ' + dtoc(edate)
          @15,1 say data12 + '              ' + etime
          @16,1 say data13 + '              ' + size
          @17,1 say str(data14) + '              ' + str(cycle)
          @18,1 say str(data15) + '              ' + str(resid)
          @19,1 say data16 + '              ' + notes1
          @20,1 say data17 + '              ' + notes2
          @21,1 say data18 + '              ' + notes3
          @22,1 say data19 + '              ' + notes4
          @23,1 say dtoc(data21) + '              ' + dtoc(clock_date)
          @24,1 say data22 + '              ' + clock_time
          yesno=' '
          @ 25,1  get yesno
          read
//          set device to screen
          endif
          history->itemid:=data1
          if data20 # ' '
          history->trans_type:=data20
          endif
          history->purity:=data3
          history->assay:=data4
          history->location:=data5
          history->grosswt:=data6
          history->tarewt:=data7
          history->netwt:=data8
          history->edate:=data11
          history->etime:=data12
          history->size:=data13
          history->resid:=data15
          history->notes1:=data16
          history->notes2:=data17
          history->notes3:=data18
          history->notes4:=data19
          history->clock_date:=data21
          history->clock_time:=data22
          exit
        else
          skip 1
        endif
      end
      if eof()
      if pr = 'y'
//        set device to printer
      @1,1 say 'no match for error correction record'
      @2,1 say 'listing of error correction record'
      @4,1 say data1
      @5,1 say data2
      @6,1 say str(data3)
      @7,1 say str(data4)
      @8,1 say data5
      @9,1 say  str(data6)
      @10,1 say str(data7)
      @11,1 say str(data8)
      @12,1 say data9
      @13,1 say data10
      @14,1 say data11
      @15,1 say data12
      @16,1 say data13
      @17,1 say str(data14)
```

```
         @18,1 say str(data15)
         @19,1 say data16
         @20,1 say data17
         @21,1 say data18
         @22,1 say data19
         @23,1 say data20
         @24,1 say data21
         @25,1 say data22
//         set device to screen
         endif
         end
      select inventor
      go nrecord
      endif
      end
      zap
      close inventor
      return nil
```

7) CLIPPER5 source code for the LISTCYL program
file c:\snri\src\prepro\listcyls.prg

```
function listcyl

field itemid, size, resid, trans_type
field cylinder, sizes, resids
local data1, data2, data3, yesno, jj
set confirm on
// cls
// yesno = ' '
// @ 1,1 say 'listcyls'
// @ 2,1 get yesno
// read
use c:\snri\dbf\cyl.dbf new alias cyl exclusive
zap
select history
go top
// yesno = ' '
// @ 1,1 say 'listcyls'
// ? reccount()
// if eof()
// @ 3,1 say 'eof on history.dbf'
// endif
// @ 4,1 get yesno
// read


do while .not. eof()
cls
if trans_type = 'c'
data1 = itemid
data2 = size
data3 = resid
yesno = ' '
// @ 1,1 say "itemid" + ' ' + data1
// @ 2,1 get yesno
// read
jj = 3
select cyl
go top
do while .not. eof()
// @ jj,1 say "cylinder" + ' ' + cylinder
// jj = jj + 1
if cylinder = data1
exit
else
skip 1
endif
end
// @ jj,1 say 'eof on cyl.dbf'
// @ jj+1,1 get yesno
// read

if eof()
append blank
cyl->cylinder:=data1
cyl->sizes:=data2
cyl->resids:=data3
// @ jj+2,1 say 'cylinder ' + ' ' + cylinder
```

```
// @ jj+3,1 get yesno
// read
endif
// @ jj+3,1 get yesno
// read
endif
select history
skip 1
loop
enddo

select cyl
go top
sort to tmp on sizes, resids
close cyl
use c:\snri\dbf\cyl2.dbf new alias cyl2 exclusive
zap
append from tmp
delete file tmp.dbf
close cyl2

return nil
```

Appendix P

8) CLIPPER5 source code for the CYCLO program
file c:\snri\src\prepro\cyclos.prg

```
function  cyclo

field cylinder
field itemid, trans_type, cycle, edate, etime
local match, save, rec_type, jj, yesno, nrec, olddate, oldtime
set date british
cls
yesno = ' '

/* Sort the history.dbf file */
select history
go top
sort to tmp on  itemid, edate, etime
zap
append from tmp
delete file tmp.dbf

  /* work through the list of cylinders that have been processed */
  use c:\snri\dbf\cyl.dbf new alias cyl exclusive
  do while .not. eof()
  match = cylinder
  rec_type = ' '
  save = 0
//  @ 3,1 say 'cylinder = ' + ' ' + cylinder get yesno
//  read
  jj = 5

     /* work through the list of transactions */
     select history
     go top
     do while .not. eof()
//     nrec = recno()
//     @ 4,1 say 'nrec, match, itemid'+' '+str(nrec)+' '+match+' '+itemid get
                yesno
//     read
        if match = itemid
//        @ jj,1 say 'itemid trans_type cycle rec_type '+' '+itemid+' '+
                trans_type+' '+str(cycle)+' '+rec_type+' '+dtoc(edate)+' '+etime
//        jj = jj + 1
           if trans_type = 'c'
             if(trans_type#rec_type)
             history->cycle:=save + 1
             save = cycle
             else
             @ jj,1 say 'error - two consecutive c records for '+itemid
             jj = jj+1
             @ jj,1 say dtoc(olddate)+' '+oldtime
             jj = jj + 1
             @ jj,1 say dtoc(edate)+' '+etime
             jj = jj + 1
             @ jj,1 get yesno
             read
             endif
           endif
           if trans_type = 'd'
             if(trans_type#rec_type)
```

```
            history->cycle:=save
            else
            @jj,1 say 'error - two consecutive d records for '+itemid
            jj = jj + 1
            @jj,1 say dtoc(olddate)+' '+oldtime
            jj = jj + 1
            @jj,1 say dtoc(edate)+' '+etime
            jj = jj + 1
            @jj,1 get yesno
            read
            endif
          endif
//        @jj,1 say 'itemid trans_type cycle rec_type ' + ' ' + itemid + ' '
            + trans_type + ' '+str(cycle)+ ' ' + rec_type+' '+dtoc(edate)+
            ' '+etime
//        jj = jj + 1
        rec_type =  trans_type
        olddate = edate
        oldtime = etime
        endif
     /* go to next item in transaction file */
     skip 1
     loop
     enddo

//  @jj+1,1 get yesno
//  read
//  cls
  /* go to the next cylinder in the list */
  select cyl
  skip 1
  loop
  enddo
  close cyl
return nil
```

9) Listing of input data to the CLIPPER RMAKE utility
file c:\snri\src\prepro\prepro.rmk

```
obj1  :=  \snri\obj\prepro.obj  \snri\obj\fillup.obj  \snri\obj\correct.obj
obj2  :=  \snri\obj\elimdups.obj  \snri\obj\copyinv.obj
obj3  :=  \snri\obj\phyinv.obj  \snri\obj\listcyls.obj  \snri\obj\cyclos.prg
// Inference Rules to Compile obj's from prg's
.prg.obj:
  c:\clippper5\bin\clipper  $**  /a/b/m/n/w/o\SNRI\obj\$*
.c.obj:
  cl /c /AL /Dalt /Z1 /Fpa /Gs $*.c
  copy $*.obj ..\..\obj

// Dependency list for prg modules
SNRI\obj\prepro.obj:  \SNRI\SRC\prepro\prepro.prg
SNRI\obj\fillup.obj:  \SNRI\SRC\prepro\fillup.prg
SNRI\obj\copyinv.obj:  \SNRI\SRC\prepro\copyinv.prg
SNRI\obj\correct.obj:  \SNRI\SRC\prepro\correct.prg
SNRI\obj\listcyls.obj:  \SNRI\SRC\prepro\listcyls.prg
SNRI\obj\cyclos.obj:  \SNRI\SRC\prepro\cyclos.prg
SNRI\obj\phyinv.obj:  \SNRI\SRC\prepro\phyinv.prg
SNRI\obj\elimdups.obj:  \SNRI\SRC\prepro\elimdups.prg

// Dependency list for exe module
\SNRI\exe\prepromn.exe:  $(obj1)  //  $(obj2)  $(obj3)
  c:\blinker2\blinker  @\snri\src\prepro\prepro.lnk
```

10) Listing of input to the BLINKER2 loader
file c:\snri\src\prepro\prepro.lnk

```
NOBELL
FILE \snri\obj\prepro
FILE \snri\obj\fillup
FILE \snri\obj\correct
FILE \snri\obj\listcyls
FILE \snri\obj\cyclos
FILE \snri\obj\phyinv
FILE \snri\obj\elimdups
FILE \snri\obj\copyinv
OUTPUT prepromn
LIBRARY c:\lib\v51\sgit51
LIBRARY c:\clipper5\lib\clipper
LIBRARY c:\clipper5\lib\extend
LIBRARY c:\clipper5\lib\terminal
LIBRARY c:\clipper5\lib\dbfntx
LIBRARY c:\clipper5\lib\nt250
```

11) Batch file to execute the PREPRO program
file c:\prepro.bat

```
@echo off
c:\snri\src\prepro\prepromn
```

12) Structure of the database file INVENTOR.DBF
file c:\snri\dbf\inventor.dbf

Structure for database : C:INVENTOR.dbf
Number of data records : 0
Date of last update : 12/01/97

Field	Field Name	Type	Width	Dec
1	ITEMID	Character	6	
2	TRANS_TYPE	Character	1	
3	PURITY	Numeric	7	4
4	ASSAY	Numeric	7	4
5	LOCATION	Character	2	
6	GROSSWT	Numeric	15	3
7	TAREWT	Numeric	15	3
8	NETWT	Numeric	15	3
9	CDATE	Date	8	
10	CTIME	Character	8	
11	EDATE	Date	8	
12	ETIME	Character	8	
13	SIZE	Character	2	
14	CYCLE	Numeric	2	
15	RESID	Numeric	3	
16	NOTES1	Character	80	
17	NOTES2	Character	80	
18	NOTES3	Character	80	
19	NOTES4	Character	80	
20	CORR_TYPE	Character	1	
21	CLOCK_DATE	Date	8	
22	CLOCK_TIME	Character	8	
** TOTAL **			445	

13) Structure of the database file CYL.DBF
file c:\snri\dbf\cyl.dbf

```
Structure for database : C:CYL.dbf
Number of data records :        0
Date of last update    :  12/02/97
Field   Field Name   Type         Width      Dec
    1   CYLINDER     Character        6
    2   SIZES        Character        2
    3   RESIDS       Numeric          3
** TOTAL **                         12
```

14) Structure of the database file CYL2.DBF
file c:\snri\dbf\cyl2.dbf

```
Structure for database : C:CYL2.dbf
Number of data records :       12
Date of last update    :  02/05/98
Field   Field Name   Type         Width      Dec
    1   CYLINDER     Character        6
    2   SIZES        Character        2
    3   RESIDS       Numeric          3
** TOTAL **                         12
```

Appendix Q

The Report Generator Programs

1) CLIPPER5 source code for the Report Generator main program:
file c:\snri\src\iaea\iaea.prg

2) CLIPPER5 source code for the program to generate reports for cylinders that should be attached to feed stations because the residence time has not been exceeded
file c:\snri\src\iaea\one.prg

3) CLIPPER5 source code for the program to generate reports for cylinders that should be attached to feed stations because there is no disconnect recorded
file c:\snri\src\iaea\two.prg

4) CLIPPER5 source code for the program to generate detailed reports of the accumulated amount of material removed from cylinders over a specified period
file c:\snri\src\iaea\three.prg

5) CLIPPER5 source code for the program to generate reports of actual residence time compared with declared residence time for normal refeed
file c:\snri\src\iaea\four.prg

6) CLIPPER5 source code for the program to generate a compact printed table of the material removed from all cylinders over a specified period
file c:\snri\src\iaea\five.prg

7) Listing of input data to the CLIPPER RMAKE utility
file c:\snri\src\iaea\iaea.rmk

8) Listing of input to the BLINKER2 loader
file c:\snri\src\iaea\iaea.lnk

9) Batch file to execute the report generator program
file c:\reports.bat

Appendix Q

1) CLIPPER5 source code for the Report Generator main program
file c:\snri\src\iaea\iaea.prg

```
*=====================================================================
* Module (File) Name :  iaea.prg
*          Programmer :  Arnold L. Aronson, Brookhaven National Laboratory
*                Date :  10/07/97
*
* Description :   Main program for Short Notice Random Inspection System
*                 Inspector's Report Generator and Printer
*
* Modification History -
*    By:                                    Date:
*    Description:
*
*=====================================================================
Function iaea( bw )                                    /* Black & White */

      local yesno, datey, term
      local datex:=space(6)
      local timex:=space(4)
      public history
      public dateinsp, timeinsp, begdate, begtime, enddate, endtime, pr
      public grand1, grand2, grand3
      set date british
      set confirm on

cls
yesno = ' '
@ 1,1 say "Portsmouth Gaseous Diffusion Plant"
@ 2,1 say "Inspector's Report Generation and Printing Program"
@ 3,1 say dtoc(date())+'  '+time()
@ 4,1 say "Do you wish to continue ?"  get yesno
read
if(yesno#'y'.and.yesno#'Y')
quit
end

use c:\snri\dbf\history.dbf new alias history exclusive
    cls
    yesno = ' '
    @ 1,1 say 'Do you want to have reports for'
    @ 2,1 say 'cylinders that should be attached to feed stations'
    @ 3,1 say 'because their residence time has not been exceeded'
    @ 4,1 say 'and reports for cylinders that should be attached to'
    @ 5,1 say 'feed stations because there is no disconnect recorded? (y/n)'
    @ 6,1 get yesno
    read
    if yesno='y'.or.yesno='Y'
    @ 7,1 say 'date on which this inspection started (format is ddmmyy)'
    @ 8,1 get datex
    read
    datey = dodateb(datex)
    dateinsp = ctod(datey)
    @ 9,1 say 'time at which this inspection started (format is military time
              = xxxx)'
    @ 10,1 get timex
    read
    timeinsp = dotime(timex)
    @ 11,1 say 'date' + ' ' + dtoc(dateinsp) + ' ' + 'time' + ' ' + timeinsp
```

```
yesno = ' '
@ 12,1 say 'Please press [ENTER] to continue.' get yesno
read
pr = ' '
@ 13,1 say 'Do you want printed reports in addition to the screen reports?
          (y/n)'
@ 14,1 get pr
read
do one
do two
endif

cls
yesno = ' '
@ 1,1 say 'Do you want to have detailed reports of'
@ 2,1 say 'the accumulated amount of material removed'
@ 3,1 say 'from cylinders over a specified period? (y/n)'
@ 4,1 get yesno
read
if yesno='y'.or.yesno='Y'
@ 6,1 say 'starting date of period (format is ddmmyy)'
@ 7,1 say '(000000 means Dec. 17, 1997)'
datex = '       '
@ 8,1 get datex
read
if datex = '000000'
datex = '171297'
endif
datey = dodateb(datex)
begdate = ctod(datey)
@ 9,1 say 'starting time of period (format is military time = xxxx)'
timex = '     '
@ 10,1 get timex
read
begtime = dotime(timex)
@ 11,1 say 'ending date of period (format is ddmmyy)'
datex = '       '
@ 12,1 get datex
read
datey = dodateb(datex)
enddate = ctod(datey)
@ 13,1 say 'ending time of period (format is military time = xxxx)'
timex = '     '
@ 14,1 get timex
read
endtime = dotime(timex)
@ 15,1 say 'starting date '+' '+dtoc(begdate)+' '+'time'+' '+begtime
@ 16,1 say 'ending date    '+' '+dtoc(enddate)+' '+'time'+' '+endtime
yesno = ' '
@ 17,1 say 'Please press [ENTER] to continue.' get yesno
read
pr = ' '
@ 18,1 say 'Do you want printed reports in addition to the screen reports?
          (y/n)'
@ 19,1 get pr
read
cls
do three
endif

cls
yesno =  ' '
@ 1,1 say 'Do you want to have a compact printed table of the material'
@ 2,1 say 'removed from all cylinders over a specified time period? (y/n)'
```

```
    @ 3,1 get yesno
    read
    if  yesno='y'.or.yesno='Y'
    @ 6,1 say 'starting date of period (format is ddmmyy)'
    @ 7,1 say '(000000 means Dec. 17, 1997)'
    datex = '       '
    @ 8,1 get datex
    read
    if datex = '000000'
    datex = '171297'
    endif
    datey = dodateb(datex)
    begdate = ctod(datey)
    @ 9,1 say 'starting time of period (format is military time = xxxx)'
    timex = '    '
    @ 10,1 get timex
    read
    begtime = dotime(timex)
    @ 11,1 say 'ending date of period (format is ddmmyy)'
    datex = '       '
    @ 12,1 get datex
    read
    datey = dodateb(datex)
    enddate = ctod(datey)
    @ 13,1 say 'ending time of period (format is military time = xxxx)'
    timex = '    '
    @ 14,1 get timex
    read
    endtime = dotime(timex)
    @ 15,1 say 'starting date '+' '+dtoc(begdate)+' '+'time'+' '+begtime
    @ 16,1 say 'ending date    '+' '+dtoc(enddate)+' '+'time'+' '+endtime
    yesno = ' '
    @ 17,1 say 'Please press [ENTER] to continue.' get yesno
    read
    do five
    endif
    cls
    yesno = ' '
    @ 1,1 say 'Do you want to have reports of'
    @ 2,1 say 'actual residence time of cylinders compared with'
    @ 3,1 say 'declared residence time for normal refeed? (y/n)'
    @ 4,1 get yesno
    read
    if  yesno='y'.or.yesno='Y'
    pr = ' '
    @ 5,1 say 'Do you want printed reports in addition to the screen reports?
            (y/n)'
    @ 6,1 get pr
    read
    cls
    do four
    endif
    close history
    cls
Return NIL

function  restim(date2,time2,date1,time1)
local  date3,date4,a,aa,b,bb,c,cc,t1,t2,t3,  elapse

      DATE3 = DATE2-DATE1
      DATE4 = DATE3*24
//       @ 2,1 SAY DTOC(DATE2)+' '+time2+' '+DTOC(DATE1)+' '+time1+' '+
                STR(DATE3,6,1)+' '+ STR(DATE4,6,1)
      A = SUBSTR(TIME2,1,2)
```

```
      AA  =  VAL(A)
      b  =  substr(time2,4,2)
      bb  =  val(b)
      c  =  substr(time2,7,2)
      cc  =  val(c)
      t2  =  aa+bb/60+cc/3600
//      @ 3,1 say time2+'  '+a+'  '+str(aa,6,1)+'  '+b+'  '+str(bb,6,1)+'  '+c+
                '  '+str(cc,6,1)+'  '+str(t2,12,3)
      A  =  SUBSTR(time1,1,2)
      AA  =  VAL(A)
      b  =  substr(time1,4,2)
      bb  =  val(b)
      c  =  substr(time1,7,2)
      t1  =  aa+bb/60+cc/3600
      // @ 4,1 say time1+'  '+a+'  '+str(aa,6,1)+'  '+b+'  '+str(bb,6,1)+'  '+c+
                '  '+str(cc,6,1)+'  '+str(t1,12,3)
      t3  =  abs(t2-t1)
      if(t2>=t1)
      elapse  =  date4+t3
      else
      elapse  =  date4-t3
      endif
//      @ 5,1 say str(t3,12,3)
//      @ 6,1 say 'elapsed time  (hours)  =  '  +  str(elapse,12,3)
//      yesno  =  ' '
//      @ 7,1 get yesno
//      read
//      cls
      return  elapse
function  dodateb(datex)
local  d,m,y,dd,mm,yy
   d  =  substr(datex,1,2)
   m  =  substr(datex,3,2)
   y  =  substr(datex,5,2)
   dd  =  stuff('   /   /   ',1,2,d)
   mm  =  stuff(dd,4,2,m)
   yy  =  stuff(mm,7,2,y)
return  yy
function  dotime(timex)
local  a,b,d,e
   a  =  substr(timex,1,2)
   b  =  substr(timex,3,2)
   d  =  stuff('   :   :00',1,2,a)
   e  =  stuff(d,4,2,b)
return  e
```

2) CLIPPER5 source code for the program to generate reports for cylinders that should be attached to feed stations because the residence time has not been exceeded
file c:\snri\src\iaea\one.prg

```
function one
field itemid, trans_type, purity, assay, location, grosswt, tarewt
field netwt, edate, etime, size, cycle, resid, notes1
field notes2, notes3, notes4
local yesno, uvalu, u235valu, jj, howlong, wo, eo
set confirm on
set date british
select history
go top
   cls
   yesno = ' '
  do while .not. eof()
  if trans_type = 'c'
  howlong =   restim(dateinsp,timeinsp,edate,etime)
//   @ 1,1 say str(resid) + ' ' + str(howlong)
//   @ 2,1 get yesno
//   read
 cls
 if howlong>0.0 .and. resid > howlong
 @ 1,1 say 'The following cylinder should be attached to a feed station'
 @ 2,1 say 'because the residence time has not been exceeded:'
 @ 4,1 say 'Cylinder number                                    '+' '+itemid
 @ 5,1 say 'Date and time inspection began      '+' '+dtoc(dateinsp)+
           ' '+timeinsp
 @ 6,1 say 'Date and time cylinder connected   '+' '+dtoc(edate)+' '+etime
 @ 7,1 say 'Residence time (hours)                        '+' '+str(resid,9,2)
 @ 8,1 say 'Elapsed time (hours)                          '+' '+str(howlong,9,2)
 @ 9,1 say 'Refeed station number                                '+' '+location
 @ 10,1 say 'Cylinder adjusted gross weight (grams)       '+' '+str(grosswt,9,2)
 @ 11,1 say 'Cylinder adjusted tare weight (grams)        '+' '+str(tarewt,9,2)
 @ 12,1 say 'Cylinder adjusted net weight (grams)         '+' '+str(netwt,9,2)
 wo = purity*100.0
 @ 13,1 say 'Uranium (wt%)                                '+' '+str(wo,9,2)
 eo = assay*100.0
 @ 14,1 say 'Enrichment (wt%)                             '+' '+str(eo,9,2)
 uvalu = netwt * purity
 u235valu = uvalu * assay
 @ 15,1 say 'Calculated U total (grams)                   '+' '+str(uvalu,9,2)
 @ 16,1 say 'Calculated U235 (grams)                      '+' '+
           str(u235valu,9,2)
 jj = 17
 if !empty(notes1)
 @ jj,1 say notes1
 jj = jj + 1
 endif
 if !empty(notes2)
 @ jj,1 say notes2
 jj = jj + 1
 endif
 if !empty(notes3)
 @ jj,1 say notes3
 jj = jj + 1
 endif
 if !empty(notes4)
 @ jj,1 say notes4
 jj = jj + 1
```

```
endif
@ jj,1 get yesno
read
if pr='y'.or.pr='Y'
set device to printer
@ 1,1 say 'The following cylinder should be attached to a feed station'
@ 2,1 say "because the residence time has not been exceeded:"
@ 4,1 say 'Cylinder number                                    '+' '+itemid
@ 5,1 say 'Date and time inspection began      '+' '+dtoc(dateinsp)+
          '  '+timeinsp
@ 6,1 say 'Date and time cylinder connected   '+' '+dtoc(edate)+' '+etime
@ 7,1 say 'Residence time (hours)                        '+' '+str(resid,9,2)
@ 8,1 say 'Elapsed time (hours)                          '+' '+str(howlong,9,2)
@ 9,1 say 'Refeed station number                                '+' '+location
@ 10,1 say 'Cylinder adjusted gross weight (grams)       '+' '+str(grosswt,9,2)
@ 11,1 say 'Cylinder adjusted tare weight (grams)        '+' '+str(tarewt,9,2)
@ 12,1 say 'Cylinder adjusted net weight (grams)         '+' '+str(netwt,9,2)
wo = purity*100.0
@ 13,1 say 'Uranium (wt%)                                '+' '+str(wo,9,2)
eo = assay*100.0
@ 14,1 say 'Enrichment (wt%)                             '+' '+str(eo,9,2)
uvalu = netwt * purity
u235valu = uvalu * assay
@ 15,1 say 'Calculated U total (grams)                   '+' '+str(uvalu,9,2)
@ 16,1 say 'Calculated U235 (grams)                      '+' '+
          str(u235valu,9,2)
jj = 17
if !empty(notes1)
@ jj,1 say notes1
jj = jj + 1
endif
if !empty(notes2)
@ jj,1 say notes2
jj = jj + 1
endif
if !empty(notes3)
@ jj,1 say notes3
jj = jj + 1
endif
if !empty(notes4)
@ jj,1 say notes4
endif
set device to screen
endif
endif
endif
skip 1
enddo
 return nil
```

3) CLIPPER5 source code for the program to generate reports for cylinders that should be attached to feed stations because there is no disconnect recorded
file c:\snri\src\iaea\two.prg

```
function two
field itemid, trans_type, purity, assay, location, grosswt, tarewt
field netwt, edate, etime, size, cycle, resid, notes1
field notes2, notes3, notes4
local yesno,uvalu,u235valu,match,oldloc,oldcyc,nrecord,disc,recno
local howlong, longhow, jj, wo, eo, a, b, c
set confirm on
set date british
yesno = ' '
  select history
  go top
    cls
  do while .not. eof()        // #1
    longhow = restim(dateinsp,timeinsp,edate,etime)

    if trans_type='c' .and. longhow > 0.0      // #2
// A connect record exists for this cylinder.
// The inspection date is after the connection date.
    nrecord = recno()
    match = itemid
    oldloc = location
    oldcyc = cycle
    disc='n'
    cls
//    @ 17,1 say match +' '+str(nrecord)+' '+itemid+' '+trans_type+' '+
                dtoc(edate)+' '+etime  get yesno
//    read
    skip 1

     do while .not. eof()                              // #3
     nrec = recno()
//      @ 18,1 say 'recno' +' '+str(nrec)+' '+dtoc(dateinsp)+' '+timeinsp+' '+
                dtoc(edate)+' '+etime get yesno
//      read
      howlong = restim(dateinsp,timeinsp,edate,etime)
//      @ 19,1 say 'itemid trans_type howlong '+' '+itemid+' '+trans_type+' '+
                str(howlong) get yesno
//      read

        if itemid = match .and. trans_type='d'      // #4
// A disconnect record exists for this cylinder.
        if howlong>0.0
// The disconnect date is before the inspection date.
// The cylinder is not connected on the inspection date.
        disc = 'y'
        exit
        endif
// The disconnect date is after the inspection date.
// The cylinder is still connected on the inspection date.
            cls
            if substr(oldloc,2,1)=' '
            a = substr(oldloc,1,1)
            b = stuff(oldloc,1,1,"")
            c = stuff(b,2,1,a)
            oldloc = c
            endif
```

```
            newloc = location
            if substr(newloc,2,1) = ' '
            a = substr(newloc,1,1)
            b = stuff(newloc,1,1,"")
            c = stuff(b,2,1,a)
            newloc = c
            endif
            if newloc # oldloc
            @ 1,1 say 'Refeed station number does not match for cylinder '
                      +match
            @ 2,1 say 'Connect refeed station number      '+oldloc
            @ 3,1 say 'Disconnect refeed station number '+newloc
            @ 4,1 get yesno
            read
            endif
            cls
            if cycle # oldcyc
            @ 5,1 say 'cycles do not match'+' '+match+' '+str(oldcyc)+' '
                      +itemid+' '+str(cycle)
            @ 6,1 get yesno
            read
            endif
            disc = 'n'
         exit
        endif
      skip 1
      enddo
        if .not. disc='y'
// Cylinder is connected on inspection date.
      go nrecord
      nrec = recno()
//      @ 21,1 say 'nrecord' + ' ' + str(nrec)  get yesno
//      read
      @ 1,1 say 'The following cylinder should be attached to a feed station'
      @ 2,1 say 'because there is no disconnect recorded:'
      @ 4,1 say 'Cylinder number                                  ' + ' ' +itemid
      @ 5,1 say 'Date and time inspection began     '+' '+dtoc(dateinsp)+
                ' '+ timeinsp
      @ 6,1 say 'Date and time cylinder connected  ' + ' ' + dtoc(edate) +
                ' ' + etime
      @ 7,1 say 'Residence time (hours)                       '+' '+str(resid,9,2)
      @ 8,1 say 'Refeed station number                         ' + ' ' + location
      @ 9,1 say 'Cylinder adjusted gross weight (grams)        ' + ' ' +
                str(grosswt,9,2)
      @ 10,1 say 'Cylinder adjusted tare weight (grams)        ' + ' ' +
                str(tarewt,9,2)
      @ 11,1 say 'Cylinder adjusted net weight (grams)         ' + ' ' +
                str(netwt,9,2)
      wo = purity*100.0
      @ 12,1 say 'Uranium (wt%)                                ' + ' ' +
                str(wo,9,2)
      eo = assay*100.0
      @ 13,1 say 'Enrichment (wt%)                             ' + ' ' +
                str(eo,9,2)
      uvalu = netwt * purity
      u235valu = uvalu * assay
      @ 14,1 say 'Calculated U total (grams)                   ' + ' ' +
                str(uvalu,9,2)
      @ 15,1 say 'Calculated U235 (grams)                      ' + ' ' +
                str(u235valu,9,2)
      jj = 16
      if !empty(notes1)
      @ jj,1 say notes1
      jj = jj + 1
```

```
endif
if !empty(notes2)
@ jj,1 say notes2
jj =jj + 1
endif
if !empty(notes3)
@ jj,1 say notes3
jj = jj + 1
endif
if !empty(notes4)
@ jj,1 say notes4
jj = jj + 1
endif
@ jj,1 get yesno
read
if pr = 'y'
set device to printer
@ 1,1 say 'The following cylinder should be attached to a feed station'
@ 2,1 say 'because there is no disconnect recorded:'
@ 4,1 say 'Cylinder number                                  ' + ' ' + itemid
@ 5,1 say 'Date and time inspection began      '+' '+dtoc(dateinsp)+
          ' '+timeinsp
@ 6,1 say 'Date and time cylinder connected   ' + ' ' + dtoc(edate) +
          ' ' + etime
@ 7,1 say 'Residence time (hours)                          ' + ' ' +
          str(resid,9,2)
@ 8,1 say 'Refeed station number                           ' + ' ' + location
@ 9,1 say 'Cylinder adjusted gross weight (grams)          ' + ' ' +
          str(grosswt,9,2)
@ 10,1 say 'Cylinder adjusted tare weight (grams)          ' + ' ' +
          str(tarewt,9,2)
@ 11,1 say 'Cylinder adjusted net weight (grams)           ' + ' ' +
          str(netwt,9,2)
wo = purity*100.0
@ 12,1 say 'Uranium (wt%)                                  ' + ' ' +
          str(wo,9,2)
eo = assay*100.0
@ 13,1 say 'Enrichment (wt%)                               ' + ' ' +
          str(eo,9,2)
uvalu = netwt * purity
u235valu = uvalu * assay
@ 14,1 say 'Calculated U total (grams)                     ' + ' ' +
          str(uvalu,9,2)
@ 15,1 say 'Calculated U235 (grams)                        ' + ' ' +
          str(u235valu,9,2)
jj = 16
if !empty(notes1)
@ jj,1 say notes1
jj = jj + 1
endif
if !empty(notes2)
@ jj,1 say notes2
jj =jj + 1
endif
if !empty(notes3)
@ jj,1 say notes3
jj = jj + 1
endif
if !empty(notes4)
@ jj,1 say notes4
endif
set device to screen
endif
endif
```

```
    endif
if  eof()
//  @ 20,1 say 'end of outer loop recno' get yesno
//  read
  go nrecord
  endif
  skip 1
  enddo
    return nil
```

4) CLIPPER5 source code for the program to generate detailed reports of the accumulated amount of material removed from cylinders over a specified period
file c:\snri\src\iaea\three.prg

```
function three
field itemid, trans_type, purity, assay, location, grosswt, tarewt
field netwt,edate,etime,size,cycle,resid,notes1,notes2,notes3,notes4

field clock_date, clock_time
field cylinder
local yesno, uvalu, u235valu, match, oldloc, oldcyc, nrecord, now, recno
local fdate,ftime,notesa,notesb,notesc,notesd,dlock_date,dlock_time
local diff, discon, jj, before, after, total1, total2, total3
local totala, totalb, totalc, iditem, wo, eo, a, b, c
set date british
set confirm on
grand1 = 0.0
grand2 = 0.0
grand3 = 0.0
use c:\snri\dbf\cyl.dbf new alias cyl exclusive

do while .not. eof()   // start # 1
match = cylinder
select history
go top
   cls
   yesno = ' '
   now = 'n'
   total1 = 0.0
   total2 = 0.0
   total3 = 0.0
   totala = 0.0
   totalb = 0.0
   totalc = 0.0

 do while .not. eof()       // start # 2

    if match =  itemid .and. trans_type = 'c'        // start # 3
//   "c" record
// if the connect time is after the end of the sampling period
// do not include the cycle !
    test1 = restim(enddate,endtime,edate,etime)
    nrecord = recno()
//    @ 13,1 say match+'  '+str(nrecord)+'  '+trans_type+'  '+str(test1)+'  '+
                dtoc(edate)+ ' '+etime get yesno
//    read

    if test1  > 0.0      // start # 2.5
    nrecord = recno()
    oldloc = location
    oldcyc = cycle
//    @ 14,1 say match+'  '+str(nrecord)+'  '+trans_type+'  '+str(test1) get
                yesno
//    read
//    cls
    skip 1


      do while .not. eof()      // start # 4
//    "d" record
```

```
      nrec = recno()
//      @ 15,1 say 'recno'+' '+itemid+' '+trans_type+' '+str(nrec) get yesno
//      read


        if itemid = match .and. trans_type = 'd'     // start # 5
// There is a disconnect record.
          before = restim(edate,etime,begdate,begtime)
          after = restim(enddate,endtime,edate,etime)
// does the disconnect occur during the sampling period ?
// @ 16,1 say str(before)+' '+str(after)+' '+dtoc(edate)+' '+etime get yesno
// read
          if before>0.0.and.after>0.0          // start # 6
// Yes.
            cls
            if substr(oldloc,2,1)='  '
            a = substr(oldloc,1,1)
            b = stuff(oldloc,1,1,"")
            c = stuff(b,2,1,a)
            oldloc = c
            endif
            newloc = location
            if substr(newloc,2,1) = '  '
            a = substr(newloc,1,1)
            b = stuff(newloc,1,1,"")
            c = stuff(b,2,1,a)
            newloc = c
            endif
            if newloc # oldloc        // start # 7
            @ 1,1 say 'Refeed station number does not match for cylinder '
                      +match
            @ 2,1 say 'Connect refeed station number      '+oldloc
            @ 3,1 say 'Disconnect refeed station number '+newloc
            @ 4,1 get yesno
            read
            endif        // end # 7
            cls

            if cycle # oldcyc       // start # 8
            @ 2,1 say '  #8  '+match+'  '+str(oldcyc)+'  '+itemid+'  '+str(cycle)
            @ 3,1 get yesno
            read

            endif      // end # 8
          fdate = edate
          ftime = etime
          notesa = notes1
          notesb = notes2
          notesc = notes3
          notesd = notes4
          dlock_date = clock_date
          dlock_time = clock_time
          discon = grosswt
          now = 'y'
          exit
          endif    // end # 6

        endif    // end # 5
      skip 1
      loop
      enddo      // end # 4

// @ 17,1 say 'enddo 4' get yesno
// read
```

```
      if .not.eof() .and. now = 'y'            // start # 9
      go nrecord
      nrec = recno()
      iditem = itemid
// @ 18,1 say 'nrecord' + ' ' + str(nrec)   get yesno
// read
     cls
     @  1,1 say 'Accumulated amount of material removed from cylinders'
     @  3,1 say 'Cylinder number                                  ' + ' ' + itemid
     @  4,1 say 'Sampling period beginning date and time          ' + ' ' +
                dtoc(begdate)+'  '+begtime
     @  5,1 say 'Sampling period ending date and time             ' + ' ' +
                dtoc(enddate)+'  '+endtime
     @  6,1 say 'Refeed station number                                  '+ location
     @  7,1 say 'Date and time cylinder connected                 ' + ' ' +
                dtoc(edate) + ' ' + etime
     @  8,1 say 'Mailbox input date and time                      ' + ' ' +
                dtoc(clock_date) + ' ' + clock_time

        jj = 9
        if !empty(notes1)
     @  jj,1 say notes1
        jj = jj + 1
        endif
        if !empty(notes2)
     @  jj,1 say notes2
        jj = jj + 1
        endif
        if !empty(notes3)
     @  jj,1 say notes3
        jj = jj + 1
        endif
        if !empty(notes4)
     @  jj,1 say notes4
        jj = jj + 1
        endif
     @  jj,1 say 'Date and time cylinder disconnected             ' + ' ' +
                 dtoc(fdate) + ' ' + ftime
        jj = jj + 1
     @  jj,1 say 'Mailbox input date and time                     ' + ' ' +
                 dtoc(dlock_date) + ' ' + dlock_time
        jj = jj + 1
        if !empty(notesa)
     @  jj,1 say notesa
        jj = jj + 1
        endif
        if !empty(notesb)
     @  jj,1 say notesb
        jj = jj + 1
        endif
        if !empty(notesc)
     @  jj,1 say notesc
        jj = jj + 1
        endif
        if !empty(notesd)
     @  jj,1 say notesd
        jj = jj + 1
        endif
     @ jj,1 say 'Connected cylinder adjusted gross weight (grams)    ' + ' ' +
                str(grosswt,9,2)
       jj = jj + 1
     @ jj,1 say 'Disconnected cylinder adjusted gross weight (grams)' + ' ' +
                str(discon,9,2)
       jj = jj + 1
```

```
  diff = grosswt - discon
  total1 = total1 + diff
  grand1 = grand1 + diff
@ jj,1 say 'Amount of material removed from cylinder (grams)    ' + ' ' +
           str(diff,9,2)
  jj = jj + 1
  wo = purity*100.0
@ jj,1 say 'Uranium (wt%)                                        ' + ' ' +
           str(wo,9,2)
  jj = jj + 1
  eo = assay*100.0
@ jj,1 say 'Enrichment (wt%)                                     ' + ' ' +
           str(eo,9,2)
  jj = jj + 1
  uvalu = diff * purity
  total2 = total2 + uvalu
  grand2 = grand2 + uvalu
@ jj,1 say 'Calculated U total (grams)                           ' + ' ' +
           str(uvalu,9,2)
  jj = jj + 1
  u235valu = uvalu * assay
  total3 = total3 + u235valu
  grand3 = grand3 + u235valu
@ jj,1 say 'Calculated U235 (grams)                              ' + ' ' +
           str(u235valu,9,2)
  jj = jj + 1
@ jj,1  get yesno
read

if pr='y'.or.pr='Y'        // start # 10
set device to printer
@ 1,1 say 'Accumulated amount of material removed from cylinders'
@ 3,1 say 'Cylinder number                                  ' + ' ' + itemid
@ 4,1 say 'Sampling period beginning date and time          ' + ' ' +
          dtoc(begdate)+'  '+begtime
@ 5,1 say 'Sampling period ending date and time             ' + ' ' +
          dtoc(enddate) +'  '+endtime

iditem = itemid
@ 6,1 say 'Refeed station number                                ' + location
@ 7,1 say 'Date and time cylinder connected                 ' + ' ' +
          dtoc(edate) + ' ' + etime
@ 8,1 say 'Mailbox input date and time                      ' + ' ' +
          dtoc(clock_date) + ' ' + clock_time

   jj = 9
   if !empty(notes1)
@ jj,1 say notes1
  jj = jj + 1
  endif
  if !empty(notes2)
@ jj,1 say notes2
  jj = jj + 1
  endif
  if !empty(notes3)
@ jj,1 say notes3
  jj = jj + 1
  endif
  if !empty(notes4)
@ jj,1 say notes4
  jj = jj + 1
  endif
@ jj,1 say 'Date and time cylinder disconnected             ' + ' ' +
          dtoc(fdate) + ' ' + ftime
```

```
        jj = jj + 1
      @ jj,1 say 'Mailbox input date and time                    ' + ' ' +
                 dtoc(dlock_date) + ' '+ dlock_time
      jj = jj + 1
      if !empty(notesa)
      @ jj,1 say notesa
      jj = jj + 1
      endif
      if !empty(notesb)
      @ jj,1 say notesb
      jj = jj + 1
      endif
      if !empty(notesc)
      @ jj,1 say notesc
      jj = jj + 1
      endif
      if !empty(notesd)
      @ jj,1 say notesd
      jj = jj + 1
      endif
      @ jj,1 say 'Connected cylinder adjusted gross weight (grams)   ' + ' ' +
                 str(grosswt,9,2)
        jj = jj + 1
      @ jj,1 say 'Disconnected cylinder adjusted gross weight (grams)' + ' ' +
                 str(discon,9,2)
        jj = jj + 1
        diff = grosswt - discon
        totala = totala + diff
      @ jj,1 say 'Amount of material removed from cylinder (grams)   ' + ' ' +
                 str(diff,9,2)
        jj = jj + 1
        wo = purity*100.0
      @ jj,1 say 'Uranium (wt%)                                      ' + ' ' +
                 str(wo,9,2)
        jj = jj + 1
        eo = assay*100.0
      @ jj,1 say 'Enrichment (wt%)                                   ' + ' ' +
                 str(eo,9,2)
        jj = jj + 1
        uvalu = diff * purity
        totalb = totalb + uvalu
      @ jj,1 say 'Calculated U total (grams)                         ' + ' ' +
                 str(uvalu,9,2)
        jj = jj + 1
        u235valu = uvalu * assay
      totalc = totalc + u235valu
      @ jj,1 say 'Calculated U235 (grams)                            ' + ' ' +
                 str(u235valu,9,2)
      set device to screen

      endif       // end # 10

      endif        // end # 9
//    @ 19,1 say 'endif 2.5' get yesno
//    read
     endif         // end # 2.5
//    @ 20,1 say 'endif 3' get yesno
//    read

    endif          // end # 3
   skip 1
   loop

   enddo              // end # 2
```

```
   cls
// if .not. eof()
// @ 21,1 say 'enddo 2' get yesno
// read
// else
// @ 21,1 say 'enddo 2 eof()' get yesno
// read
// endif
if now = 'y'
   jj = 1
   @ jj,1 say 'Summary for cylinder                                  ' + '   ' + iditem
   jj = jj + 1
   @ jj,1 say 'Amount of material removed from cylinder (grams)' + ' ' +
              str(total1,9,2)
   jj = jj + 1
   @ jj,1 say 'U total removed from cylinder (grams)            ' + ' ' +
              str(total2,9,2)
   jj = jj + 1
   @ jj,1 say 'U235 removed from cylinder (grams)               ' + ' ' +
              str(total3,9,2)
   jj = jj + 1
   @ jj,1 get yesno
   read
if  pr='y'.or.pr='Y'
set device to printer
   jj = 1
   @ jj,1 say 'Summary for cylinder                                ' + ' ' + iditem
   jj = jj + 1
   @ jj,1 say 'Amount of material removed from cylinder (grams)' + ' ' +
              str(totala,9,2)
   jj = jj + 1
   @ jj,1 say 'U total removed from cylinder (grams)            ' + ' ' +
              str(totalb,9,2)
   jj = jj + 1
   @ jj,1 say 'U235 removed from cylinder (grams)               ' + ' ' +
              str(totalc,9,2)
set device to screen
endif
endif
select cyl
skip 1
loop

end do            // end # 1
cls
   jj=1
   @ jj,1 say 'Summary for all cylinders'
   jj = jj + 1
   @ jj,1 say 'Amount of material removed from all cylinders (grams)' + ' ' +
              str(grand1,9,2)
   jj = jj + 1
   @ jj,1 say 'U total removed from all cylinders (grams)            ' + ' ' +
              str(grand2,9,2)
   jj = jj + 1
   @ jj,1 say 'U235 removed from all cylinders (grams)               ' + ' ' +
              str(grand3,9,2)
   jj = jj + 1
   @ jj,1 get yesno
   read
if  pr='y'.or.pr='Y'
set device to printer
   jj=1
   @ jj,1 say 'Summary for all cylinders'
   jj = jj + 1
```

```
   @ jj,1 say 'Amount of material removed from all cylinders (grams)' + ' ' +
              str(grand1,9,2)
   jj = jj + 1
   @ jj,1 say 'U total removed from all cylinders (grams)           ' + ' ' +
              str(grand2,9,2)
   jj = jj + 1
   @ jj,1 say 'U235 removed from all cylinders (grams)              ' + ' ' +
              str(grand3,9,2)
   for jj=5 to 60
   @ jj,1 say ' '
   next
set device to screen
endif
   close cyl
   return nil
```

5) CLIPPER5 source code for the program to generate reports of actual residence time compared with declared residence time for normal refeed
file c:\snri\src\iaea\four.prg

```
function four
field itemid, trans_type, purity, assay, location, grosswt, tarewt
field netwt, edate, etime, size, cycle, resid,notes1,notes2,notes3,notes4

field clock_date, clock_time
field cylinder
local yesno, uvalu, u235valu, match, oldloc, oldcyc, nrecord, now, recno
local fdate, ftime, notesa, notesb, notesc, notesd, dlock_date, dlock_time
local diff, diffa, diffb, howlong, jj, gdate, gtime, oldresid, newloc
local a,b
set date british
set confirm on
use c:\snri\dbf\cyl2.dbf new alias cyl2 exclusive
do while .not. eof()
match = cylinder
select history
go top
    cls
    yesno = ' '
    now = 'n'
  do while .not. eof()
    if match=itemid .and. trans_type = 'c'
//  "c" record
    nrecord = recno()
    oldresid = resid
    oldloc = location
    oldcyc = cycle
    gdate = edate
    gtime = etime
    cls
//    @ 19,1 say match + ' ' + str(nrecord) get yesno
//    read
//    cls
    skip 1

      do while .not. eof()
//    "d" record
      nrec = recno()
//      @ 20,1 say 'recno' + ' ' + str(nrec) get yesno
//      read
      now = 'n'
        if itemid = match .and. trans_type = 'd'
             cls
             if substr(oldloc,2,1)=' '
             a = substr(oldloc,1,1)
             b = stuff(oldloc,1,1,"")
             c = stuff(b,2,1,a)
             oldloc = c
             endif
             newloc = location
             if substr(newloc,2,1) = ' '
             a = substr(newloc,1,1)
             b = stuff(newloc,1,1,"")
             c = stuff(b,2,1,a)
             newloc = c
             endif
```

```
            if newloc # oldloc
            @ 1,1 say 'Refeed station number does not match for cylinder'+
                      ' '+match
            @ 2,1 say 'Connect refeed station number      '+oldloc
            @ 3,1 say 'Disconnect refeed station number '+newloc
            @ 4,1 get yesno
            read
            endif
            cls
            if cycle # oldcyc
            @ 2,1 say 'cycle does not match'+' '+match+ ' ' + str(oldcyc) +
                      ' ' + itemid + ' ' + str(cycle)
            @ 3,1 get yesno
            read
            endif
          howlong = restim(edate,etime,gdate,gtime)
          fdate = edate
          ftime = etime
          notesa = notes1
          notesb = notes2
          notesc = notes3
          notesd = notes4
          dlock_date = clock_date
          dlock_time = clock_time
          diff = howlong - oldresid
//          @ 4,1 say 'howlong resid diff'+' '+str(howlong,9,2)+' '+
                    str(oldresid,9,2)+' '+str(diff,9,2)
//          @ 5,1 get yesno
//          read
          now = 'y'
          exit
        endif
      skip 1
      loop
      enddo
      if .not.eof() .and. now = 'y'
      go nrecord
      nrec = recno()
//      @ 21,1 say 'nrecord' + ' ' + str(nrec)  get yesno
//      read
      @ 1,1 say 'Comparison of the actual residence time of a cylinder'
      @ 2,1 say 'with the declared residence time for normal refeed:'
      @ 4,1 say 'Cylinder number                                  ' +' ' + itemid
      @ 5,1 say 'Date and time cylinder connected          ' + ' ' + dtoc(edate)
                + ' ' + etime
      diffa = restim(clock_date,clock_time,edate,etime)
      @ 6,1 say 'Mailbox input date and time               ' + ' ' +
                dtoc(clock_date) + ' ' + clock_time
      @ 7,1 say 'Connect declaration time difference (hours)       ' + ' ' +
                str(diffa,9,2)
        jj = 8
        if !empty(notes1)
      @ jj,1 say notes1
        jj = jj + 1
        endif
        if !empty(notes2)
      @ jj,1 say notes2
        jj = jj + 1
        endif
        if !empty(notes3)
      @ jj,1 say notes3
        jj = jj + 1
        endif
        if !empty(notes4)
```

```
@  jj,1 say notes4
   jj = jj + 1
   endif
@  jj,1 say 'Date and time cylinder disconnected       ' + ' ' +
            dtoc(fdate) + ' '+ ftime
   diffb  =  restim(dlock_date,dlock_time,fdate,ftime)
   jj = jj + 1
@  jj,1 say 'Mailbox input date and time               ' + ' ' +
            dtoc(dlock_date) + ' ' + dlock_time
   jj = jj + 1
@  jj,1 say 'Disconnect declaration time difference (hours) ' + ' ' +
            str(diffb,9,2)
   jj = jj + 1
   if  !empty(notesa)
@  jj,1 say notesa
   jj = jj + 1
   endif
   if  !empty(notesb)
@  jj,1 say notesb
   jj = jj + 1
   endif
   if  !empty(notesc)
@  jj,1 say notesc
   jj = jj + 1
   endif
   if  !empty(notesd)
@  jj,1 say notesd
   jj = jj + 1
   endif
@ jj,1 say 'Connected cylinder adjusted gross weight (grams) ' + ' ' +
           str(grosswt,9,2)
  jj = jj + 1
@ jj,1 say 'Connected cylinder adjusted tare weight (grams)  ' + ' ' +
           str(tarewt,9,2)
  jj = jj + 1
@ jj,1 say 'Connected cylinder adjusted net weight (grams)   ' + ' ' +
           str(netwt,9,2)
  jj = jj + 1
@ jj,1 say 'Declared residence time (hours)                  ' + ' ' +
           str(resid,9,2)
  jj = jj + 1
@ jj,1 say 'Actual residence time (hours)                    ' + ' ' +
           str(howlong,9,2)
  jj = jj + 1
@ jj,1 say 'Residence time difference (hours)                ' + ' ' +
           str(diff,9,2)
jj = jj + 1
@ jj,1 get yesno
read
if pr='y'.or.pr='Y'
set device to printer
@  1,1 say 'Comparison of the actual residence time of a cylinder'
@  2,1 say 'with the declared residence time for normal refeed:'
@  4,1 say 'Cylinder number                                  ' +' '+ itemid
@  5,1 say 'Date and time cylinder connected         ' + ' ' + dtoc(edate)
           + ' ' + etime
diffa  =  restim(clock_date,clock_time,edate,etime)
@  6,1 say 'Mailbox input date and time              ' + ' ' +
           dtoc(clock_date) + ' ' + clock_time
@  7,1 say 'Connect declaration time difference (hours)      ' + ' ' +
           str(diffa,9,2)
   jj = 8
   if  !empty(notes1)
@  jj,1 say notes1
```

```
      jj = jj + 1
      endif
      if !empty(notes2)
   @  jj,1 say notes2
      jj = jj + 1
      endif
      if !empty(notes3)
   @  jj,1 say notes3
      jj = jj + 1
      endif
      if !empty(notes4)
   @  jj,1 say notes4
      jj = jj + 1
      endif
   @  jj,1 say 'Date and time cylinder disconnected     ' + ' ' +
               dtoc(fdate) + ' ' + ftime
      diffb = restim(dlock_date,dlock_time,fdate,ftime)
      jj = jj + 1
   @  jj,1 say 'Mailbox input date and time             ' + ' ' +
               dtoc(dlock_date) + ' ' + dlock_time
      jj = jj + 1
   @  jj,1 say 'Disconnect declaration time difference (hours) ' + ' ' +
               str(diffb,9,2)
      jj = jj + 1
      if !empty(notesa)
   @  jj,1 say notesa
      jj = jj + 1
      endif
      if !empty(notesb)
   @  jj,1 say notesb
      jj = jj + 1
      endif
      if !empty(notesc)
   @  jj,1 say notesc
      jj = jj + 1
      endif
      if !empty(notesd)
   @  jj,1 say notesd
      jj = jj + 1
      endif
   @ jj,1 say 'Connected cylinder adjusted gross weight (grams) ' + ' ' +
             str(grosswt,9,2)
     jj = jj + 1
   @ jj,1 say 'Connected cylinder adjusted tare weight (grams)  ' + ' ' +
             str(tarewt,9,2)
   jj = jj + 1
   @ jj,1 say 'Connected cylinder adjusted net weight (grams)   ' + ' ' +
             str(netwt,9,2)
     jj = jj + 1
   @ jj,1 say 'Declared residence time (hours)                  ' + ' ' +
             str(resid,9,2)
     jj = jj + 1
   @ jj,1 say 'Actual residence time (hours)                    ' + ' ' +
             str(howlong,9,2)
     jj = jj + 1
   @ jj,1 say 'Residence time difference (hours)                ' + ' ' +
             str(diff,9,2)
   set device to screen
   endif
   endif
  endif
skip 1
enddo
select cyl2
```

```
skip 1
loop
enddo
  close cyl2
  return nil
```

6) CLIPPER5 source code for the program to generate a compact printed table of the material removed from all cylinders over a specified period
file c:\snri\src\iaea\five.prg

```
function five
field itemid, trans_type, purity, assay, location, grosswt, tarewt
field netwt,edate,etime,size,cycle,resid,notes1,notes2,notes3,notes4

field clock_date, clock_time
field cylinder
local yesno, uvalu, u235valu, match, oldloc, oldcyc, nrecord, now, recno
local fdate, ftime, nrec
local diff, discon, jj, before, after, total1, total2, total3
local wo, eo, howlong, newloc, a, b, c

set date british
set confirm on

yesno = ' '
grand2 = 0.0
grand3 = 0.0
set device to printer
 @ 2,1 say 'Sampling period beginning date and time      '+' '+
           dtoc(begdate)+'  '+begtime
 @ 3,1 say 'Sampling period ending date and time         '+' '+
           dtoc(enddate)+'  '+endtime
 @ 5,1 say 'Cyl.  Ref.  Date  Date Elapsed  Mat.      U  U235    U     U235'
 @ 6,1 say ' No.  Sta.  Conn. Disc. time  Removed '
 @ 7,1 say '                            (hours) (grams) (wt%)(wt%)(grams)(grams)'
set device to screen
  jj = 8
use c:\snri\dbf\cyl.dbf new alias cyl exclusive
// @ 1,1 say 'start loop 1 through all cylinders' +' '+cylinder get yesno
// read
do while .not. eof() // start # 1
match = cylinder
select history
go top
   yesno = ' '
   now = 'n'
// @ 2,1 say 'start loop2' +' '+itemid+' '+trans_type get yesno
// read
   do while .not. eof() // start # 2
// look for a 'c' record for this cylinder
// @ 3,1 say 'start loop 3' get yesno
// read
    if match =  itemid .and. trans_type = 'c'    // start # 3
//  "c" record found
// if the connect time is after the end of the sampling period
// do not include the cycle !
    test1 = restim(enddate,endtime,edate,etime)
// @ 4,1 say 'start loop 3.5'+' '+'c record'+' '+match + ' ' + str(nrecord)
//           get yesno
// read
    if test1  > 0.0  // start # 3.5
    nrecord = recno()
    cls
    skip 1
// @ 5,1 say 'start loop 4'+' '+'d record recno' + ' ' + str(nrec) get yesno
// read
```

```
      do while .not. eof()   // start # 4
//    look for the "d" record that matches the 'c' record
      nrec = recno()
// @ 6,1 say 'start loop 5' get yesno
// read

        if itemid = match .and. trans_type = 'd'    // start # 5
          before = restim(edate,etime,begdate,begtime)
          after = restim(enddate,endtime,edate,etime)
// @ 17,1 say itemid+' '+str(before)+' '+dtoc(edate)+' '+etime+' '+
//            dtoc(begdate)+' '+begtime
// @ 18,1 say itemid+' '+str(after)+' '+dtoc(enddate)+' '+endtime+' '+
//            dtoc(edate)+' '+etime
// @ 19,1 get yesno
// read
// @ 7,1 say 'start loop 6' get yesno
// read
// does the disconnect occur during the sampling period ?
         if before>0.0.and.after>0.0        // start # 6
// yes
         fdate = edate
         ftime = etime
         discon = grosswt
         now = 'y'
         exit
         else
// no
         now = 'n'
         exit
// @ 8,1 say 'end loop 6' get yesno
// read
         endif     // end # 6
// @ 9,1 say 'end loop 5' get yesno
// read
        endif // end # 5
      skip 1
      loop
// @ 10,1 say 'end loop 4' get yesno
// read
enddo     // end # 4
// @ 11,1 say 'start loop 9' + ' ' + 'nrecord' + ' ' + str(nrec)   get yesno
// read
      if .not.eof() .and. now = 'y'   // start # 9
      go nrecord
      nrec = recno()
      howlong = restim(fdate,ftime,edate,etime)
      diff = grosswt - discon
      wo = purity*100.0
      eo = assay*100.0
      uvalu = diff * purity
      u235valu = uvalu * assay
      grand2 = grand2 + uvalu
      grand3 = grand3 + u235valu
      newloc = location
      if substr(newloc,2,1) = ' '
      a = substr(newloc,1,1)
      b = stuff(newloc,1,1,"")
      c = stuff(b,2,1,a)
      newloc = c
      endif
set device to printer
@ jj,1 say ' '+itemid+'     '+newloc+' '+dtoc(edate)+' '+dtoc(fdate)+' '+
           str(howlong,7,0)+' '+str(diff,8,0)+' '+str(wo,5,2)+' '+
           str(eo,5,2)+' '+str(uvalu,8,0)+' '+str(u235valu,8,0)
```

```
set device to screen
      jj = jj + 1

// @ 12,1 say 'end loop 9' get yesno
// read
      endif    // end # 9

// @ 13,1 say 'end loop 3.5' get yesno
// read
   endif      // end # 3.5

// @ 14,1 say 'end loop 3' get yesno
// read
    endif      // end # 3
   skip 1
// @ 15,1 say 'loop in loop 2' get yesno
// read
  loop

// @ 15,1 say 'end loop 2'  get yesno
// read
  enddo      // end # 2
select cyl
skip 1
// @ 16,1 say 'loop in loop 1' get yesno
// read
loop

// @ 16,1 say 'end loop 1' get yesno
// read
end do       // end # 1
// @ 22,1 get yesno
// read
set device to printer
      @ jj+1,1 say 'Total material removed:'+' '+str(grand2,9,0)+' ' +
                   'U (grams)'+' '+str(grand3,9,0)+' '+'U235 (grams)'
set device to screen
   close cyl
   return nil
```

7) Listing of input data to the CLIPPER RMAKE utility
file c:\snri\src\iaea\iaea.rmk

```
obj1  :=  \snri\obj\iaea.obj  \snri\obj\one.obj  \snri\obj\two.obj
obj2  :=  \snri\obj\three.obj  \snri\obj\four.obj
obj3  :=  \snri\obj\five.obj

// Inference Rules to Compile obj's from prg's
.prg.obj:
  c:\clippper5\bin\clipper  $**  /a/b/m/n/w/o\SNRI\obj\$*
.c.obj:
  cl /c /AL /Dalt /Z1 /Fpa /Gs $*.c
  copy $*.obj ..\obj

// Dependency list for prg modules
SNRI\obj\iaea.obj:  \SNRI\SRC\iaea\iaea.prg
SNRI\obj\one.obj:  \SNRI\SRC\iaea\one.prg
SNRI\obj\two.obj:  \SNRI\SRC\iaea\two.prg
SNRI\obj\three.obj:  \SNRI\SRC\iaea\three.prg
SNRI\obj\four.obj:  \SNRI\SRC\iaea\four.prg
SNRI\obj\five.obj:  \SNRI\SRC\iaea\five.prg

// Dependency list for exe module
\SNRI\exe\iaeamain.exe:  $(obj1)  $(obj2)  $(obj3)
  c:\blinker2\blinker  @\snri\src\iaea\iaea.lnk
```

8) Listing of input to the BLINKER2 loader
file c:\snri\src\iaea\iaea.lnk

```
NOBELL
FILE \snri\obj\iaea
FILE \snri\obj\one
FILE \snri\obj\two
FILE \snri\obj\three
FILE \snri\obj\four
FILE \snri\obj\five
OUTPUT iaeamain
LIBRARY c:\lib\v51\sgit51
LIBRARY c:\clipper5\lib\clipper
LIBRARY c:\clipper5\lib\extend
LIBRARY c:\clipper5\lib\terminal
LIBRARY c:\clipper5\lib\dbfntx
LIBRARY c:\c510\lib\llibca
```

9) Batch file to execute the report generator program
file c:\reports.bat

```
@echo off
reptmain %1
```

(THIS PAGE INTENTIONALLY LEFT BLANK)

Appendix R

Sample Reports

1) Sample listing of declarations for cylinder operations

2) Report of a cylinder that should be attached to a feed station because its residence time has not been exceeded

3) Report of a cylinder that should be attached to a feed station because there is no cylinder disconnection reported for it

4) Detailed report of the accumulated amount of uranium removed from cylinders over a specified period

5) Compact printed table of the uranium removed from all cylinders over a specified period

6) Report of actual residence time of a cylinder compared with its declared residence time for normal refeed operations

Appendix R

1) Sample listing of declarations for cylinder operations

Cylinder Number	Refeed Station No.	Event Type	Event Date	Event Time	Mailbox Input Date	Mailbox Input time	Cylinder Adusted Gross Weight (grams)	Cylinder Adjusted Tare Weight (grams)
st0144	05	d	01/20/98	09:18:00	01/20/98	11:32:05	26120.000	25618.000
050099	06	d	01/20/98	09:19:00	01/20/98	11:32:05	28870.000	28331.000
n54019	05	c	01/20/98	10:37:00	01/20/98	11:32:05	35180.000	28037.000
051874	06	c	01/20/98	10:35:00	01/20/98	11:32:05	43890.000	26827.000
st0142	02	d	01/21/98	14:27:00	01/21/98	15:21:14	25790.000	25306.000
st0298	02	c	01/21/98	15:09:00	01/21/98	15:21:14	42280.000	25943.000
st0120	03	d	01/22/98	09:36:00	01/22/98	14:14:51	25800.000	25475.000
051938	08	d	01/22/98	09:37:00	01/22/98	14:14:51	29820.000	28896.000
st0100	10	d	01/22/98	09:38:00	01/22/98	14:14:51	26560.000	26166.000
st0138	03	c	01/22/98	12:32:00	01/22/98	14:14:51	36400.000	25686.000
st0273	08	c	01/22/98	12:36:00	01/22/98	14:14:51	39100.000	25417.000
st0275	10	c	01/22/98	12:40:00	01/22/98	14:14:51	32650.000	26214.000
st0085	07	d	01/23/98	08:50:00	01/23/98	15:28:45	26420.000	25615.000
st0355	09	d	01/23/98	08:45:00	01/23/98	15:28:45	26180.000	25742.000
st0167	07	c	01/23/98	10:05:00	01/23/98	15:28:45	26270.000	26038.000
st0396	09	c	01/23/98	10:09:00	01/23/98	15:28:45	34040.000	25679.000
n54019	05	d	01/24/98	09:00:00	01/24/98	11:41:53	28410.000	28037.000
051874	06	d	01/24/98	09:05:00	01/24/98	11:41:53	27170.000	26827.000
st0038	05	c	01/24/98	10:20:00	01/24/98	11:41:53	41250.000	25256.000
051528	06	c	01/24/98	10:15:00	01/24/98	11:41:53	38630.000	27524.000
st0298	02	d	01/25/98	09:15:00	01/25/98	11:28:59	26320.000	25943.000
st0275	10	d	01/25/98	09:20:00	01/25/98	11:28:59	26340.000	26214.000
st0122	02	c	01/25/98	10:00:00	01/25/98	11:28:59	38260.000	25075.000
st0269	10	c	01/25/98	10:05:00	01/25/98	11:28:59	36900.000	26395.000

2) Report of a cylinder that should be attached to a feed station because its residence time has not been exceeded

The following cylinder should be attached to a feed station because the residence time has not been exceeded:

Cylinder number		st0038
Date and time inspection began	26/01/98	13:30:00
Date and time cylinder connected	24/01/98	10:20:00
Residence time (hours)		66.00
Elapsed time (hours)		51.17
Refeed station number		05
Cylinder adjusted gross weight (grams)		41250.00
Cylinder adjusted tare weight (grams)		25256.00
Cylinder adjusted net weight (grams)		15994.00
Uranium (wt%)		67.34
Enrichment (wt%)		96.42
Calculated U total (grams)		10770.36
Calculated U235 (grams)		10384.78

3) Report of a cylinder that should be attached to a feed station because there is no cylinder disconnection reported for it

The following cylinder should be attached to a feed station because there is no disconnect recorded:

Cylinder number		st0122
Date and time inspection began	26/01/98	13:30:00
Date and time cylinder connected	25/01/98	10:20:00
Residence time (hours)		66.00
Refeed station number		02
Cylinder adjusted gross weight (grams)		38260.00
Cylinder adjusted tare weight (grams)		25075.00
Cylinder adjusted net weight (grams)		13185.00
Uranium (wt%)		67.35
Enrichment (wt%)		93.06
Calculated U total (grams)		8880.10
Calculated U235 (grams)		8263.82

Appendix R

4) Detailed report of the accumulated amount of uranium removed from cylinders over a specified period

Accumulated amount of material removed from cylinders

Cylinder number		st0298
Sampling period beginning date and time	17/12/97	00:01:00
Sampling period ending date and time	26/01/98	13:30:00
Refeed station number		02
Date and time cylinder connected	21/01/98	15:09:00
Mailbox input date and time	21/01/98	15:21:14
Date and time cylinder disconnected	25/01/98	09:15:00
Mailbox input date and time	25/01/98	11:28:59
Connected cylinder adjusted gross weight (grams)		42280.00
Disconnected cylinder adjusted gross weight (grams)		26320.00
Amount of material removed from cylinder (grams)		15960.00
Uranium (wt%)		67.30
Enrichment (wt%)		96.61
Calculated U total (grams)		10741.08
Calculated U235 (grams)		10376.96

Summary for cylinder	st0298
Amount of material removed from cylinder (grams)	15960.00
U total removed from cylinder (grams)	10741.08
U235 removed from cylinder (grams)	10376.96

Summary for all cylindera	
Amount of material removed from all cylinders (grams)	45760.00
U total removed from all cylinders (grams)	30830.09
U235 removed from all cylinders (grams)	23748.64

5) Compact printed table of the uranium removed from all cylinders over a specified period

Sampling period beginning date and time 17/12/97 00:01:00
Sampling period ending date and time 26/01/98 13:30:00

Cyl. No.	Ref. Sta.	Date Conn.	Date Disc.	Elapsed Time (hours)	Mat. Removed (grams)	U (wt%)	U235 (wt%)	U (grams)	U235 (grams)
n54019	05	20/01/98	24/01/98	94	6770	67.33	92.86	4558	4233
051874	06	20/01/98	24/01/98	95	16720	67.47	45.92	11281	5180
st0298	02	21/01/98	25/01/98	90	15960	67.30	96.61	10741	10377
st0275	10	22/01/98	25/01/98	69	6310	67.35	93.15	4250	3959

Total material removed: 30830 U (grams) 23749 U235 (grams)

6) Report of actual residence time of a cylinder compared with its declared residence time for normal refeed operations

Comparison of the actual residence time of a cylinder with the declared residence time for normal refeed:

Cylinder number		st0298
Date and time cylinder connected	21/01/98	15:09:00
Mailbox input date and time	21/01/98	15:21:14
Connect declaration time difference (hours)		0.20
Date and time cylinder disconnected	25/01/98	09:15:00
Mailbox input date and time	25/01/98	11:28:59
Disconnect declaration time difference (hours)		2.22
Connected cylinder adjusted gross weight (grams)		42280.00
Connected cylinder adjusted tare weight (grams)		25943.00
Connected cylinder adjusted net weight (grams)		16337.00
Declared residence time (hours)		66.00
Actual residence time (hours)		90.10
Residence time difference (hours)		24.10

(THIS PAGE INTENTIONALLY LEFT BLANK)