COMPUTER Program and User Documentation
Medical Data Tape Retrieval System

TN516

NASA-11579

Anderson (Philco-Ford Corp.)
1 Nov. 1971 547 p
CSCL 09B G3/08 13382

Submitted to
National Aeronautics and Space Administration
Manned Spacecraft Center
Information Systems Branch
Houston, Texas

Philco-Ford Corporation
Houston Operation
COMPUTER PROGRAM AND USER DOCUMENTATION

MEDICAL DATA TAPE RETRIEVAL SYSTEM

Contract NAS 9-11579

Submitted to the

BIOMEDICAL DATA SYSTEMS OFFICE

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
MANNED SPACECRAFT CENTER
INFORMATION SYSTEMS BRANCH
Houston, Texas

Submitted by:

Anderson, Cognizant Engineer

Approved by:

M. Brinkman, Supervisor
Data Processing Applications Section

N. Hines, Manager
Support Engineering Department

G. Straty, Manager
System Engineering Activity

PHILCO-FORD CORPORATION
AEROSPACE AND DEFENSE SYSTEMS OPERATIONS
WDL DIVISION
HOUSTON OPERATION
1002 GEMINI AVENUE
HOUSTON, TEXAS
<table>
<thead>
<tr>
<th>TABLE OF CONTENTS</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 SCOPE</td>
<td>1-1</td>
</tr>
<tr>
<td>2.0 BIOMEDICAL INFORMATION SYSTEM OVERVIEW</td>
<td>2-1</td>
</tr>
<tr>
<td>2.1 OVERVIEW SUMMARY</td>
<td>2-6</td>
</tr>
<tr>
<td>3.0 MDTRS SYSTEM</td>
<td>3-1</td>
</tr>
<tr>
<td>3.1 GENERAL SPECIFICATIONS</td>
<td>3-1</td>
</tr>
<tr>
<td>3.1.1 Background</td>
<td>3-1</td>
</tr>
<tr>
<td>3.1.2 Functions of the System</td>
<td>3-2</td>
</tr>
<tr>
<td>3.2 TECHNICAL SPECIFICATIONS</td>
<td>3-4</td>
</tr>
<tr>
<td>3.2.1 System Description</td>
<td>3-4</td>
</tr>
<tr>
<td>3.2.2 Input</td>
<td>3-4</td>
</tr>
<tr>
<td>3.2.3 Processing</td>
<td>3-5</td>
</tr>
<tr>
<td>3.2.4 Output</td>
<td>3-8</td>
</tr>
<tr>
<td>3.2.5 Buffers and Tables</td>
<td>3-9</td>
</tr>
<tr>
<td>3.2.6 System Flow</td>
<td>3-15</td>
</tr>
<tr>
<td>3.2.7 Hardware Configuration</td>
<td>3-18</td>
</tr>
<tr>
<td>3.2.8 System Block Diagram</td>
<td>3-18</td>
</tr>
<tr>
<td>3.3 SUBROUTINES</td>
<td>3-20</td>
</tr>
<tr>
<td>3.3.1 BLOO - Setup Boolean Tree</td>
<td>3-20</td>
</tr>
<tr>
<td>3.3.2 CDOO - Convert Date</td>
<td>3-46</td>
</tr>
<tr>
<td>3.3.3 CMOO - Compare</td>
<td>3-52</td>
</tr>
<tr>
<td>3.3.4 CP00 - Control Program</td>
<td>3-56</td>
</tr>
<tr>
<td>3.3.5 CT00 - Count</td>
<td>3-66</td>
</tr>
<tr>
<td>3.3.6 CYOO - COPY Data</td>
<td>3-73</td>
</tr>
<tr>
<td>3.3.7 EROO - Output Message</td>
<td>3-83</td>
</tr>
<tr>
<td>3.3.8 FAOO - FLOATING POINT - ASCII Conversion</td>
<td>3-88</td>
</tr>
<tr>
<td>3.3.9 FILL - Fill Buffer</td>
<td>3-102</td>
</tr>
<tr>
<td>3.3.10 FLOO - ASCII to Floating Point</td>
<td>3-106</td>
</tr>
<tr>
<td>3.3.11 HM00 - Magnetic Tape Handler</td>
<td>3-117</td>
</tr>
<tr>
<td>3.3.12 IAAR - Input TTY or Modem</td>
<td>3-132</td>
</tr>
<tr>
<td>3.3.13 IM00 - Input Message</td>
<td>3-142</td>
</tr>
<tr>
<td>3.3.14 ISO0 - Initialize System</td>
<td>3-152</td>
</tr>
<tr>
<td>3.3.15 ISB1 - Input a character from KSR-35</td>
<td>3-168</td>
</tr>
<tr>
<td>3.3.16 L100 - List</td>
<td>3-172</td>
</tr>
<tr>
<td>3.3.17 MA00 - Match</td>
<td>3-211</td>
</tr>
<tr>
<td>3.3.18 MC00 - Match Condition</td>
<td>3-255</td>
</tr>
<tr>
<td>3.3.19 MD00 - Match Date</td>
<td>3-263</td>
</tr>
<tr>
<td>3.3.20 MW00 - Match What</td>
<td>3-272</td>
</tr>
<tr>
<td>3.3.21 OM00 - Output Message</td>
<td>3-280</td>
</tr>
<tr>
<td>3.3.22 OS00 - Operator Search</td>
<td>3-290</td>
</tr>
<tr>
<td>3.3.23 OTO0 - Output Teletype or 103 Modem</td>
<td>3-296</td>
</tr>
<tr>
<td>3.3.24 OZKC - Output Character to TTY</td>
<td>3-309</td>
</tr>
<tr>
<td>3.3.25 OZKB - Output Character to TTY</td>
<td>3-313</td>
</tr>
<tr>
<td>3.3.26 OZKA - Print Buffer on TTY</td>
<td>3-317</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS (Continued)

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.27</td>
<td>OZKR - Carriage Return and Line Feed</td>
<td>3-322</td>
</tr>
<tr>
<td>3.3.28</td>
<td>PF00 - Power Fail - Restart</td>
<td>3-326</td>
</tr>
<tr>
<td>3.3.29</td>
<td>PK00 - Pack Character</td>
<td>3-330</td>
</tr>
<tr>
<td>3.3.30</td>
<td>RA00 - Request Action</td>
<td>3-335</td>
</tr>
<tr>
<td>3.3.31</td>
<td>RC00 - Request Condition/What</td>
<td>3-341</td>
</tr>
<tr>
<td>3.3.32</td>
<td>RD00 - Request Date</td>
<td>3-377</td>
</tr>
<tr>
<td>3.3.33</td>
<td>RM00 - Record Match</td>
<td>3-392</td>
</tr>
<tr>
<td>3.3.34</td>
<td>RQ00 - Request</td>
<td>3-406</td>
</tr>
<tr>
<td>3.3.35</td>
<td>RRO0 - Request Record and Type</td>
<td>3-423</td>
</tr>
<tr>
<td>3.3.36</td>
<td>RS00 - Request Social Security Number</td>
<td>3-433</td>
</tr>
<tr>
<td>3.3.37</td>
<td>SEO0 - System Error</td>
<td>3-439</td>
</tr>
<tr>
<td>3.3.38</td>
<td>TA00 - Tabulate and Analyze</td>
<td>3-442</td>
</tr>
<tr>
<td>3.3.39</td>
<td>TC00 - Trace Condition</td>
<td>3-473</td>
</tr>
<tr>
<td>3.3.40</td>
<td>UPO0 - Unpack Character</td>
<td>3-479</td>
</tr>
<tr>
<td>3.3.41</td>
<td>$KLF - TTY Carriage Return, Line Feed</td>
<td>3-484</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0</td>
<td>PROGRAM UTILIZATION</td>
<td>4-1</td>
</tr>
<tr>
<td>4.1</td>
<td>COMPUTER OPERATOR INSTRUCTIONS</td>
<td>4-3</td>
</tr>
<tr>
<td>4.2</td>
<td>OPERATIONAL PROCEDURES</td>
<td>4-8</td>
</tr>
<tr>
<td>4.3</td>
<td>INPUT DESCRIPTION</td>
<td>4-8</td>
</tr>
<tr>
<td>4.4</td>
<td>OUTPUT DESCRIPTION</td>
<td>4-9</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Reports</td>
<td>4-9</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Tapes</td>
<td>4-9</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Messages</td>
<td>4-16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPENDIX A</td>
<td>(TAPE DESCRIPTION)</td>
<td>A-1</td>
</tr>
<tr>
<td>APPENDIX B</td>
<td>(REQUEST TABLE)</td>
<td>B-1</td>
</tr>
<tr>
<td>APPENDIX C</td>
<td>(BOOL BUFFER)</td>
<td>C-1</td>
</tr>
<tr>
<td>APPENDIX D</td>
<td>(OPERAND BUFFER)</td>
<td>D-1</td>
</tr>
<tr>
<td>APPENDIX E</td>
<td>(REQUEST BUFFER)</td>
<td>E-1</td>
</tr>
<tr>
<td>APPENDIX F</td>
<td>(CONDITION TABLE)</td>
<td>F-1</td>
</tr>
<tr>
<td>APPENDIX G</td>
<td>(WHAT TABLE)</td>
<td>G-1</td>
</tr>
<tr>
<td>APPENDIX H</td>
<td>(READING THE EXECUPORT 300 RETRIEVAL SYSTEM)</td>
<td>H-1</td>
</tr>
<tr>
<td>APPENDIX I</td>
<td>(SAMPLE INPUT)</td>
<td>I-1</td>
</tr>
<tr>
<td>APPENDIX J</td>
<td>(SAMPLE OUTPUT)</td>
<td>J-1</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>Varian/MEDATA Storage and Retrieval System</td>
<td>2-7</td>
</tr>
<tr>
<td>2-2</td>
<td>Medical Data Input System</td>
<td>2-8</td>
</tr>
<tr>
<td>2-3</td>
<td>Medical Data Update System</td>
<td>2-9</td>
</tr>
<tr>
<td>2-4</td>
<td>Medical Data Tape Retrieval System (MDTRS)</td>
<td>2-10</td>
</tr>
<tr>
<td>3-1</td>
<td>Buffer Diagram for SS No, Record, Type, and Date Responses</td>
<td></td>
</tr>
<tr>
<td>3-2</td>
<td>Buffer Diagram for Simple Condition Response</td>
<td>3-12</td>
</tr>
<tr>
<td>3-3</td>
<td>Buffer Diagram for Complex Condition Response</td>
<td>3-13</td>
</tr>
<tr>
<td>3-4</td>
<td>MDTRS System Block Diagram</td>
<td>3-14</td>
</tr>
<tr>
<td>3-5</td>
<td>Boolean Trees</td>
<td>3-19</td>
</tr>
<tr>
<td>3-6</td>
<td>Use of Stacks in the Boolean Algorithm</td>
<td>3-24</td>
</tr>
<tr>
<td>3-7</td>
<td>Month Conversion Table (CDTB)</td>
<td>3-25</td>
</tr>
<tr>
<td>4-1</td>
<td>Seven Retrieval Questions</td>
<td>3-47</td>
</tr>
<tr>
<td>4-2</td>
<td>Sample Identification Lists</td>
<td></td>
</tr>
<tr>
<td>4-3</td>
<td>Character String Search</td>
<td>4-17</td>
</tr>
<tr>
<td>4-4</td>
<td>Sample Ranging</td>
<td>4-18</td>
</tr>
<tr>
<td>4-5</td>
<td>Sample Boolean</td>
<td></td>
</tr>
<tr>
<td>4-6</td>
<td>Sample Tabulate</td>
<td>4-19</td>
</tr>
<tr>
<td>4-7</td>
<td>Sample Analyze</td>
<td>4-20</td>
</tr>
<tr>
<td>A-1</td>
<td>Master File Structure</td>
<td></td>
</tr>
<tr>
<td>A-2</td>
<td>Record Structure</td>
<td>4-21</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4-22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4-23</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A-2</td>
</tr>
</tbody>
</table>
1.0 SCOPE

The purpose of this volume is to provide the reader with several levels of documentation for this program module of the NASA Medical Directorate Mini-computer Storage and Retrieval System.

The Biomedical Information System Overview (Section 2) describes some of the reasons for the development of the mini-computer storage and retrieval system. It briefly describes all of the program modules which constitute the system.

The General Specifications Section (Paragraph 3.1) describes the purpose and function of the specific program module documented in this volume.

The Technical Specifications (Paragraph 3.2) is oriented to the programmer. It is a technical discussion of the same processing described in general terms in the previous section, but is a high enough level as not to be redundant with the very detailed analysis described in the Subroutine Section 3.3.

The Subroutine Section (Paragraph 3.3) describes each subroutine in enough detail to permit an in-depth understanding of the routines and facilitate program modifications.

The Program Utilization Section (Paragraph 4.0) may be used as a "Users Guide" and is as non-technical as possible.
To eliminate unnecessary reproduction, the program listings are maintained in a separate document which may be obtained from the VARIAN library of computer programs currently maintained in Building 32 at NASA MSC. In addition, another document is to be developed for the Medical Directorate management. The document will describe the mini-computer system on a higher functional level and will illustrate how the mini-computer storage and retrieval system interfaces with the total directorate data management plan.
2.0 BIOMEDICAL INFORMATION SYSTEM OVERVIEW

For several years, the NASA Medical Directorate has been developing a Medical Information Management System (MEDATA). The System, as implemented at NASA, utilized an off-line IBM 1050 for preparation of data input via paper tape and required card deck runs for retrieval of data from the data files. A new phase of development has been implemented utilizing remote terminals and mini-computers. This section describes the development rationale of the remote terminal and mini-computer approach.

Several features of the old MEDATA system presented serious difficulties to the system user and made the system unresponsive. These deficiencies in the system included:

- Input data prepared on punched paper tape
- No online data input capability with the data base (i.e., no remote data entry)
- Data corrections via punched paper tape
- Preparation of retrieval requests via a punched card system
- 48-hour turnaround time required for retrieval outputs.

To improve the overall responsiveness of the system and eliminate these deficiencies, Philco-Ford developed the Varian/MEDATA Storage and Retrieval System (VMSARS) which utilizes the Varian 620I computer systems in Building 32 or 37.
VMSARS consists of the Medical Data Input System (MDIS), the Medical Data Update System (MDUS), and the Medical Data Tape Retrieval System (MDTRS). The Medical Data Input System (MDIS) is used to input data via a CRT. If updates or reviews are desired, the output of MDIS may be updated on the CRT by the Medical Data Update System (MDUS). The data tape (MDIS or MDUS output) is entered into the CAAD MEDATA system where it is sorted and merged into the appropriate Medical Data File. The updated CAAD MEDATA Medical Data Tape File is sent back to the Varian computer system and used as input for the Medical Data Tape Retrieval System (MDTRS) which processes data retrieval requests from remote terminals. The primary advantages of the VMSARS are as follows:

- Use of state-of-the-art input devices such as CRT's and portable acoustic-coupled teleprocessing terminals. (CRT's operate at 2400 bits per second on the telephone lines.)

- Elimination of paper tapes from the system. (Data storage is on magnetic tape.)

- Online data input from remote input station via the telephone lines.

- Error correction capabilities on CRT. (Limited error checking is performed by the computer.)

- Update capabilities on Varian system before data is entered into the CAAD MEDATA system.
- Ease of creation and updating of background forms.

- Capability of transmitting MEDATA retrievals over telephone lines to CRT or TTY.

- Capability of performing online retrievals.

VMSARS was used for the Flight Crew Health Stabilization Program for three months preceding the launch of Apollo 14 and Apollo 15. A Computer Communications Inc. (CCI) CRT and keyboard and an Execuport Typewriter were installed in the surveillance command post at KSC, and a surveillance master file was created. The VMSARS was used to collect data, perform updates, and retrieve data from the surveillance master file.

The overall system, including the CAAD MEDATA system interface, is illustrated in Figure 2-1. The individual programs are described as follows:

MEDICAL DATA INPUT SYSTEM

The Medical Data Input System (MDIS) is designed to collect data from a CRT input station and store the data on a magnetic tape. The medical questionnaire forms are maintained on magnetic tape and are read into computer memory at run time. The user selects the appropriate form to be displayed on the CRT. As each form is completed, the data is stored on magnetic tape. This tape may be further updated, transmitted to another terminal, or input to the 1108 MEDATA system. Hardcopies of any form are produced at the
user's request. The system requires only one tape unit and operates either via the phone lines or directly online with the computer.
MEDICAL DATA UPDATE SYSTEM

The Medical Data Update System (MDUS) updates any tape created by MDIS. Two tape-drives, a CRT, and printer are required. The old data is read from the MDIS output tape and displayed on the CRT. Changes may be made to the data on the CRT; comments or recommendations may be added to the record, and the new updated record written to the update tape. The updated tape may be either transmitted to another terminal or input to the MEDATA system, or both. Hardcopies of the records are produced if requested by the user. Refer to Figure 2-3 for an illustration of the MDUS components.
MEDICAL DATA TAPE RETRIEVAL SYSTEM (MDTRS)

MDTRS permits the user to make data retrievals from the MEDATA master tapes created by the 1108 MEDATA system. The 1108 system builds the MEDATA master tape from several sources, one of which is the MDIS or MDUS created input tapes. MDTRS outputs preprogrammed retrieval questions from the Varian computer to the requester's CRT or typewriter. The user builds his retrieval request by answering these questions. When the retrieval request is complete, MDTRS searches the MEDATA master tape for the data, formats the selected data for output, and outputs the data to the terminal. The CRT may operate on a private telephone line at 2000 bits per second and the typewriter operates on any commercial telephone line at 300 bits per second. Refer to Figure 2-4 for an illustration of MDTRS components.

2.1 OVERVIEW SUMMARY

In summary, the Varian/MEDATA Storage and Retrieval System provides the user with an online input and retrieval capability previously unavailable. The response time is significantly improved over the old paper tape system. There are still two weak points in the system due to hardware constraints. These are (1) the requirement to update the MEDATA master file on the CAAD 1108 computer system, and (2) the use of tape instead of disk for storage of the data base. A more comprehensive storage and retrieval system is being designed to operate on a Varian 620 mini-computer system utilizing a disk memory storage device and an input/output multiplexing device. The new system will permit immediate update of files, faster response for retrieval requests and multiple terminal users operating simultaneously.
Figure 2-1 Varian/MEDATA Storage and Retrieval System
Figure 2-2 Medical Data Input System
Figure 2-3 Medical Data Update System
Figure 2-4  Medical Data Tape Retrieval System (MDTRS)
This document describes in detail one of the three components of VMSARS, the Medical Data Tape Retrieval System (MDTRS).
3.0 MDTRS SYSTEM

3.1 GENERAL SPECIFICATIONS

3.1.1 Background

After the input system (MDIS) and the update system (MDUS) were implemented in the latter part of 1970 at the Manned Spacecraft Center, it became apparent that a retrieval system on the Varian 6201 computer would be a worthwhile complement to the overall storage and retrieval operations at MSC. At the time, all retrievals were done in a batch processing environment which did not lend itself to the near-real-time requirements of certain individuals in the Medical Directorate. Once they had identified the types of information they required from the data base, these scientists and doctors were not at all satisfied with the twenty-four hour delay that was necessary before they could receive their data. Thus, at the start of 1971, work began on the Medical Data Tape Retrieval System (MDTRS) which was modeled after the system used in the batch processing environment. The principle difference was to be in the use of a computer dedicated to retrieving data from a master file, and operated either locally at the computer or from a remote station over an ordinary telephone line. On July 1, 1970, it was implemented.

The MDTRS permits a user to retrieve specific information from his data base. What information is selected, and the format in which it is output, is determined by the request the user submits via a keyboard connected to the computer. There are a number of different types of outputs the user may specify, as well as a limited amount of statistical information on the data.
3.1.1 Background (Continued)

During the twenty-one day preflight quarantine of Apollo 15, the MDTRS got its first real test in the Flight Crew Health Stabilization Program. In order to minimize the possibility of any prime crew member contacting a disease, all persons coming in contact with the crew during this quarantine period were put under medical surveillance. Personal and family medical information was collected on each individual. This information was available for retrieval via the MDTRS as any need arose. Rapid access to the data base and the statistics the retrieval system provides were the key components of the surveillance program which is planned for use in all remaining Apollo missions.

3.1.2 Functions of the System

In order to understand any further discussion concerning the MDTRS, several terms should first be defined.

- **operator** - the individual who loads the computer with the programs necessary for operation of the MDTRS
- **user** - the individual who wishes to retrieve data, and will operate the local or remote station that controls the MDTRS
- **request** - a set of seven questions and answers completed by the user at his station, which direct the MDTRS in its operation
- **response (request response)** - answers to the seven questions of the request; keyed in by the user at his station
- **retrieval** - all inputs, outputs, and processing necessary to fulfill all phases of a user-input request
3.1.2.1 Input

Input to the MDTRS is of two types:

1. Keyboard manual data entries, and

Keyboard entries are made both by the operator and the user. Operator entries define the computer location and the user input device. User inputs are in the form of request responses. All processing is done on these inputs or the data tape inputs described in detail in Appendix A.

3.1.2.2 Processing

A request specifies three things: (1) identification of the record or records to be dealt with, (2) what portion of that record to be output, and (3) what format to be used for output. Once this request has been input and validated, the data tape is searched for the record identified. The specified output is then begun. Processing continues until all requested data has been output, or until the user aborts the retrieval.

3.1.2.3 Output

Two types of output, magnetic tape or printed pages, are available with the MDTRS. Tape output is used to make a duplicate of portions of the master file. This may be helpful when many retrievals are to be done using the same small portion of the data over and over again. The printed output applies to all or a portion of a record and may be in several formats.
3.2 TECHNICAL SPECIFICATIONS

3.2.1 System Description

The MDTRS can be divided into four functional modules: Initialization, Request, Record Match, and Output. Each module plays a critical role in the successful operation of all succeeding modules.

<table>
<thead>
<tr>
<th>CONTROL</th>
<th>INITIALIATION</th>
<th>I/O Device Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>REQUEST</td>
<td>Retrieval Definition</td>
</tr>
<tr>
<td></td>
<td>RECORD MATCH</td>
<td>Record Selection</td>
</tr>
<tr>
<td></td>
<td>OUTPUT</td>
<td>Data Formatting and Output</td>
</tr>
</tbody>
</table>

3.2.2 Input

There are three types of input data used in the MDTRS: operator initialization data, user request data, and master tape data. The first type is used during Initialization when three questions concerning the computer configuration are answered via the system input device, the teletype. These three answers tell (1) on what computer configuration the system is running, (2) what device is to be used for user input and output, and (3) on what unit (10 or 11) the Master File is mounted.

After Initialization the MDTRS is ready to accept the user's inputs. These inputs define the retrieval to be performed, and make up the data input for
the Request module. There are seven questions that must be answered and they are:

- SS NO:
- RECORD:
- TYPE:
- DATE
- CONDITION:
- ACTION:
- WHAT:

The first five questions identify a specific record from the master; the last question defines what portion of that record is to be output; and the sixth question specifies the output format. There exist strict rules for the input of request data. These rules are stated in Section 4 of this document.

Appendix A details the exact layout of the Master File and the records on the file.

3.2.3 Processing

Initialization

The first thing done after the system is loaded into the computer is the initialization of all I/O handlers based on the computer configuration and the location of the user terminal. This initialization includes inserting the correct device codes into the input/output instructions and inserting any other device-dependent coding necessary for proper interface with all peripheral devices. All routines using the I/O handlers are to assume that the primary input and output device is a teletype. The appropriate handlers will take the necessary action with special characters.
Request

In the Request module, the seven questions of the Retrieval Request are processed separately. In each case, there is a limited amount of error checking. This is done to prevent a retrieval from being rendered useless after several minutes of processing, due to some error in the format of a user's inputs which makes it impossible to determine what he actually wants. Each question is checked for a blank response. In this event there is one, the default response can be assumed for each question.

These default responses are:

- SS NO: ALL
- RECORD: ALL
- TYPE: ALL
- DATE: ALL
- CONDITION: NONE
- ACTION: LIST
- WHAT: ALL

As a Request Response is processed, it is placed in a buffer called the Request Buffer (CPRB). In the Request Table (CPRT), the beginning location of each response is saved and used later by the Record Match and the Output modules.

Record Match

Once all responses have been successfully input, the tape is searched for a record that matches the selection criteria (the first five questions) of the Request. The processing in the Record Match module is relatively simple for
the first four questions. These make up the ID section of the record (see Appendix A). In most cases a straight comparison can be made between the Request Response in the Request Buffer and a fixed area of the Tape Input Buffer. If the two are exact, the match is true. However, the response to CONDITION may have Boolean operators, which allow the user to specify complex conditions. These responses are made up of Headings and Answers from the Body Section of a record. In the Request module, the response to CONDITION was placed in the Request Buffer, and a tree network was created to facilitate easy and rapid determination of the logical conclusion of the Boolean expression in this response. The Headings and Answers make up the base of the tree. In the Record Match module, as each Heading-Answer pair is matched, a flag is set that allows the program to proceed further up the tree from the base. Once the top of the tree is reached, the Boolean expression is true. If the expression is true, the record is said to match the CONDITION response.

Output
A record is selected when the first five questions of the selection criteria of the Request are fulfilled. The sixth question specifies the format of the output. The appropriate routine is called and collects the data specified by the response to the WHAT question. As soon as enough data for one line of output is collected, the Output Message routine (OMOO) is called. This routine selects the necessary output handler, based on the operator input data of the Initialization module. When all the requested data on the record is processed, control is again passed to the Record Match module which searches
for another record to match the selection criteria of the Request.

**End Action**

Three things will terminate the retrieval processing:

1. the SS NO on record exceeds the one specified in the Request;
2. the end of the tape data (end-of-file) is reached; and
3. the user elects to abort the retrieval from his remote station.

Each Output subroutine in the Output module has some specific action that must be performed in the event one of the above events occurs. Once this action is completed, control is transferred to the Request module and the processing begins again.

### 3.2.4 Output

There are three types of output in the MDTRS: user I/O device output, system I/O device output, and user-requested tape output. The three types will fall into one of three categories:

1. user-requested output may go to either tape or the user I/O device;
2. error messages may go to the system or user I/O devices;
3. advisory messages may go to the user I/O device during retrievals, or to the system I/O device during initialization.

The user I/O device may be any one of the following:

1. CLINC Teletype #1
2. CLINC Teletype #2
3. DOC Teletype
4. 103A Modem

The system I/O device is the Teletype in all cases.
Examples of user-requested output may be found in Appendix J. These outputs are in one of four ACTION formats:

(1) LIST
(2) COUNT
(3) COPY
(4) TABULATE/ANALYZE.

3.2.5 Buffers and Tables
The function of the Request module is to accept the user's inputs, save the responses in the Request Buffer (CPRB), and organize a set of buffers and tables that will be used in the remaining two modules of the MDTRS. To completely comprehend the processing done in the last two modules, these buffers and tables, and their interrelationships must be fully understood. Appendices B through G define the layout of each of the specific buffers of concern in the Request module.

Request Buffer and Table
In Figure 3-1, there is a diagram showing the relationship between the Request Table and the Request Buffer. All Request responses are placed in the Request Buffer, with the starting addresses of the first four saved in the Request Table. Should any Request question not have a response, a zero is placed in the position of the Request Table associated with that response. This zero indicates the Default Condition for that question.

Operand Buffer - simple response
For the response to CONDITION, one of two situations may exist: a complex response with Boolean operators or a simple response without Boolean operators.
Figure 3-2 is a buffer diagram for the simple response. In this example, the first location in the Operand Buffer contains the beginning location of the CONDITION response. The second location contains a -1 to indicate that there are no more parameters in the response.

Condition Table

Each simple response may be one of four forms:

1. Heading only - flag = 0;
2. Heading plus alpha Answer - flag = 1;
3. Heading plus numeric Answer - flag = 2;
4. Heading plus range of numeric Answer - flag = 3.

To signify which form each simple response may be, a table (Condition Table) has been created containing a flag to signify the form of the response, and a pointer to locate the answer associated with each simple response. In the sample in Figure 3-2, the flag is two, indicating a Heading with a numeric Answer. The pointer is next in the table and locates the Answer in the Request Buffer.

Bool Buffer and Operand Buffer - complex response

A complex response is a series of simple responses separated by the Boolean operators AND and OR, and possibly grouped using parentheses as required. Figure 3-3 is an example of the buffer arrangement associated with a complex response. The Operand Buffer is now a series of two word sets - the first word of the set contains a pointer to the Heading portion of the simple response in the Request Buffer; the second word contains a pointer to the Boolean operator associated with that simple response. Every simple response of the Operand Buffer is linked to an operator. This operator is located in the Bool
Buffer and may link to other operators in the Bool Buffer depending on the degree of complexity of the CONDITION response.

What Table

The responses to the question WHAT are handled in exactly the same manner as the CONDITION responses, with the exception that the What Table is used in place of the Condition Table.

The WHAT response results in a flag being set in the sixth location of the Request Table. For a list of these flags and their meanings see Appendix B.
FIGURE 3-1 BUFFER DIAGRAM FOR SS NO, RECORD, TYPE, AND DATE RESPONSES
FIGURE 3-2 BUFFER DIAGRAM FOR SIMPLE CONDITION RESPONSE
FIGURE 3-3 BUFFER DIAGRAM FOR COMPLEX CONDITION RESPONSE
3.2.6 System Flow

MDIRS

ISO00
INITIALIZE
SYSTEM

REWIND
MASTER

RO00
INPUT USER
REQUEST & SET
UP BUFFERS &
TABLES

HAS
USER REQUESTED
TO ABORT
RETRIEVAL?

YES
PROCESS
END ACTION

NO
READ
MASTER

3-15
ACTION = COUNT

YES -> COUNT DATA OR RECORDS

NO -> 10

ACTION = TABULATE

YES -> OUTPUT TABULATE REPORT

NO -> TAOO

OUTPUT ANALYZE REPORT

10
3.2.7 Hardware Configuration

Following is a minimum hardware configuration for operation of the MDUS:

1 - Varian 620/i computer with 20K of core memory
2 - tape drives
1 - teletype
3 - Buffer Interlace Controllers (BIC)
1 - Priority Interrupt Module with the following interrupts.
   End of Transmission interrupts on all BIC's
   CRT keyboard interrupt
1 - 103A Modem

The two tape drives should be connected to separate BIC's.

3.2.8 System Block Diagram

See Figure 3-4.
FIGURE 3-4 MDTIR SYSTEM BLOCK DIAGRAM
3.3 SUBROUTINES

3.3.1 BLOO - Setup Boolean Tree

3.3.1.1 Purpose
The purpose of the subroutine is to set up a tree network for CONDITION and WHAT responses, which can be used to facilitate rapid matching of data.

3.3.1.2 Technical Description
A thorough knowledge of both the Operand Buffer and the Bool Buffer is necessary for a complete understanding of the BLOO routine. Appendices C and D detail the layout of these buffers. Before any explanation is made of the processing, a few definitions are necessary.

DEFINITIONS

stack
a stack functions as a last in - first out storage mechanism. Physically the stack constitutes a data table. There is a pointer designating the current end of stack. Whenever an item is taken from the stack, it is the last item that had been inserted. An item may be added to the stack or removed from the stack. To stack an item means to insert the item in the location indicated by the pointer and then increment the stack pointer.

popstack
to pop (or remove) an item from the stack is to decrement the stack pointer and then remove the item indicated by the pointer.
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>link</td>
<td>this is a function whereby two arguments (or parameters) are associated by having the first argument point to the second.</td>
</tr>
<tr>
<td>pointer</td>
<td>a pointer is merely an address of the location of some specific item in memory.</td>
</tr>
<tr>
<td>atom</td>
<td>the smallest element of an operand in a Boolean expression. In the case of the MDTRS, an atom represents one Heading or one Heading-Answer pair.</td>
</tr>
<tr>
<td>simple operand</td>
<td>an atom to the left or right of the operator in a Boolean expression.</td>
</tr>
<tr>
<td>complex operand</td>
<td>a simple Boolean expression made up of an operator and two operands (simple or complex), the expression itself to be used as an operand in another Boolean expression.</td>
</tr>
<tr>
<td>operator</td>
<td>symbol representing a Boolean function of either union or intersection.</td>
</tr>
<tr>
<td>terminal</td>
<td>the last call to BLOO indicating the end of the Boolean expression, is the terminal parameter call.</td>
</tr>
</tbody>
</table>
| parameter    | an element that makes up the Boolean expression. It may be one of the following:  
|              | (1) left parenthesis  
|              | (2) right parenthesis |
(3) atom (simple operand)

(4) operator

(5) terminal.

Node a collection of memory locations containing the data necessary to determine the logical conclusion of a simple Boolean expression and the link to the next level of the expression. See Appendix C.

Conclusion the logical occurrence of an operand or expression, whereby, if X occurs, X union Y is true; or if Y occurs, X union Y is true; both X and Y must occur for X intersection Y to be true.

Basic Processing

There are five types of parameters in a legal Boolean expression in MDTRS (see definition of parameter). BLOO processes the entire expression one parameter at a time. Each type of parameter has its own specific processing which is composed of some combination of the following three functions: inserting an item onto a stack, removing an item from the stack, or linking two items together.

For use with the stack operations two stacks, A and B, are maintained in the MDTRS. Stack A contains items that are to be linked to the next operator in the Boolean expression. These items may be simple operands or operators.
representing complex operands. Stack B contains left parentheses and operators that will be linked to by the next simple operand or complex operand.

ALGORITHM

The algorithm for the processing of the parameters contains five segments, one for each type of parameter. The algorithm will be listed here and will be followed by an example of its use.

1. \[\text{left parenthesis}\]. Stack B (left parenthesis)

2. \[\text{right parenthesis}\]. If Popstack B = left parenthesis, continue. Otherwise Link (Popstack A to Popstack B).

3. \[\text{atom}\]. If Popstack B empty or left parenthesis, Stack A (atom) and Stack B (left parenthesis). Otherwise Link (atom to Popstack B)

4. \[\text{op}\] Link (Popstack A to operator), Stack A (op) and Stack B (op)

5. \[\text{terminal}\]. a) If Popstack B is empty Link (Popstack A to NULL). b) Otherwise, if Popstack B is left parenthesis, Go to a. c) Otherwise Link (Popstack A to Popstack B) and Go to a.
FIGURE 3-5 BOOLEAN TREES
PARAMETER

\[ (A \cup B) \cap (X' \cup A \cap B) \]

\[ X \rightarrow \cap \]

\[ X \rightarrow \cap \rightarrow Y \]

\[ (X \cap Y) \cup Z \]

FIGURE 3-6 USE OF STACKS IN THE BOOLEAN ALGORITHM
Figure 3-6 illustrates four different Boolean trees that can be set up by this algorithm. In Figure 3-6 there is a step by step example of the algorithm. In each portion of the figure you can see the parameter that is being processed, the contents of each stack after the processing, and the new links, if any, between operand and operator. The links are illustrated by arrows, and the direction of the link by the direction of the arrow.

Initially, both Stacks A and B are empty and there are no links between parameters.

a. The left parenthesis is the parameter and it is added to Stack B. Eventually, this left parenthesis will be removed by a right parenthesis. However, it is saved to prevent the linking of any atom within the parenthesis to an operator outside the parenthesis.

b. The next parameter is an atom, X. Stack B is popped. If the last item on B is a left parenthesis, this atom is the atom to the left of the operator. Since the atom to the left is always linked to the operator immediately following, X will be saved in Stack A and will wait for the operator that is to follow. It is important that the left parenthesis that was removed from Stack B be replaced, since we have not yet found the right parenthesis that will remove it permanently.
c. With this operator, ∧, we will create our first link. The last item on Stack A is popped. This item is the atom X which has been waiting for the operator. The first link is set by linking X to the operator ∧. We must now add the operator to both stacks. It is added to B because it is waiting for the next simple or complex, operand which will link to ∧. The operator is added to A because represents a complex operand (X Y), and must be linked to an operator as all operands must.

d. Our next parameter is an atom Y. Stack B is popped and since the operator ∧ has been waiting for the operand Y, Y is linked to ∧. Looking at the links we can see that we have a complete expression X ∧ Y. This expression is used as a complex operand in the expression (X ∧ Y) ∨ Z.

e. The right parenthesis is found now. This means that our simple expression X Y has been completely linked and there is no danger of illegal linking of the parameters that were within the parentheses. In some cases, when the Boolean expression consists of many nested expressions (e.g., ((X ∨ Y) ∨ Z) ∨ (R ∨ (S ∨ T))) the last item on Stack B will not be the left parenthesis that must be removed. In this event, the terminal processing will remove them. However, in our example here the left parenthesis is the last item and is removed.
f. The next parameter is the operator \( \lor \). At any time we have an operator we will also have an operand (simple or complex) on Stack A. Thus, we link Stack A to the operator \( \lor \). Then \( \lor \) is placed on both stacks; on Stack B to wait for the next operand, and on Stack A because it represents a complex operand (in this case the entire expression) that must be linked to an operator.

g. The last operand is now received. It is immediately linked to the operator on Stack B.

h. There are no more parameters but we have some cleaning up to do in our stacks. This is done by examining Stack B. If B is empty we must link the item on Stack A to NULL (\( \times \)). This null link indicates the top of the Boolean tree. If B had not been empty the last item on A would have been linked to the last item on B and the next item on B would have been examined as before.
### 3.3.1.2.1 Calling Sequence

```call bloo```

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Parameter to be processed:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = left parenthesis</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 = right parenthesis</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 = operator</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 = atom (operand)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 = terminal</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>link location in operand</td>
<td></td>
</tr>
<tr>
<td></td>
<td>buffer if A-reg = 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>type of operator 0 = OR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = AND</td>
<td></td>
</tr>
<tr>
<td></td>
<td>if A-reg = 3</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

3-29
CALL SA00

REGISTER

<table>
<thead>
<tr>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>item to be added to Stack A</td>
<td>N/A</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
</tr>
</tbody>
</table>

CALL SB00

REGISTER

<table>
<thead>
<tr>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>item to be added to Stack B</td>
<td>N/A</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
</tr>
</tbody>
</table>

CALL PA00

REGISTER

<table>
<thead>
<tr>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
</tr>
</tbody>
</table>

item removed from Stack A

N/A
CALL PBOO

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

CALL LK00, LKA, LKB

PARAMETER

<table>
<thead>
<tr>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>LKA</td>
</tr>
<tr>
<td>LKB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
3.3.1.2.2 General Flow Chart

- **BLOO**
  - **PARAMETER = "("**
    - **YES** PROCESS LEFT PARENTHESIS
    - **NO**
      - **PARAMETER = ")"**
        - **YES** PROCESS RIGHT PARENTHESIS
        - **NO**
          - **PARAMETER = ATOM**
            - **YES** PROCESS ATOM
            - **NO**
              - **PARAMETER = OPERATOR**
                - **YES** PROCESS OPERATOR
                - **NO**
                  - PROCESS TERMINAL

- RETURN

3-32
3.3.1.3 Label Description

3.3.1.3.1 Local
BLA First argument in call to LK00
BLB Second argument in call to LK00
BLSA Stack A Buffer
BLSV Save location
BLXA Stack A Pointer
BLXB Stack B Pointer
BLSB Stack B Buffer
LKA First argument in call
LKB Second argument in call

3.3.1.3.2 Global
   None

3.3.1.3.3 Entry Points
BL00
LK00
PA00
PB00
SA00
SB00

3-33
3.3.1.3.4 External References

CPXB  Bool Buffer pointer
$SE  routine to pick up a calling sequence
A

3-35

SBOO
STACK B
-1

RETURN

B

BL10

PBOO
POPSTACK B

B = -1

YES

NO

PAOO
POPSTACK A

LKOO
LINK A TO B

RETURN

3-36
SAVE OPERATOR IN BITS 3-5 OF NEXT AVAILABLE WORD IN BOOL BUFFER (CPBB)

SET B = BOOL BFR PTR (CPXB)

INCREMENT BOOL BUFFER PTR BY 2

SAVE THE OPERATOR

POPSTACK A

D
3-40
LK00

$SE
PICK UP CALLING SEQ.

STORE THE SECOND ARGUMENT IN A LOCATION INDICATED BY THE FIRST ARGUMENT

RETURN
STORE CONTENTS OF THE A-REG. IN NEXT AVAILABLE LOCATION IN STACK B

INCREMENT STACK B POINTER (BLXB)

RETURN
STORE CONTENTS OF A-REGISTER IN NEXT AVAILABLE LOCATION IN STACK A

INCREMENT STACK A POINTER (BLXA)

RETURN
YES

STACK B
POINTER = BEGINNING
OF STACK B

NO

DECREMENT
STACK B
POINTER (BLXB)

PICK UP LAST
AVAILABLE VALUE
IN STACK B

RETURN
-PAOO

IS STACK A POINTER = BEGINNING OF STACK A

YES

NO

DECREMENT STACK A POINTER (BLXA)

PICK UP LAST AVAILABLE VALUE IN STACK A

RETURN
3.3.2 CDOO - Convert Date

3.3.2.1 Purpose

CDOO is a subroutine whose purpose is to take a month which is stored in the Tape Input Buffer in integer form and convert that number to a corresponding three character alpha equivalent and store it into a buffer for later processing.

3.3.2.2 Technical Description

CDOO picks up the integer number corresponding to the month from the Tape Input Buffer. Using the integer as an index, the corresponding three alpha character entry in the Month Conversion Table CDTB is accessed and output, a character at a time, by calling the pack routine PK01 three times in succession. If the value of the integer is greater than 15, an entry of 'UNK' is assigned. The Month Conversion Table CDTB is given in Figure 3-7.

3.3.2.2.1 Calling Sequence

CALL CDOO

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>Contents unpredictable</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>Contents unpredictable</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
MONTH CONVERSION TABLE (CDTB)

<table>
<thead>
<tr>
<th>Integer</th>
<th>Alpha Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>UNK (unknown)</td>
</tr>
<tr>
<td>1</td>
<td>JAN</td>
</tr>
<tr>
<td>2</td>
<td>FEB</td>
</tr>
<tr>
<td>3</td>
<td>MAR</td>
</tr>
<tr>
<td>4</td>
<td>APR</td>
</tr>
<tr>
<td>5</td>
<td>MAY</td>
</tr>
<tr>
<td>6</td>
<td>JUN</td>
</tr>
<tr>
<td>7</td>
<td>JUL</td>
</tr>
<tr>
<td>8</td>
<td>AUG</td>
</tr>
<tr>
<td>9</td>
<td>SEP</td>
</tr>
<tr>
<td>10</td>
<td>UNK</td>
</tr>
<tr>
<td>11</td>
<td>UNK</td>
</tr>
<tr>
<td>12</td>
<td>UNK</td>
</tr>
<tr>
<td>13</td>
<td>OCT</td>
</tr>
<tr>
<td>14</td>
<td>NOV</td>
</tr>
<tr>
<td>15</td>
<td>DEC</td>
</tr>
<tr>
<td>&gt;15</td>
<td>UNK</td>
</tr>
</tbody>
</table>

Figure 3-7
3.3.2.2.2 General Flow Chart

1. LOAD AND STORE FIRST ALPHA MONTH CHARACTER
2. LOAD AND STORE SECOND ALPHA MONTH CHARACTER
3. LOAD AND STORE THIRD ALPHA MONTH CHARACTER

EXIT
3.3.2.3 Label Description

3.3.2.3.1 Local

CDTB - A 48 word table containing the alpha designation for each month of the year. In addition, space has been allocated for those additional entries which are to be designated as 'UNK' (i.e., unknown).

3.3.2.3.2 Global

None

3.3.2.3.3 Entry Point

CDOO - primary entry point

3.3.2.3.4 External References

PK01 - pack character routine
3.3.2.4 Detailed Flow Chart

CI00

LOAD BINARY MONTH NUMBER

MONTH = ZERO?

MONTH >15?

YES

LOAD TABLE ADDRESS OF 'UNKNOWN'

NO

POSITION TO MONTH IN DATE TABLE

CD 05
LOAD FIRST ALPHA CHARACTER OF MONTH

PK01
STORE IT

LOAD SECOND ALPHA Character

PK01
STORE IT

LOAD THIRD ALPHA CHARACTER

PK01
STORE IT

EXIT
3.3.3 CM00 - Compare

3.3.3.1 Purpose
CM00 is a subroutine whose purpose is to compare two strings of alphanumeric characters.

3.3.3.2 Technical Description
Starting locations of the two strings of alphanumeric characters to be compared are passed to the routine via the A and B registers. CM00 then does a word by word compare on the two character strings. If the two strings are identical, control is returned to the calling program with a zero in the A-register; otherwise a non zero value is stored in the A-register. The two alphanumeric character strings are assumed to be the same length.

3.3.3.2.1 Calling Sequence
CALL CM00

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>beginning location of first data value</td>
<td>zero if values equal, otherwise, unpredictable</td>
</tr>
<tr>
<td>B</td>
<td>beginning location of second data value</td>
<td>unpredictable</td>
</tr>
<tr>
<td>X</td>
<td>Number of characters to compare</td>
<td>unpredictable</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
3.3.3.2.2 General Flow Chart

CMOO

SAVE A-VALUES
BEGINNING LOCATION (A-REG)

SAVE B-VALUES
BEGINNING LOCATION (B-REG)

NO.
OF WORDS IN STRING = 0

A

SUBTRACT A-
VALUE FROM B-VALUE

RESULT = 0

YES

A

RETURN

RETURN
3.3.3.3 Label Description

3.3.3.3.1 Local
AVAL  storage location containing the beginning address of one of the values to be compared

BVAL  storage location containing the beginning address of the second value to be compared

3.3.3.3.2 Global
N/A

3.3.3.3.3 Entry Points
CM00 - primary entry point

3.3.3.3.4 External References
N/A
3.3.3.4 Detailed Flow Chart

CMOO

SAVE A-VALUES BEGINNING LOCATION (A-REG)

SAVE B-VALUES BEGINNING LOCATION (B-REG)

NO. OF WORDS IN STRING = 0

YES

RETURN

SUBTRACT A-VALUE FROM B-VALUE

RESULT = 0

YES

DECREMENT INDEX

RETURN

NO

RETURN

3-55
3.3.4 CPOO - Control Program

3.3.4.1 Purpose

The purpose of CPOO is to control the execution of all processing and input/output routines. In this manner, CPOO provides a modular system which simplifies development, checkout, and maintenance of the software.

3.3.4.2 Technical Description

CPOO initializes the system by calling the Initialize System Routine IS00, rewinding the input tape, and clearing the request buffers. The Request User Input (RQ00) routine is executed to accept the retrieval request from the user. The data tape is read by the Magnetic Tape Handler (HMO0) and the Record Match Routine (RMO0) is executed. If a record match occurs, CPOO calls the appropriate format routine LIST, COPY, COUNT, TABULATE OR ANALYZE. When a tape end of file or a request-greater-than-master-record condition occurs, the appropriate wrap-up routine for the requested output format is executed and the "OUTPUT COMPLETE" message is displayed. The next request is then accepted by the RQ00 routine again. If no matches are found, the entire file will be searched (except when "GREATER THAN" condition on control data occurs) and the only output to the requestor terminal will be the "OUTPUT COMPLETE" message.

If the user aborts the retrieval in progress, the messages "RETRIEVAL ABORTED" and "OUTPUT TERMINATED" are displayed after the appropriate wrap-up routine has been executed.

3.3.4.2.1 Calling Sequence

None
3.3.4.3 Label Description

3.3.4.3.1 Local

CPAF  Abort Flag

CPBK  Backspace-constant value 6

CPCU  Maximum number of words to be read from tape; value 2880

CPEA  End function; Indicator for What Format Wrap-up Routine to call.

CPFU  Function Table; Indicator for What Format Routine to call. See Table 2 in 3.3.4.2 for details.

CPMC  Table containing addresses of processing to be done for each of the three record match conditions, low, equal, or greater.

CPRD  Tape read function code of 0

CPRW  Tape function code of 7

CPT E  Tape read error count

CPTP  Table containing addresses of processing to be done for each of the possible tape status codes returned by the tape handler HM00.

3.3.4.3.2 Global

CPBB  Bool buffer; 40 words in length; See Appendix F for details; Referenced by RQ00, RCO0

CPNC  Number of characters in retrieval question. Referenced by RQ00.

CPOB  Operator buffer; 40 words in length; See Appendix D for details referenced by RQ00, RCO0

CPRB  Request buffer; 480 words in length; See Appendix E for details; Referenced by RQ00, RS00

3-58
3.3.4.3.2 Global (Continued)

CPRT  Request Table; See Appendix B for details; Referenced by RQ00, RS00, RR00, and RDD0, RC00, RA00, RM00, MD00, MC00, MW00, CT00, TA00, LI00

CPTB  Tape buffer, 2880 words in length; See Appendix A for layout, referenced by RA00, MA00, LI00, CY00, CT00, TA00.

CPSW  Status Word for user I/O device. Number of characters input or output referenced by RQ00, RS00, RR00, RA00 and CY00

CPTS  Tape status set to negative value when abnormal condition arises, set to positive value is the number of words written, referred by CY00

CPWT  What Buffer; 20 words in length; See Appendix G for detail, referenced by RQ00, RC00

CPXB  Bool buffer pointer. Referenced by RQ00, RC00

CPXC  Condition buffer pointer, referenced by RQ00, RC00

CPXO  Operand buffer pointer, referenced by RQ00, RC00

CPXR  Request buffer pointer, referenced by RQ00, RS00, RR00, RD00

3.3.4.3.3 Entry Point

N/A

3.3.4.3.4 External References

CT00  Subroutine to COUNT

CT10  Subroutine ENTRY POINT to close COUNT subroutine

CY00  Subroutine to Copy Data Tape

CY20  Subroutine ENTRY POINT to Close Copy subroutine

CY00  Subroutine to Copy Data Tape
3.7.4.3.4 External References (Continued)

- **GY20** Subroutine ENTRY POINT to Close Copy subroutine
- **ER00** Subroutine to Output Message to User
- **FILL** Subroutine to Clear Buffer
- **HM00** Subroutine to handle the input tape
- **IS00** Subroutine to initialize the program
- **LI00** Subroutine to output LIST
- **LI$7** Subroutine to close ENTRY POINT LIST subroutine
- **OMHC** Defined in OMHC; Header switch, clear to zero; CPO0 at end of the retrieval
- **RM00** Subroutine to match record buffer with request buffer
- **RQ00** Subroutine to accept request data from the user
- **TAA0** Subroutine ENTRY POINT to close TABULATE subroutine
- **TAA1** Subroutine ENTRY POINT to close ANALYZE subroutine
- **TA00** Subroutine to tabulate and analyze
3.3.4.4 Detailed Flow Chart

CALL IS00
INITIALIZE SYSTEM

CALL HM00
REWIND INPUT TAPE

CALL FILL
ZERO CONTROL BUFFER

CALL RQ00
REQUEST USER INPUT

CALL EROO
OUTPUT RETRIEVAL IN PROGRESS

CALL CP0A
CHECK ABORT

ABORT

LOAD CPTE WITH -5
NO. OF TIMES TO REREAD TAPE

CALL HM00
READ TAPE

CP00
PAGE 1
The diagram shows a flowchart with nodes connected by arrows. The nodes are labeled with conditions or actions, such as 'CP06', 'EOF', 'BUSY', 'EOF', 'RENDER', 'SOFTWARE', 'EOF', 'CALL RMOO RECORD MATCH', 'RECORD REQUEST', and 'CP08'. The flowchart starts with 'CP04' and 'CPTS' at the top, with branches leading to various states or actions, including 'CP06', 'CP02', 'CP20', 'CP23', and others. The diagram uses standard flowchart symbols for decision points and process actions.
CPCA

SENSE WRITE BUFFER =

LOAD REGX 7

CLEAR AND INPUT CHAR

SET PARITY

SUBTRACT 0300 FROM CHAR

RETURN
3.3.5 CTOO Count

3.3.5.1 Purpose

CT00 is a subroutine whose purpose is to count the number of records whenever "COUNT" is specified as the answer to the "ACTION" query. The subroutine will only count those records which meet the criteria specified in the response to the "WHAT" query.

3.3.5.2 Technical Description

The subroutine checks the What entry in the Request Table. If the entry is zero, the default value of "count all records" is assumed. If the entry is non-zero, only those records that meet the 'What' response criteria are counted. To determine if the record meets the 'What' response criteria, the subroutine Match What (MWOO) is called to check all question/answer pairs in tape input buffer.

If the entry meets the specified criteria, the counter (CTCT) is incremented. When an End of File is encountered by the Control Program, CPO0, control is given to the entry point CT10. The count is converted from binary to ASCII, displayed on the output device and the counter (CTCT) cleared to zero. Control is then returned to CPO0.
3.3.5.2.1 Calling Sequence

CALL CT00

CALL CT10

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>Modified</td>
</tr>
</tbody>
</table>
3.3.5.3 Label Description

3.3.5.3.1 Local

CTMS  Message Display 'Count Is'
CTNO  Four Work Areas used to convert the count to ASCII
CTOC  Display Count
CTQN  Question number to be checked by Match What
CTST  Status Word

3.3.5.3.2 Global

CTCT  Binary Count used also by CY00

3.3.5.3.3 Entry Points

CTSO  Count the Records
CTT0  Output the Count

3.3.5.3.4 External References

CPRT  Defined in Appendix B, Request Table
CPTB  Defined in Appendix A, Tape Buffer. Second word in buffer contains number of questions in buffer
MW00  Subroutine Match What
OM00  Subroutine to output count to User Terminate
PK00  Subroutine to Initialization Pack Output Count
PK01  Subroutine to Pack Count
CT10

LOAD INDEX WITH 4

LOAD REGB WITH CTCT

DIVIDE CTCT BY 10 AND ADD 60 OCT TO MAKE ASCII

SUBTRACT 1 FROM INDEX

STORE ASCII CTNO + INDEX

INDEX ≠ ZERO

ZERO CTCT

PKOO PACK SPACE BUFFER

LOAD LOOP CONTROL WITH 4

ZERO INDEX

CT 13
CT13

- Pick up char from CTNO + INDEX

PK01 PACK CHAR IN BUFFER

- Subtract 1 from loop control

- Add 1 to index

- Loop control 0

- PK01 PACK SPACE BUFFER

- Call OMOO DISPLAY COUNT

- Busy

- Error

- Return
3.3.6 CYO0 - COPY Data

3.3.6.1 Purpose
Copy is a subroutine whose purpose is to copy data from the input file whenever "COPY" is specified as the response to "ACTION" query.

3.3.6.2 Technical Description
The Copy Subroutine initializes the output unit number based on the current assignment of the input unit number. A message to the computer operator requesting that a tape be mounted is generated. The computer operator responds with a 'G' when the tape is mounted. The tape mount message is repeated until the operator responds with a 'G'; subject to a maximum repetition of fifty times in which case a message, 'NO RESPONSE FROM COMPUTER OPERATOR, REQUEST CANCELLED', is displayed at the user terminal.

After the computer operator responds, the first record is written and the counter (CTCT) is set to one. Thereafter each time CYO0 is entered, the record just read by the control program (CP00) is written to the output tape and the counter (CTCT) incremented.

When an End of File is encountered by CP00, control is given to CY20. CY20 writes a software End of File and hardware End of File. The output count CTCT is displayed to the user terminal and the computer operator is requested to dismount the tape. Control is returned to the control program.

3-73
### 3.3.6.2.1 Calling Sequence

**CALL CY00**

**CALL CY20**

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>Modified</td>
</tr>
</tbody>
</table>
3.3.6.2.2 General Flow Chart

CYOO

1ST TIME ENTER

WAIT ON COMPUTER OPERATOR RESPONSE

WRITE RECORD

ADD 1 TO RECORD COUNT

RETURN

NO

YES
3.3.6.3.2 Global

None

3.3.6.3.3 Entry Points

CY00      COPY TAPE
CY20      WRITE END OF FILE

3.3.6.3.4 External References

CPCA      Subroutine to check for abort
CPTB      Defined in Appendix A Tape Buffer
CPTS      Defined in CPOO; number of characters to be written
CP20      Transfer control when abort
CTCT      Defined in CTOO, counter
CT10      Subroutine to output count to user
HM00      Subroutine to write to tape
IS90      Subroutine to output message to computer operator
OM00      Subroutine to output message to user
ADD 1 TO CYC1

CYC1 = 50

CALL OMOO
DISPLAY NO RESPONSE FROM COMPUTER

BUSY

ERROR

SET CYSW TO NON ZERO

RETURN

ZERO CYSW

CY06
CALL HMOO
WRITE RECORD

BUSY
≠
ADD 1 TO CTCT
RETURN

CALL HMOO
WRITE EOF
BUSY
≠
CALL CTOO
OUTPUT COUNT
SET CYSW TO NON-ZERO

CY26

CY00
PAGE 3
CALL HM00
REWIND TAPE

BUSY

CALL IS90
DISPLAY DISMOUNT

RETURN

CY50

READ BUFFER FULL

= CY52

SUBTRACT 1
FROM REGX

OVERFLOW

RETURN

CY00
PAGE 4
CLEAR AND INPUT TO REGA

SENSE WRITE

ECHO CHARACTER

RETURN +2
3.3.7.1 Purpose

ER0O is a subroutine whose purpose is to output error or advisory messages to the user at the terminal.

3.3.7.2 Technical Description

Whenever an error occurs or status information is to be output, subroutine ER0O is employed to select the proper message from table ERTB and output it. Register X is used as an index into ERTB to indicate the message desired to be output. The address of the start of the message is retrieved and stored in the call to OM00, which outputs the message.

3.3.7.2.1 Calling Sequence

CALL ER0O

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>Same</td>
</tr>
<tr>
<td>X</td>
<td>Number of Message to be displayed</td>
<td>Modified</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>Same</td>
</tr>
</tbody>
</table>
3.3.7.2.2 General Flow Chart

EROO

PICKUP ADDRESS
OF MESSAGE TO
BE DISPLAYED

DISPLAY
MESSAGE

RETURN
3.3.7.3 Label Description

3.3.7.3.1 Local

ERCT contains 24 for number of characters to be output.

ERTB is the table of address for messages. The table has 20 entries.

Each entry specifies the address of the first character of the message.

ERROR 'INVALID SS NO'

ERR1 'INVALID RECORD'

ERR2 'INVALID TYPE'

ERR3 'INVALID DATA'

ERR4 'INVALID CONDITION'

ERR5 'INVALID ACTION'

ERR6 'INVALID WHAT'

ERR7 'CANNOT ANALYZE PROSE'

ERR8 'ILLEGAL PARENTHESES'

ERR9 'RECORD BAD-SKIPPED'

ERR10 'NO ANSWER WITH COLON'

ERR11 'TOO MANY PARAMETERS'

ERR12 'INVALID WHAT'

ERR13 'POWER FAILURE, PLEASE'

ERR14 'RESTART REQUEST*****'

ERR15 'RETRIEVAL WORKING'

ERR16 'RETRIEVAL ABORTED'

ERR17 'OUTPUT TERMINATED'

3.3.7.3.2 Global

None
3.3.7.3.3 Entry Point

ER00

3.3.7.3.4 External References

CPOC is output message in CPOO 'OUTPUT COMPLETE'

CPET is output message in CPOO 'TAPE ERROR RECORD SKIP'

OM00 call to output the message.
3.3.8 FA00 - FLOATING POINT - ASCII Conversion

3.3.8.1 Purpose
The purpose of the subroutine FA00 is to translate a floating point number from its internal representation to alpha codes into a buffer for subsequent printout.

3.3.8.2 Technical Description
The overall approach used to encode the floating point number utilizes the Varian-provided subroutine ($HS) which converts floating point numbers to binary integers. $HS is obviously subject to the limitation of 32,767; i.e., the maximum integer that may be represented in the 620i. The maximum number of characters permitted to be encoded excluding the decimal point is 7.

The integer part and fractional part of the floating point number are encoded separately. The integer part of the number to be encoded is compared with the upper limit of 32,767; should it exceed the maximum, it is divided by 1,000, resulting in a quotient equal to the n significant digits greater than 1,000. The resulting floating point integer is converted to binary integer via $HS. The resulting binary integer is then encoded to a string of ASCII characters. The integer part of the number which is less than 1,000 is then converted to a binary integer via $HS and subsequently converted to its ASCII equivalent.
If a fractional part exists, the decimal point is inserted and the fractional part is multiplied by 10,000 to make it an integer. The conversion process used for the integer part is then utilized for the fractional part also. A second multiplication by 10,000 of the remaining portion of the fractional number may be necessary to acquire the remaining digits.

### 3.3.8.2.1 Calling Sequence

**CALL** FA00

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Most signification part of number to be converted.</td>
<td>Modified</td>
</tr>
<tr>
<td>B</td>
<td>Least significant part of number to be converted.</td>
<td>Modified</td>
</tr>
<tr>
<td>X</td>
<td>Address when to store result.</td>
<td>Modified</td>
</tr>
</tbody>
</table>

Overflow: N/A  Modified
3.3.8.2.2 General Flow Chart

FA00

NO. POSITIVE

MAKE THE NUMBER POSITIVE

STORE MINUS SIGN IN BUFFER

NO. GREATER THAN 32767

DIVIDE NO. BY 1000 RESULT

CONVERT RESULT TO INTEGER

CONVERT INTEGER TO ASCII

STORE ASCII IN BUFFER

FA00
1
CONVERT INTEGER TO FLOATING

MULTIPLY BY 1,000

SUBTRACT FROM ORG NO.

2

NO. LESS THAN 1

YES 3

NO

CONVERT INTEGER

CONVERT INTEGER TO ASCII

STORE IN BUFFER

END DATA

YES RETURN

FA00
3

STORE DEC POINT IN BUFFER

END OF DATA

YES

RETURN

NO

CONVERT INTEGER TO FLOATING

SUBTRACT FROM NO.

MULTIPLY BY 10,000

CONVERT TO INTEGER

CONVERT TO ASCII

STORE IN BUFFER

3-92
3.3.8.3 Label Description

3.3.8.3.1 Local

<table>
<thead>
<tr>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FABF</td>
<td>Binary working storage used to convert integer</td>
</tr>
<tr>
<td>FACT</td>
<td>Counter</td>
</tr>
<tr>
<td>FADG</td>
<td>Digit counter</td>
</tr>
<tr>
<td>FADX</td>
<td>Address of buffer to store result</td>
</tr>
<tr>
<td>FAH1</td>
<td>Working storage for high order floating point number</td>
</tr>
<tr>
<td>FAH2</td>
<td>Working storage for low order floating point number</td>
</tr>
<tr>
<td>FAMX</td>
<td>Floating Point number 32,767.0</td>
</tr>
<tr>
<td>FAON</td>
<td>Floating Point number 1.0</td>
</tr>
<tr>
<td>FARA</td>
<td>Most significant part of number to be converted</td>
</tr>
<tr>
<td>FARB</td>
<td>Least significant part of number to be converted</td>
</tr>
<tr>
<td>FARD</td>
<td>Floating Point number 0.5 x 10^-7</td>
</tr>
<tr>
<td>FATH</td>
<td>Floating Point number 1000.0</td>
</tr>
<tr>
<td>FATN</td>
<td>Floating Point number 10,000.0</td>
</tr>
<tr>
<td>FATP</td>
<td>5 word working storage used to convert to ASCII code</td>
</tr>
<tr>
<td>FASW</td>
<td>Switch used to tell if a leading zero is present</td>
</tr>
</tbody>
</table>

3.3.8.3.2 Global

None

3.3.8.3.3 Entry Points

FA00
3.3.8.3.4 External References

PK00  Subroutine entry point to pack first character in output buffer.
PK01  Subroutine entry point to pack character in output buffer.
SIGN  Subroutine to copy sign.
$HS   Subroutine to convert floating point number to binary integer.
$QL   Subroutine to perform floating point subtraction.
$QM   Subroutine to perform floating point multiply.
$QN   Subroutine to perform floating point divide.
$QS   Subroutine to convert binary integer to floating point number.
3.3.8.4 Detailed Flow Chart

FA00

SAVE FLOATING NO. IN FAR A, FAR B

SAVE BUFFER ADDRESS FABX

TURN OFF FAS W WITH NON ZERO

NO. POSITIVE = FA02

CALL SIGN MAKE NO. POSITIVE

STORE POSITIVE NO. IN FAR A

PK00 PUT MINUS SIGN IN BUFFER

SET FADG TO 6 DIGIT COUNT

FA00

PAGE 1
FA01

SET FADG TO 7 DIGIT COUNT

PK00
PUT SPACE 1ST CHAR OF BUFFER

FA02

FA04

FARA 32767 ≤

CALL $QN DIVIDE BY 1000

CALL $HS CONVERT TO INTEGER

LOAD REGB WITH INTEGER

LOAD REGX WITH 5

CALL FA90 CONVERT INTEGER TO ASCII

LOAD REGX WITH 1

SET FACT TO 4

1
CALL FA92
STORE VALUE IN BUFFER

CALL $QS
CONVERT INTEGER TO FLOATING

CALL $QM
MULTIPLY FLOATING INTEGER BY 1000

CALL $QL
SUBTRACT FLOATING INTEGER FROM FARA, FARH

STORE RESULT IN FARA, FARH

STORE 3 IN FA05+1 NO. OF CHAR TO MOVE

STORE 2 IN FA05+1 WHN TO START

FA00
PAGE 3

3-97
CALL $HS CONV\text{ERT FLOAT-ING TO INTEGER

LOAD \text{REGX WITH 5

CALL \text{FA90 CONVERT INTEGER TO ASCII

LOAD FACT WITH NO. OF CHAR-\text{TO MOVE

LOAD \text{REGX WITH WHEN TO START

CALL \text{FA92 PACK IN BUFFER

ZERO \text{FABF

\text{FA07
3.3.9 FILL - Fill Buffer

3.3.9.1 Purpose

FILL is a subroutine whose purpose is to store the contents of the A-register into a number of consecutive core locations.

3.3.9.2 Technical Description

FILL stores the contents of the A-register beginning at a location whose address is specified in the B-register, into a number of consecutive locations indicated by the count contained in the X-register. A practical use of this subroutine would be in the "blank" filling of a buffer.

3.3.9.2.1 Calling Sequence

CALL FILL

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>word to be stored</td>
<td>same</td>
</tr>
<tr>
<td>B</td>
<td>beginning location of destination buffer</td>
<td>address +1 of where the last word was stored</td>
</tr>
<tr>
<td>X</td>
<td>number of locations to fill</td>
<td>zero (0)</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
3.3.9.2.2 General Flow Chart

- **FILL**
- **STORE WORD**
- **DECREMENT WORD INDEX**
- **INCREMENT LOCATION COUNTER**
- **ALL WORDS STORED?**
  - **NO**
  - **YES**
- **EXIT**
3.3.9.3 Label Description

3.3.9.3.1 Local

N/A

3.3.9.3.2 Global

N/A

3.3.9.3.3 Entry Points

FILL - primary entry point

3.3.9.3.4 External References

N/A
3.3.9.4 Detailed Flow Chart

- Fill
- Store Word
- Decrement Word Index
- Increment Location Counter
- All Words Stored?
- Yes: Exit
- No: Repeat
3.3.10 FLOO - ASCII to Floating Point

3.3.10.1 Purpose

The purpose of the subroutine FLOO is to convert a decimal number stored in a pseudo floating point format into its Varian Internal Floating Point representation.

3.3.10.2 Technical Description

The decimal number to be converted is stored in a pseudo-floating point format which uses the following four parameters to define a number: sign, total number of digits, number of digits to left of the decimal point and the address of the string of ASCII numeric characters.

The overall technical approach used in the conversion involves the repetitive multiplication and partial summing by the positional powers of 10, i.e., starting with the most significant decimal digit, a partial sum is accumulated by taking each digit, adding it to the partial sum and multiplying by 10. However, since the word size on the 620i limits us to 15 bits of accuracy, and since the number of digits input may be considerably larger, overflow may possibly result from both the multiplication and addition used in the conversion process. This overflow problem is handled in the following manner: After each multiplication or addition in the conversion process, a test for overflow is made. Should overflow occur, the overflow register indicates the multiples of $2^{15}$, that is 32,768, that should be added into the partial sum. Therefore, a separate
floating point partial sum containing only the overflow multiples of 32,768 is maintained. This procedure is followed until all of the input characters have been processed. The remainder of the number which is not a multiple of 32,768, and thus not counted in the overflow partial sum, must then be converted to floating point and added to the overall floating point sum.

Should the input number to be converted contain a fractional part, that is, digits to the right of the decimal, the partial sum is adjusted by a division by the appropriate power of 10. The sign of the input number is then affixed.
3.3.10.2.1 Calling Sequence

CALL FLOO, A, B, C, D

A    Address of Sign Word
B    Address of Pseudo Exponent in Binary
C    Address of total count of number of characters to be converted
D    Address of number to be converted

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents Upon Entry</th>
<th>Contents Upon Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>High order of result</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>Low order of result</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>Modified</td>
</tr>
</tbody>
</table>
3.3.10.2.2 General Flow Chart

FLOW
ZERO BINARY ACCUMULATOR
ZERO FLOATING ACCUMULATOR

NO. OF CHARACTER ZERO

MULTIPLY BINARY ACCUMULATOR BY 10

HAS THERE BEEN PREVIOUS OVERFLOW

MULTIPLY FLOATING ACC. BY 10

FLOO
DID BINARY MULTI. OVERFLOW

SET PREV. OVERFLOW TO YES

SET OVERFLOW CNT TO OVERFLOW

ADD 32,768 TO FLOATING ACCUMULATOR

SUBTRACT 1 FROM OVERFLOW COUNT

OVERFLOW CNT ZERO

GET A CHARACTER FROM BUFFER

ADD CHARACTER TO BINARY ACCUMULATOR

OVERFLOW

ADD 32,768 TO FLOATING ACCUMULATOR

ALL CHARACTERS PROCESSED

3-110
CONVERT BINARY ACCUMULATOR TO FLOATING

ADD CONVERT BINARY FLOATING TO FLOATING ACCUMULATOR

NO.
OF CHARACTER
EQUAL EXPEONENT

YES

NO

FLOATING DIFFERS

MULTIPLY FLOATING ACCUMULATOR BY 10

MAKE FLOATING SIGN

3-111
### 3.3.10.3 Label Description

#### 3.3.10.3.1 Local

<table>
<thead>
<tr>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLCH</td>
<td>Binary character to be converted</td>
</tr>
<tr>
<td>FLMA</td>
<td>Floating point 32,768.0</td>
</tr>
<tr>
<td>FLM3</td>
<td>Working storage for floating point number</td>
</tr>
<tr>
<td>FLSW</td>
<td>Overflow switch</td>
</tr>
<tr>
<td>FLS1</td>
<td>Binary accumulator</td>
</tr>
<tr>
<td>FLS2</td>
<td>Floating Accumulator</td>
</tr>
<tr>
<td>PLTC</td>
<td>Address of count</td>
</tr>
<tr>
<td>FLTE</td>
<td>Address of pseudo exponent</td>
</tr>
<tr>
<td>FLTM</td>
<td>Address of first character of number to be converted</td>
</tr>
<tr>
<td>FLTN</td>
<td>Floating point 10.0</td>
</tr>
<tr>
<td>FLTS</td>
<td>Address of sign word</td>
</tr>
<tr>
<td>FLXI</td>
<td>Total count on the number</td>
</tr>
<tr>
<td>FLZO</td>
<td>Floating zero</td>
</tr>
</tbody>
</table>

#### 3.3.4.3.2 Global

None

#### 3.3.4.3.3 Entry Points

- FLOO
3.3.10.3.4 External References

SIGN Subroutine to copy Sign.

UPOO Subroutine to unpack first character.

UP01 Subroutine to unpack character.

$QE Subroutine to perform floating exponentiation.

$QK Subroutine to perform floating point add.

$QM Subroutine to perform floating point multiply.

$QS Subroutine to convert integer number to floating number.

$SE Subroutine to save parameters.
3.3.10.4 Detailed Flow Chart

ZERO FLS1, FLSW, FLX2, FLX2+1

FLTC COUNT: 0

STORE FLTC COUNT → FLX1

LOAD REQB WITH BUFFER ADDRESS OF FLTM

CALL UPOO SET 1ST CHAR FROM BUFFER

FL01

MASK LAST 4 BIT → FLCH

MULTIPLY FLS1 BY 10

HAS THERE BEEN PREV OVERFLOW

CALL $QM
MULTIPLY FLS2 BY 10

FL03
FL00
PAGE 2

WAS OVERFLOW SET

SET OVERFLOW SW FLSW
SET REGX TO AMOUNT OF OVERFLOW

CALL $QK
ADD 32,768 TO FLS2

SUBTRACT +1 FROM REGX

YES REGX EQUALS ZERO

NO

ADD FLCH TO FLS1

OVERFLOW

CALL $QK
ADD 32,768 TO FLS2

SUBTRACT 1 FROM FLXI

FLXI 0

CALL UP01
GET NEXT CHARACTER

FL01

FL00
PAGE 2

3-115
CALL $QS
FLOAT FLS1
→ FLS3

CALL $QK
ADD FLS2 TO FLS3

SUBTRACT FLTC FROM FLTC ←
REGA, REGX

CALL $QS
FLOAT REGA
← FLS2

LOAD REGA, REGB WITH FLS3

REGX 0

= ≠

CALL $QE
10 REGX →
FLS2

CALL $QM
MULTIPLY FLS3
BY FLS2

CALL SIGN
COPY SIGN

RETURN
3.3.11 HMOO - Magnetic Tape Handler

3.3.11.1 Purpose

HMOO is a general purpose tape routine capable of handling all available tape functions. The purpose of this routine is to reduce the duplication of effort that is inherent when individual users develop their own coding to handle each specific tape I/O operation. The routine currently does not return control to the calling program until I/O is complete or an error is detected, but as soon as additional interrupts are available on the DOC system, the routine will be modified to allow overlapped processing during tape operations.

3.3.11.2 Technical Description

HMOO allows input or output of variable length tape records. The Model 620i-31 Magnetic Tape System Controller is capable of performing 8 functions (see Calling Sequence 3.3.8.2.1) all of which are handled by HMOO. The BIC is used for input and output operations. HMOO first sets up the BIC instructions by using the MTU number received in the call and saves the MTU number for later use in building the actual tape instruction. The starting buffer address is picked up from the call; the ending address is computed by using the number of words to be transferred in the call, and the BIC initial and BIC final addresses are output to the BIC. The tape function is picked up from the arguments and combined with the MTU number to create the tape instruction. The I/O operation is then executed. After completion of the I/O, HMOO checks the status of the MTU and returns that status to the calling program as the fifth parameter of the call.
3.3.11.2.1 Calling Sequence

CALL HMOO,A,B,C,D,STAT

PARAMETER

FUNCTION

A

Address of location containing tape function to be performed.

0 = input binary
1 = input BCD
2 = output binary
3 = output BCD
4 = Write End of File
5 = Forward one record
6 = Backspace one record
7 = Rewind

B

Beginning location of buffer (not to exceed (377778)

C

Address of location containing number of words to transfer

D

Address of location containing unit number (108 or 118)

STAT

Address of location containing Status of I/O operation.
(Output by HMOO to calling program.)

\[
\text{STAT} = \geq 0 \quad \text{I/O completed successfully. Number indicates number of words transferred.}
\]

\[
\text{STAT} = -1 \quad \text{User requested I/O still in progress. (to be implemented)}
\]
3.3.11.2.1 Calling Sequence (Continued)

PARAMETER

FUNCTION

STAT = -2  End of File
STAT = -3  Tape Error
STAT = -4  End of Tape
STAT = -5  Beginning of Tape

Busy Return          CALL+7
Handler busy processing a previous I/O request.

Normal Return       Call +9
Requested I/O complete. User must check STAT
for errors or number of words transferred.

REGISTER

CONTENTS UPON ENTRY   CONTENTS UPON EXIT

A       Saved       Same as Entry
B       Saved       Same as Entry
X       Saved       Same as Entry
Overflow               Not effected       Same as Entry
3.3.11.2.2 General Flow Chart

- **HMOO**
- **HANDLER BUSY?**
  - **YES**
  - **A**
  - **NO**
    - **INITIALIZE PROPER BIC**
    - **BUILD UNIT AND FUNCTION IN HMFF**
    - **IS FUNCTION A TAPE POSITION OR REWIND COMMAND**
      - **YES**
      - **RETURN**
      - **NO**
        - **LOAD BIC INITIAL AND FINAL**
        - **ACTIVATE BIC**
          - **B**
          - **POSITION RETURN TO CALL+7**
          - **B**
### 3.3.11.3 Label Description

#### 3.3.11.3.1 Local

<table>
<thead>
<tr>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HM1 - HM5, HM15</td>
<td>Locations of BIC instructions initialized by HMOO</td>
</tr>
<tr>
<td>HMOI</td>
<td>Location where tape I/O instruction is constructed and executed.</td>
</tr>
<tr>
<td>HMA</td>
<td>Save area for A-register.</td>
</tr>
<tr>
<td>HMB</td>
<td>Indirect address for unit number.</td>
</tr>
<tr>
<td>HMA3</td>
<td>Initialize BIC 2 instruction. Used for initialization.</td>
</tr>
<tr>
<td>HMA4</td>
<td>Initialize BIC 4 instruction.</td>
</tr>
<tr>
<td>HMB</td>
<td>Save area for B-register.</td>
</tr>
<tr>
<td>HMBU</td>
<td>Beginning address of buffer. Used to compute number of words transferred.</td>
</tr>
<tr>
<td>HMBV</td>
<td>Previous I/O in progress (to be implemented).</td>
</tr>
<tr>
<td>HMB3</td>
<td>Sense BIC 2 not busy instruction. Used for initialization.</td>
</tr>
<tr>
<td>HMB4</td>
<td>Sense BIC 4 not busy instruction. Used for initialization.</td>
</tr>
<tr>
<td>HMC3</td>
<td>Set BIC 2 starting address instruction. Used for initialization.</td>
</tr>
<tr>
<td>HMC4</td>
<td>Set BIC 4 starting address instruction. Used for initialization.</td>
</tr>
<tr>
<td>HMD3</td>
<td>Set BIC 2 final address instruction. Used for initialization.</td>
</tr>
<tr>
<td>HMD4</td>
<td>Set BIC 4 starting address instruction. Used for initialization.</td>
</tr>
<tr>
<td>HME3</td>
<td>Enable BIC 2 instruction. Used for initialization.</td>
</tr>
<tr>
<td>HME4</td>
<td>Enable BIC 4 instruction. Used for initialization.</td>
</tr>
<tr>
<td>HMFF</td>
<td>Tape function being performed.</td>
</tr>
</tbody>
</table>
3.3.11.3.1 Local (Continued)

**HM3**
Location containing a "load BIC 2 initial register" instruction. Used for initialization.

**HM4**
Location containing a "load BIC 4 initial register" instruction. Used for initialization.

**HMUN**
Unit number.

**HMx**
Save area for X-register.

**HMXX**
Indirect argument pointer.

3.3.11.3.2 Global

**HM00**
Enter point into subroutine.

3.3.11.3.3 Entry Points

**HM00** primary entry point

3.3.11.3.4 External References

None.
3.3.11.4 Detailed Flow Chart

HMO0

HMSR
SAVE
REGISTERS

HANDLER
BUSY?

YES

A

HMO
INITIALIZE
UNIT 10 BIC
INSTRUCTIONS

NO

SET HMOO
BUSY
-1 HMBU

SET UP HNAB AS
INDIRECT ADDRESS
FOR UNIT NUMBER

UNIT
NUMBER 10?

NO

B

HM30
POSITION
RETURN TO
BUSY RETURN.
CALL +7

HMRR
RESTORE
REGISTERS

RETURN
SENSE STATUS OF APPROPRIATE MTU.

I

I7

SET HANDLER NOT BUSY

SET STATUS FOR CALLING PROGRAM

SET HANDLE NOT BUSY

RETURN
B

HMI

INITIALIZE BIC

NO

BIC NOT BUSY?

YES

PICK UP TAPE FUNCTION FROM ARGUMENTS

SHIFT FUNCTION FOR I/O COMMAND FORMAT

STORE FUNCTION IN HMFF

FUNCTION (HMFF) > 3?

YES

NO

C

C1

3-125
SET UP A REG WITH ENDING BUFFER ADDRESS

LOAD B WITH BEGINNING BUFFER ADDRESS

HM3
OUTPUT (OBR) BUFFER STARTING LOCATION TO BIC INITIALIZE

HM4
OUTPUT (OAR) BUFFER ENDING LOCATION TO BIC FINAL

HM5
ACTIVATE ENABLE BIC

C1
HMBB
SAVE UNIT NUMBER IN HMUN

D

3-126
SET USER STATUS BUSY (-1)

BUILD TAPE EXC INST FROM HMFF AND HMUN

STORE EXC INSTRUCTION AT HM01

POSITION HM00 FOR NORMAL RETURN

HM20 MTU BUSY

HMRR RESTORE REGISTERS

E
OR UNIT NUMBER INTO SENSE INSTRUCTIONS

HM01

TAPE EXC INSTRUCTION

HM03

PROCESS END OF OPERATION

RETURN

HM03

PROCESS END OF I/O OPERATION

OR UNIT NUMBER INTO SENSE INSTRUCTIONS

HM12

SENSE MTU REWINDING

HM04

SENSE FILE MARK

HM05

SET USER STATUS TO -2

SET USER STATUS TO 0

HM01

F

G

I

3-128
OR UNIT NUMBER INTO SENSE INSTRUCTIONS

HM03

PROCESS END OF I/O OPERATION

RETURN

HM03

HM01

TAPE EXC INSTRUCTION

HM03

PROCESS END OF I/O OPERATION

HM04

SENSE FILE MARK

YES

HM05

SET USER STATUS TO -2

NO

HM12

SENSE MTU REWINDING

YES

HM07

SET USER STATUS TO 0

NO

G

3-129
HM15
CHECK MTU READY ROUTINE

HM16
SET HANDLER NOT BUSY
0 → HMBU

HMRR
RESTORE REGISTERS

RETURN

HM15
COMPUTE NUMBER OF WDS TRANSFERRED

CHECK MTU READY ROUTINE

HM20
OR UNIT NUMBER INTO SENSE INSTRUCTION

NO

SENSE MTU READY

YES

RETURN

3-130
HMSR

SAVE REGISTERS IN HMA, HMB, AND HMX

RETURN

HMRR

RESTORE REGISTERS FROM HMA, HMB, AND HMX

RETURN

3-131
3.3.12 IAAR Input TTY or Modem

3.3.12.1 Purpose

The purpose of IAAR is to input a string of characters into a user specified buffer from the CLINC TTY's, DOC TTY, or DOC 103 Modem.

3.3.12.2 Technical Description

IAAR gains control when it is called by the generalized Input Message (IM00) Routine. The complete input string is stored in the user buffer before the user status word is updated and control is returned to IM00. As soon as character interrupts are available on the DOC system, IM00 may be modified to return control to the original calling program when input of the first character is initiated. IAAR calls the Input Character Routine (ISB1) to input, edit, and echo each input character. The input instructions in ISB1 are set up for the assigned input device by the Initialize System (ISO0) routine.

In addition to inputting and packing the input message, IAAR checks for several input control characters and terminates the input accordingly. If an input complete code (*) is received, the user status word is set to the number of characters packed in the input buffer and control is returned to the user via IM00.

If a cancel line ($) character is detected, a (-3) status is returned. If a cancel retrieval request (@) character is received, a (-4) status is returned. Finally, if the maximum number of input characters allowed by the user program is reached, the operation status word is set to that number and control returned to the calling program. (Note that the *, $, @, CR, and LF are not stored or counted by IAAR).
3.3.12.2.1 Calling Sequence

CALL IAAR, A,B

PARAMETER

A
B

FUNCTION

Address of input buffer.
Maximum number of characters allowed for the input buffer.

REGISTER

CONTENTS UPON ENTRY
CONTENTS UPON EXIT

A
N/A
Modified

B
N/A
Modified

X
N/A
Modified

Overflow
N/A
Modified
SAVE BUFFER ADDRESS AND MAX CHARACTER ALLOWED

ISBl INPUT CHARACTER

INPUT COMPLETE CODE (*)

YES

E

NO

CANCEL LINE CHAR.? (\$)

YES

SET USER STATUS TO CANCEL LINE CODE (-3)

IAA8

NO

CANCEL REQUEST CHAR. (@)

YES

SET USER STATUS TO CANCEL REQUEST CODE (-4)

IA10

NO

FIRST OR SECOND CHAR.

NO

FIRST CHAR. OF PACK WORD.
STORE PACKED CHARACTERS INTO USER BUFFER

MAX. INPUT CHAR. COUNT? 

YES

SET USER'S STATUS TO INPUT CHAR. COUNT.

RETURN

NO

A

SHIFT INPUT CHARACTER INTO LEFT HALF OF B-REGISTER

MAX. NUMBER OF CHARACTERS INPUT?

NO

RETURN

YES

SET USER'S STATUS TO INPUT CHAR. COUNT

A

3-135
FIRST OR SECOND CHARACTER OF PACK WORD BEING PROCESSED.

FIRST CHAR.

SECOND CHAR.

MASK OUT THE """

STORE LAST CHARACTER IN USER BUFFER

C
3.3.12.3 Label Description

3.3.12.3.1 Local

IAAC - Location containing actual input character count.
IAA9 - Location containing incremented buffer address.
IA12 - Save area for maximum input characters allowed.

3.3.12.3.2 Global

None

3.3.12.3.3 Entry Points

IAAR

3.3.12.3.4 External References

- ISB1 - Subroutine called to input, edit, and echo each input character.
3.3.12.4 Detailed Flow Chart

- IAAR
- SAVE BUFFER STARTING ADDRESS IN IA12
- SAVE CHARACTER COUNT IN IAAC
- A
- IAA5
  - SOF
  - SET CHARACTER SWITCH
- B
- ISB1
  - INPUT CHARACTER
- INPUT COMPLETE CODE? (*)
  - YES
  - H
  - NO
- CANCEL LINE CHAR.? ($)
  - YES
  - F
  - NO
  - C

OVERFLOW INDICATOR IS USED FOR INDICATING WHICH HALF OF PACK WORD IS BEING PROCESSED.
C

CANCEL REQUEST CHAR. (@) YES

NO

JOF FIRST OR SECOND CHAR. OF PACK WORD?
FIRST CHAR.
SECOND CHAR.

STORE PACKED INPUT WORD INTO USER BUFFER

INCREMENT BUFFER ADDRESS

INCREMENT CHARACTER COUNT

MAX. CHAR. COUNT INPUT? YES

NO

A

3-139
SET USER STATUS TO INPUT CHAR. COUNT

INCREMENT CHARACTER COUNT

MAX. NUMBER OF CHAR. INPUT?

YES

SHIFT INPUT CHARACTER INTO LEFT HALF OF B

RETURN

NO

SET USER STATUS TO CANCEL LINE CODE (-3)

RETURN
IA10
SET USER STATUS TO CANCEL REQUEST CODE (-4)

RETURN

IA20

H
JOE
FIRST OR SECOND CHAR.? 

FIRST CHARACTER 

SECOND CHARACTER 

MASK OUT THE 

STORE LAST CHARACTER IN USER BUFFER 

E

E
3.3.13 IM00 - Input Message

3.3.13.1 Purpose

The purpose of the Input Message Routine is to provide MDTRS with a
generalized input routine which will service all input message requests
regardless of the specific hardware configuration in which MDTRS is
operating.

3.3.13.2 Technical Description

IM00 is designed to require a common calling sequence for input message
requests. IM00 serves as an interface (or link) to the specific I/O
handler which has been assigned by the Initialize System (IS00) Routine.
The common arguments passed to IM00 by the calling program are converted
by IM00 to the proper calling sequence required by the specific I/O
handler.

IM00 utilizes a table (IMDT) which contains the addresses of 'set up'
processing to be done for each particular I/O handler. The device code
(IMDC) is used to index into the IMDT table to obtain the address of the
'set up' processing for each I/O handler. After the appropriate handler
has been called, IM00 does not regain control until either the input
character count has been satisfied or a terminating condition has been
detected by the I/O handler. The I/O handler may return control to the
user for concurrent processing while the I/O is taking place. See the
individual I/O handler documentation for specific input processing.
When IM00 regains control it returns control to the calling program.
This status word will be initialized by the specific handler to (-1) as soon as the first input command is issued. As soon as the first character is received by the computer, the status word is changed to (-2). If a cancel line character ($) is received, a (-3) will be set in the status word. If a cancel request character (@) is received, a (-4) will be returned. Refer to the calling sequence for additional error codes to be implemented at a later date.

If an input complete code (*) is received by the I/O handler or the maximum number of characters to be input is reached, the operation status word will be changed to a positive value indicating the number of data characters which have been placed in the user specified buffer.

The special characters: Carriage Return (CR), Line Feed (LF), Asterisk (*), Dollar Sign ($), at sign (@), EOM, LINE or END are not placed in the user's buffer but are echoed back to the output device by the assigned I/O handler. This status word address, in addition to the beginning address of the input buffer, address of character count, and address of cursor position word if required, is defined in IM00 but used by the individual I/O handlers.

3.3.13.2.1 Calling Sequence

CALL IM00, A,B,C,D
PARAMETER | FUNCTION
---|---
A | The beginning address where the message is to be stored. (Packed two characters per word.)
B | The address of a word containing the maximum number of characters to be accepted as input.
C | Always zero (0).
D | The address of an operation status word.

The status returned by the input message routine IM00 will be as follows:

Positive value = Input complete. The value of the positive number will be the number of characters received. (May be zero if no data characters are transmitted but operation is complete.)

-1 = Input initiated - first character not received yet.
-2 = Input initiated - first character received.
-3 = Cancel line character ($) received.
-4 = Cancel request character (@) received.
-5 = Parity error (can character) (CRT only)
-6 = 103 modem disconnected.
-7 = CRT not ready.

The return addresses will be as follows:

CALL+6 = previous I/O still in progress.
CALL+8 = I/O initiated. User must check operation status word at this point.

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Saved</td>
<td>Restored</td>
</tr>
<tr>
<td>B</td>
<td>Saved</td>
<td>Restored</td>
</tr>
<tr>
<td>X</td>
<td>Saved</td>
<td>Restored</td>
</tr>
<tr>
<td>Overflow</td>
<td>Unknown</td>
<td>Modified</td>
</tr>
</tbody>
</table>
3.3.13.2.2 General Flow Chart

IMOO

IMOO BUSY?
IMBU=-1

YES

NO

SET IMOO BUSY
-1 -> IMBU

SET UP CALL TO
APPROPRIATE
INPUT HANDLER

CALL INPUT
HANDLER

RETURN

SET IMOO TO
BUSY RETURN
CALL+6

RETURN

A

A
3.3.13.3 Label Description

3.3.13.3.1 Local

IMA  A-Register Save Area
IMB  B-Register Save Area
IMBU  Busy indicator for IM00 Routine (IMBU = -1 for Busy)
IMDT  Table containing address of set up processing for each individual Input Handler call.
IMX  X-Register Save Area
IM05  Address of call to TTY or Modem Input Handler. Used to initialize calling parameters.

3.3.13.3.2 Global

IMBA  Beginning Address of Input Buffer (Referenced by Input Handler)
IMCC  Address of Input Character Count (Referenced by Input Handler)
IMCP  Address of Cursor Position Word if required (Referenced by Input Handler)
IMDC  Device code. Set by IS00.
IMOS  Address of Operation Status Word (Referenced by Input Handler)

3.3.13.3.3 Entry Points

IM00  
IMSR  - Entry point for Saving Registers. Called by Output Message Routine (OM00).
IMRR  - Entry point for Restoring Registers. Called by Output Message Routine (OM00).

3-147
3.3.13.3.4 External References

IAAR - Input handler for both CLINC TTY's, DOC TTY, and DOC 103 Modem.
3.3.13.4 Detailed Flow Chart

Using device code IMDC as an index, pick up the address where processing is to continue for setting up the call for the appropriate I/O handler.
NO

TO BE DEVELOPED

DEVICE CODE (IMDC) 1 THROUGH 4? NO

IM02

STARTING ADDRESS OF BUFFER (IMBA) → IAAR CALL+2

CHARACTER COUNT (IMCC*) → IAAR CALL+3

IM05

IAAR INPUT MESSAGE HANDLER

CLEAR IM00 BUSY INDICATOR 0 → IMBU

IMRR RESTORE REGISTERS

RETURN

HALT

INPUT MESSAGE FROM CLINC TTY 1 OR 2, DOC TTY, OR DOC 103 MODEM.
SAVE REGISTERS

STORE A, B, AND X IN IMA, IMB, IMX

RETURN

RESTORE REGISTERS

RESTORE A, B, X FROM IMA, IMB, IMX

RETURN

3-151
3.3.14 ISOO - Initialize System

3.3.14.1 Purpose
The primary purpose of the ISOO subroutine is to permit the same MDTRS object program to operate for various configurations on various computer systems. This capability eliminates the tedious, costly, and sometimes confusing task of maintaining many "configuration tailored" object programs, operating instructions, and program documentation. ISOO accomplishes this by initializing all input/output routines with the operator requested configuration device numbers and device dependent coding.

3.3.14.2 Technical Description
The ISOO routine requires the operator to type in a description of the configuration that MDTRS must satisfy. Refer to the Users Guide for specific computer/operator dialogue. ISOO types the code numbers and instructions on the operator console before requiring the operator to answer questions about the configuration. ISOO contains tables of device numbers, PIM numbers, interrupt masks, and interrupt locations associated with each computer system and each I/O device. In addition, tables containing I/O instruction addresses in each of the I/O handlers are maintained so that those instructions may be modified if necessary. These tables are used to set up the appropriate I/O handlers with proper device numbers, PIM numbers, etc. ISOO initializes the Input Message Routine (IMOO) and the Output Message Routine (OMOO) to call the appropriate handlers based upon the requested hardware configuration.
All inputs from the operator are checked for validity. The operator is required to assign the tape unit number for the Master Data Tape and HCO0 verifies that the tape is on-line. In addition, HCO0 initializes the 103 modem and waits for a telephone connection if it is part of the requested configuration.

### 3.3.14.2.1 Calling Sequence

**CALL** ISOO

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>Modified</td>
</tr>
</tbody>
</table>
3.3.14.2.2 General Flow Chart

ISO0

TYPE OUT INITIALIZATION INSTRUCTIONS TO OPERATOR

ACCEPT AND EDIT COMPUTER & DEVICE CODES FROM OPERATOR

SET UP APPROPRIATE HANDLERS WITH DEVICE CODES

"MOUNT MASTER TAPE" MESSAGE

ACCEPT AND EDIT MTU NUMBER FROM OPERATOR

ASSIGNED MTU ON-LINE?

YES

NO

A

B

3-154
3.3.14.3 Label Description

3.3.14.3.1 Local

**ISCC** Temporary location for computer code. (for future use only)

**ISDT** Table containing actual computer device codes associated with device code numbers used by IS00 to communicate with the operator.

**ISD3** Table of addresses which must have device number placed in them for TTY and 103 Modem I/O.

**ISHT** Table of addresses where table of addresses for modifying each handler can be found.

**ISNW** Number of words in ISTM buffer used for TTY call.

**ISN1** Number of words in ISQ1 buffer - used for TTY call.

**ISN2** Number of words in ISQ2 message buffer.

**ISN3** Number of words in ISQ3.

**ISN4** Number of words in ISQ4.

**ISN5** Number of words in ISQ5.

**ISQ1** "WHAT COMPUTER" message buffer.

**ISQ2** "PRIMARY RETRIEVAL DEVICE" message buffer.

**ISQ3** "MOUNT MASTER TAPE" message buffer.

**ISQ4** "SYSTEM INITIALIZED" message buffer.

**ISQ5** "MTU NOT READY" message buffer.

**ISTM** "INITIALIZE SYSTEM" message buffer.

**IST1** "ILLEGAL CODE" message buffer.

**IST2** Number of words in IST1

**ISZZ** End Flag for Device instruction address table (ISD3).
3.3.14.3.3 Entry Points

IS00
3.3.14.3.4 External References

CPC1     TTY or 103 Modem instruction requiring device number to be initialized.
CPC2     TTY or 103 Modem instruction requiring device number to be initialized.
ISB2     TTY or 103 Modem instruction requiring device number to be initialized.
ISB3     TTY or 103 Modem instruction requiring device number to be initialized.
OZK2     TTY or 103 Modem instruction requiring device number to be initialized.
OZK3     TTY or 103 Modem instruction requiring device number to be initialized.
OZK4     TTY or 103 Modem instruction requiring device number to be initialized.
OZK5     TTY or 103 Modem instruction requiring device number to be initialized.
OZN2     TTY or 103 Modem instruction requiring device number to be initialized.
OZW1     TTY or 103 Modem instruction requiring device number to be initialized.
3.3.14.4 Detailed Flow Chart

ISO0

FILL NOP INT. LOCATIONS

INITIALIZE OPERATOR TTY CALLS WITH WORD COUNTS

ISO1

TYPE OUT INITIALIZATION INSTRUCTIONS TO OPR.

ISO2

"WHAT COMPUTER IS THIS?"

ISOIA INPUT ANSWER

ANSWER > 0 < 4

NO

ISO3

ISO90

OUTPUT ERROR MESSAGE

YES

B
IS04
STORE COMPUTER
CODE IN ISCC

C

IS05
"PRIMARY
RETRIEVAL
DEVICE?"

ISIA
INPUT
ANSWER

ANSWER
>0 <5

YES
STORE DEVICE
CODE IN IMDC AND
OMDC

NO
IS90
OUTPUT ERROR
MESSAGE

IS08

D

IS14
LOAD A WITH
INSTRUCTION TO
RECEIVE DEVICE
NUMBER

IMDC AND OMDC ROUTINES
REQUIRE THE DEVICE NUMBER

FROM DEVICE INSTRUCTION TABLE

3-161
IS20
STORE INSTRUCTION WITH PROPER DEVICE NUMBER INTO APPROPRIATE LOCATION

IS25
"MOUNT MASTER TAPE" MESSAGE

ISIA
INPUT FIRST CHARACTER OF MTU NUMBER FROM OPERATOR

CONVERT TO BINARY

ADDRESS OBTAINED FROM DEVICE INSTRUCTION TABLE
INPUT SECOND CHARACTER OF MTU NUMBER FROM OPERATOR

ISIA

CONVERT TO BINARY AND MERGE WITH FIRST CHARACTER

MTU 10 OR 11?

YES

IS30

STORE SEN MTU READY IN IS31, IS3B

IS31

SEN MTU READY

YES

IS3A

"MTU NOT READY" MESSAGE

NO

IS27

IS90 OUTPUT ERROR MESSAGE

NO

F

H

3-163
IS48

CHARACTER INPUT?

YES

IS54

INPUT CHARACTER DO NOT ECHO

RETURN

NO

INPUT ONE CHARACTER FROM OPERATOR CONSOLE (STANDARD DEVICE 1); CHARACTER RETURNED IN A-REGISTER

ISIA

READ BUFFER FULL?

YES

INPUT CHARACTER INTO THE A-REGISTER

NO

WRITE BUFFER READY

YES

OUTPUT CHARACTER FROM A-REGISTER (ECHO)

RETURN
OUTPUT N NUMBER OF WORDS TO THE OPERATOR'S CONSOLE (STANDARD DEVICE 1)

GET NUMBER OF WORDS AND BUFFER ADDRESS FROM CALLING PROGRAM ARGUMENTS

K

IS91
LOAD WORD FROM BUFFER INTO B-REGISTER

SHIFT MOST SIGNIFICANT 8 BITS INTO A

IS93
OUTPUT A-REG.

SHIFT NEXT CHARACTER INTO A-REGISTER

IS93
OUTPUT A-REG

L
LAST WORD PROCESSED?

YES
RETURN

NO
K
3.3.15 ISB1 - Input a character from KSR-35

3.3.15.1 Purpose
This routine inputs a character from the KSR-35 teletype, checks it for validity, and outputs it to the TTY. When the character is illegal it is deleted.

3.3.15.2 Technical Description
N/A

3.3.15.2.1 Calling Sequence
CALL ISB1

PARAMETER
None

REGISTER | CONTENTS UPON ENTRY | CONTENTS UPON EXIT
--- | --- | ---
A | destroyed | 
B | destroyed | 
X | undisturbed | 
Overflow | undisturbed |
3.3.15.2.2 General Flow Chart

1. Is byte 1 (ISBI) present?
2. If not, STOP.
3. If present, go to step 4.
4. Input character from TTY.
5. If not a legal character, go to step 1.
6. If yes, output char to TTY and set byte 3 (OZKB).
7. Go back to step 1.
3.3.15.3 Label Description

3.3.15.3.1 Local
ISB5 - ASCII character mask

3.3.15.3.2 Global
None

3.3.15.3.3 Entry Points
ISB1 - primary entry point

3.3.15.3.4 External References
OZKB - output one character to TTY
3.3.15.4 Detailed Flow Chart

ISB1

IS B2

READ READY

YES

INPUT CHAR FROM KSR-35

SET PARITY BIT

CARRIAGE RET OR LINE FEED?

YES

OUTPUT CR OR LINE FEED

NO

LEGAL CHARACTER?

NO

OUTPUT CHARACTER

YES

OZKB

DELETE ILLEGAL CHARACTER

IS B2

OZKB

3-171
3.3.16  L100 - List

3.3.16.1  Purpose

The purpose of this subroutine is to format, in outline form, the data specified by the WHAT response in the Retrieval Request.

3.3.16.2  Technical Description

A match to a WHAT parameter indicates that the parameter is to be output. Both the Heading and its associated Answer are output, even if both were not specified in the Retrieval Response.

Example:

WHAT: ASTHMA

If the Heading ASTHMA is found on the record, then ASTHMA will be output as well as the Answer associated with it, which, in all probability, is either NEG or POS.

Associated with each Heading-Answer pair is a "level code". This code is a number from 0-9, and it specifies two characteristics of the Heading-Answer pair: the level of the pair in the hierarchical structure of the record format, and the relation among pairs. Each of these characteristics will have an effect on the output. The level of the pair will be shown on output by indenting each Heading a number of spaces equal to the value of the level code. The relation among Heading-Answer pairs will be indicated by outputting, in addition to the matched parameter, all those pairs related to the one matched. Example: if ASTHMA is related to the Headings PAST HX and GENERAL, then these Headings and their associated Answers should also be output.
Relation among Headings is determined in the following manner: starting at the level code of the matched Heading and proceeding back through the codes, find a level code that is one less in value than the matched Headings level code. The Heading associated with this level code is related to the matched Heading. From this point, proceed backwards again until a code that is two less is found. This continues until a level code of zero has been reached. Now examine each level code forward from the matched Heading's code. If the level code is greater than the matched code, its Heading is related. When the first code less than or equal to the level code of the matched Heading is found, the search is finished.

As each Heading-Answer pair is matched by MW00 it and all related pairs are flagged for eventual output. This flagging consists of setting the word preceding the level code of each pair to -1. When the entire record has been considered, the flagged data is output. In the case of the default condition for WHAT, all data on the record is output.

If the user input "ID ONLY" as his response to the WHAT question, only the line of data containing the ID information is output.

3.3.16.2.1 Calling Sequence

CALL LI00

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

3-173
General Flow Chart

1. PARAMETERS
   - FLAG MATCHED PARAMETERS AND RELATED PARAMETERS

2. COMPUTE BEGINNING ADDRESSES FOR SEGMENTS OF TAPE BUFFER

3. OUTPUT HEADER

4. PICK UP FLAGGED HEADING

5. PICK UP ANSWER

6. OUTPUT LINE OF DATA

7. END OF RECORD
   - YES
   - NO

A

RETURN

3-174
3.3.16.3 Label Description

3.3.16.3.1 Local

LIAN  beginning address in the tape buffer of the IANS section (or the section of the record containing the Answers).

LIBC  blank counter. Counts the number of consecutive blanks before they are stored for output.

LIBL  (40\textsubscript{8}) blank

LICC  character counter. Used to count characters in special situations (Headings, Answers)

LICL  (72\textsubscript{8}) colon

LICR  (15\textsubscript{8}) carriage return

LIEA  ending address of the current Answer.

LIEQ  (75\textsubscript{8}) equal sign

LIFW  beginning address in the tape buffer of the IFWAA section (or the section of the record containing the beginning addresses of the Answers).
3.3.16.3.1 Local (Continued)

LIID  ID only flag. If set, indicates that the user response to WHAT was "ID ONLY."

LIIQ  beginning address in the tape buffer of the IQ section (or the section of the record containing the Headings).

LILA  address of the current level code

LILC  the value of the current level code

LILL  line character-counter. Counts the number of characters on the current line.

LILQ  beginning address in the tape buffer of the LCQ section (or the section of the record containing the level codes).

LILX  a dummy level code

LIQN  question number. The number of the Heading-Answer pair currently being dealt with.

LIQX  temporary question number. Dummy location for the question number

LISA  starting address of the current Answer

LIST  status word returned set by OMOO on output
3.3.16.3.1 Local (Continued)

LISV utility save location

LITB (74) tab character

3.3.16.3.2 Global

None

3.3.16.3.3 Entry Points

L100

3.3.16.3.4 External References

CDOO routine to decode the date on the record.

CPRT Request Table

CPTB Tape input buffer

MWOO routine to match responses to WHAT request, to record Headings.

OMOO routine to control output to primary output device.

PKBX beginning address of buffer into which characters are to be packed.

PKIX index into buffer into which characters are being packed.

PKSW switch indicating which half of word to pack next character.
3.3.16.3.4 External References (Continued)

PK01 routine that packs characters two per word.

SE00 routine to output error message to system I/O device, the teletype.

TABF buffer used to pack a line of data for output.

UPO0 routine to unpack characters from buffer.
3.3.16.4 Detailed Flow Chart

L100

IQ = 151

LCQ = IQ + 12(IQNDX)

IFWAA = LCQ + 3(IQNDX)

IANS = IFWAA + 3(IQNDX)

CPRT+6 = 0

WHAT = DEFAULT

YES

ID ONLY FLAG SET

YES

SET ADDRESS OF CURRENT LEVEL CODE = 0

NO

NO

100

130

10
COMPARE A RECORD QUESTION TO ALL WHAT RESPONSE OPERANDS

SET QUESTION NUMBER = 5 (LIQN)

MATCH WHAT

MATCH

YES

SET WORD PRECEDING LEVEL CODE OF MATCHED HEADING TO -1

SAVE MATCHED HEADING LEVEL CODE (LILC) (LILX)

NO

90

30
FLAG RELATED LEVEL CODES (ASSOCIATED WITH HEADING-ANSWER PAIRS) PRECEDING MATCHED LEVEL CODE (HEADING-ANSWER PAIR).

DECREMENT TEMPORARY LEVEL CODE ADDRESS (X-REG)

DECREMENT LEVEL CODE ADDRESS TO TEMPORARY LEVEL CODE

ARE TWO CODES EQUAL

YES

NO

60

40
50

SET WORD PRECEDING CODE AT LEVEL CODE ADDRESS TO -1.

30

60 LI30

RESTORE X-REG TO ADDRESS OF MATCHED LEVEL CODE

SET TEMPORARY QUESTION NO. TO ACTUAL NO. OF MATCHED QUESTION

70

TEMPORARY QUES. NO. = LAST QUESTION NO.?

YES

90

NO

80

FLAG RELATED LEVEL CODES (ASSOCIATED WITH HEADING-ANSWER PAIRS) FOLLOWING MATCHED LEVEL CODE (HEADING-ANSWER PAIR)

3-182
80

INCREMENT TEMPORARY QUESTION NO.

INCREMENT ADDRESS OF LEVEL CODE (X-REG)

COMPARE CODE AT LEVEL CODE ADDRESS (X-REG) TO MATCHED LEVEL CODE

NEW CODE > MATCHED CODE

YES

SET WORD PRECEDING CODE AT LEVEL CODE ADDRESS (X-REG) TO -1

NO

90

70

3-183
END OF HEADINGS?

90 LI40

QUESTION NO. = LAST QUESTION NO.? YES

NO

INCREMENT QUESTION NO. (LIQN)

20

120

SET TEMPORARY QUESTION NO. = 5 (LIQX)

100 LI41

FLAG ALL LEVEL CODES (ASSOCIATED WITH HEADING-ANSWER PAIRS)

COMPUTE ADDRESS OF LEVEL CODE ASSOCIATED WITH QUESTION NO. 5

110

3-184
120 LI45

LEVEL CODE ADDRESS = 0

YES

NO

RETURN

130 LI46

LI$1

INITIALIZE PK00 ROUTINE

CLEAR LINE CHARACTER-COUNTER (LILL)

ID

ONLY FLAG
SET?

NO

YES

140 LI48

PK01

PACK 2 (CARRIAGE RETURNS - LINE FEEDS)

PK01

PACK FORM CHARACTER (2148)

150
PICK-UP DAY OF MONTH

UNPACK DAY OF MONTH

PK01
PACK 1ST DIGIT

INCREMENT LINE CHAR-CTR (LILL)

PK01
PACK 2ND DIGIT

170
180

INCREMENT LINE CHAR-CTR (LILL)

PK01
CARRIAGE RETURN LF

INCREMENT LINE CHAR-CTR BY 2 (LILL)

LIS3
OUTPUT THE LINE

ID ONLY FLAG (LIID) SET?

YES

NO

190

RETURN

3-190
SET QUESTION NO (LIQN) TO 4

INCREMENT QUESTION NO. (LIQN)

COMPUTE ADDRESS OF FLAG PRECEDING LEVEL CODE FOR CURRENT QUESTION

FLAG SET?

QUESTION NO. = LAST QUESTION NO.

RETURN
220

SAVE LEVEL CODE

CLEAR BLANK COUNTER (LIBC)

CLEAR LINE CHAR-CTR (LILL)

CLEAR CHARACTER CTR (LICC)

COMPUTE BEGINNING ADDRESS OF HEADING

230
270

PK01
PACK
CHARACTER

INCREMENT LINE CHARACTER (LILL)

250

280

LI85

COMPUTE
BEGINNING ADDRESS
OF ANSWER

ADDRESS = 0

YES

NO

PK01
PACK COLON

INCREMENT
LINE CHAR-CTR

SAVE TEMP
QUEST. NO.

LI83
OUTPUT LINE

210

290

3-195
LICO

370
LICO

BLANK CTR=0

YES

NO

380
LIC1

PK01
PACK CHAR

INCREMENT LINE CHAR-CTR

LI$6
TEST FOR END OF LINE

360

LI$4
STORE BLANKS

390
LIC5

SET CNT = 7

400

3-199
LIDO

TEST END OF LINE

END OF ANSWER

UNPACK CHAR

INCREMENT CHAR CTR

CHAR = TAB

CHAR = CARRIAGE RETURN

POSITION TO TAB POSITION

OUTPUT LINE
430

CHAR = BLANK

INCREMENT CHAR COUNT

BLANK CTR = 0

OUTPUT BLANKS

PK01
PACK CHAR

INCREMENT LINE CHAR CTR

410

ANSWER
INITIALIZE PACK ROUTINE BUT
DO NOT CHARACTER

LI$1

PK00

CLEAR SWITCH (PKSW)

SET INDEX TO 1 (PKIX)

DECREMENT BUFFER ADDRESS (PKBX)

RETURN
STORE BLANKS AS INDICATED BY BLANK CTR

SAVE CHAR

PK01
PACK BLANK

DECREMENT BLANK CTR

INCREMENT LINE CHAR-CTR

LI$6
TEST FOR END OF LINE

NO
BLANK CTR=0?

YES
PICK-UP SAVED CHARACTER

RETURN
POSITION TO TAB POSITION (25)

LI$5

LINE
CTR ≥ 25

YES

NO

PK01
STORE BLANK

INCREMENT
LINE CHAR CTR

RETURN
TEST FOR END OF LINE

LINE
CHAR CTR > 119

YES

LI$3
OUTPUT LINE

CLEAR LINE CHAR-CTR

RETURN
WRAP UP LIST OPTION

LI$7

CLEAR ID ONLY FLAG

RETURN
3.3.17 MAOO - Match

3.3.17.1 Purpose.
The purpose of the MATCH Subroutine (MAOO) is to find a match in the Request Buffer and in the Tape Buffer.

3.3.17.2 Technical Description
MAOO compares the Tape Buffer to the Request Buffer. There are four types of matches that can be requested by the user: Heading Only, Headings and Prose Data, Headings and Numeric Values, and Headings and Range of Numeric Data.

MAOO checks for a match by heading in the Request Buffer and the Tape Buffer. If a request header matches, it makes a further test to ascertain whether the request is heading only, heading-prose, heading numeric value, or heading range of numeric value. If heading only has been requested, the 'match' exit is taken.

On heading-prose data the subroutine checks the answer of the Tape Buffer against the string of alphanumeric characters in the Request Buffer.

For heading-numeric value request, the numeric value in the Tape Buffer is converted to a pseudo floating point number. The value in the Request Buffer is also converted to the same pseudo floating point format. Then the pseudo floating point numbers are compared on the sign, the number of characters in pseudo floating point number, the number of characters to the left of the
decimal point, and the numeric values.

On a heading range request, MAOO converts the Tape Buffer value to the pseudo floating point format. The high and the low in the Request Buffer are converted to the pseudo floating point format and a range test made.

MABO is the second entry point into MAOO subroutine. The purpose for the second entry point is to convert numeric value in the Tape Buffer to pseudo floating point number format. MABO is called after MAOO is called to specify match of heading only.
3.3.17.2.1 Calling Sequence

CALL MAOO, HEAD, OP, FLAG, SW

PARAMETER

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEAD</td>
<td>The heading number to be checked in the Tape Buffer.</td>
</tr>
<tr>
<td>OP</td>
<td>Pointer to Operand Buffer.</td>
</tr>
<tr>
<td>FLAG</td>
<td>FLAG 0. Check only Heading:</td>
</tr>
<tr>
<td></td>
<td>1. Check Heading and Prose.</td>
</tr>
<tr>
<td></td>
<td>2. Check Heading and Numerical Data.</td>
</tr>
<tr>
<td></td>
<td>3. Check Heading and Range of Numerical Data.</td>
</tr>
<tr>
<td>SW</td>
<td>Switch to indicate whether Tape Buffer has been moved. MAOO set to -1 when moving the Tape Buffer, the user must set to zero when the new Heading number is checked.</td>
</tr>
</tbody>
</table>

CALL MABO

PARAMETER

A call to MAOO must have been made to set up parameters for MABO

REGISTER

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents Upon Entry</th>
<th>Contents Upon Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>Modified</td>
</tr>
</tbody>
</table>

Overflow       | N/A                 | N/A               |

3-213
3.3.17.2.2 General Flow Chart

MAOO

IS HEADING NO. GREATER THAN MAX.
YES
NO

HAS TAPE BUFFER BEEN MOVED
YES
NO

SET TAPE BUFFER BEEN MOVED TO YES

COMPUTE BEGIN ADDRESS OF HEADING

MOVE HEADING TO TEMP BUFFER OMITTING SPACE

COUNT THE NO. OF CHARACTERS IN TAPE BUFFER

DISPLAY BAD RECORD

ADD 2 TO RETURN ADDRESS

RETURN NO MATCH

3-214
COUNT THE NO. OF CHARACTERS IN REQUEST BUFFER

COUNT IN REQUEST EQUAL COUNT IN TAPE BFR
NO

REQUEST BUFFER EQUAL TEMP BUFFER
NO

COMPARE ON HEADING ONLY
YES

RETURN MATCH

PROSE DATA
NO

FIND START & ENDING ADDRESS OF ANSWER
YES

ANSWER FOUND
NO
DOES REQUEST BUFFER EQUAL TAPE BUFFER POSE

RETURN MATCH

CONVERT TAPE BUFFER NUMBER TO PSEUDO FLOATING NUMBER

WAS NUMBER CONVERT

CONVERT REQUEST NUMBER TO PSEUDO FLOATING NUMBER

RANGE

MA00
CONVERT UPPER RANGE NUMBER TO PSEUDO FLOATING NUMBER

TAPE SIGN LOWER SIGN

TAPE EXPONENT LOWER EXP.

TAPE NO. LOWER NO.

RETURN MATCH
10

TAPE SIGN POSITIVE

NO 1

YES

RETURN MATCH

11

TAPE SIGN POSITIVE

YES 1

NO

RETURN
3.3.17.3 Label Description

3.3.17.3.1 Local

MABA Address of the Number to Convert Pseudo Floating Point
MABI \((\text{Heading Number} - 1) \cdot 12\) Used to calculate Base Address of Heading
    in Tape Buffer
MACT Working Counter
MADS Decimal Point Switch for Pseudo Floating Point
MAEC Terminate Character to Check for in Subroutine MA55
MAFP Address of Flag
MAFW Pointer to Heading Answer in Tape Buffer
MAF6 Two Characters of Pack ASCII Zero
MAHB Area where Tape Header is Stored
MAHD Address of Heading Number
MAHN Heading Number
MAI1 Number of Character of Heading in the Tape Buffer
MAI2 Number of Characters of Heading in Request Buffer
MAI3 Working Storage used to Store Total Count Address for
    Pseudo Floating Point
MAI4 Working Storage used to Store Exponent Address for Pseudo
    Floating Point
MALM Ending Address of Answer in Tape Buffer
MANS Beginning Address of Answer in Tape Buffer
MAOP Address of Pointer to Operand Buffer
MAPP Set to 1 when there is an Odd Number of Characters in Heading
3.3.17.3.1 Local (Continued)

MAPS  Working Storage

MAQ1  (Heading Number 01)*3  Used to Calculate Base Address of
      Heading Answer in Tape Buffer

MASI  Address of Sign of Pseudo Floating Point

MASW  Address of Packet Switch

MAW1  Working Storage for Temporary Calculation

MAW2  Working Storage for Temporary Calculation

MAXX  Index when Odd Character is in Request Buffer and MAHB

MASW  Switch Indicate which part to Unpack

MAZS  Leading Zero Switch for Pseudo Floating Point

3.3.17.3.2 Globe

MAHE  High Exponent for Floating Point Number

MAHL  High Count for Pseudo Floating Point Number

MAHM  High Pseudo Floating Point Number

MAHS  High Sign for Pseudo Floating Point Number

MALC  Low Count for Pseudo Floating Point Number

MALE  Low Exponent for Pseudo Floating Point Number

MALF  Low Pseudo Floating Point Number

MALS  Low Sign for Pseudo Floating Point Number

MAS1  Switch set to Non Zero on No Match Condition. Referenced by TA00

MATE  Tape Exponent, Number of Digits to Left of Decimal Point

MATL  Tape Count of Number of Digits of Pseudo Floating Point Number

MATM  Tape Pseudo Floating Point Number

MATS  Tape Sign Word 0, Positive; -1 Negative

3-222
3.3.17.3.3 Entry Point

MAB0
MAOO

3.3.17.3.4 External References

CMOO Compare Subroutine
CPTB Table Buffer, See Appendix for Details
FILL Subroutine to Fill Buffer
PK00 Entry Point of Subroutine to Pack First Character of Buffer
PK01 Entry Point of Subroutine to Pack Characters in Buffer
UP00 Entry Point of Subroutine to Unpack First Character of Buffer
UP01 Entry Point of Subroutine to Unpack Characters in a Buffer
$SE Subroutine to Store Parameter List
HAS TAPE BUFFER BEEN MOVED

YES → MA 07

NO

SET NO. MATCH SW TO NO

SET TAPE BUFFER-SW TO YES

COMPUTE QUESTION INDEX (QUESTION NO-D*3)

1

3-225
1

COMPUTE BEGIN ADDRESS OF HEADING

LOAD LOOP CONTROL WITH 24

CHARACTER BLANK

YES

UFO1
GET NEXT CHARACTER

SUBTRACT 1 FROM LOOP CONTROL

NO LOOP CONTROL ZERO

YES

MA 04

ADD 2 TO RETURN ADDRESS

RETURN NO MATCH

3-226
MA 05

SET MA11 TO 1

LOAD BREG WITH TEMP BUFFER OF HEADING

PK00
PACK 1ST CHARACTER IN TEMP BUFFER

SUBTRACT 1 FROM LOOP CONTROL

LOOP CONTROL ZERO

YES

MA 07

NO

UP01
GET NEXT CHARACTER

YES

CHARACTER BLANK

NO

PK01
PACK CHARACTER IN TEMP BUFFER

ADD 1 TO MA11
PICK THE ADDRESS OF REQUEST BUFFER FROM OP BUFFER

UP00
GET FIRST CHARACTER FROM REQUEST BUFFER

SET MAI2 TO 1

UP01
GET NEXT CHARACTER FROM REQUEST BUFFER

CHARACTER END

MAI1 EQUAL MAI2

ADD 1 TO MAI2

MAI1

MA 07

MA 04

MA 09
COMPUTE NO WORDS TO COMPARE MA12/2

REQUEST EQUAL TAPE HEADING

YES

ODD

NO

NO

LAST CHARACTER REQUEST TAPE

YES

MA 04

JUST HEADING

YES

RETURN MATCH

NO

PROSE DATA

YES MA 12

NO

MA 20

3-229
GET 1ST CHARACTER FROM REQUEST

TURN OFF 1ST CHARACTER SW 0 - MAW1

PICK UP REQUEST BUFFER

UPQD

GET 1ST CHARACTER FROM REQUEST

MA 15

MA 20

MAS1

OFF

ON

MABO

CONVERT TAPE FLOATING

MA 25

3-233
MABO

GET START & STOP ADDR. OF TAPE BUFFER

INDEX 0

YES

NO

RETURN

FILL ZERO FILL TAPE MANTISSA

ZERO → LEADING ZERO SW MAZS
DEC POINT SW MADS
TAPE EXPONENT MATE
TAPE COUNT MATC
CHARACTER SW MUSW
MAW1

LOAD ADDRESS OF TAPE MANTISSA IN MABA

LOAD ADDRESS OF TAPE EXPONENT IN MA14

LOAD ADDRESS OF TAPE COUNT IN MA13

LOAD ADDRESS OF TAPE SIGN IN MASI

2

3-234
2

MUOO
GET A
CHARACTER
FROM TAPE
BUFFER

CHARACTER
( = )

YES

ADD 4 TO TAPE
BUFFER
ADDRESS

TAPE
BUFFER
ADDRESS
LIMIT

YES

MA
24

NO

MUOO
GET A
CHARACTER
FROM TAPE
BUFFER

MA
22

3-235
LOAD ADDRESS OF LOWER EXPONENT IN MAI4

LOAD ADDRESS OF LOWER COUNT IN MAI3

LOAD ADDRESS OF LOWER MANTISSA IN MABA

LOAD ADDRESS OF LOWER SIGN IN MASI

PICK UP ADDRESS OF REQUEST BUFFER

UPO0 GET 1ST CHARACTER FROM RQST

MA50 CHECK FOR SIGN

END DATA

YES MA 04

SIGN PRESENT

YES MA 28

NO MA 27
HIGH

MATC
MALC

LOAD MATC
IN REGB

LOAD MALC
IN REGB

LOW

COMPUTE NO. OF
WORD TO COMPARE
OF MANTISSA

TAPE
MANTISSA EQUAL
LOWER MANTISSA

NO
MA 04

YES

MATCH RETURN

3-241
MA 43

TAPE SIGN POS

YES → MA 04 NO MATCH

NO

MA 46

MA 45

TAPE SIGN POS

NO → MA 04 NO MATCH

YES

MA 46

NO MATCH

MA 04

TAPE SIGN HIGH SIGN

MA 04

RETURN MATCH

MA 64

TAPE EXPONENT HIGH EXP.

MA 63

MA 48
MULTIPLY IQNDEX BY 15

ADD BASE ADDRESS OF IQUES

ADD MAQI → MAFW ADDRESS OF INDEX START ADDRESS

ADD 3 TO MAFW ← MAEM ADDRESS OF INDEX+1

MULTIPLY IQUES BY 18

ADD BASE ADDRESS - 2 → MAWI

ADD CONTENT OF MAFW → MANS

MA 81 YES QUESTION MAX NO MA 82

3-250
MA 81

ADD NO. OF
OF QUESTION TO
MAW1 \rightarrow MALM

MA 80

NEXT
QUESTION
ZERO

YES

NO

ADD MAW1
TO MALM

ADD 1 TO
LIMIT

END
OF IANS

NO

MA 81

YES

MA 82

3-251
MA90

STORE BASE OF BUFFERS

PICK CHARACTER FROM BUFFER1

SHIFT 8 BITS

PICK CHARACTER FROM BUFFER2

SHIFT 8 BITS

SUBTRACT CHARACTER2 FROM CHARACTER1

3-252
M4U00

PICK UP FULL WORD IN MANTISSA

WORK WITH UPPER OR LOWER

UPPER

LOWER

MASK OUT ONLY LOWER PART OF WORD IN REGA

MASK OUT ONLY UPPER

ADD 1 TO MANTISSA ADDRESS

RIGHT JUSTIFY IN REGA

TURN SW TO UPPER

TURN SW TO LOWER

CHARACTER TAB

NO

YES

CHARACTER CR

NO

YES

CHARACTER BLANK

NO

RETURN

3-254
3.3.18 MC00 - Match Condition

3.3.18.1 Purpose

MC00 is a subroutine whose purpose is to determine if a match exists between a user specified alphanumeric character string which constitutes the condition parameter and an entry on the tape record.

3.3.18.2 Technical Description

The routine initially checks for the default selection (5th word of Request Table equal to zero). If the default option has been selected, a 'MATCH' return is initiated; otherwise, the Bool buffer (CPBB) is cleared and each heading:answer pair on the current master record is then compared to each condition response parameter in the user's response to the question "CONDITION". If no match is found between a master record heading:answer pair and a condition response parameter, a 'NO MATCH' return is initiated. If, however, a match does occur, control is passed to TC00 to determine if the entire condition response parameter string is matched. On finding the Boolean expression true (See Section 3.3.39) control is returned to 'match' return address of the calling program. If TC00 returns a 'no-match' then the next condition response parameter is processed. This procedure is repeated until either TC00 returns a 'match' status or all conditions response parameters have been processed.
3.3.18.2.1 Calling Sequence

CALL MCOO,A,B

PARAMETER
A
B

FUNCTION
Memory location whose contents specify where to return in the calling program in case of a match.

Memory location whose contents specify where to return in the calling program in case of a non-match.

REGISTER CONTENTS UPON ENTRY CONTENTS UPON EXIT
A N/A Unpredictable
B N/A Unpredictable
X N/A Unpredictable
Overflow N/A N/A

3-256
3.3.18.3 Label Description

3.3.18.3.1 Local.

MCCT - storage location which contains the starting address of the condition table.

MCMT - storage location which contains the address to which this routine returns in case of a match.

MCNM - storage location which contains the address to which this routine returns in case of a non-match.

MCQN - storage location which contains the current tape question number being processed.

MCSA - storage location which contains the pointer to the operand buffer.

3.3.18.3.2 Global

MCSW - storage location which is set to zero each time this routine is referenced.

3.3.18.3.3 Entry Point

MC00 - primary entry point.
3.3.18.3.4 External Reference

3.3.18.3.4.1 External Labels

CPBB - starting address of the Bool buffer.

CPCT - starting address of the condition table.

CPRT - starting address of the request table.

CPTB+2 - storage location which contains the number of questions on the record being processed.

CPXD - storage location which is used as an index into the operand buffer.

3.3.18.3.4.2 External Subroutines

MAOO - match routine

TCOO - trace condition routine
MC 06

INCREMENT REQUEST TABLE POINTER BY 2

ALL DATA PROCESSED?

MC 20

DATA POINTER = 0?

INCREMENT CONDITION ADDRESS

DATA POINTER = 3?

INCREMENT CONDITION ADDRESS BY 3

INCREMENT CONDITION ADDRESS BY 2

MC 02

3-261
MC 20

ALL QUESTIONS PROCESSED?

YES → EXIT

NO

INCREMENT QUESTION NUMBER

RESET REQUEST TABLE POINTER

MC 01

MC 30

TCoo
TRACE PATH

MATCH?

YES → EXIT

NO

MC 06
3.3.19 MDOO - Match Date

3.3.19.1 Purpose
MDOO is a subroutine whose purpose is to compare the date on the current tape record to the date or range of dates in the user response to the "DATE" question.

3.3.19.2 Technical Description
The fourth word of the Request Table contains a pointer to the beginning of the date response in the Request Buffer. If the pointer is zero, implying the default condition, control is passed to the calling program with a match. Otherwise, the date on the current tape record is compared to the user requested date. The user may request a date range (10MAR70-20JUN70), in which case a master record date need only fall within the two values to constitute a match.

3.3.19.2.1 Calling Sequence
CALL MDOO

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>Unpredictable</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>Unpredictable</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>Unpredictable</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

3-263
3.3.19.2.2 General Flow Chart

MDOO

DEFAULT DATE

YES

RETURN MATCH

NO

RECORD DATE : REQUEST DATE

<

10

>

20

RANGE OF DATES SPECIFIED

NO

YES

RECORD DATE : UPPER LIMIT REQUEST DATE

<

20

>

10

3-264
RETURN NO MATCH

RETURN MATCH
3.3.19.3 Label Description

3.3.19.3.1 Local

MDCT - storage location used as a counter which causes the routine not to have to make an extra year-month match check if a match has already been found previously.

3.3.19.3.2 Global

N/A

3.3.19.3.3 Entry Points

MDOO - primary entry point

3.3.19.3.4 External References

CPRT+3 - date portion of the Request Table

CPTB+119 - that word in the Tape Input Buffer which contains the year part of the tape record date

CPTB+122 - that word in the Tape Input Buffer which contains the month part of the tape record date

CPTB+125 - that word in the Tape Input Buffer which contains the day part of the tape record date

RDDF - flag which is set if a date range has been specified
3.3.19.4 Detailed Flow Chart

MDOO

PICK UP
POINTER TO
DATE(CPRT+3)

POINTER =
0

YES

NO

DEFAULT

CLEAR
COUNTER
(MDCT)

RETURN
MATCH

COMPARE LOWER DATE LIMIT

PICK UP YEAR
FROM RECORD
(CPTB+119)

<

RECORD YR.: REQUEST YR.

>

40

= 10

20
Was range specified?

Date flag = 0

Counter (MDCT) = 0

Pick up year from record (CPTB+119)

Compare upper date limit

Record year : request year

<

50

= 30

40

3-268
7H FROM
CORD
REQUEST
UP DAY
RECORD

PICK UP MONTH FROM RECORD

RECORD
MONTH : REQUEST MONTH

<

50

>

40

PICK UP DAY FROM RECORD

RECORD
DAY : REQUEST DAY

<

≤

> 

50

MD70

40

RETURN NO MATCH

3-270
3.3.20 MWOO - Match What

3.3.20.1 Purpose

MWOO is a subroutine whose purpose is to determine if a match exists between a user specified WHAT response and an entry on the current tape record.

3.3.20.2 Technical Description

The routine initially checks for the default selection (7th word of Request Table equal to zero). If this selection has been made, a 'match' return is initiated; otherwise, the WHAT Table parameters are successively compared to a current tape record heading-answer pair until either a match is found or all WHAT parameters in the user response to the question "WHAT" have been exhausted.
3.3.20.2.1 Calling Sequence

CALL MW00,A,B,C

**PARAMETER**

**FUNCTION**

**A**
Memory location whose contents specify which master file question is being referenced.

**B**
Memory location whose contents specify where to return in the calling program in case of a match.

**C**
Memory location whose contents specify where to return in the calling program in case of a non-match.

**REGISTER**

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>Unpredictable</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>Unpredictable</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>Unpredictable</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
3.3.20.2.2 General Flow Chart

LOAD OPERAND
BUFFER ENTRY

DEFAULT?
NO
A

LOAD OPERAND
BUFFER ENTRY

IS
IT ZERO?
YES
EXIT
NO MATCH
NO

CHECK FOR
MATCH

MATCH
FOUND?
YES
EXIT
MATCH
NO

INCREMENT
WHAT TABLE
POINTER
A

EXIT
MATCH
3.3.20.3 Label Description

3.3.20.3.1 Local

MWMT - storage location which contains the return address in case of a match.

MWNM - storage location which contains the return address in case of a non-match.

MWQN - storage location which contains the current tape question number being processed.

MWSA - storage location which contains the pointer to the operand buffer.

MWXP - storage location which contains the beginning address of the WHAT TABLE.

3.3.20.3.2 Global

MWFG - storage location which contains the number of the current tape question being processed.

MWSW - storage location which is set to zero each time this routine is referenced.
3.3.20.3.3 Entry Point

MW00 - primary entry point

3.3.20.3.4 External References

3.3.20.3.4.1 External Labels

CPRT+6 - storage location which contains either a zero (default condition) or a value pointing to the operand buffer.

CPWT - beginning address of the WHAT table.

3.3.20.3.4.2 External Subroutines

MA00 - match routine
3.3.20.4 Detailed Flow Chart

MWOO

CLEAR FLAGS, SWITCHES

INITIALIZE QUESTION NUMBER

SAVE WHAT TABLE ADDRESS

SAVE REQUEST TABLE WHAT POINTER

DEFAULT? YES

NO

EXIT

MW 02
MATCH?

YES

INCREASE REQUEST TABLE POINTER BY 2

EXIT

NO

IS WHAT POINTER = 0?

YES

INCREMENT WHAT TABLE ADDRESS

MW 02

NO

IS WHAT POINTER = 3?

YES

INCREMENT WHAT TABLE ADDRESS

MW 02

NO

INCREMENT WHAT TABLE ADDRESS BY 2

MW 02
3.3.21 OMOO - Output Message

3.3.21.1 Purpose

The purpose of the Output Message Routine is to provide MDTRS with a generalized output routine which will service all output message requests regardless of the specific hardware configuration in which MDTRS is operating.

3.3.21.2 Technical Description

OMOO is designed to require a common calling sequence for output message requests. OMOO serves as an interface (or link) to the specific I/O handler which has been assigned by the Initialize System (ISOO) routine. The common arguments passed to OMOO by the calling program are converted by OMOO to the proper calling sequence required by the specific I/O handlers.

OMOO utilizes a table (OMDT) which contains the addresses of 'set-up' processing to be done for each particular I/O handler. The device code (OMDC) is used to index into the OMDT table to obtain the 'set-up' processing address. After the appropriate handler has been called, OMOO does not regain control until the output character count has been satisfied; however, the assigned I/O handler may return control directly to the calling program for concurrent processing while I/O is taking place. See the individual I/O handler documentation for specific output processing. The operation status word is initialized by the I/O handler to a (-1) as soon as the first character is output. Refer to the calling sequence for additional status codes to be implemented at a later date. When the output is complete...
(i.e., the number of characters to be output is satisfied), the
operation status word defined by the calling program is changed to a
positive value indicating the number of data characters output. If a
cancel request has been received during output of the buffer, OM00 will
return control to the control program at CP20 in order for a new request
to be initiated at the terminal.
3.3.21.2.1 Calling Sequence

CALL  OM00,A,B,C,D

PARAMETER FUNCTION

A  The beginning address where the message is stored. (Packed two characters per word.)

B  The address of a word containing the number of characters to be sent.

C  Always zero (0).

D  The address of an operation status word.

The status returned by the output message routine will be as follows:

Positive value = output complete. The value will be the number of characters transmitted. (May be zero if no data characters are transmitted but operation is complete.)

-1 = output initiated (first character started out)

-2 = parity error (CRT only)

-3 = 103 modem disconnected (TTY output only)

-4 = CRT not ready (CRT only)

Return addresses will be as follows:

Call +6 = previous I/O request in progress.

Call +8 = normal return. I/O initiated. User must check status word at this point.

3-282
<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Saved</td>
<td>Restored</td>
</tr>
<tr>
<td>B</td>
<td>Saved</td>
<td>Restored</td>
</tr>
<tr>
<td>X</td>
<td>Saved</td>
<td>Restored</td>
</tr>
<tr>
<td>Overflow</td>
<td>Unknown</td>
<td>Modified</td>
</tr>
</tbody>
</table>
3.3.21.2.2 General Flow Chart

OM00

OM00 BUSY? OMBU= -1?

SET OM00 BUSY -1 → OMBU

IS MESSAGE A HEADING?

YES

OM16

SET OM00 FOR BUSY RETURN CALL +6

RETURN

NO

OM06

SET UP CALL TO APPROPRIATE OUTPUT HANDLER

CALL OUTPUT HANDLER

SET OM00 NOT BUSY 0 → OMBU

NO

SAVE CHAR. COUNT IN OMHC FOR O/P HANDLER

SAVE HEADING IN OMHB FOR O/P HANDLER

CLEAR LINE COUNT (OTLC) IN O/P HANDLER

YES

B

B

C

D

3-284
CANCEL OUTPUT CHAR. RECEIVED?

YES

EXIT TO CP20

RETURN

NO
3.3.21.3 Label Description

3.3.21.3.1 Local

OMBA  Beginning address of Output Buffer (Referenced by O/P handler)
OMCC  Address of Output Character Count (Referenced by O/P handler)
OMCP  Address of Cursor Position Word if required (Referenced by O/P handler)
OMDC  Device code. Set by ISOO. (Referenced by O/P handler)
OMHB  Heading Buffer (Referenced by O/P handler)
OMHC  Heading Character Count (Referenced by O/P handler)
OMOS  Address of Operation Status Word (Referenced by O/P handler)

3.3.21.3.2 Global

None

3.3.21.3.3 Entry Points

OMOO

3.3.21.3.4 External References

OTLC  Line count for each page output - defined in output teletype (OT00)
OT00  Output handler routine for both CLINC TTY's, DOC TTY, and DOC 103 Modem.
3.3.21.4 Detailed Flow Chart

OM06

USE THE DEVICE CODE (OMDC) AS IN INDEX INTO THE ADDRESS TABLE.

OM00

IMSR
SAVE
REGISTERS

OM00
BUSY?
OMBU = -1?

SET OM00 BUSY
-1 → OMBU

SET UP FOR
NORMAL RETURN
CALL+8

IS THERE A
PAGE HEADING
TO BE SAVED?

NO

A

A1

OM06

PICK UP ADDRESS
OF SET UP
PROCESSING FOR
OUTPUT HANDLER
CALL

STORE ADDRESS
IN JUMP

B

OM16

SET UP OM00
FOR BUSY RETURN
CALL+6

IMRR
RESTORE
REGISTERS

RETURN

D
OTHER DEVICE I/O HANDLERS TO BE DEVELOPED.

OM08
SET BEGINNING ADDRESS OF BUFFER IN OT00 CALL

SET NUMBER OF CHARACTERS IN OT00 CALL

OM12
OT00 OUTPUT MESSAGE

SET OM00 NOT BUSY 0 → OMBU

CPGA CHECK FOR CANCEL O/P

IF USER HAS PRESSED THE AT SIGN (@) CPGA WILL DETECT IT AND SET THE A-REGISTER TO 0. OTHERWISE, A-REGISTER WILL BE NON-ZERO.
YES
CANCEL MESSAGE CHAR RECEIVED?
YES
EXIT TO CP20
NO
IMRR RESTORE REGISTERS
RETURN

D
OM20
SAVE HEADER CHARACTER COUNT IN OMHC FOR OTOO

OM22
MOVE HEADER FROM USER BUFFER TO OMHB BUFFER

OM24
CLEAR LINE COUNT (OTLC) IN OUTPUT HANDLER OTOO

THE TOP OF PAGE HEADING WILL BE PRINTED FROM OMHB WHEN A TOP OF PAGE CONDITION IS ENCOUNTERED BY OTOO.
3.3.22 OSOO - Operator Search

3.3.22.1 Purpose
The Operator Search Subroutine (OSOO) checks the temporary input buffer for the operators 'AND' and 'OR'.

3.3.22.2 Technical Description
OSOO checks the temporary input buffer for the operators 'AND' and 'OR'. If the operator 'AND' is found, Register-A is set to 1, and normal exit is taken. If the operator 'OR' is found, Register-A is set to zero and a normal exit is taken. If neither operator is found, the characters that were checked are packed in the request buffer; the condition count, RCCC, is incremented and an alternate exit is taken.

3.3.22.2.1 Calling Sequence
CALL OSOO, Normal, Alternate

PARAMETER
Normal Address to be taken when match is found
Alternate Address to be taken when no match is found

REGISTER CONTENTS UPON ENTRY CONTENTS UPON EXIT
A N/A 0 for 'OR'; 1 for 'AND'
B N/A Modified
X N/A Modified
Overflow N/A Same

3-290
3.3.10.2.2 General Flow Chart

START

OPERATOR 'AND'

YES

OPERATOR 'OR'

NO

STORE THE CHARACTERS CHECKED IN THE REQUEST BUFFER

ALTERNATE RETURN

OPERATOR 'OR'

NO

TERMINATE BY SPACE

YES

SET RECA 0

RETURN

SET REGB 1

NO
3.3.22.3 Label Description

3.3.22.3.1 Local

OSAR   Address of no match exit
OSBL   Table of five word to store check characters
OSNR   Address of match exit

3.3.22.3.2 Global

None

3.3.22.3.3 Entry Point

OSO0

3.3.22.3.4 External References

PK01   Entry point of subroutine to pack character into buffer
RCCC   Condition count defined in RCO0
UP01   Entry point of subroutine to unpack character from buffer
$SE    Subroutine Save parameter list
C

CHAR = 'R'

SAVE CHAR

D

OS15

UP01

CHAR = BLANK

NO

OP = AND

YES

RETURN ALTERNATE

NO

E

OS20

PK01

PACK ALL SAVED CHARACTERS

SET A-REG TO 0

SET A-REG TO 1

RETURN NORMAL

3-295
3.3.23 OT00 - Output Teletype or 103 Modem

3.3.23.1 Purpose
The purpose of OT00 is to output a buffer to the CLINC TTY's, DOC TTY, or DOC 103 Modem.

3.3.23.2 Technical Description
OT00 gains control when it is called by the generalized Output Message (OM00) Routine. OT00 uses OM00 as a link back to the calling program to obtain the address of the data buffer, the address of the character count, and the address of an operation status word. The complete buffer is transmitted before the status word is set with a completed status and control is returned to OM00. As soon as character interrupts are available on the DOC computer systems, OT00 may be modified to return control to the original calling program when output of the first character is initiated.

OT00 is presently designed to unpack the user buffer (packed three characters per word) and output a character at a time via the output character routine (OZKC). The output instructions in OZKC are set up for the output device by the Initialize System (IS00) Routine. The write buffer is sensed for a ready status by OZKC to determine when the next character may be output.

In addition to transmitting the output buffer, OT00 performs several other functions. A line count (OTLC) is maintained by OT00 for the purpose of paging the output. The line count is decremented every time
a line feed (LF) is detected in the output buffer. If the line count reaches zero, OTOO checks to see if a heading is to be printed at the top of the new page. If there is a heading requirement, OTOO outputs 5 carriage returns and line feeds, and prints the heading from the heading buffer (OMHB) where it was previously saved by OMOO. The original calling program may have OMOO save the heading for subsequent output by placing an ASCII FORM character as the first character of the heading in the output buffer. If there is not a heading requirement, five carriage returns - line feeds are output by OTOO. The original calling program may cause a top of page to occur on any line by placing an ASCII SOM character in the output buffer. Upon detecting the SOM character, OTOO performs the same processing as that performed when the line count reaches zero.

Upon completion of the output, the calling program operation status word is updated to contain the number of characters processed from the user supplied buffer. Control is returned to the OMOO routine which checks for a cancel output request, restores the user's registers, and returns to the user.

3.3.23.2.1 Calling Sequence

CALL OTOO,A,B

PARAMETER FUNCTION

A

Beginning address of output buffer.

B

Number of characters in output buffer to be processed.
<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>Modified</td>
</tr>
</tbody>
</table>
3.3.23.2.2 General Flow Chart

- **HEADING?**
  - YES → **OT50 TOP OF PAGE ROUTINE**
  - NO → **SAVE BUFFER ADDRESS AND CHARACTER COUNT**

- **GET PACKED WORD FROM USER OUTPUT BUFFER**

- **SHIFT CHARACTER IN A-REGISTER**

- **TOP OF PAGE HEADING?**
  - NO
  - YES → **SAVE BUFFER ADDRESS AND CHARACTER COUNT**

- **RETURN**
3.3.23.3 Label Description

3.3.23.3.1 Local

OTA  Save A location
OTB  Save B location
OTBA Output Buffer Address
OTCC Number of characters in output buffer
OTCP Number of characters processed
OTOF Save overflow location
OTSV Save character location
OTX  Save X location

3.3.23.3.2 Global

OTLC Line count - initialized by OM00

3.3.23.3.3 Entry Points

OT00

3.3.23.3.4 External References

OMHB Top of Page output buffer
OMHC Number of characters in user top of page output buffer (OMHB)
OMOS Address of user operation status word - defined in OM00
OZKA Subroutine used to output OMHB
OZKC Subroutine to output one character to TTY or Modem
OZKR Subroutine to output a CR, LF to TTY or Modem
$KLF Subroutine to output n number of CR, LF to TTY or Modem
3.3.23.4 Detailed Flow Chart

- OTO0

  - SAVE BUFFER ADDRESS IN OTBA

  - SAVE NUMBER OF CHARACTERS TO OUTPUT IN OTCC

  - SET NUMBER OF CHARACTERS PROCESSED TO 0, 0 → OTCP

  - OTO5

    - SET CHARACTER SWITCH USING OVFL

    - GET WORD FROM OUTPUT BUFFER (J CHARs)

  - A

  - OTO7

    - SHIFT CHAR. (LEFT-HALF) INTO A-REGISTER

OVERFLOW INDICATOR IS USED TO DETERMINE WHETHER TO OUTPUT LEFT OR RIGHT HALF OF PACKED WORD FROM BUFFER.
IS THIS BUFFER A TOP OF PAGE HEADING?

SET USER STATUS TO NUMBER OF CHARACTERS O/P

RETURN

TOP OF PAGE TO BE ISSUED?

SOM CHAR.?

LINE FEED?

RETURN

3-304
OUTPUT HEADING FROM OMHB

$KL\text{F}
\text{OUTPUT 5 CR, LF}

OT60
OZKA
\text{OUTPUT HEADING}

OZKR
\text{OUTPUT CR, LF}

LOAD A-REG.
WITH LINE COUNT (54)

OT62
STORE LINE COUNT (A-REG) IN OILC

QTRR
\text{RESTORE REGISTERS}

RETURN

3-307
OT66

$SKLF
OUTPUT 5 CR, LF

SET LINE COUNT OF 56 IN A-REGISTER

G1

OTSRS

SAVE A, B, X, OVFL REGISTERS IN OTA, OTB, OTX, OTOF

RETURN

OTRR

RESTORE A, B, X, OVFL REGISTERS FROM OTA, OTB, OTX, OTOF

RETURN
3.3.24 OZKC - Output Character to TTY

3.3.24.1 Purpose
This routine outputs one character to the ASR-35 teletype.

3.3.24.2 Technical Description
NA

3.3.24.2.1 Calling Sequence
CALL OZKC

The A-register contains the character to be output.
3.3.24.2.2 General Flow Chart

- OZKC
- OUTPUT ONE CHARACTER
- RETURN
3.3.24.3 Label Description

3.3.24.3.1 Local Labels
None

3.3.24.3.2 Global
None

3.3.24.3.3 Entry Point
Primary entry point OZKC

3.3.24.3.4 External References
None
3.3.24.4 Detailed Flow Chart

OZKC

DEVELOPMENT
READY TO WRITE

YES

NO

OUTPUT
A-REGISTER TO
ASR-35

RETURN
3.3.25 OZKB - output character to TTY

3.3.25.1 Purpose
This routine outputs the contents of the B-register to the KSR-35 each time it is called.

3.3.25.2 Technical Description
N/A

3.3.25.3 Calling Sequence
CALL OZKB

PARAMETER
None

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>undisturbed</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>ASCII character code</td>
<td>same</td>
</tr>
<tr>
<td></td>
<td>to be output</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>undisturbed</td>
<td></td>
</tr>
<tr>
<td>Overflow</td>
<td>undisturbed</td>
<td></td>
</tr>
</tbody>
</table>

3-313
3.3.25.2.2 General Flow Chart

OZKB

OUTPUT
B-REGISTER TO
KSR-35

3-314
3.3.25.3.1 Local

None

3.3.25.3.2 Global

None

3.3.25.3.3 Entry Points

OZKB - primary entry point

3.3.25.3.4 External References

OZWW - check write buffer empty (from routine OZKR)
3.3.25.4 Detailed Flow Chart

- **OZKB**
  - WRITE BUFFER EMPTY
    - OUTPUT B-REGISTER TO KSR-35
3.3.26 OZKA - Print Buffer on TTY

3.3.26.1 Purpose

This routine outputs characters to the ASR-35 teletype.

3.3.26.2 Technical Description

This routine outputs a buffer to the ASR-35 teletype. Each buffer word should contain two eight bit ASCII characters. Both the beginning location to be output and the number of characters to print are specified by the calling routine.

3.3.26.2.1 Calling Sequence

CALL OZKA,START,N

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>FUNCTION</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td></td>
<td>beginning address to be printed</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td></td>
<td>number of characters to print</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Saved</td>
<td>Restored</td>
</tr>
<tr>
<td>B</td>
<td>Saved</td>
<td>Restored</td>
</tr>
<tr>
<td>X</td>
<td>Saved</td>
<td>Restored</td>
</tr>
<tr>
<td>Overflow</td>
<td>Destroyed</td>
<td></td>
</tr>
</tbody>
</table>
3.3.26.3 Label Description

3.3.26.3.1 Local Labels

Symbol  Function
OZ$1  contains address of next word to be output

3.3.26.3.2 Global Labels

Symbol  Function
SAVE  buffer used to save volatile registers

3.3.26.3.3 Entry Point

primary entry point - OZKA

3.3.26.3.4 External References

OZKC - routine which outputs one character to the TTY
3.3.26.4 Detailed Flow Chart

OZKA

SAVE REGISTERS

PICKUP ARGUMENTS: BUFFER ADDRESS AND WORD.COUNT

A

OUTPUT A CHARACTER

ALL CHARACTERS OUT?

NO

YES

RETURN

OZKC
3.3.26.4 Detailed Flow Chart

- OZKA
  - SAVE REGISTERS
  - PICKUP ARGUMENTS: BUFFER ADDRESS AND WORD COUNT
    - A
      - OUTPUT A CHARACTER
        - OZKC
          - ALL CHARACTERS OUT?
            - NO
              - RETURN
            - YES
    - RETURN
3.3.27 OZKR - carriage return and line feed

3.3.27.1 Purpose

This routine outputs a carriage return and line feed to the KSR-35 teletype.

3.3.27.2 Technical Description

N/A

3.3.27.2.1 Calling Sequence

CALL OZKR

PARAMETER

None

REGISTER CONTENTS UPON ENTRY CONTENTS UPON EXIT

A destroyed
B undisturbed
X saved
Overflow destroyed

3-322
3.3.27.2.2 General Flow Chart

OZKR

CHECK WRITE BUFFER EMPTY

CARRIAGE RETURN LINE FEED TO TTY

DELAY FOR EXECUTPORT TIMING

3-323
3.3.27.3 Label Description

3.3.27.3.1 Local
OZNX - contents of X-register

3.3.27.3.2 Global
None

3.3.27.3.3 Entry Points
OZKR - primary entry point
OZNN - entry to output null characters to TTY
OZWW - entry to sense write buffer empty

3.3.27.3.4 External References
None
3.3.27.4 Detailed Flow Chart

```
OZKR

WRITE BUFFER EMPTY? NO

OUTPUT CARRIAGE RETURN

WRITE BUFFER EMPTY? NO

OUTPUT LINE FEED

WRITE BUFFER EMPTY? NO

OUTPUT 4 NULL CHARACTERS
```

3-325
3.3.28 PFOO - power fail - restart

3.3.28.1 Purpose
Once a power failure occurs and power is restored, this routine will initiate the action for the restart of the retrieval system.

3.3.28.2 Technical Description
When a power failure occurs, the program being executed is terminated and prevailing program conditions are lost. Once power is restored, this routine notifies the operator that the system has gone off-line and must be reinstated. The teletype bell is rung several times and an appropriate message is printed.

3.3.28.2.1 Calling Sequence

CALL PFOO

PARAMETER
None

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>lost</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>lost</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>lost</td>
<td></td>
</tr>
<tr>
<td>Overflow</td>
<td>lost</td>
<td></td>
</tr>
</tbody>
</table>
3.3.28.3 Label Description

3.3.28.3.1 Local

PFSX     contents of X-register.

3.3.28.3.2 Global

HMBU     mag tape handler busy indicator
IMBU     input message busy indicator
ISN6     count of words to be printed
ISQ6     address of message buffer to be printed
ISTU     mag tape unit to be readied
OMBU     output message busy indicator
OMHC     output message heading indicator
TAPU     output message heading indicator

3.3.28.3.3 Entry Points

location 040 is power fail entry; PF00 - primary entry point for restart

3.3.28.3.4 External References

CP00     control program entry
CP20     end-action in CP00
ERO0     error message output routine
IS90     output to teletype routine
3.3.28.4 Detailed Flow Chart

PFOO

PF 04

MAG TAPE READY?

NO

YES

POWER FAIL MESSAGE

RESTART REQUEST MESSAGE

CP20

OUTPUT TAPE NOT READY MESSAGE

15 TIMES?

IS90

NO

YES

111

PF 04

3-329
3.3.29 PK00 - Pack Character

3.3.29.1 Purpose

PK00 is a subroutine whose purpose is to accept ASCII characters one at a time and pack them into a buffer two characters per word in order to conserve computer memory.

3.3.29.2 Technical Description

PK00 has two entry points, PK00 and PK01. The first time the routine is called, control must be passed to PK00 with the first character to be packed in the A-register and the starting address of the destination buffer in the B-register. For all subsequent calls, control must be passed to PK01 with the data character in the A-register. The data will be packed two characters per word starting in the buffer address specified in the first call.

3.3.29.2.1 Calling Sequence

CALL PK00 (for initial call)
CALL PK01 (for all subsequent calls)
### Calling Sequence (Continued)

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Character to store</td>
<td>Restored to original value</td>
</tr>
<tr>
<td>B</td>
<td>Destination Buffer address (initial call only)</td>
<td>Restored to original value</td>
</tr>
<tr>
<td>X</td>
<td>Saved upon entry</td>
<td>Restored to original value</td>
</tr>
</tbody>
</table>

Overflow: N/A

3-331
3.3.29.2.2 General Flow Chart

PK00

SAVE DESTINATION BUFFER ADDRESS

PK01

ONE CHARACTER ALREADY STORED IN WORD?

YES

MERGE CHARACTER INTO ONE WORD

EXIT

NO

STORE CHARACTER IN UPPER HALF OF WORD
3.3.29.3 Label Description

3.3.29.3.1 Local

PKSA - temporary storage location for the contents of the A-register.

PKSB - temporary storage location for the contents of the B-register.

PKSX - temporary storage location for the contents of the X-register.

PKXX - storage location for the starting address of the destination buffer.

3.3.29.3.2 Global

PKBX - starting address of the destination buffer

PKIX - index into the destination buffer

PKSW - switch which controls into which half of a data word a character is to be put.

3.3.29.3.3 Entry Points

PK00 - entry point for the first call to this routine.

PK01 - entry point for all subsequent calls to this routine.

3.3.29.3.4 External References

None
3.3.29.4 Detailed Flow Chart

PK00

CLEAR COUNTERS, SWITCHES

SAVE DESTINATION BUFFER ADDRESS

PK01

SAVE REGISTERS

CHARACTER SWITCH SET?

NO

YES

MERGE CHARACTERS, STORE

INCREMENT BUFFER INDEX

CLEAR CHARACTER SWITCH

LEFT JUSTIFY DATA CHARACTER

STORE CHARACTER

SET CHARACTER SWITCH

RESTORE REGISTERS

EXIT

3-374
3.3.30 RAOO - Request Action

3.3.30.1 Purpose

RAOO is a subroutine whose purpose is to check the requested ACTION for List, Copy, Count, Tabulate or Analyze, and sets the Request Table entry accordingly.

3.3.30.2 Technical Description

RAOO validates the response from the user to the "ACTION" query. RAOO checks to see if less than four characters were input to the temporary input buffer. If less than four characters have been input, an error message, 'INVALID ACTION', is displayed, the question number (to permit a reissuance of "ACTION" query) is decremented, and control is returned to the calling routine (RQOO).

After the character count check is complete, the 3rd and 4th characters of the response are compared with the entries in the valid response table (RAPR). If a match is not found, the error message "INVALID ACTION" is displayed. If a match is found, an index equal to the relative position of the match entry in RAPR is placed in the sixth word of the Request Table for subsequent processing and control returned to CPOO.
### Calling Sequence

CALL RAOO

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>Same as Entry</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>Same as Entry</td>
</tr>
</tbody>
</table>
RA00

CHAR. INPUT LESS 4
YES 1
NO

ZERO INDEX

TEMP BUFFER REQUEST? TABLE-ACTION
YES
NO

INDEX EQUAL 4
YES 1
NO

ADD 1 TO INDEX

SET REQUEST TABLE TO INDEX
RETURN

DISPLAY INVALID ACTION
RETURN

3-337
3.3.30.3 Label Description

3.3.30.3.1 Local

RAPR is a five word table containing the following:

<table>
<thead>
<tr>
<th>Word</th>
<th>Content</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>'ST'</td>
<td>LIST</td>
</tr>
<tr>
<td>1</td>
<td>'PY'</td>
<td>COPY</td>
</tr>
<tr>
<td>2</td>
<td>'UN'</td>
<td>COUNT</td>
</tr>
<tr>
<td>3</td>
<td>'BU'</td>
<td>TABULATE</td>
</tr>
<tr>
<td>4</td>
<td>'AL'</td>
<td>ANALYZE</td>
</tr>
</tbody>
</table>

3.3.30.3.2 Global

None

3.3.30.3.3 Entry Point

RA00

3.3.30.3.4 External References

CPRT Request Table defined in CP00. The 6 word will be set as follows:

<table>
<thead>
<tr>
<th>Word</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LIST</td>
</tr>
<tr>
<td>1</td>
<td>COPY</td>
</tr>
<tr>
<td>2</td>
<td>COUNT</td>
</tr>
<tr>
<td>3</td>
<td>TABULATE</td>
</tr>
<tr>
<td>4</td>
<td>ANALYZE</td>
</tr>
</tbody>
</table>

CPSW Status word found in CP00 containing number of characters input by user
3.3.30.3.4 External References (Continued)

ERQ00  Subroutine to output the error message

RQQN  Request question number defined in RQ00. On entry value, will be 5 and is changed to 4 only when error occurs.

RQTb  Request temporary buffer defined in RQ00

RQ37  Mask 7 bits of request defined in RQ00
3.3.31 RCOO - Request Condition/What

3.3.31.1 Purpose
The purpose of this subroutine is to process the user's response to CONDITION and WHAT, and set the appropriate flags and pointers.

3.3.31.2 Technical Description
There are five legal parameters which can be input as responses to the CONDITION (WHAT) question. They are (1) left parenthesis, (2) right parenthesis, (3) operator (AND or OR), (4) operand, and (5) terminal. This subroutine has two major functions with respect to those parameters. First, it is to identify each parameter, passing on this information to BLOO. Second, it is to store all operands in the Request Buffer. Associated with these two functions is the setting of flags and pointers and a limited amount of error checking.

To RCOO the user response is merely a string of characters packed two per word in the Temporary Input Buffer. A carriage return character is placed at the end to signify the end of the data. Between the beginning of the Buffer and this carriage return can be a complicated organization of the five legal parameters, making up a network of Boolean strings with up to ten operands.

There are certain rules that the parameters must follow:
(1) left parenthesis can follow operators or another left parenthesis.
(2) right parenthesis can follow operands or another right parenthesis.
(3) operators must be preceded by and followed by a blank
(4) operands may be one of four forms:
   a. heading alone
   b. heading plus alpha answer
   c. heading plus numeric answer

When an operand is placed in the Request Buffer, several pointers and
flags must be set based on the following:
(1) The first word of the next available two word set in the Operand
    Buffer must be set to the beginning location of the operand in the
    Request Buffer.
(2) The next available location in the Condition (What) Table is a flag
    that describes the form of the operand. If the operand is of the form:
    a. heading alone, the flag is zero
    b. heading plus alpha answer, the flag is one and the location
       following the flag is an address which is the beginning location
       in the Request Buffer of the answer portion of the operand.
    c. heading plus numeric answer, the flag is two and the location
       following the flag is an address which is the beginning location
       in the Request Buffer of the answer portion of the operand.
    d. heading plus a range of numeric answer, the flag is three and
       the next two locations following the flag contain addresses
       which are the beginning locations in the Request Buffer of the
       lower limit and upper limit, respectively, of the range of
       numeric answer.
In the case when the WHAT response is being processed, there are a couple of special considerations. First, if the response is "ID ONLY" the ID flag, LIID, for L100 routine is set. Secondly, if the ACTION response is Tabulate or Analyze, the number of operands is limited, based on the device to which the printed data is to be output. A check must be made with each parameter to verify that the maximum number of operands has not been exceeded. In addition, for Analyze, only numeric data can be specified in the What response.

3.3.31.2.1 Calling Sequence

CALL RCOO

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

CALL OSO0

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

3-343
3.3.31.2.2 General Flow Chart

1. **Initialize**
2. **Pick up and store heading**
3. **Set appropriate flag**
4. **End of response**
   - **Yes**: Go to step 6
   - **No**: Go to step 5
5. **Operator**
   - **Yes**: Go to step 7
   - **No**: Go to step 8
6. **End of response**
   - **Yes**: Go to step 9
   - **No**: Go to step 8
7. **PICK UP AND STORE ANSWER**
8. **PICK UP AND STORE ANSWER**
9. **PICK UP AND STORE ANSWER**
B

SAVE PARAMETER

A

C

SET APPROPRIATE FLAGS AND POINTERS

RETURN
3.3.1.3 Label Description

3.3.1.3.1 Local

RCBL (40_B) blank
RCCC character counter. Counts the number of characters processed so far in heading and/or answer portion of operand.
RCCH save location for character currently being processed.
RCCL (72_B) colon
RCCR (15_B) carriage return
RCCS colon switch. Flag that indicates a colon has been input and an answer must follow.
RCDT (10,5,5,5,5,2,2,2,2,9) device table. The maximum number of operands that can be analyzed or tabulated for each output device.
RCID (445048, 475168, 461318) "ID ONLY"
RCLP (50_B) left parenthesis
RCPA maximum number of operand parameters for current output device
RCPC location used to verify that an equal number of left parentheses and right parentheses are present. Incremented for left paren; decremented for right paren; zero at end of response if equal.
RCPM parameter code passed off to BLOO indicating the type of parameter currently being processed.
RCFX actual number of operand parameters.
RCRP (51_B) right parenthesis
RCSX save location for Request Buffer Pointer
3.3.31.3.1 Local (Continued)

RCTM (1778) terminal character in Request Buffer

RCXX condition or What table pointer. Contains address of next available location or set of locations for flags and pointers associated with an operand.

3.3.31.3.2 Global

None

3.3.31.3.3 Entry Points

RCOO

3.3.31.3.4 External References

ELGO routine that builds the tree from the Boolean strings in the CONDITION response

CMOO routine that compares two sets of characters and returns equal to, greater than, or less than.

CPBB Bool Buffer

CPCT Condition Table

CPOB Operand Buffer

CFRB Request Buffer

CPRT Request Table

CPSW Status word from input. Equals number of characters input.

CFWT What Table

CFXB Bool Buffer pointer

CPXC Condition Table pointer
3.3.3.1.3.4 External References (Continued)

CPXO Operand Buffer pointer.
CPXR Request Buffer pointer
CPXW What Table pointer
EROO routine used to output a precoded error message to the primary output device
FILL routine to store a character or number in consecutive locations
LIID ID only flag. Indicates that the WHAT response was "ID ONLY."
OMDC output device code
OSBL table containing the characters stored in the Request Buffer by OSOO
OSOO routine that searches the input stream for an operator ("AND or "OR")
PKBX buffer address for PKOO
PKIX index used with PKOO
PKSW switch indicating into which half of the word the next character is to be packed
PKOO routine that packs characters two per word
RQQN request question number
RQTB Temporary Input Buffer
SI00 routine to clear stacks used by BLOO
UPBX buffer address for UPOO
UPIX index used with UPOO
UPSW switch indicating which half of the word the next character is to be unpacked from
UPOO routine to unpack characters from a buffer
3.3.31.4 Detailed Flow Chart

RCOO

CLEAR PAREN CTR AND COLON SWITCH

QUESTION = WHAT

YES

STORE OP BFR PTR IN 7TH LOCATION OF REQUEST TABLE

CLEAR OP BUFFER

CLEAR CONDITION TABLE

CLEAR BOOL BUFFER

INITIALIZE BOOL BUFFER PTR

INITIALIZE RCXX TO 1ST LOCATION OF CONDITION TABLE

A

NO

STORE OP BFR PTR IN 5TH LOCATION OF REQUEST TABLE

CLEAR WHAT TABLE

CLEAR ID FLAG

INITIALIZE RCXX TO 1ST LOCATION OF WHAT TABLE
STORE CR AS TERMINAL CHAR.
IN TEMP. INPUT BUFFER

S100
INITIALIZE STACKS

INITIALIZE UNPACK W/O UNPACKING 1ST CHARACTER

INITIALIZE PACK W/O PACKING 1ST CHAR. LOCATION

QUESTION COND.

YES

NO

A1

A2

3-350
A1

ACTION TAB OR ANALYZE

YES

SET MAX. NUMBER OF PARAMETERS BASED ON PRIMARY OUT DEVICE

A3

A2

SET MAX. NUMBER OF PARAMETERS AT 10

A3

RC09
INCREASE PARAMETER COUNT (RCPX)

PARAMETER COUNT > MAX. PARAMETERS

YES

SAVE REQUEST BUFFER PTR (CPXR)

CLEAR CHARACTER COUNTER

B

ER00

13

Y

A3

RC10

A4

RC12
SET HEADING
FLAG IN COND.
(WHAT) TABLE

SET PARA =
4

RC15
C

UP01
UNPACK
CHARACTER

SAVE CHARACTER

CHAR
= '(('

YES

NO

CHAR-CTR.
= 0

NO,

YES

INCREMENT
PAREN CTR
(RCPC)

SET PARA
= 1

BLO0
BOOL (PARA)

C

3-353
HEADING

Flowchart:

1. D: RC20
   - CHAR. = '1':
     - NO: D1
     - YES: D2
   - CHAR. = COLUMN
     - NO: H
     - YES: K3

2. D1: RC25
   - CHAR. = COLUMN
     - NO: E
     - YES: SET COLON SWITCH (RCCS) TO 1

INPUT COMPLETE?

E

CHAR. = BLANK

YES

NO

F

CHAR. = CARRIAGE RETURN

YES

NO

W

PK01

STORE CHAR.

INCREMENT CHARACTER COUNTER

C
CHAR. IN A-REGISTER
UPON RETURN FROM
OSOO

CHAR. = '('

CHAR. = '

CHAR. = ')'

3-356
HEADING
NO ANSWER

INCREDNT
CONDITION (WHAT)
TABLE PTR.

FIRST
TABLE ENTRY
= 0

YES

NO

K

INCREmnt
TABLE PRINTER

FIRST
TBL ENTRY
= 1

YES

NO

K1

FIRST
TBL ENTRY
= 2

YES

NO

INCREDNT
TBL PTR

K

3-358
K1  RC40

QUESTION = "CONDITION"  YES  K

ACTION ANALYZE  YES

NO

K  RC45

INCREMENT OP BUFFER PRINTER

BLOO  BOOL (PARA)

RC70

INCREMENT OP BUFFER PTR (CPXO)

K2

ER00  7

Y
ALL PARAMETERS

K2

SAVED CHAR. = '(

NO

SAVED CHAR. = ')

NO

SAVED CHAR. = CARRIAGE RETURN

NO

SAVED CHAR. = OPERATOR

NO

RC70

A3

P1

K3

X

SET PARA = 3

M

3-360
RC85

RC80

RC70

SET NUMERIC FLAG IN CONDITION (WHAT) TABLE

STORE REQUEST BFR PTR IN NEXT AVAILABLE LOCATION OF CONDITION (WHAT) TABLE

UP01

UNPACK A CHARACTER

SAVE CHARACTER

CHAR = '('

YES

NO

CHAR = ')'

YES

NO

Q

P1

P2
Q  RC90

CHAR = BLANK

YES

S

OS00
SEARCH FOR OPERATOR

OP ?

YES

G

NO

CHAR = '('

YES

P1

NO

CHAR = ')'

YES

H

NO

NUM OR RANGE FLAG SET

NO

Q

SEARCH OSBL (FROM OS00) FOR NON DIGIT

R

3-365
T
RCB5

CHAR = ' ' NO YES

V

R1

T2

NO

SET ALPHA FLAG IN CONDITION (WHAT) TABLE

U

PK01
PACK A CHARACTER

INCREMENT CHARCTR (RCCC)

P

3-369
ANSWER

SET RANGE FLAG IN COND (WHAT) TABLE

STORE CPXR IN COND (WHAT) TBL AT [CPXG OR CPXM] +2

P
Y2  RCE5

RESET OPERAND
BUFFER PTR
(CPX0)

CLEAR REQUEST
BUFFER CPR3

DECREMENT
QUESTION NO.
RQQN

RETURN

3-375
1. INITIALIZE STACK A POINTER (BLXA)
2. INITIALIZE STACK B POINTER (BLXB)
3. FILL ZERO STACK A
4. FILL ZERO STACK B
5. RETURN
3.3.32 RDOO - Request Date

3.3.32.1 Purpose
RDOO is a subroutine whose purpose is to process the user response for the date portion of the user request criteria.

3.3.32.2 Technical Description
After the "DATE" question is output, the user inputs his response to the question. The answer is moved character by character into a Temporary Input Buffer. RDOO first picks up the day representation from the Temporary Input Buffer. If the day portion of the response is not numeric, less than one, or greater than thirty-one, it is ruled invalid, an error message is sent to the user and control is passed to the calling program. If the day is valid, it is packed into the Request Buffer. The same procedure is followed for the month and year entries. Valid entries are stored in the Request Buffer and invalid entries cause an exit with an appropriate error message. Note, however, that the month is stored in binary and not alphabetic form, i.e., January is stored as 01, February is 02, etc. The only exception to this rule are the months October, November, and December, which are represented as 13, 14, and 15 which is necessary for MEDATA compatibility. If a date range is requested (for example, 05Jan71-10Mar71) both dates are processed in the same fashion and a flag pointing to the beginning address of the second date in the Request Buffer is set.
3.3.32.2.1 Calling Sequence

CALL RDOO

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS</th>
<th>RTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>Overflow</td>
<td></td>
<td>N/A</td>
</tr>
</tbody>
</table>
3.3.32.2.2 General Flow Chart

RDOO

INITIALIZE
FLAGS
COUNTERS

A

UNPACK
CHARACTER

SPACE?

YES

A

NO

DAY
≥ 1 OR ≤ 31?

YES

MONTH
VALID?

YES

YEAR
= 6 OR 7?

YES

PACK IN
REQUEST
BUFFER

B

NO

ERR00
ERROR
MESSAGE

EXIT.

3-379
DATE RANGE?

SET DATE RANGE FLAG

SAVE REQUEST BUFFER POINTER

ALL OF DATA?

EXIT
3.3.32.3 Label Description

3.3.32.3.1 Local

RDCT - storage location used to keep count of how many data characters have been processed.

RDDF - storage location which is set (incremented) upon detecting a range of dates as a user response.

RDMK - (value 177) storage location containing the sentinel character which follows the data.

RDMT - starting address of the twelve word date table. This table is used during the month validation processing.

3.3.32.3.2 Global

N/A
3.3.32.3.3 Entry Points
RDOO - primary entry point

3.3.32.3.4 External References

3.3.32.3.4.1 External Labels
CPRT - starting address of the Request Table

CPSW - storage location containing the number of data characters input

CPXR - storage location for the request buffer pointer

RQQN - storage location for the question number index

RQTB - starting address of the Temporary Input Buffer

3.3.32.3.4.2 External Subroutines
EROO - output error message routine

PKOO - pack character routine

UPOO - unpack character routine
3.3.32.3.3 Entry Points
RDO0 - primary entry point

3.3.32.3.4 External References

3.3.32.3.4.1 External Labels
CPRT - starting address of the Request Table
CPSW - storage location containing the number of data characters input
CPXR - storage location for the request buffer pointer
RQQN - storage location for the question number index
RQTB - starting address of the Temporary Input Buffer

3.3.32.3.4.2 External Subroutines
ER00 - output error message routine
PK00 - pack character routine
UP00 - unpack character routine
3.3.32.4 Detailed Flow Chart

1. RDOO
   - INITIALIZE
     - FLAGS, COUNTERS

2. UPOO
   - GET CHARACTER
     - OF DATE

3. RD70
   - CHECK FOR
     - SPACE

4. PK00
   - STORE
     - CHARACTER

5. RD02

6. RD60
   - VALIDATE
     - CHARACTER

7. UP01
   - GET NEXT
     - CHARACTER

8. RD70
   - CHECK FOR
     - SPACE

9. RD60
   - VALIDATE
     - CHARACTER

10. A
A

IS DAY < 1?

NO

IS DAY > 31?

NO

PK01
STORE CHARACTER

UP01
GET NEXT CHARACTER

RD70
CHECK FOR SPACE

UP01
GET NEXT CHARACTER

RD70
CHECK FOR SPACE

PK01
STORE ZERO

B
To ADD 3 to BINARY MONTH COUNT if MONTH 10-12.

PK01
STORE MONTH

UP01
GET NEXT CHARACTER

RD70
CHECK FOR SPACE

YEAR VALID?

YES

NO

RD 50

ADD 3 TO BINARY MONTH COUNT

NO

RD 18
LOAD SENTINEL CHARACTER

PK01 STORE IT

UPDATE NEXT ANSWER ADDRESS

EXIT

RD 30

SET DATE FLAG

UP01 GET NEXT CHARACTER

RD70 CHECK FOR SPACE

PK01 STORE CHARACTER

RD 02
3.3.33 RMOO - Record Match

3.3.33.1 Purpose
RMOO is a subroutine whose purpose is to determine if the current master record matches the user selection criteria of the retrieval request.

3.3.33.2 Technical Description
Each response in the selection criteria of the retrieval request must be matched before a record is accepted for further processing by other modules of the system. If a match does not occur on any one of the responses, the tape record is rejected. If, however, a match is found for all five responses, the record is accepted and control is transferred to the Control Program (CPOO). RMOO performs the match test in the following order: SS NO, RECORD, TYPE, DATE, CONDITION. The results of the match are passed to the Control Program via the X-register. If a match is found, a one is returned in the X-register; if the ID portion of the request (SS NO, RECORD, TYPE, DATE) is less than the corresponding entry in the current master record, a two is returned; otherwise, a zero is returned.
### 3.3.3.2.1 Calling Sequence

**CALL RM00**

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Saved upon entry</td>
<td>Restored upon exit</td>
</tr>
<tr>
<td>B</td>
<td>Saved upon entry</td>
<td>Restored upon exit</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>Status of the compare</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

3-393
3.3.18.2.2 General Flow Chart

1. **RM00**
   - **SS NO. MATCH?**
     - **NO**
     - **YES**
       - **RECORD MATCH?**
         - **NO**
         - **YES**
           - **TYPE MATCH?**
             - **NO**
             - **YES**
               - **MD00 MATCH DATE**
                 - **MATCH?**
                   - **NO**
                   - **YES**
                     - **MC00 MATCH CONDITION**
                       - **MATCH?**
                         - **NO**
                         - **YES**
                           - **EXIT NO MATCH**
                             - **EXIT MATCH**

3.3.3.3 Label Description

3.3.3.3.1 Local

RMBF - a 13 word buffer used for temporary storage for each request parameter as it is being processed.

RMCT - storage location used to save the starting address of each request parameter as it is being processed.

RMFG - flag that is set if request data has been found.

RMKA - temporary starting location for the contents of the A-register.

RMKX - temporary starting location for the contents of the X-register.

RMPT - counter which signifies how many characters have been processed for this request parameter.

RMSA - temporary starting location for the contents of the B-register.

RMSB - temporary starting location for the contents of the B-register.

3.3.3.3.2 Global
N/A

3.3.3.3.3 Entry Point

RM00 - primary entry point
3.3.3.3.4 External References

3.3.3.3.4.1 External Labels
CPRB - beginning address of the Request Buffer

CPRT - storage location which contains the first word of the request SS NO

CPTB - beginning address of the Tape Input Buffer

3.3.3.3.4.2 External Subroutines
CMOO - Compare two values

MCOO - Match condition

MDOO - Match date

PKOO - Pack character

UPOO - Unpack character
3.3.3 Detailed Flow Chart

RM00

SAVE REGISTERS

SS NO DEFAULT? YES

NO

INVALID SS NO. YES

NO

INITIALIZE COUNTERS, ADDRESSES

CM00 COMPARE SS NO.

SS NO. MATCH? YES

NO

RM 10

RM 50

RM 10

RM 50
INCREMENT CHARACTER COUNT

ALL DATA PROCESSED?

YES

LOAD_SPACE

PK01 STORE IT

RESTORE REGISTERS

EXIT
3.3.34 RQOO Request

3.3.34.1 Purpose
RQOO, the Request Subroutine, outputs all the questions to be answered and accepts the answer from the user. Each response is edited for line error, abort, and input complete codes. If the default parameter has been entered, the subroutine sets the default flag and proceeds to the next question. If the default parameter is not the response, the subroutine sends control to the subroutine corresponding to question number to edit the answer. When all questions have been answered, the subroutine returns control to RPOO, the Control Program.

3.3.34.2 Technical Description
The subroutine saves the A, B and X-registers, and the question number pointer is initialized to zero. The following pointers are initialized as follows:

<table>
<thead>
<tr>
<th>POINTER</th>
<th>INITIALIZED TO</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPXB</td>
<td>CPBB, address of first word of BOOL Buffer</td>
</tr>
<tr>
<td>CPXO</td>
<td>CPOB, address of first word of Operand Buffer</td>
</tr>
<tr>
<td>CPXR</td>
<td>CPRB, address of first word of Request Buffer</td>
</tr>
<tr>
<td>CPXC</td>
<td>CPCT, address of first word of Condition Table</td>
</tr>
<tr>
<td>CPXW</td>
<td>CPWT, address of first word of What Table</td>
</tr>
</tbody>
</table>
3.3.34.2 Technical Description (Continued)

The subroutine displays the question based on question number by accessing the address of the appropriate message from RQAM. The messages are listed below:

<table>
<thead>
<tr>
<th>QUESTION NUMBER</th>
<th>MESSAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>'SS-NO'</td>
</tr>
<tr>
<td>1</td>
<td>'RECORD'</td>
</tr>
<tr>
<td>2</td>
<td>'TYPE'</td>
</tr>
<tr>
<td>3</td>
<td>'CONDITION'</td>
</tr>
<tr>
<td>4</td>
<td>'DATE'</td>
</tr>
<tr>
<td>5</td>
<td>'ACTION'</td>
</tr>
<tr>
<td>6</td>
<td>'WHAT'</td>
</tr>
</tbody>
</table>

The Input Message Routine (IMO00) is called to accept the response from the user. The status word set by IMO00 is checked to determine the action to be taken.

<table>
<thead>
<tr>
<th>STATUS WORD</th>
<th>ACTION TO BE TAKEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;0</td>
<td>Input complete - process data</td>
</tr>
<tr>
<td>-1</td>
<td>Input busy - wait for completion</td>
</tr>
<tr>
<td>-2</td>
<td>Line Error - repeat question</td>
</tr>
<tr>
<td>-3</td>
<td>Abort - Repeat SS-NO.</td>
</tr>
<tr>
<td>0</td>
<td>Default value - set Request Table entry to zero</td>
</tr>
</tbody>
</table>
3.3.34.2 Technical Description (Continued)

When the status word is greater than zero, the response is compared against the corresponding entry in the default table (RQDT). See Table 1. If the default condition is true, zero is set in the word of the Request Table corresponding to question number, and the next question is processed.

<table>
<thead>
<tr>
<th>QUESTION</th>
<th>DEFAULT CONDITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>'ALL'</td>
</tr>
<tr>
<td>1</td>
<td>'ALL'</td>
</tr>
<tr>
<td>2</td>
<td>'ALL'</td>
</tr>
<tr>
<td>3</td>
<td>'ALL'</td>
</tr>
<tr>
<td>4</td>
<td>'NONE'</td>
</tr>
<tr>
<td>5</td>
<td>'LIST'</td>
</tr>
<tr>
<td>6</td>
<td>'ALL'</td>
</tr>
</tbody>
</table>

Table 1

If the response is not the default condition, the address of appropriate subroutine needed to process the answer is retrieved from end-action table (RQET) and the subroutine called. See Table 2. On return RQOO determines if all questions have been answered. If they have, the registers are restored and control is returned to CP00. If they have not, the next question is displayed.
Table 2

<table>
<thead>
<tr>
<th>QUESTION NUMBER</th>
<th>SUBROUTINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RSOO</td>
</tr>
<tr>
<td>1</td>
<td>RROO</td>
</tr>
<tr>
<td>2</td>
<td>RROO</td>
</tr>
<tr>
<td>3</td>
<td>RDOO</td>
</tr>
<tr>
<td>4</td>
<td>RCOO</td>
</tr>
<tr>
<td>5</td>
<td>RAOO</td>
</tr>
<tr>
<td>6</td>
<td>RCOO</td>
</tr>
</tbody>
</table>

3.3.34.2.1 Calling Sequence

CALL RQOO

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>Same</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>Same</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>Same</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>Modified</td>
</tr>
</tbody>
</table>
33.34.2.2 General Flow Chart

RQ00

1

ZERO QUESTION NUMBER

SET BUFFER ADDRESS

2

DISPLAY QUESTION

ACCEPT INPUT

LINE ERROR?

YES

NO

ABORT?

YES

NO

DEFAULT?

YES

NO

ZERO ENTRY IN REQUEST TABLE

3-410
EDIT THE REQUEST

ADD 1 TO QUESTION NUMBER

END OF QUESTION?

YES

NO

2

3

4
3.3.34.3 Label Description

3.3.34.3.1 Local

RQAC 'ACTION' Message

RQAM 7 word Table of Address of Message (See Section 3.3.34.2)

RQCN 7 word Table containing Length of Messages

Entry as follows:

8 SS-NO
10 RECORD
8 TYPE
8 DATE
13 CONDITION
10 ACTION
8 WHAT

RQCO 'CONDITION' Message

RQCU Max. number of 240, the max input character

RQDA 'DATE' Message

RQDT 7 word table of address of default test. See Table 1.

RQD5 'SS NO' Message

RQET 7 word Table of Address of end action. See Table 2.

RQRE 'RECORD' Message

3-412
3.3.3.4.3.1 Local (Continued)

RQSA  Temporary location to save register A on entry.
RQSB  Temporary location to save register B on entry.
RQSX  Temporary location to save Register X on entry.

RQTP  'TYPE' Message
RQWH  'WHAT' Message

RQXT  Temporary Index
RQ37  Mask to mask out two characters - 7 bits.

3.3.3.4.3.2 Global

RQQN  Question number, referenced by RSOO, RR00, RD00, RC00, and RA00.

RQTB  Temporary input buffer; referenced by following subroutines:
        RSOO, RR00, RD00, RC00, and RA00.

3.3.3.4.3.3 Entry Point

RQ00
3.3.34.3.4 External References

CPBB  Defined in Appendix C, Bool Buffer
CPCT  Defined in Appendix F, Condition Table
CPNC  Defined in CPOO, the number of characters input through IMOO
CPOB  Defined in Appendix D, Operand Buffer
CPRB  Defined in Appendix E, Request Buffer
CPSW  Defined in CPOO, Status Word for Input and Output Routine
CPWT  Defined in Appendix G, What Table
CPXB  Defined in CPOO, Pointer into Bool Buffer
CPXC  Defined in CPOO, Pointer into Condition Table
CPXO  Defined in CPOO, Pointer into Operand Buffer
CPXR  Defined in CPOO, Pointer into Request Buffer
CPXW  Defined in CPOO, Pointer into What Table
ER00  Subroutine to Output Error Message
IM00  Subroutine to accept input for User
OM00  Subroutine to Output Question
RA00  Subroutine to process 'ACTION' Response
RC00  Subroutine to process 'CONDITION' and 'WHAT' Response
RD00  Subroutine to process 'DATE' Response
RR00  Subroutine to process 'RECORD and 'TYPE' Response
RS00  Subroutine to process 'SS-NO' Response
3.3.34.4 Detailed Flow Chart

RQ00

SAVE REGA, B, X IN RQSA, RQSB, &RQSB

RQ01

ZERO QUESTION NUMBER

LOAD ADDRESS OF BOOL BUFFER IN CPXB

LOAD ADDRESS OF OP BUFFER IN CPXO

LOAD ADDRESS OF REQUEST BUFFER IN CPXR

LOAD ADDRESS OF CONDITION BUFFER IN CPXC

LOAD ADDRESS OF WHAT BUFFER IN CPXW

RQ02

RQ00

1 OF 8
3-415
RQ02

ZERO TEMP BUFFER INDEX

LOAD INDEX WITH QUESTION NO.

PICK # OF CHARACTER TO OUTPUT FROM RQLN INDEX

PICK ADDRESS OF BUFFER TO OUTPUT FROM RQAM, INDEX

RQ03
1. OM00 DISPLAY QUESTION

2. PICK UP STATUS WORD

3. STATUS POS.
   - YES: RQ 05
   - NO: RQ 04

4. STATUS -1
   - YES: RQ 04
   - NO: RQ 03
ZERO ENTRY IN REQUEST TABLE

 ACTION TAB OR ANALYZE

 EROO DISPLAY 'INVALID WHAT'

RQ 15

QUESTION WHAT

YES

RQ 17

NO

RQ 16

RQ 52

RQ 17

NO

RQ 16

YES

RQ 02

RQ00

6 OF 8

3-420
RQ 52

LAST QUESTION PROCESSED

YES

NO

ADD 1 TO QUESTION NO.

RESTORE REGISTER

RETURN

RQ 02
3.3.35 RROO - Request Record and Type

3.3.35.1 Purpose
RROO subroutine packs the user's response to the "RECORD" and "TYPE" queries into the request buffer.

3.3.35.2 Technical Description
The subroutine packs the response from "record" and "type" queries into the request buffer. If the number of characters is greater than twenty-four, an error message is output to the user, the question number is decremented by one, and control returned to RQ00.

If the number of characters is 24 or less, the next available address in the request buffer is picked up from CPXR to initialize the pack subroutine, PK00. Only non-space characters are packed into the request buffer. When the complete response has been packed in the request buffer, 1778 is stored at the end of the buffer.

The address in CPXR is stored in the second word of Request-Table for record, or in the third word of the Request-Table for type. The next available address is computed by dividing (the number of character inputs +2) by 2 and storing it in CPXP. Control is then returned to RQ00.
3.3.35.2.1 Calling Sequence

CALL RR00

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>Modified</td>
</tr>
</tbody>
</table>
3.35.2.2 General Flow Chart

RRO0

CHAR.
INPUT GREATER
 THAN 24

NO

SET CNT
TO ZERO

YES

DISPLAY
INVALID
TYPE OR
RECORD

SUBTRACT 1
FROM QUESTION
NO.

GET A
CHARACTER
FROM TEMP.
BUFFER

ADD 1 TO
COUNT

CHARACTER
BLANK

YES

PACK
CHARACTER
IN REQUEST
BUFFER

NO

A.L.
CHARACTER
PROCESSES

YES

1

RRO0

3-425
PACK 0177
IN REQUEST BUFFER

STORE START ADDRESS IN REQUEST TABLE

COMPUTE ENDING ADDRESS

1
3.3.35.3 Label Description

3.3.35.3.1 Local
None

3.3.35.3.2 Global
None

3.3.35.3.3 Entry Point
RROO

3.3.35.3.4 External References
CPRT Defined in CPOO, Request Table
CPSW Defined in CPOO, contains number of character input
CPXR Defined in CPOO, contains the next available location in the Request Buffer
ER00 Subroutine to output error message
PK00 Subroutine entry used to initialize the packing into the Request Buffer
PK01 Subroutine entry used to pack into the Request Buffer
RQQN Defined in EQOO, question number (i.e., 1 or Record, or 2 for Type.)
RQTB Defined in RQ00, temporary Input Buffer
RSMK Defined in RSO0, 7 bit mask for ASCII code.
UP00 Subroutine entry used to initialize the unpacking of the characters from the temporary Request Buffer
UP01 Subroutine entry used to unpack characters from the temporary Request Buffer
3.3.35.4 Detailed Flow Chart

RRO0

CHARACTER INPUT > 24

YES

EERO0
DISPLAY INVALID TYPE OR RECORD

SUBTRACT 1 FROM QUESTION NO.

RETURN

NO

UERO0
GET 1ST CHARACTER FROM TEMP BUFFER

RR15
SUBTRACT 1 FROM CHARACTER COUNT

PICK ADDRESS FROM CPXR THE ADDRESS OF REQUEST BUFFER

PACK 1ST CHARACTER IN REQUEST BUFFER

SUBTRACT 1 FROM CHARACTER COUNT

CHARACTER COUNT ZERO

YES

CHARACTER BLANK

YES

NO

RR 15

RR 45

RR 25

RR 16

RR 26
UP01
GET NEXT CHARACTER FROM TEMP BUFFER

CHARACTER BLANK

YES

RR 30

NO

PK01
PACK NEXT CHARACTER IN REQUEST BUFFER

SUBTRACT 1 FROM CHARACTER COUNT

CHARACTER COUNT ZERO

NO

RR 16

YES

RR 25
PK01
PACK 0177 IN REQUEST BUFFER

STORE START ADDRESS OF TYPE OR RECORD IN REQUEST-TABLE

COMPUTE ENDING ADDRESS OF TYPE OR RECORD

RETURN
SUBTRACT 1 FROM CHARACTER COUNT

CHARACTER COUNT ZERO

STORE ZERO IN REQUEST TABLE

RETURN

RR 30

RR 20

RR 45

RR 50

RR 15

RR 50

RETURN

5 OF 5
3.3.36 RSOO - Request Social Security Number

3.3.36.1 Purpose
RSOO is a subroutine whose purpose is to edit the Social Security Number and move it from the temporary input buffer to the request buffer.

3.3.36.2 Technical Description
The subroutine edits Social Security Number for 11 digits including dashes. If the Social Security does not meet this criteria, the error message 'INVALID SS-NO' is displayed, the question is decremented and control is returned to RQ00.

The Social Security Number is moved from the temporary input buffer to the beginning of the request buffer. The first word of the request table (CPRT) is set to the address of the first word of the request buffer (CPRB). The next available address, CPXR is set to the address of the 7th word of CPRB.

3.3.36.2.1 Calling Sequence
CALL RSOO

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>Same</td>
</tr>
</tbody>
</table>
3°3.36.2.2 General Flow Chart

RS00

NO
CHARACTER
INPUT: 11

YES

NO
CHARACTER
'-' 'Q' - '9'

YES
PACK SS NO.
IN REQUEST
BUFFER

NO
CHECK:
ALL SS NO.

YES
RETURN

RETURN

DISPLAY
INVALID
SS-NO.
3.3.36.3 Label Description

/ 

3.3.36.3.1 Local
None

3.3.36.3.2 Global
RSMK 7 bit mask used to mask out ASCII code. The subroutine RR00 also uses it as a mask.

3.3.36.3.3 Entry Point
RS00

3.3.36.3.4 External References
CPRB Defined in CR00, Request Buffer
CPRT Defined in CP00, Request Table
CPSW Defined in CP00, contains number of characters input
CPXR Defined in CP00, next available address in Request Buffer
ER00 Subroutine called to output error message
PK00 Subroutine entry used to initialize the pack of the Request Buffer
PK01 Subroutine entry used to pack the Request Buffer
RQTB Defined in RQ00, Temporary Input Buffer
UPO0 Subroutine entry used to initialize the unpacking of the Temporary Input Buffer
UPO1 Subroutine entry used to unpack the Temporary Input Buffer
3.3.36.4 Detailed Flow Chart

RSOO

NO
CHARACTER
INPUT: 11

YES

UPOO
GET 1ST
CHARACTER FROM
TEMP BUFFER

CHARACTER
0-9

YES

PKOO
PACK 1ST
CHARACTER IN
REQUEST
BUFFER

NO

LOAD LOOP
CONTROL WITH
10

RS 50

RS 02

RSOO
1 OF 3
RS04

FK01
PACK NEXT CHARACTER IN REQUEST BUFFER

SUBTRACT 1 FROM LOOP CONTROL

LOOP CONTROL = 0
NO → RS02
YES

STORE NEXT AVAILABLE ADDRESS IN REQUEST BUFFER IN CPXR

STORE START ADDRESS OF SS-NO OF REQ BUFFER IN THE REQ TABLE

RETURN

RS50

ER00
DISPLAY 'INVALID SS-NO'

RETURN
RS02

UP01
GET NEXT CHARACTER FROM TEMP BUFFER

CHARACTER ( - )

YES → RS04
NO

CHARACTER 0 - 9

YES → RS04
NO → RS50

RS04
3.3.37 SE00 - System Error

3.3.37.1 Purpose

The purpose of this subroutine is to output an error message to the system input/output device, the teletype.

3.3.37.2 Technical Description

Error messages are precoded and a table of pointers (SETB) can be indexed to access the beginning locations of each message buffer. The appropriate message may then be output through a call to IS90. All messages are twelve words (24 characters) in length. The calling routine passes off the number of the error message to be output which is used as the index into the pointer table.

3.3.37.2.1 Calling Sequence

CALL SE00

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>X</td>
<td>number of error message to be output</td>
<td>N/A</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

3.3.37.2.2 General Flow Chart

See Section 3.3.37.4

3-439
3.3.37.3 Label Description

3.3.37.3.1 Local

SENC - (24) the number of characters in each error message.

SETB - (SE10, SE11, SE12) table of pointers to beginning locations of each error message buffer.

SE10 - (carriage return, line feed, "PARITY ERROR", carriage return, line feed)

SE11 - (carriage return, line feed, "MODEM DISCONNECTED", carriage return, line feed)

SE12 - (carriage return, line feed, "CRT NOT READY", carriage return, line feed)

3.3.37.3.2 Global

None

3.3.37.3.3 Entry Points

SE00

3.3.37.3.4 External References

IS90 - output handler to system input/output device, the teletype.
3.3.37.4 Detailed Flow Chart

- S900
- Pick up address of error message
- IS90
- Output to teletype
- Return
3.3.38  TAOO - Tabulate and Analyze

3.3.38.1 Purpose

TAOO is a subroutine whose purpose is to tabulate or analyze those parameters specified in a Retrieval Request and then output those values in an ordered sequence.

3.3.38.2 Technical Description

TAOO initially collects all user specified WHAT headings which are to be tabulated or analyzed and outputs those in the form of a heading line. Six headings are allowed for a line being output to Teletype while only two headings are allowed per CRT line. After all headings have been output, MATCH WHAT is called to check for matches between the current tape record and the Retrieval Request WHAT parameter. If a match is found, the data is moved to a print buffer. Four types of data are possible: heading with no answer, heading with alpha answer, heading with numeric answer, and heading with a range of numeric data. If the tabulate action is requested, all numeric data is collected and stored for future use in calculating the Mean and Standard Deviation. The formulae used in calculating those are as follows:

The Mean of N numbers

\[ \text{Mean} = \frac{\sum_{i=1}^{N} X_i}{N} \]

The Standard Deviation of N numbers

\[ \text{S.D.} = \sqrt{\frac{\sum_{i=1}^{N} X_i^2 - \left( \frac{\sum_{i=1}^{N} X_i}{N} \right)^2}{N(N-1)}} \]
For each WHAT heading answer, an associated social security number and date are shown with the data. A sample analyze request is shown below. The tabulate request would be identical except for an omission of the Mean and Standard Deviation lines.

<table>
<thead>
<tr>
<th>SSN</th>
<th>Date</th>
<th>Height</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-45-6789</td>
<td>01JAN68</td>
<td>71</td>
<td>160</td>
</tr>
<tr>
<td>234-56-7890</td>
<td>10MAR68</td>
<td>68.5</td>
<td>150</td>
</tr>
<tr>
<td>345-67-8901</td>
<td>30MAR68</td>
<td>69</td>
<td>170</td>
</tr>
</tbody>
</table>

- **MEAN**: 69.5
- **SD**: 2.33
- **MEAN**: 160
- **SD**: 9.85

### 3.3.38.2.1 Calling Sequence

CALL TAA00 (for all data values to be output)

CALL TAA1 (for outputting of the Mean and Standard Deviation)

### REGISTER

<table>
<thead>
<tr>
<th></th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Saved upon entry</td>
<td>Restored upon exit</td>
</tr>
<tr>
<td>B</td>
<td>Saved upon entry</td>
<td>Restored upon exit</td>
</tr>
<tr>
<td>X</td>
<td>Saved upon entry</td>
<td>Restored upon exit</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

---

3-443
3.3.38.2.2 General Flow Chart

- TAO0
  - INITIALIZE COUNTERS, POINTERS, BUFFERS
  - LOAD ADDRESS OF FIRST HEADING
  - MOVE HEADING TO PRINT LINE
  - INCREMENT TO NEXT HEADING
  - ALL HEADINGS PROCESSED?
    - NO
    - OUTPUT HEADER LINE
    - INITIALIZE QUESTION NO.
    - A

3-444
A

MATCH FOUND?

NO
B

YES

ANALYZE REQUESTED?

NO

YES

SUMS, MATH, ETC., FOR ANALYZE

RECORD HEADING = PREVIOUS HEADING?

NO

YES

MOVE SS NO. & DATE TO PRINT LINE

OUTPUT PRINT LINE

MOVE DATA TO PRINT LINE

B

3-445
ALL QUESTIONS PROCESSED?

OUTPUT LAST DATA LINE

EXIT

TAAIL

ANALYZE REQUESTED?

BUILD MEAN

OUTPUT MEAN LINE

BUILD STANDARD DEVIATION

OUTPUT STANDARD DEVIATION LINE

EXIT
3.3.38.3 Label Description

3.3.38.3.1 Local

TAAD   a ten word table containing values by which headings are to be biased in the output buffer.
TABF   temporary storage location for the contents of the 61 word print output buffer
TABL   nine word table used to flag headings which have been processed and are ready to be output
TACR   individual data field output character count
TACW   storage location which contains the status of the I/O function being performed
TAPG   flag which is set whenever a line of data has been output
TAPU   flag which is set whenever a header line has been output
TAQN   storage location containing the number of the question being processed
TASA   temporary storage location for the contents of the A-register
TASB   temporary storage location for the contents of the B-register
TASS   an eighteen word table used to store sums of squared data values which will be used in the Standard Deviation calculations
TASW   storage location which contains the status of the I/O function being performed
TASX   temporary storage location for the contents of the X-register
TAUC   storage location which is used to keep count of the number of characters output for a particular heading. A maximum of eight characters are allowed per heading.
3.3.38.3.1 Local (Continued)

**TAXC** a nine word table used to store various integer values that are to be converted to floating point at a later time.

**TAXG** flag which is set whenever data has been found for a particular parameter.

**TAXI** an eighteen word table used to store sums of numbers which will be used in the Standard Deviation calculations.

**TAXS** a two word buffer used to store miscellaneous floating point data values.

**TAXX** storage location for the contents of the X-register.

3.3.38.3.2 Global

**TAPU** flag which is set whenever a header line has been output.

3.3.38.3.3 Entry Points

**TA00** - primary entry point.

**TAA1** - entry point to the routine which calculates and outputs the Mean and Standard Deviation.

3.3.38.3.4 External References

3.3.38.3.4.1 External Labels

**CPRT** starting address of the seven word Request Table

**CPTB** starting address of the Tape Input Buffer

**MAS1** switch which is set if any matched data has been found on the master.
3.3.38.3.4.1 External Labels (Continued)

MATC  storage location which contains the number of coded data
       characters found in an answer area
MATE  storage location which signifies where the decimal is to be
       placed in a fractional number
MATM  starting address of the buffer where the answer data is stored
MATS  storage location which contains the sign of the numeric data
       being processed (0-positive, 1-negative)
MWFG  a storage location containing the tape question number currently
       being processed
MWXP  a storage location containing the beginning address of the
       WHAT Table
PKSW  switch which controls into which half of a data word a character
       is to be put

3.3.38.3.4.2 External Subroutines

CDOO  Convert and Store Month
FAOO  Floating Point to ASCII Conversion
FLOO  ASCII to Floating Point Conversion
MABO  Find Matched Data Routine
MWOO  Match WHAT Routine
OM00  Output Header/Data Line
PK00  Pack Character
SQRT  Square Root Calculation
UF00  Unpack Character
$QK  Floating Point Add
3.3.38.3.4.2 External Subroutines (Continued)

$QL$  Floating Point Subtract
$QM$  Floating Point Multiply
$QN$  Floating Point Divide
$QS$  Integer to Floating Point Conversion
TA06

CALCULATE NUMBER OF SPACES BETWEEN HEADINGS

TA08

TA95 OUTPUT SPACES BETWEEN HEADINGS.

INCREMENT TO NEXT ENTRY IN OPERAND BUFFER

IS POINTER ZERO?

YES

TA10

NO

TA02
LOAD ANSWER TYPE FLAG

HEADING ONLY? YES -> TA 50

HEADING PLUS ANSWER RANGE? YES -> TA 51

TA 85 ANSWER TO PRINT LINE

TA 72 ANALYZE ROUTINE

TA 50
TA 85

NEW HEADER?

NO

CALCULATE BUFFER ADDRESS OF ANSWER

PK00
STORE CHARACTER

UPO0
PICK UP CHARACTER

PK01
STORE CHARACTER

TA 87

INCREMENT CHARACTER COUNT

...
TA95
SPACE BETWEEN SS NO. AND DATE

PK01
STORE DAY

CDO0
CONVERT AND STORE MONTH

PK01
STORE YEAR

UPDATE CHARACTER COUNT

EXIT
TA0

CLEAR HEADER FLAG

EXIT

TA1

CLEAR HEADER FLAG, CHARACTER COUNT

PK00
STORE CARRIAGE RETURN

PK01
STORE LINE FEED

PK00
STORE SPACE

TA95
CLEAR PRINT LINE

PK01
STORE "MEAN"

TA2
3-469
TA97
CALCULATE CHARACTER COUNT

ANY DATA TO OUTPUT?

OUTPUT COMPLETE?

OUTPUT SD

CLEAR SD TABLES

EXIT

NO

TA A9

TA A7
3.3.39 TCOO - Trace Condition

3.3.39.1 Purpose
The purpose of this routine is to trace through the tree network in the Bool Buffer and return a true status if the top of the tree is reached, thus indicating that the entire Boolean expression of the CONDITION response is true.

3.3.39.2 Technical Description
Appendices D and C explain in detail the layout of the Operand and Bool Buffers respectively. A thorough knowledge of these buffers and the purpose and function of BLOO will be very helpful in understanding the operation of TCOO.

An address (link) from the Operand Buffer is passed in the calling sequence. From the point of this address TCOO begins its tracing until one of two conditions exists: (1) the null link (-1) is reached indicating the top of the tree, or (2) a Bool Buffer nodal operation is found to be false.

Tracing is done in the following manner; as each nodal operation is found to be true, the link associated with the node is picked up and the logical conclusion of the next node is tested. This test is done in the following manner: the conclusion (left or right) pointed to by the link is set true. If the nodal operator is OR, the logical conclusion of the node is true. If the nodal operator is AND, the other conclusion (left or right) is tested. If that conclusion is true, the nodal operation
3.3.39.2 Technical Description (Continued)

is true. When the logical conclusion of the node is found to be false, the subroutine returns control to the calling routine, two locations past the normal return point. The normal return point is the location immediately following the calling sequence parameter.

3.3.39.2.1 Calling Sequence

CALL TCO0, LINK

PARAMETER  FUNCTION
LINK  The contents of LINK is a link address from the Operand Buffer.

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
3.3.39.2.2 General Flow Chart

TCOO

PICK UP LINK

NULL

SET CONCLUSION TRUE

OPERATOR = OR

OTHER CONCLUSION TRUE

RETURN INCOMPLETE PATH

RETURN COMPLETE PATH

3-475
3.3.39.3 Label Description

3.3.39.3.1 Local
None

3.3.39.3.2 Global
None

3.3.39.3.3 Entry Points
TC00

3.3.39.3.4 External References
CPBB    Bool Buffer
3.3.39.4 Detailed Flow Chart

TC00

PICK UP AND SAVE LINK

LINK NULL (-1) YES

RETURN COMPL. PATH

NO

A

DETERMINE LEFT OR RIGHT CONCLUSION

SET CONCLUSION TRUE

B
3.3.40 UPOO - Unpack Character

3.3.40.1 Purpose.

UPOO is a subroutine whose purpose is to return ASCII characters which have been packed two per word from a specified buffer into the A-register. The routine returns one character per call.

3.3.40.2 Technical Description

This routine has two entry points, UPOO and UP01. The first time that the routine is called control must be passed to UPOO and the starting address of the source buffer must be in the B-register. For all subsequent calls control must be passed to UP01. In both cases, the character will be returned in the A-register.
### 3.3.40.2.1 Calling Sequence

**CALL** UPO0 (for initial call)

**CALL** UPO1 (for all subsequent calls)

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N/A</td>
<td>Unpacked data character</td>
</tr>
<tr>
<td>B</td>
<td>Source buffer address</td>
<td>original data restored (initial call only)</td>
</tr>
<tr>
<td>X</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Overflow</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

3-480
3.3.40.3 Label Description

3.3.40.3.1 Local

UPBX - starting address of the source buffer

UPIX - index into the source buffer

UPSB - temporary storage location for the contents of the B-register

UPSW - switch which controls from which half of a data word a character is to be extracted.

UPXX - storage location initially containing the starting address of the source buffer; thereafter, its contents are incremented by the buffer index.

3.3.40.3.2 Global

None

3.3.40.3.3 Entry Points

UP00 - entry point for the first call to this routine.

UP01 - entry point for all subsequent calls to this routine.

3.3.40.3.4 External References

None
3.3.40.4 Detailed Flow Chart

UPOO

INITIALIZE SWITCHES, BUFFER INDEX

SAVE SOURCE BUFFER ADDRESS

UP01

SAVE REGISTERS

CHARACTER SWITCH SET?

YES

PICK UP LOWER CHARACTER

CLEAR CHARACTER SWITCH

INCREMENT SOURCE BUFFER INDEX

RESTORE REGISTERS

EXIT

NO

PICK UP UPPER CHARACTER

RIGHT JUSTIFY CHARACTER

SET CHARACTER SWITCH
3.3.41 $KLF - TTY carriage return, line feed

3.3.41.1 Purpose

This routine outputs to the TTY the specified number of carriage returns and line feeds.

3.3.41.2 Technical Description

N/A

3.3.41.2.1 Calling Sequence

CALL #KLF, CNT

PARAMETER

CNT

the number of desired carriage returns and line feeds to be output to the TTY.

FUNCTION

the number of desired carriage returns and line feeds to be output to the TTY.

REGISTER

<table>
<thead>
<tr>
<th>CONTENTS UPON ENTRY</th>
<th>CONTENTS UPON EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>destroyed</td>
</tr>
<tr>
<td>B</td>
<td>destroyed</td>
</tr>
<tr>
<td>X</td>
<td>destroyed</td>
</tr>
<tr>
<td>Overflow</td>
<td>destroyed</td>
</tr>
</tbody>
</table>
3.3.41.2.2 General Flow Chart

```
  $KLF

OUTPUT
CARRIAGE RETURN
AND LINE FEEDS
```

3-485
3.3.41.3.1 Local

None

3.3.41.3.2 Global

None

3.3.41.3.3 Entry Points

$KLF primary entry point

3.3.41.3.4 External References

OZKR output one carriage return and line feed to teletype.
3.3.41.4 Detailed Flow Chart

- $KLF$
- **PICKUP COUNT TO BE OUTPUT**
- **OUTPUT CR, LF**
- **ALL OUTPUT?**
- **YES**
4.0 PROGRAM UTILIZATION

The Medical Data Tape Retrieval System (MDTRS) is a user oriented information retrieval system designed to provide on-line data retrieval capabilities. The NASA CAAD UNIVAC 1108 MEDATA system creates a master tape containing records of various medical information which is used as input to MDTRS. The user makes a retrieval request by answering pre-programmed retrieval questions. MDTRS responds to the retrieval request by searching the MEDATA master tape for the data satisfying the request criteria, formatting the selected data for output, and displaying the data on the terminal.

4.1 COMPUTER OPERATOR INSTRUCTIONS

- Load the MDTRS program from either paper tape or magnetic tape.
- Turn on the Magnetic Tape Unit.
- Mount the retrieval master tape.
- If the primary retrieval device is to be the 103A modem, it must be placed in AUTO mode.
- Begin program execution at location zero (0).
- The computer will respond with the message:

  PLEASE INITIALIZE SYSTEM

  COMPUTER CODES ARE AS FOLLOWS

  DOC*A = 1
  DOC*B = 2
  CLINC = 3
4.1 COMPUTER OPERATOR INSTRUCTIONS (Continued)

DEVICE CODES ARE AS FOLLOWS

0 = NOT REQUIRED
1 = CLINC TTY#1
2 = CLINC TTY#2
3 = DOC TTY
4 = 103A MODEM
5 = CLINC CRT 73
6 = CLINC CRT 74
7 = DOC CRT 71
8 = DOC CRT 74 (REMOTE)
9 = CLINC LINE PRINTER

WHAT COMPUTER IS THIS (MUST ANSWER 1, 2, OR 3)

The operator must input the proper computer code at this time.

The computer will then respond with the message:

PRIMARY RETRIEVAL DEVICE
(MUST ANSWER 1, 2, 3, OR 4)

The operator must input the proper device code at this time. Note that only device codes 1 thru 4 are operational at the present time.

After the device code has been input, the computer will output the following message:

PLEASE MOUNT MASTER TAPE
TYPE UNIT NUMBER (10 OR 11)

The operator must input the tape unit number on which the retrieval master is to be placed.
4.1 COMPUTER OPERATOR INSTRUCTIONS (Continued)

- If the user requests the COPY option, the computer will output the message "MOUNT OUTPUT TAPE ON UNIT XX". The operator must mount a scratch tape on the magnetic tape unit not reserved for the retrieval master. On program termination, the message "DISMOUNT AND LABEL TAPE UNIT XX" will be output, requiring the operator to wrap up the retrieval.

4.2 OPERATIONAL PROCEDURES

There are two processes involved in retrieving: definition (defining the specific record(s) or information of interest) and action (indicating how that information is to be retrieved or output). A series of seven questions are posed to the user. See Figure 1. The answers entered in response to these questions provide the definitions of the record(s) to be considered as well as the action to be performed on the records.

The first four questions allow the user to identify the record(s) of interest. They are the same four questions or headings which comprise the identification portion of all MEDATA records in the computer, namely SS NO, RECORD, TYPE, and DATE.

SS NO
The user may specify one social security number example (SS NO: 123-45-6709) or he may answer "ALL", in which case all social security numbers in the file would be accepted. If a social security number is specified, it must be in the form of XXX-XX-XXXX, where X is a digit. If no answer is given to this question, the "default" condition "ALL" is assumed.
4.2 OPERATIONAL PROCEDURES (Continued)

RECORD

The user may specify one particular record of interest example (RECORD: SURVEY) or he may answer "ALL". The RECORD response may not exceed twenty-four characters. If no answer is given to the question, "ALL" is assumed.

TYPE

The user may specify one particular type of record example (TYPE: LABORATORY) or he may answer "ALL". The TYPE response may not exceed twenty-four characters. If no answer is given to this question, "ALL" is assumed.

DATE

The user may specify one specific date example (DATE:01JAN71), a range of dates example (DATE:01JAN70-04MAR71), or "ALL". DATE must always be answered in the form of day, month, year with a three letter alpha month. If no answer is given to this question, "ALL" is assumed.

See Figure 2 for sample lists.

CONDITION

The fifth retrieval question, CONDITION, further limits the records searched to those containing specific information so that the user may specify portions of records or items from records as selection criteria. These selection criteria may be particular heading(s) with or without data. The user indicates whether a particular character string is heading
or data by utilizing the colon just as it is used in the basic file structure; characters preceding the colon are interpreted as heading and characters following the colon are understood to be data. The user may wish to specify only a heading and no data, but if he wishes to specify a data string, it must always be preceded by a heading and colon. The data itself may be either of two types: prose or coded. Prose data is alphabetic or alphanumeric, whereas coded data is numeric data and enclosed in parenthesis.

The CONDITION question offers the user three types of capabilities: string search, ranging, and Boolean logic.

STRING SEARCH

If the user specifies data as part of his selection criteria, and that data is prose, a search is made of the data field corresponding to each matching heading in every pertinent record in the file to find the exact string of characters that the user has specified. See Figure 3.

In a string search, prose and coded data may not be mixed in the same character string. For example, PURPOSE:(71)ANNUAL must be retrieved in the following way: PURPOSE:71 AND PURPOSE: ANNUAL.

Furthermore, in a string search the user may not retrieve prose numeric data (numeric data not enclosed in parentheses) unless at least one alphabetic character is included. For example, WEIGHT:160 LBS will not be retrieved if the user specifies only WEIGHT:160. He must include at least one alpha character since 160 is not coded, but is prose numeric data.
4.2 OPERATIONAL PROCEDURES (Continued)

RANGING

The user has the capability to specify a range of numeric values as part of his selection criteria. Any record containing a data value which falls within the range will be retrieved if that record also meets all other specifications as well. Only coded data will be searched under the ranging selection. See Figure 4.

BOOLEAN

This feature adds another dimension to the search definition AND/OR control by offering the user the capability to link selection parameters logically together with the Boolean operators AND and OR. Several sets of headings with or without data may be linked by these terms into complex retrieval requests. AND linkage requires that both criteria be met in order to constitute a valid retrieval. The OR linkage is an inclusive OR and requires that either one or both of the criteria be met. See Figure 5. The user may specify no more than ten Boolean strings in response to the CONDITION and WHAT questions. The CONDITION question may be answered with "NONE"; indicating that there is no specific selection criteria within the body of the record(s) of interest. If the question is unanswered, "NONE" is assumed.

ACTION

The first five questions (SS NO, RECORD, TYPE, DATE and CONDITION) comprise the definition segment of the retrieval. So far, the computer has been told only which records are to be searched. The sixth question indicates the action to be taken, i.e., how the computer is to output the selected information.
4.2 OPERATIONAL PROCEDURES (Continued)

ACTION (Continued)

Valid actions are LIST, COUNT, COPY, TABULATE and ANALYZE. For a description of each output type see Section 4.4.1, Reports. Note that the LIST option is assumed if the user leaves this question unanswered.

WHAT

The seventh retrieval question, WHAT, allows the user to specify exactly which items in the record(s) are to be retrieved if he does not wish to retrieve the whole record. The user may specify data (including numeric ranges) as well as headings. If the whole record is desired, WHAT is answered with "ALL or left unanswered.

SPECIAL CHARACTERS

The three characters described below are used to control the query-response interaction between the computer and the user.

* - An asterisk indicates the final character of each of the seven answers.

$ - The dollar sign character causes the current question to be output over again.

@ - The at sign character causes the current retrieval to be aborted.

EXECUPORT OPERATING INSTRUCTIONS

See Appendix H.
4.3 INPUT DESCRIPTION
MDTRS requires two types of input, namely; (1) Master tape records and (2) User responses to questions. MDTRS accepts the tape in MEDATA format and extracts any user requested information required. The user communicates his retrieval request to the computer by answering pre-programmed retrieval questions output by MDTRS.

4.4 OUTPUT DESCRIPTION

4.4.1 Reports
LIST
The LIST option reproduces upon output the specified record(s) or portions of records exactly as they were input. The format in which the data is retrieved is identical to the format of the stored records.

COPY
The COPY option takes those records selected for retrieval from the master tape and outputs them to a second magnetic tape, thus allowing the user to interrogate a subset of the master file as opposed to interrogating the larger master file.

COUNT
The COUNT option displays a count of all records which satisfy the selection criteria or a count of specific items within records. If the retrieval question "WHAT" is answered with "ALL", or left unanswered, all records which satisfy the selection criteria are counted, whereas, if "WHAT" is answered with a specific heading or heading/answer pair, that specific item is counted.
4.4.1 Reports (Continued)

**TABULATE**

The TABULATE option prints out the specified information in a tabular column form. The headings corresponding to the tabular columns of data are printed above their respective columns. Each horizontal line of tabular output begins with the social security number and the date associated with that row of data. The user may tabulate coded or prose data. See Figure 6.

**ANALYZE**

The ANALYZE option is identical to TABULATE with one exception. At the end of the tabulation a mean and standard deviation is provided for each heading. See Figure 7.

4.4.2 Tapes

If the user request the COPY option, the system creates as output a magnetic tape containing those records selected for retrieval. This tape may itself be used as a retrieval master tape for subsequent retrievals.

4.4.3 Messages

4.4.3.1 Diagnostics

<table>
<thead>
<tr>
<th>Message</th>
<th>Explanation</th>
<th>Corrective Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILLEGAL CODE</td>
<td>Operator has specified an incorrect computer code, device code, or magnetic tape unit number.</td>
<td>Input another code or tape unit number.</td>
</tr>
</tbody>
</table>
4.4.3.1 Diagnostics

Message | Explanation | Corrective Measure
---|---|---
INVALID SS NO | User has requested an invalid Social Security number; possibly an incorrect format or an alpha character in the number. | Input another Social Security number.
INVALID RECORD | User has requested a Record Name which exceeds twenty-four characters. | Input another Record name.
INVALID TYPE | User has requested a record Type name which exceeds twenty-four characters. | Input another Type name.
INVALID DATE | User has requested an invalid Date, possibly due to an alpha character in the day or year or an invalid format. | Input another Date.
INVALID CONDITION | User has requested an invalid Condition; possibly a format. | Input another Condition.
4.4.3.1 Diagnostics (Continued)

Message        Explanation       Corrective Measure

INVALID ACTION User has requested something other than a valid Action (LIST, COPY, TABULATE, ANALYZE or COUNT).

INVALID WHAT User has requested an invalid What condition.

CANNOT ANALYZE User has requested the Analyze action for prose data.

ILLEGAL PARENTHESIS A parenthesis has occurred in an illegal position in the answer or number of right parentheses unequal to number of left parentheses.

Input another Action.

Input another What.

Request another Action or different What parameters.

Make correction and input another response to the CONDITION question.

4-11
### 4.4.3.1 Diagnostics (Continued)

<table>
<thead>
<tr>
<th>Message</th>
<th>Explanation</th>
<th>Corrective Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECORD BAD-SKIPPED</td>
<td>There is some illegal data on the record being processed. The tape is advanced to the next record and the bad record is disregarded.</td>
<td>None required.</td>
</tr>
<tr>
<td>TAPE ERROR RECORD SKIP</td>
<td>The computer is unable to read a master tape record. The tape is advanced to the next record and the bad record disregarded.</td>
<td>None Required.</td>
</tr>
<tr>
<td>NO ANSWER WITH COLON</td>
<td>A WHAT or CONDITION heading followed by a colon has been found, but no answer following the colon has been specified.</td>
<td>Make correction and input another response to the question in error.</td>
</tr>
<tr>
<td>TOO MANY PARAMETERS</td>
<td>Too many WHAT parameters specified for the requested output device for the ANALYZE or TABULATE option.</td>
<td>Make correction. Only five headings are allowed for the teletype or 103A Modem.</td>
</tr>
</tbody>
</table>
4.4.3.2 Advisory/System Queries

Message User Action

OUTPUT COMPLETE Output for this request is complete. User may key in new request if desired.

PLEASE INITIALIZE No user action. These are the corresponding code
SYSTEM COMPUTER CODES numbers of the computers on which the retrieval
ARE AS FOLLOWS may take place.

DOC*A = 1
DOC*B = 2
CLINC = 3

DEVICE CODES ARE AS No user action. These are the primary retrieval
FOLLOWS device codes.

0 = NOTE REQUIRED
1 = CLINC TTY #1
2 = CLINC TTY #2
3 = DOC TTY
4 = 103A MODEM
5 = CLINC CRT 73
6 = CLINC CRT 74
7 = DOC CRT 71
8 = DOC CRT 74 (REMOTE)
9 = CLINC LINE PRINTER

WHAT COMPUTER IS THIS Operator must specify on which computer the
(MUST ANSWER 1, 2 retrieval is taking place (see above for computer
OR 3) codes).

PRIMARY RETRIEVAL Operator must specify on which primary I/O device
DEVICE (MUST ANSWER the retrieval is taking place (see above for device
1, 2, 3 OR 4) codes). Presently only codes 1 thru 4 are operational.
4.4.3.2 Advisory/System Queries (Continued)

<table>
<thead>
<tr>
<th>Message</th>
<th>User Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLEASE MOUNT MASTER TAPE TYPE UNIT NUMBER (10 OR 11)</td>
<td>Operator must specify the unit number of the tape drive on which the master file tape has been mounted.</td>
</tr>
<tr>
<td>SYSTEM INITIALIZED READY</td>
<td>No user action required. The system has been initialized properly and the user may now input his retrieval request.</td>
</tr>
<tr>
<td>MTU IS NOT READY. AFTER YOU READY MTU, PROGRAM WILL CONTINUE</td>
<td>Ready the magnetic tape unit on which the retrieval master tape is mounted.</td>
</tr>
<tr>
<td>PUT MAG TAPE UNIT(S) ON LINE</td>
<td>Due to a power failure, the magnetic tape units must be initialized to an on-line status. This is done by pressing LOAD twice and ON LINE once.</td>
</tr>
<tr>
<td>SS NO:</td>
<td>User must specify one Social Security number on which he wants to retrieve data or he may answer &quot;ALL&quot; which tells the computer to accept all Social Security numbers in the file.</td>
</tr>
<tr>
<td>RECORD:</td>
<td>User must specify one Record name on which he wants to retrieve data or he may answer &quot;ALL&quot;.</td>
</tr>
<tr>
<td>TYPE:</td>
<td>User must specify one record TYPE on which he wants to retrieve data or he may answer &quot;ALL&quot;.</td>
</tr>
</tbody>
</table>
4.4.3.2 Advisory/System Queries (Continued)

**Message**

**DATE:**
User must specify one date, a range of dates, or "ALL" dates on which he wants to retrieve data.

**CONDITION:**
User must specify the condition parameter(s) on which he wants to retrieve data.

**ACTION:**
User must specify the action (COPY, COUNT, LIST, TABULATE, or ANALYZE) he wants taken on the retrieved data.

**WHAT:**
User must specify upon what parameter(s) he wants the above action taken.

**RETRIEVAL WORKING**
No user response necessary. The retrieval is in progress.

**RETRIEVAL ABORTED**
No user action required. This message occurs when the user has voluntarily aborted the retrieval currently in progress by pressing the @ key.

**OUTPUT TERMINATED**

**POWER FAILURE. PLEASE RESTART REQUEST *******
Power failure has occurred. User must re-enter the retrieval request.
4.4.3.2 Advisory/System Queries (Continued)

<table>
<thead>
<tr>
<th>Message</th>
<th>User Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRT NOT READY</td>
<td>No user action required. The program will continue ignoring the last I/O request.</td>
</tr>
<tr>
<td>PARITY ERROR</td>
<td>No user action required. The program will continue, ignoring the last I/O request.</td>
</tr>
<tr>
<td>MODEM DISCONNECTED</td>
<td>No user action required. The program will continue, ignoring the last I/O request.</td>
</tr>
</tbody>
</table>

4.5 RESTRICTIONS

- Due to space limitations, a maximum of five headings for the teletype are allowed as WHAT answer responses when the TABULATE or ANALYZE actions are requested.

- The user may not specify a range or a series of Social Security numbers.

- The user may specify only one type of data per heading. In a string search, prose and coded data may not be mixed in the same character string. For example,

  PURPOSE:(71)ANNUAL must be retrieved in the following way:

  PURPOSE:71 AND PURPOSE:ANNUAL.
<table>
<thead>
<tr>
<th>Function</th>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search</td>
<td>1. SS NO</td>
<td>= WHO?</td>
</tr>
<tr>
<td>Criteria</td>
<td>2. RECORD</td>
<td>= WHAT group of records?</td>
</tr>
<tr>
<td></td>
<td>3. TYPE</td>
<td>= WHAT subset of that group of records</td>
</tr>
<tr>
<td></td>
<td>4. DATE</td>
<td>= WHEN?</td>
</tr>
<tr>
<td></td>
<td>5. CONDITION</td>
<td>= WHAT limitations narrow that subset?</td>
</tr>
<tr>
<td>Type of retrieval</td>
<td>6. ACTION</td>
<td>= HOW are these items to be output?</td>
</tr>
<tr>
<td>Retrieval parameters</td>
<td>7. WHAT</td>
<td>= WHAT specific items are to be output?</td>
</tr>
</tbody>
</table>

**FIGURE 4-1 - SEVEN RETRIEVAL QUESTIONS**
FIGURE 4-2 - SAMPLE IDENTIFICATION LISTS

4-18
SAMPLE RETRIEVAL REQUEST

SS NO. 111-11-1111*
RECORD REPORT*
TYPE EKG*
DATE 01JAN69*
CONDITION PURPOSE: APOLLO 13*
ACTION LIST*
WHAT ALL*

SAMPLE RETRIEVAL OUTPUT

111-11-1111 REPORT EKG 01JAN69
NAME: DOE, JOHN
PURPOSE: APOLLO 13 F-5
MEASUREMENTS

RHYTHM: (xxx)
RATE: (xxx)
INTERVALS
P-R: (xxx)
QRS: (xxx)
QT: (xxx)
AXIS: (xxx)

DESCRIPTIONS
Limb leads: xxx
Precordial leads: xxx
Interpretation: xxx
Comments: xxx
Examiner: xxx

FIGURE 4-3 - CHARACTER STRING SEARCH

4-19
<table>
<thead>
<tr>
<th>SS NO</th>
<th>ALL*</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECORD</td>
<td>SURVEY*</td>
</tr>
<tr>
<td>TYPE</td>
<td>MEASUREMENT*</td>
</tr>
<tr>
<td>DATE</td>
<td>ALL*</td>
</tr>
<tr>
<td>CONDITIONS</td>
<td>WEIGHT: 150-200*</td>
</tr>
<tr>
<td>ACTION</td>
<td>LIST*</td>
</tr>
<tr>
<td>WHAT</td>
<td>ALL*</td>
</tr>
</tbody>
</table>

**FIGURE 4-4 - SAMPLE RANGING**

4-20
<table>
<thead>
<tr>
<th>SS NO</th>
<th>ALL*</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECORD</td>
<td>REPORT*</td>
</tr>
<tr>
<td>TYPE</td>
<td>LABORATORY*</td>
</tr>
<tr>
<td>DATE</td>
<td>ALL*</td>
</tr>
<tr>
<td>CONDITIONS</td>
<td>EXAM: HEMATOLOGY AND PURPOSE: APOLLO 13 AND WBC: 3800-4400 AND (HB: 15-16 OR HCT: 0-45)*</td>
</tr>
<tr>
<td>ACTION</td>
<td>LIST*</td>
</tr>
<tr>
<td>WHAT</td>
<td>ALL*</td>
</tr>
</tbody>
</table>

**FIGURE 4-5 - SAMPLE BOOLEAN**
### SAMPLE RETRIEVAL REQUEST

<table>
<thead>
<tr>
<th>SS NO</th>
<th>RECORD</th>
<th>TYPE</th>
<th>DATE</th>
<th>CONDITIONS</th>
<th>ACTION</th>
<th>WHAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL*</td>
<td>SURVEY*</td>
<td>MEASUREMENT*</td>
<td>15JAN69-15MAR69*</td>
<td>NONE*</td>
<td>TABULATE*</td>
<td>HEIGHT AND WEIGHT AND BUILD AND TEMP:99-100</td>
</tr>
</tbody>
</table>

### SAMPLE RETRIEVAL OUTPUT

<table>
<thead>
<tr>
<th>SS NO</th>
<th>DATE</th>
<th>HEIGHT</th>
<th>WEIGHT</th>
<th>BUILD</th>
<th>TEMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-45-6789</td>
<td>23JAN69</td>
<td>70.50</td>
<td>175.00</td>
<td>MEDIUM</td>
<td>99</td>
</tr>
<tr>
<td>456-78-9123</td>
<td>3FEB69</td>
<td>68.75</td>
<td>167.00</td>
<td>MEDIUM</td>
<td>99</td>
</tr>
<tr>
<td>678-23-4567</td>
<td>15MAR69</td>
<td>68.00</td>
<td>152.00</td>
<td>SMALL</td>
<td>100</td>
</tr>
</tbody>
</table>

**FIGURE 4-6 - SAMPLE TABULATE**
### SAMPLE RETRIEVAL REQUEST

<table>
<thead>
<tr>
<th>SS NO.</th>
<th>RECORD</th>
<th>TYPE</th>
<th>DATE</th>
<th>CONDITIONS</th>
<th>ACTION</th>
<th>WHAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>111-11-1111</td>
<td>SURVEY*</td>
<td>MEASUREMENT*</td>
<td>ALL*</td>
<td>WEIGHT*</td>
<td>ANALYZE*</td>
<td>ALL*</td>
</tr>
</tbody>
</table>

### SAMPLE RETRIEVAL OUTPUT

<table>
<thead>
<tr>
<th>SS NO.</th>
<th>DATE</th>
<th>WEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>111-11-1111</td>
<td>01JAN68</td>
<td>180</td>
</tr>
<tr>
<td>111-11-1111</td>
<td>01FEB68</td>
<td>181</td>
</tr>
<tr>
<td>111-11-1111</td>
<td>01MAR68</td>
<td>183</td>
</tr>
<tr>
<td>111-11-1111</td>
<td>01APR68</td>
<td>180</td>
</tr>
<tr>
<td>111-11-1111</td>
<td>01SEP68</td>
<td>185</td>
</tr>
</tbody>
</table>

**Mean** 181.8  
**SD** 2.46

**FIGURE 4-7 - SAMPLE ANALYZE**

4-23
APPENDIX A

TAPE DESCRIPTION

FIGURE A-1  MASTER FILE STRUCTURE

The Master File is composed of N records of variable length, not to exceed 2880 words. At the end of the data a special end of file record is written to avoid any incompatibility between drives that would prevent the hardware end of file from being recognized.

A-1
MASTER FILE INFORMATION

The Master File information is made up of three parts: Heading, Answer, and Level Code. The Heading is that background data that asks a question (e.g., Blood Pressure, Temperature, etc.). In some cases, however, it merely identifies the category of the Headings (or questions) to follow (e.g., Cardiovascular, Musculoskeletal, etc.). The Answer is the response made to the question. This response may be narrative or numeric (coded) data. The Level Code defines the relationship among all the Headings and Answers on a record. The use of Level Codes breaks down a particular record into paragraphs of related information.
A data record consists of a Fixed Length Identification and Variable Length Body. FIGURE A-2 illustrates the organization of a record. The segments are identified by mnemonics which were used in addressing these same segments in the original MEDATA programs. Six of the seven segments of a record are composed of word blocks and their size in words is specified in the parenthesis following each mnemonic. The number following the parenthesis is the number of blocks in each segment of the Identification. (The blocks of the segment IANS are variable length.) The number of blocks in each segment of the Body is dependent upon the contents of the IQNDEX segment.

IQNDEX

This segment of the Identification is a number that specifies the total number of Headings that are found in the record. All records have at least eight Headings, so IQNDEX will be greater than eight.
IAINDEX
This segment of the Identification specifies the length of the segment IANS.

ID
The ID segment is made up of four 36-word blocks. These blocks contain the Answers to the first four Headings of a record. These Headings are: SS NO, RECORD, TYPE, and DATE. The Answers begin in the fourth word of the block.

IQ
This segment of the Body contains all the Headings of the record, one in each twelve word block. The number of blocks is equal to the value in the IQNDEX segment.

LCQ
This segment of the Body contains the Level Codes which are found in the third word of each three word block. The number of blocks is equal to the value in the IQNDEX segment.

IFWAA
This segment of the Body contains pointers relative to the beginning of the IANS segment. These pointers address the beginning of each respective Answer to the Heading in IQ. The number of three word blocks is equal to the value in the IQNDEX segment.
IANS

This segment of the Body is made up of variable length blocks which contain the Answers associated with the Headings in IQ. There are three types of data allowed in IANS and they have priorities:

(1) date data
(2) numeric data
(3) narrative data

The date is a fixed length of seven characters and is preceded by an equal sign. The narrative data is variable in length and is always preceded by a colon. The numeric data is found between the two, and is also variable in length.
The end-of-file record has an IQNDEX value of one. This is the only record on the entire record with such a value.
APPENDIX B
REQUEST TABLE

The Request Table is a seven word buffer that correlates each Request question either to some location in memory where the user response is saved, or to a flag which represents certain standard responses. The seven Request questions are:

SS NO
RECORD
TYPE
DATE
CONDITION
ACTION
WHAT

Responses are saved in the Request Buffer, and, for the first four questions, the Request Table relates directly to this buffer through pointers. These pointers are indices into the buffer, and they locate the beginning of each saved response.

The fifth and seventh questions relate indirectly to the Request Buffer in that they point to a location in the Operand Buffer whose contents addresses the saved response in the Request Buffer.

STANDARD RESPONSES

All the responses to the sixth question and one response to the other questions are called standard responses. In the case of these responses
the Request Table location contains a flag rather than a pointer.

The standard responses are:

- SS NO: ALL*
- RECORD: ALL*
- TYPE: ALL*
- DATE: ALL*
- CONDITION: NONE*
- ACTION: LIST, COPY, COUNT, TABULATE, OR ANALYZE*
- WHAT: ALL*

The flags for each of these responses are:

- ALL = 0
- NONE = 0
- LIST = 0
- COPY = 1
- COUNT = 2
- TABULATE = 3
- ANALYZE = 4

**DEFAULT RESPONSES**

If no response is given to a question, the standard response whose flag is zero is assumed as follows:

- SS NO: ALL*
- RECORD: ALL*
- TYPE: ALL*
- DATE: ALL*
- CONDITION: NONE*
- ACTION: LIST*
- WHAT: ALL*

See FIGURES 3-1, 3-2, and 3-3 for examples of the Request Table.

B-2
The Bool Buffer consists of several nodes. Each node is made up of four parts: left conclusion, right conclusion, operator, and link.

The node is two memory locations which represent a simple Boolean expression, which is an expression in Boolean algebra that consists of one operator and two operands (however, the operand may itself be a simple expression). The logical conclusion of any node is the Boolean result of the nodal operation of that node.

The link, the second word in a node (node 1), is a pointer to another node (node 2). If the link of node 1 addresses the first word or second word of node 2, then node 1 represents the left operand or right operand, respectively, of node 2. Thus, if the logical conclusion of node 1 is found to be true, and if the link of node 1 addresses the second word of node 2, then the right conclusion of node 2 must be set true. Figure 3-3 is a diagram of the relation of the Bool Buffer to a Retrieval Request. Section 3.3.1 gives a thorough description of how the links are created in the Bool Buffer, and Section 3.3.39 describes the use of this buffer in evaluating a Boolean expression.
APPENDIX D

OPERAND BUFFER

The Operand Buffer is a buffer, forty words in length, that is utilized in two-word sets. Each set contains two pointers; the first pointer addresses the Request Buffer and the second links to the Bool Buffer.

The responses to the Retrieval Request questions CONDITION and WHAT may consist of a Boolean expression, where the operand of the expression is saved in the Request Buffer, and a coded version of the operator is saved in the Bool Buffer. To maintain the original form of the expression, the Operand Buffer is used to link each operand to an operator.

See FIGURES 3-2 and 3-3 for a diagram of the relation of the sets in the Operand Buffer
APPENDIX E
REQUEST BUFFER

The Request Buffer (CPRB) is a 480 word buffer into which all non-standard responses are saved (see Appendix B for explanation of standard responses). The data is packed two ASCII characters per word, with all blanks removed.

SS NO
The response to the first question in the Retrieval Request (SS NO) is saved in the first six locations of the Request Buffer, unless there is no response or the standard response ALL is found. This response is not terminated by the terminal character (177) as is the case with other responses.

RECORD and TYPE
If a non-standard response is made to the second and/or third questions of the Retrieval Request, it/they will be saved beginning at the next available new word boundary. The last character in both cases will be the terminal character (177).

DATE
The non-standard response to the fourth question of the Retrieval Request will be one of two forms: a single data or two dates representing a range of dates. If a single date is the response, the data is stored beginning at the next available word boundary, and a terminal character
APPENDIX E (Continued)

DATE (Continued)

is stored at the end. If two dates are input, both begin at the next available word boundary, and end with a terminal character.

CONDITION and WHAT

Non-standard responses to the fifth and seventh questions of the Retrieval Request may be one of four forms:

(1) Heading alone

(2) Heading plus alpha Answer

(3) Heading plus numeric Answer

(4) Heading plus a range of numeric Answer.

In each case the Heading is stored beginning at the next available word boundary, and terminated with a terminal character. For the second and third forms the Answer is also begun on a new word boundary and ended with a terminal character. For the fourth form each Answer, representing the upper and lower limit of the range, is stored in the same manner as the second and third forms.

ACTION

All responses to the sixth question of the Retrieval Request are standard responses, and since only non-standard responses are stored in the Request Buffer, the response to this question is not saved.

See FIGURES 3-1, 3-2, and 33 for diagrams of the Request Buffer.
The Condition Table is broken down into one, two, or three word sets. The first word of each set is a flag and is associated with one operand in the response to the fifth question of the Retrieval Request, CONDITION. This response may be made up of several operands and they may be one of four different forms:

1. Heading alone
2. Heading plus alpha Answer
3. Heading plus numeric Answer
4. Heading plus a range of numeric Answer.

**HEADING ALONE**

If the operand is a Heading alone, the Condition Table will supply one word whose contents is zero.

**HEADING PLUS ALPHA ANSWER**

The set associated with this form of operand is two words. The first word is a flag of one; the second word is a pointer to the beginning of the Answer in the Request Buffer.

**HEADING PLUS NUMERIC ANSWER**

The set associated with this form of operand is two words. The first word is a flag of two; the second word is a pointer to the beginning of the Answer in the Request Buffer.
**HEADING PLUS A RANGE OF NUMERIC ANSWER**

The set associated with this form of operand is three words. The first word is a flag of three; the second and third words are pointers to the lower limit and upper limit respectively of the Answer stored in the Request Buffer.

**SUMMARY**

<table>
<thead>
<tr>
<th>FORM</th>
<th>FLAG</th>
<th>NO. OF WORDS IN SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heading alone</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Heading plus alpha Answer</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Heading plus numeric Answer</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Heading plus range of numeric Answer</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

See FIGURE 3-3 for a diagram of the Condition Table.
APPENDIX G

WHAT TABLE

The format of the What Table is the same as the Condition Table (see Appendix F).

See FIGURES 3-3 for a diagram of the What Table.
APPENDIX H
READYING THE EXECUPORT 300 RETRIEVAL SYSTEM

1. If the computer has not already been readied, call 713-483-4796 or Philco at 713-488-1270, X-564 and have the Retrieval System initialized for remote terminal use.

2. Lift the lid of the Execuport 300 and position the five switches at the front as follows:

   MODE       Line
   DUPLEX     Full
   CHAR/SEC   30
   PARITY     Odd
   QSL        Upper

3. Turn on Execuport 300.

4. From any telephone dial 713-483-6260 and wait for a continuous high-pitched tone. This should come after one ring.

5. Insert the telephone receiver, with the cord toward the front, into the receptacle on the side of the Execuport 300. Make certain that the receiver is secure.

6. Wait for the "Ready" light, on the left front panel, above the keyboard of the Execuport 300 to come on.

7. Type in "@". This will begin the output of the seven retrieval request-questions.
# APPENDIX I

## SAMPLE INPUT

<table>
<thead>
<tr>
<th>INDEX</th>
<th>INDEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>72</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>REPORT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CP LAB</td>
</tr>
<tr>
<td></td>
<td>000070</td>
</tr>
<tr>
<td>SS NO</td>
<td>8000004</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RECORD</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>COR</td>
</tr>
<tr>
<td>UPDATED</td>
<td>TYPED</td>
</tr>
<tr>
<td>PROC</td>
<td>NAME</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TEST NO</th>
<th>ROUTINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>EKG</td>
<td>DIAGNOSES</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COMMENTS</th>
<th>EXERCISE</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXAMINER</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LCQ</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IFWAA</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>42</td>
<td>66</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IANS</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>/700804</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>WAS NO DIAGNOSIS : PATIENT WAS NOT VERY ACTIVE FOR A MAN HIS AGE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: HAROLD KINCHAM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

TAPe INPUT
SAMPLE REQUESTS

SS NO: *
RECORD: REPORT*
TYPE: LABORATORY*
DATE: *
CONDITION: EXAM: SERUM*
ACTION: *
WHAT: *

SS NO: *
RECORD: *
TYPE: *
DATE: *
CONDITION: *
ACTION: COUNT*
WHAT: *

SS NO: *
RECORD: REPORT*
TYPE: LABORATORY*
DATE: *
CONDITION: EXAM: SERUM*
ACTION: TABU*
WHAT: GLU AND SGOT AND A1 AND A2 AND B*

SS NO: *
RECORD: REPORT*
TYPE: LABORATORY*
DATE: *
CONDITION: EXAM: SERUM AND GLU: 70-80*
ACTION: ANALYZE*
WHAT: GLU AND SGOT AND A1 AND A2 AND B*

SS NO: 111-22-3333*
RECORD: REPORT*
TYPE: LABORATORY*
DATE: 01JUN72-09SEP74*
CONDITION: EXAM: UA RANDOM OR (EXAM: SERUM AND GLU: 70-80)*
ACTION: LIST*
WHAT: EXAM AND NAME AND GLU*
SAMPLE INPUT (Continued)

DEFAULT

SS NO:*  
RECORD:*  
TYPE:*  
DATE:*  
CONDITION:*  
ACTION:*  
WHAT:*  

ID LIST

SS NO:*  
RECORD:*  
TYPE:*  
DATE:*  
CONDITION:*  
ACTION:*  
WHAT: ID ONLY*
APPENDIX J

SAMPLE OUTPUT

**LIST**

111-22-3456 REPORT LABORATORY 10JUN71

<table>
<thead>
<tr>
<th>UPDATED</th>
<th>COR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typed:</td>
<td>000000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PROC:</th>
<th>710706</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>NAME:</th>
<th></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>PERMANENT BASE:</th>
<th>KSC</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>PURPOSE:</th>
<th>APOLLO 15 SURVEILLANCE PC</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SEX:</th>
<th>M</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>BIRTHYEAR:</th>
<th>(TF)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>CATEGORY:</th>
<th>COMPLETE</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>EXAM:</th>
<th>SERUM</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SEROLOGY</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>CRP:</th>
<th>POS</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>HEMATOLOGY</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>HCT:</th>
<th>(36)PCT</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>WBC:</th>
<th>(10300)CU MM</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>DIFF</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>NEUT:</th>
<th>(7931)CU MM</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>LYMPH:</th>
<th>(2163)CU MM</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>MONO:</th>
<th>(0)GU MM</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>EO:</th>
<th>(206)CU MM</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>BASO:</th>
<th>(0)CU MM</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>BANDS:</th>
<th>(0)CU MM</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>MORPH:</th>
<th>MOD. HYPOCHROMIA</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>CHEMISTRY</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>GLU:</th>
<th>(74)MG PCT</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SGOT:</th>
<th>(13)MU/ML</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>BUN:</th>
<th>(21)MG PCT</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>URIC ACID:</th>
<th>(5.9)MG PCT</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ALK PHOS:</th>
<th>(28)IU</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>CREATININE:</th>
<th>(1.8)MG PCT</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>LDH:</th>
<th>(253)MU/ML</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ELECTROPHORESIS</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>PROTEIN:</th>
<th>(7.1)GM PCT</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ALBUMIN:</th>
<th>(4.0)GM PC</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>GLOBULIN</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>A1:</th>
<th>(0.3)GM PCT</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>A2:</th>
<th>(0.9)GM PCT</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>B:</th>
<th>(1.0)GM PCT</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>G:</th>
<th>(0.8)GM PCT</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>OTHER:</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>COMMENTS:</th>
</tr>
</thead>
</table>

J-1
SAMPLE OUTPUT (Continued)

COUNT

COUNT IS 0362

TABULATE

<table>
<thead>
<tr>
<th></th>
<th>GLU</th>
<th>SPOT</th>
<th>A1</th>
<th>A2</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>111-11-1111</td>
<td>74</td>
<td>13</td>
<td>.3</td>
<td>.9</td>
<td>1.0</td>
</tr>
<tr>
<td>222-22-222</td>
<td>108</td>
<td>15</td>
<td>.2</td>
<td>.5</td>
<td>1.0</td>
</tr>
<tr>
<td>333-33-3333</td>
<td>106</td>
<td>12</td>
<td>.1</td>
<td>.5</td>
<td>1.0</td>
</tr>
<tr>
<td>444-44-4444</td>
<td>100</td>
<td>15</td>
<td>.2</td>
<td>.6</td>
<td>1.0</td>
</tr>
<tr>
<td>555-55-5555</td>
<td>92</td>
<td>12</td>
<td>.1</td>
<td>.5</td>
<td>1.0</td>
</tr>
<tr>
<td>666-66-6666</td>
<td>132</td>
<td>18</td>
<td>.3</td>
<td>.7</td>
<td>1.0</td>
</tr>
<tr>
<td>777-77-7777</td>
<td>132</td>
<td>18</td>
<td>.3</td>
<td>.7</td>
<td>1.0</td>
</tr>
</tbody>
</table>

ANALYZE

<table>
<thead>
<tr>
<th></th>
<th>GLU</th>
<th>SGOT</th>
<th>A1</th>
<th>A2</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>111-11-1111</td>
<td>74</td>
<td>13</td>
<td>.3</td>
<td>.9</td>
<td>1.0</td>
</tr>
<tr>
<td>222-22-2222</td>
<td>94</td>
<td>20</td>
<td>.2</td>
<td>.6</td>
<td>1.0</td>
</tr>
<tr>
<td>333-33-3333</td>
<td>94</td>
<td>10</td>
<td>.1</td>
<td>.5</td>
<td>1.0</td>
</tr>
<tr>
<td>444-44-4444</td>
<td>116</td>
<td>14</td>
<td>.3</td>
<td>.6</td>
<td>1.0</td>
</tr>
<tr>
<td>555-55-5555</td>
<td>98</td>
<td>16</td>
<td>.2</td>
<td>.8</td>
<td>1.0</td>
</tr>
<tr>
<td>666-66-6666</td>
<td>90</td>
<td>18</td>
<td>.2</td>
<td>.6</td>
<td>1.0</td>
</tr>
<tr>
<td>777-77-7777</td>
<td>80</td>
<td>15</td>
<td>.3</td>
<td>.8</td>
<td>1.0</td>
</tr>
<tr>
<td>MEAN</td>
<td>92.28523</td>
<td>15.14262</td>
<td>.2281945</td>
<td>.6845840</td>
<td>1.014264</td>
</tr>
<tr>
<td>SD</td>
<td>13.48640</td>
<td>3.287307</td>
<td>.0754685</td>
<td>.1461453</td>
<td>.037742</td>
</tr>
</tbody>
</table>

J-2