Program GDETAP Documentation

Jack A. Briscoe
The BDM Corporation

for

ARI Field Unit at Presidio of Monterey, California
James H. Banks, Chief

TRAINING RESEARCH LABORATORY
Jack H. Hiller, Director

ARI

U. S. Army
Research Institute for the Behavioral and Social Sciences
January 1987

Approved for public release; distribution unlimited.
DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.
U. S. ARMY RESEARCH INSTITUTE
FOR THE BEHAVIORAL AND SOCIAL SCIENCES
A Field Operating Agency under the Jurisdiction of the
Deputy Chief of Staff for Personnel

EDGAR M. JOHNSON
Technical Director

WM. DARRYL HENDERSON
COL, IN
Commanding

Research accomplished under contract
for the Department of the Army

The BDM Corporation

Technical Review by
James H. Banks

This report, as submitted by the contractor, has been cleared for release to Defense Technical Information Center (DTIC) to comply with regulatory requirements. It has been given no primary distribution other than to DTIC and will be available only through DTIC or other reference services such as the National Technical Information Service (NTIS). The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other official documentation.
GDETAP is a VAX Fortran program which can read the GDE log tape produced at the National Training Center (NTC), and produce reports and files on the data the tape contains. There are two separate sources of digital data at NTC, the GDE tape, and BACKUP tapes created from files set up using the CIS VAX-780 computer.

Comparing the position/location data from the GDE tapes with the files created from the BACKUP tapes will be useful for several reasons: (OVER)
ARI Research Note 87-09

20. Abstract (continued)

a) validation of the NTC research database system
b) access to data not available currently

c) the possibility of combining data sources to clean data.

In order to access data on the log tapes, it was necessary to develop a VAX program to read them. This has been done, and documentation of the tape-reading program GDETAP is the subject of this report.
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 Background</td>
<td>1</td>
</tr>
<tr>
<td>2.0 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>3.0 Program Description</td>
<td>2</td>
</tr>
<tr>
<td>3.1 Input</td>
<td>2</td>
</tr>
<tr>
<td>3.2 Processing</td>
<td>2</td>
</tr>
<tr>
<td>3.3 Output</td>
<td>3</td>
</tr>
<tr>
<td>Appendix A - User's Guide</td>
<td></td>
</tr>
<tr>
<td>Introduction</td>
<td>A-1</td>
</tr>
<tr>
<td>1.0 Compiling and Linking</td>
<td>A-1</td>
</tr>
<tr>
<td>2.0 Tape</td>
<td>A-1</td>
</tr>
<tr>
<td>3.0 Executing the Program</td>
<td>A-2</td>
</tr>
<tr>
<td>4.0 Outputs</td>
<td>A-2</td>
</tr>
<tr>
<td>4.1 Output Processing</td>
<td>A-3</td>
</tr>
<tr>
<td>Appendix B - Sample Outputs</td>
<td>B-1</td>
</tr>
<tr>
<td>Appendix C - GDETAP Source Listing</td>
<td>C-1</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

3-1 GDETAP Flow Chart: Main Program................................. 4
3-2 GDETAP Flow Chart: Subroutine EVBRK............................ 5
3-3 GDETAP Flow Chart: Subroutine PPROC........................... 6
3-3 GDETAP Flow Chart: Subroutine EVPROC.......................... 7
4-1 Event File Format.................................................. A-4
B-1 Example of a Combined Event/Commo Report...................... B-1
B-2 Example of a Commo Report....................................... B-2
B-3 Example of an Event Report...................................... B-3
B-4 Example of a Personnel Report................................... B-4
1.0 **Background**

There are two separate sources of digital data from the NTC. For over a year, ARI-POM has received BACKUP tapes created from NTC files created on the CIS VAX 11-780 computers. Files restored from those tapes can be processed by the TRANSLATOR and LOADER programs to create INGRES tables available for research/analysis use. The restored files also directly support the use of the NTC workstations in historian mode.

The second source of digital NTC data is the GDE log tape from the SEL computer that processes RDMS data and calculates position/location data. This tape contains all data passed to the CIS, including unfiltered event data. Comparing the contents of this log tape with the files created from the BACKUP tape will be useful for several reasons, including:

1. Validation of the NTC research data base system,
2. Access to data not available currently, and
3. Possibility of combining data sources to "clean" data.

To access data on the log tapes, it was necessary to develop a VAX program to read them. This has been done: documentation of the tape-reading program GDETAP is the subject of this report.

2.0 **Introduction**

GDETAP is a VAX FORTRAN program which can read the GDE log tape produced at NTC and produce reports/files of the data contained thereon. Program GDETAP consists of the main program, GDETAP, and ten subroutines. The calling structure is represented below:

GDETAP - Main Program
- EVBRK - Process one block of messages
- EVPROC - Process event and commo messages
- REPACK - Reverses order of bytes in the block
- CSCON - Get Date & Time
- JDAY - Returns Mo, Day from Julian Date
- MSCON - Returns time of day
- ECODE - Returns description, given event code
- COMST - Compute commo time deltas
- PPROC - Process player list messages
- REPACK - Reverse order of bytes in the block
- BREP - Process player information
3.0 Program Description

The following are the details of GDETAP input/output and operation. There are three sections, Input, Processing, and Output.

3.1 Input

GDETAP requires two kinds of input, the magnetic tape from NTC, and user inputs.

The magnetic tapes must be mounted as device TAPE$GDE. The DEC Command Language (DCL) command to do this is presented as part of Appendix A, the Users' Guide.

There are two user inputs. The first lets the user specify the output report(s) that he wants. At the present time, there are three kinds of reports available. Samples are included as Appendix B. The three options are Event listings, Communication listings, and Player lists. The option is requested by entering the first letter(s) of all reports requested. For example, to get an event listing the user enters E, while to get all reports he enters ECP. The second user input specifies which tape block processing is to begin. This allows the user to skip preliminary data and get to what he wants. The program will read up to the block indicated, but will not process the data.

3.2 Processing

Program GDETAP operation is graphically represented in the flowcharts presented as figures 3-1 - 3-4. These flowcharts document the high-level logic for the main program, GDETAP, and the three principal subroutines, EVBRK, EVPROC, and PPROC. A short description of each subroutine is presented below, and a complete listing of program GDETAP source code with comments is included as Appendix C.

The GDETAP program reads a magnetic tape generated on the CIS SEL at the National Training Center (NTC) and generates one or two (user option) reports. Report one is an event listing. The user may chose all events, commo events, or just non-commo events. Report two creates a player list, including logical player number, B-Unit number, player type, and a validity code.

EVBRK isolates messages in the physical block read from the tape, then CALLs appropriate processing subroutine.

EVPROC processes all the event type messages encountered in a GDE tape block.

PPROC processes all the player type messages encountered in the GDE tape block.

CSCON gets the time of day and date from an input containing the number of centiseconds since the beginning of the year.

JDAY returns a nine character ASCII string containing the date, given an integer that contains the Julian date+1 (number of days since last 12/31).
MSCON returns an eight character ASCII string containing the time of day, given an integer that contains the number of milliseconds since midnight. COMST checks COMMO messages read from the GDE tape For a KEY DOWN message:
The time of the message is saved in the KEY array. For a KEY UP message:
If there is a time saved for this player and net, calculate the time from KEY DOWN to KEY UP. If there is no time saved for this player and net, save error message 'No Key Down Message.' ECODE decodes the event messages and returns two descriptive ASCII strings to be printed.

3.3 Output

GDETAP produces reports in two different formats. The former is used for the event and commo message reports, and the second is used for the player list reports. Appendix B contains a sample of each type of report.
Figure 3-1
GOETAP Flow Chart: Main Program
Figure 3-2
GOETAP Flow Chart: Subroutine EVBRK
Figure 3-3
GOETAP Flow Chart: Subroutine PPROC

START

REPACK ALL WORDS, THIS SUB-BLOCK
UNPACK MESSAGE COUNT
UNPACK MESSAGE TYPE

PLAYER INITIALIZATION SUB-BLOCK

UNPACK: LOGISTICAL PLAYER NUMBER
B-UNIT NUMBER
TYPE
VALIDITY CODE

BREP
ACCUMULATE PLAYER INFO

WRITE LPN-ORDERED REPORT
WRITE B-UNIT ORDERED REPORT

RETURN
Figure 3-4
GOETAP Flow Chart: Subroutine EVPROC
Appendix A - User's Guide

A-0
Appendix A User's Guide

Introduction.

This appendix was written to document the information necessary to use program GOETAP. It includes instructions for compiling and linking the program, mounting the input tape, executing the program, and processing the output.

1.0 Compiling and Linking.

Program GOETAP was written in VAX FORTRAN. It consists of the main program, GOETAP.FOR, and the following subroutines/files:

<table>
<thead>
<tr>
<th>SUBROUTINE NAME</th>
<th>FILE NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMST</td>
<td>COMST.FOR</td>
</tr>
<tr>
<td>CCON</td>
<td>CCON.FOR</td>
</tr>
<tr>
<td>ECODE</td>
<td>ECODE.FOR</td>
</tr>
<tr>
<td>EVBRK</td>
<td>EVBRK.FOR</td>
</tr>
<tr>
<td>EVPROC</td>
<td>EVPROC.FOR</td>
</tr>
<tr>
<td>JDAY</td>
<td>JDAY.FOR</td>
</tr>
<tr>
<td>MCON</td>
<td>MCON.FOR</td>
</tr>
<tr>
<td>PPROC</td>
<td>PPROC.FOR</td>
</tr>
<tr>
<td>REPACK</td>
<td>REPACK.FOR</td>
</tr>
<tr>
<td>BREP</td>
<td>PPROC.FOR</td>
</tr>
</tbody>
</table>

In addition to the files containing the FORTRAN source code, two files are INCLUDE'd in the source and must be present for the compile step. They are:

- BLOCK.INC
- ALPHA.INC

The command to compile GOETAP is:

```
FOR/LIS/CON=99 GOETAP,EVBRK,EVPROC,ECODE,COMST,PPROC,CCON,MCON,JDAY,REPACK
```

And to link:

```
LINK GOETAP,EVBRK,EVPROC,ECODE,COMST,PPROC,CCON,MCON,JDAY,REPACK
```

2.0 Tape

The tape must be mounted as device TAPE$GDE. The DCL command is:

```
MOUNT/FOREIGN/BLOCKSIZE=2200 MFAO:GDE
```
3.0 Executing the Program

The program is executed by entering:

RUN GDETAP

The program will prompt with:

Please enter the option requested for dumping the raw Tape Data you may enter from one to three letters, depending on the report(s) wanted:

For an Event listing (without commo) enter E
For a listing of commo events, enter C
For a list of player B-Units vs LPN, enter P
For a combination, simply enter the letters for all options requested

What are your choices?

The user must then enter, in any order, any combination of the letters E, P, and C. The program will parrot the selected options, then ask for the starting block number. If no number is entered, the program will start at the beginning of the tape.

4.0 Outputs

GDETAP supports two output formats. One contains MILES and communications events, and the other contains the player listings. The event listing format is described in figure 4-1.

There are two kinds of player listings, the initial load list and the changes. The load list contains the logical player number (LPN), the B-Unit number (in both octal and hex), the player type (undocumented), and a validity code (0/1). The load list is printed twice, first ordered by LPN, and then sorted by B-Unit number. The player change record is:

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-15</td>
<td>PL CH FOR PN</td>
</tr>
<tr>
<td>16-18</td>
<td>LPN</td>
</tr>
<tr>
<td>19-20</td>
<td>BLANK</td>
</tr>
<tr>
<td>21-24</td>
<td>PREVIOUS B-UNIT NUMBER (HEX)</td>
</tr>
<tr>
<td>25-26</td>
<td>BLANK</td>
</tr>
<tr>
<td>27-30</td>
<td>NEW B-UNIT NUMBER (HEX)</td>
</tr>
<tr>
<td>31-35</td>
<td>BLANK</td>
</tr>
<tr>
<td>36-39</td>
<td>PREVIOUS PLAYER TYPE</td>
</tr>
<tr>
<td>40-44</td>
<td>BLANK</td>
</tr>
<tr>
<td>45-48</td>
<td>NEW PLAYER TYPE</td>
</tr>
<tr>
<td>49-53</td>
<td>BLANK</td>
</tr>
<tr>
<td>54-57</td>
<td>PREVIOUS PLAYER VALIDITY</td>
</tr>
<tr>
<td>58-62</td>
<td>BLANK</td>
</tr>
<tr>
<td>63-66</td>
<td>NEW PLAYER VALIDITY</td>
</tr>
</tbody>
</table>
4.1 Output Processing

An auxiliary program, EV Cut, has been written to allow the user to re-process the event listing.

EVCUT automatically deletes messages that are illegal (i.e. not supported by the RDMS). These messages occur frequently as a result of garbled RDMS transmissions. EVCUT also eliminates the MILES SELF TEST message.

The second capability of EV Cut is to eliminate all messages from a given player. For instance, if logical player number (LPN) 67 was reporting a high percentage of illegal messages, the probability is that even those which are apparently valid, are not. As another example, occasionally a B-unit will "stick" and transmit the same message constantly, overwhelming the system. EVCUT can also eliminate this player. The basic operation of EV Cut is as shown below.

```
NEW FILE

ORIGINAL FILE

EVCUT

EXCEPTIONS FILE
```

All messages that are deleted are written to the exceptions file, while the others are written to the new file. The program prompts the user for the names of all these files.

The VAX environment contains a number of tools that can also be used to manipulate the files, including editors and the SORT utility.
<table>
<thead>
<tr>
<th>COLUMN</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>BLANK</td>
</tr>
<tr>
<td>3-8</td>
<td>DATE (E.G. 18 APR)</td>
</tr>
<tr>
<td>9</td>
<td>BLANK</td>
</tr>
<tr>
<td>10-17</td>
<td>TIME EVENT WAS RECEIVED BY RDMS (HH:MM:SS)</td>
</tr>
<tr>
<td>18-21</td>
<td>BLANK</td>
</tr>
<tr>
<td>22-29</td>
<td>TIME EVENT OCCURRED (HH:MM:SS)</td>
</tr>
<tr>
<td>30-32</td>
<td>BLANK</td>
</tr>
<tr>
<td>33-35</td>
<td>LOGISTICAL PLAYER NUMBER</td>
</tr>
<tr>
<td>36-38</td>
<td>BLANK</td>
</tr>
<tr>
<td>39</td>
<td>MILES FLAG (1 FOR MILES EVENT)</td>
</tr>
<tr>
<td>40-42</td>
<td>BLANK</td>
</tr>
<tr>
<td>43</td>
<td>MILES EVENT TYPE</td>
</tr>
<tr>
<td>44-45</td>
<td>BLANK</td>
</tr>
<tr>
<td>46-47</td>
<td>MILES EVENT INFO</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EVENT TYPE</th>
<th>OCTAL EVENT INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10 octal for KILL command acknowledge and 36 octal for RESURRECT.</td>
</tr>
<tr>
<td>1</td>
<td>Weapon code (see Table XIV)</td>
</tr>
<tr>
<td>2</td>
<td>Weapon code (see Table XIV)</td>
</tr>
<tr>
<td>3</td>
<td>Not used</td>
</tr>
<tr>
<td>4</td>
<td>For rounds remaining: When bit 4=1, bits 0-3 contain the most significant digit in rounds remaining count. When bit 4=0, bits 0-3 contain the least significant digit. The rounds remaining event shall follow the weapon fire event. For firer identification: When bits 4.3 = 0.0, bits 0-2 contains the least significant 3 bits (bits 0-2) of the nine bit firer identification. When bits 4.3 = 0.1, bits 0-2 contain the middle three bits (bits 3-5). When bits 4.3 = 1.0, bits 0-2 contain the most significant three bits (bits 6-8). Catenate bits 0-2, bits 3-5, and bits 6-8 to form the 9 bit firer identification (0-330 unique codes).</td>
</tr>
<tr>
<td>5</td>
<td>All 1's for self test passed and all 0's for self test failed.</td>
</tr>
</tbody>
</table>

FIGURE 4-1
Event File Format

A-4
48-49 BLANK
50-51 DISCRETE EVENT CODE

<table>
<thead>
<tr>
<th>VALUE</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not used</td>
</tr>
<tr>
<td>1</td>
<td>Man Worn Laser Detection System Kill</td>
</tr>
<tr>
<td>2</td>
<td>Man Worn Laser Detection System Near Miss</td>
</tr>
<tr>
<td>3</td>
<td>TOW, Dragon, Viper, Vulcan, and Chaparral-MILES Weapon fire</td>
</tr>
<tr>
<td>4</td>
<td>TOW-MILES Kill</td>
</tr>
<tr>
<td>5</td>
<td>TOW-MILES Near Miss</td>
</tr>
<tr>
<td>6</td>
<td>Tactical Radio No. 1 key up</td>
</tr>
<tr>
<td>7</td>
<td>Tactical Radio No. 1 key down</td>
</tr>
<tr>
<td>8</td>
<td>Tactical Radio No. 2 key up</td>
</tr>
<tr>
<td>9</td>
<td>Tactical Radio No. 2 key down</td>
</tr>
<tr>
<td>10-255</td>
<td>Not used</td>
</tr>
</tbody>
</table>

52-53 BLANK
54-123 DESCRIPTION OF EVENT

for MILES events, includes event type and info
for COMMO, contains key down/up and radio net, plus
duration of message for key up

FIGURE 4-1
Event File Format
Appendix B - Sample Outputs
FIGURE B-1
Example of a Combined Event/Commo Report
<table>
<thead>
<tr>
<th>Event Received</th>
<th>Occurred</th>
<th>LPN</th>
<th>MFL</th>
<th>MTY</th>
<th>MEV</th>
<th>EHA</th>
<th>Description</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>17 Apr 23 37 12</td>
<td>23:37:07</td>
<td>63</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>Tac Radio 1 - Key Down</td>
<td>5 sec</td>
</tr>
<tr>
<td>17 Apr 23 37 13</td>
<td>23:37:10</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td>Tac Radio 2 - Key Down</td>
<td>6 sec</td>
</tr>
<tr>
<td>17 Apr 23 37 15</td>
<td>23:37:10</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>Tac Radio 1 - Key Down</td>
<td>5 sec</td>
</tr>
<tr>
<td>17 Apr 23 37 16</td>
<td>23:37:12</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>Tac Radio 1 - Key Up</td>
<td>4 sec</td>
</tr>
<tr>
<td>17 Apr 23 37 20</td>
<td>23:37:10</td>
<td>67</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>Tac Radio 1 - Key Down</td>
<td>5 sec</td>
</tr>
<tr>
<td>17 Apr 23 37 21</td>
<td>23:37:16</td>
<td>67</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>Tac Radio 1 - Key Up</td>
<td>4 sec</td>
</tr>
<tr>
<td>17 Apr 23 37 24</td>
<td>23:37:24</td>
<td>67</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>Tac Radio 1 - Key Down</td>
<td>5 sec</td>
</tr>
<tr>
<td>17 Apr 23 37 25</td>
<td>23:37:30</td>
<td>67</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>Tac Radio 1 - Key Down</td>
<td>5 sec</td>
</tr>
<tr>
<td>17 Apr 23 37 47</td>
<td>23:37:41</td>
<td>67</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>Tac Radio 1 - Key Down</td>
<td>5 sec</td>
</tr>
<tr>
<td>17 Apr 23 37 50</td>
<td>23:37:04</td>
<td>131</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td>Tac Radio 2 - Key Down</td>
<td>7 sec</td>
</tr>
<tr>
<td>17 Apr 23 37 50</td>
<td>23:37:04</td>
<td>131</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td>Tac Radio 2 - Key Down</td>
<td>7 sec</td>
</tr>
<tr>
<td>17 Apr 23 37 50</td>
<td>23:37:48</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>Tac Radio 1 - Key Up</td>
<td>4 sec</td>
</tr>
<tr>
<td>17 Apr 23 37 50</td>
<td>23:37:48</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>Tac Radio 1 - Key Up</td>
<td>4 sec</td>
</tr>
<tr>
<td>17 Apr 23 39 01</td>
<td>23:37:51</td>
<td>167</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>Tac Radio 1 - Key Down</td>
<td>8 sec</td>
</tr>
<tr>
<td>17 Apr 23 39 01</td>
<td>23:37:51</td>
<td>167</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>Tac Radio 1 - Key Down</td>
<td>8 sec</td>
</tr>
<tr>
<td>17 Apr 23 39 02</td>
<td>23:38:00</td>
<td>167</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>Tac Radio 1 - Key Up</td>
<td>4 sec</td>
</tr>
<tr>
<td>17 Apr 23 39 15</td>
<td>23:38:12</td>
<td>111</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td>Tac Radio 2 - Key Down</td>
<td>5 sec</td>
</tr>
<tr>
<td>17 Apr 23 39 21</td>
<td>23:38:17</td>
<td>111</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td>Tac Radio 2 - Key Down</td>
<td>5 sec</td>
</tr>
<tr>
<td>17 Apr 23 39 21</td>
<td>23:38:17</td>
<td>111</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td>Tac Radio 2 - Key Down</td>
<td>5 sec</td>
</tr>
<tr>
<td>17 Apr 23 39 44</td>
<td>23:38:49</td>
<td>63</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>Tac Radio 2 - Key Down</td>
<td>5 sec</td>
</tr>
<tr>
<td>17 Apr 23 39 49</td>
<td>23:38:49</td>
<td>63</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>Tac Radio 2 - Key Down</td>
<td>5 sec</td>
</tr>
<tr>
<td>17 Apr 23 39 55</td>
<td>23:38:51</td>
<td>116</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td>Tac Radio 2 - Key Down</td>
<td>5 sec</td>
</tr>
<tr>
<td>17 Apr 23 39 55</td>
<td>23:38:51</td>
<td>116</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td>Tac Radio 2 - Key Down</td>
<td>5 sec</td>
</tr>
<tr>
<td>17 Apr 23 39 57</td>
<td>23:38:53</td>
<td>116</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td>Tac Radio 1 - Key Up</td>
<td>4 sec</td>
</tr>
<tr>
<td>17 Apr 23 39 00</td>
<td>23:39:55</td>
<td>116</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td>Tac Radio 2 - Key Up</td>
<td>4 sec</td>
</tr>
</tbody>
</table>

**FIGURE B-2**
Example of a Commo Report
Appendix C - GDETAP Source Listing
NAME - GDETAP

AUTHOR - JACK BRISCOE

DATE - 15 APRIL, 1985

DESCRIPTION - The GDETAP program reads a mag tape generated on the CIS SEL at the National Training Center (NTC) and generates one or two (user option) reports. Report one is an event listing. The user may choose all events, commo events, or just non-commo events. Report two creates a player list, including logical player number, B-unit number, player type, and a validity code.

INCLUDE FILES - BLOCK. INC - The input buffer common, containing the input buffer, the current record count, and the options selected by the user.

CALLING SEQUENCE DESCRIPTION - None - Main Program

INPUTS - Mag Tape - Logical name TAPE$GDE

USER INPUTS - Option, starting block number

OUTPUTS - Reports discussed above.

IMPLICIT INTEGER*4 (A-Z)

CHARACTER*3 U

*BLOCK INC contains the read buffer & record count

INCLUDE 'BLOCK. INC/LIST'

DATA RSTART/1/

*Set all option flags FALSE until user chooses

EVENT=. FALSE.

COMMO=. FALSE.

PLAYR=. FALSE.

*Apprise user of his options

PRINT 1010

1010 FORMAT('Please enter the option requested for dumping'
1, 'the raw Tape Data'
2, 'You may enter from one to three letters, depending'
3, 'upon the report(s) wanted'
4, 'For an Event listing (without commo) enter E'
5, 'For a listing of commo events, enter C'
6, 'For a list of player B-Units vs LPN, enter P'
7, 'For a combination, simply enter the letters for all'
8, 'options requested'///,'What are your choices? '*)
*Get his answer - look for up to three characters
ACCEPT 1011.0
1011 FORMAT(A3)
*check each character for every possibility
DO 10 I=1,3
    IF((D(I;I) EQ 'E' OR (D(I;I) EQ 'e')) EVENT= TRUE
    IF((D(I;I) EQ 'C' OR (D(I;I) EQ 'c')) COMM= TRUE
    IF((D(I;I) EQ 'P' OR (D(I;I) EQ 'p')) PLAY= TRUE
10 CONTINUE
*Tell user what he's chosen
PRINT 1012, EVENT, COMM, PLAY
1012 FORMAT(///, ' Options selected were : Event ' ,L1, ' Comm ','L1
               , ' Player list ',L1)
*Write the title on the event/commo listing, if necessary
IF(EVENT OR COMM) WRITE(8,1005)
1005 FORMAT( ' Event Received Occurred LPN MFL MTY MEY EBA ',
               ' Description',
               '---------------------- ------ ------ ------ ------ ------
               '------------------------------------------------------')
*See if the user wants to start at the beginning, or wherever
PRINT 1000
1000 FORMAT(' Which record do you want to start with? '* )
ACCEPT 1001, RSTART
1001 FORMAT(I)
*Open the tape file as FORTRAN UNIT 2. The tape must have been
*mounted as device TAPE$GDE. The command is
OPEN(2,FILE='TAPE$,GDE'
,RECORDTYPE='VARIABLE'
,STATUS='OLD'
,ORGANIZATION='SEQUENTIAL'
,FORM='UNFORMATTED'
,ACCESS='SEQUENTIAL'
,IOSTAT=IOWORD)
*Initialize record count
BLIDNO=RSTART-1
*If RSTART was set, read up to that many records
IF(BLIDNO GT 0) THEN
    DO 25 I=1, BLIDNO
        READ(2, IOSTAT=IOWORD, END=400) BLOCK
25    CONTINUE
ENDIF
*Do until end-of-file
BLIDNO=BLIDNO+1
*Read a record. Each record may be up to 2047 words
Reading
*this way will cause IOSTAT to be set to 67, but the tape will
be properly read

*At end-of-file, go to statement 900 to wrap things up

READ(2,IOSTAT=IOWORD,END=900) BLOCK

*Call appropriate subroutine(s) for selected option(s)
  CALL EVBRK

GOTO 1

900 CONTINUE
REWIND 2
CLOSE(2)
STOP 'Okay'
END
SUBROUTINE EVBRK

NAME - EVBRK

AUTHOR - JACK BRISCOE

DATE - 15 APRIL, 1985

DESCRIPTION - Isolates messages in physical block read from tape, then CALLs appropriate processing subroutine.

INCLUDE FILES

BLOCK INC - BLO COMMON: Read buffer, current block no., flags

CALLING SEQUENCE DESCRIPTION - None

INPUTS - This routine is CALLED each time a new buffer is read into BLOCK.

OUTPUTS - None

SUBROUTINES CALLED

EVPROC - Process events
PPROC - Process player info
MVBITS - Unpack a word (System routine)

IMPLICIT INTEGER*4 (A-Z)

INCLUDE 'BLOCK INC/LIST'

DATA ETYPE/ '0000001D'X/
DATA PTYPE/ '0000001F'X/

NW=1

*ISOLATE SIZE OF CURRENT MESSAGE IN BLOCK
  CALL MVBITS(BLOCK(NW),24,8,SIZE,0)

*WHILE NOT AT END-OF-BLOCK
  DO WHILE (SIZE .NE. 0)

  *ISOLATE MESSAGE TYPE
    CALL MVBITS(BLOCK(NW+1),24,8,TYP,0)

  *IF (THIS IS AN EVENT MESSAGE)
    IF (TYPE .EQ. ETYPE) THEN
      *GO PROCESS MESSAGE FOR MILES OR COMM
        IF (COMM OR EVENT) CALL EVPROC(NW,SIZE)
    ELSE
      IF (TYPE .EQ. PTYPE) THEN
        *PLAYER TYPE MESSAGE
          IF (PLAYR) CALL PPROC(NW,SIZE)
      ENDIF
    ENDIF
  ENDIF
*INCREMENT LOCATION IN BUFFER
  NW=NW+SIZE
  CALL MVBITS(BLOCK(NW), 24, B, SIZE, 0)
END DO
RETURN
END
SUBROUTINE EVPROC(SUB, LEN)

NAME - EVPROC

AUTHOR - JACK BRISCOE

DATE - 15 APRIL, 1985

DESCRIPTION - EVPROC processes all the event type messages encountered in a GDE tape block

INCLUDE FILES:
BLOCK INC - BLO COMMON, Read buffer, current block no., flags

CALLING SEQUENCE DESCRIPTION
SUB - Subscript of first word of message in array BLOCK
LEN - Length of packed message(s)

INPUTS - Array BLOCK, containing all messages

OUTPUTS - Print of each message

SUBROUTINES CALLED
CSCON - Converts centiseconds since midnight to date, time
ECODE - Returns ASCII description of events
REPACK - Swaps bytes in each word

IMPLICIT INTEGER*4 (A-Z)

CHARACTER*8 BTIME, ETIME
CHARACTER*9 BDAY, EDAY
CHARACTER*20 RETSTR
CHARACTER*70 DESC
LOGICAL GO

INCLUDE 'BLOCK INC/LIST'

*MOVE ALL BYTES
DO 100 I=1,LEN
   CALL REPACK(BLOCK(SUB+I-1))
100 CONTINUE

*GET TIME & DATE FOR BIG EVENT BLOCK
CALL CSCON(BLOCK(SUB+3), BDAY, BTIME)

*SET NEW SUBSCRIPT LOCATION - FOR FIRST SUB-BLOCK
NEWSUB = SUB+5

*GET MESSAGE COUNT FOR FIRST SUB-BLOCK
CALL MVBITS(BLOCK(NEWSUB), 8, 8, SLEN, 0)
C
*GET NUMBER OF MESSAGES IN SUB-BLOCK
NMES=(SLEN-1)/2
PRINT 1002, BLIDNO, SUB, LEN, SLEN, NMES, BDAY, BTIME
1002 FORMAT(1BLIDNO=', I12, 'SUB=', I6, 'LEN=', I4, 'SLEN=', I4
1,
   'NMES=', I4, ', ', A9, A8)
C
*DO FOR EACH SUB-BLOCK MESSAGE
DO 200 I=1, NMES
   DESC = ' 
   EVCODE=BLOCK(NEWSUB+1)
   EVTIME=BLOCK(NEWSUB+2)
   CALL ECODE(EVCODE, RETSTR, DESC, GO, EVTIME)
   CALL CSConv(EVTIME, EDAY, ETIME)
   IF (GO) WRITE(8, 1001) BDAY, BTIME, ETIME, RETSTR, DESC
1001 FORMAT(A9, A8, 4X, A8, A22, 2X, A70)
C
*RESET SUBSCRIPT
   NEWSUB=NEWSUB+2
C
200 CONTINUE
   RETURN
END
SUBROUTINE ECODE(CODENO, RETSTR, DESCR, GO, CSTIME)

NAME - ECODE

AUTHOR - JACK BRISCOE

DATE - 15 APRIL, 1985

DESCRIPTION - ECODE Decodes the event messages and returns two descriptive ASCII strings to be printed.

INCLUDE FILES
- BLOCK INC - BLO COMMON: Read buffer, current block no., flags
- ALPHA INC - Event descriptions

CALLING SEQUENCE DESCRIPTION (I=Input, O=Output)
I CODENO - Event code
O RETSTR - ASCII string returned with parsed message
O DESCR - 70-character ASCII string with event description
O GO - Flag set to indicate if post-ECODE processing will be necessary
I CSTIME - Time of event, in centiseconds since midnight

INPUTS - Calling sequence

OUTPUTS - Calling sequence

SUBROUTINES CALLED
- COMST - Calculate message length

IMPLICIT INTEGER*4(A-Z)
CHARACTER*40 WC(0:31), CODES(120)
CHARACTER*30 MILES(0:7)
CHARACTER*55 EBAD(0:9)
CHARACTER*70 DESCR
CHARACTER*20 RETSTR, RR, COMMSG
LOGICAL GO

INCLUDE 'BLOCK.INC/LIST'
INCLUDE 'ALPHA.INC/LIST'

*SET TO WRITE UNLESS TURNED OFF LATER
GO = FALSE.

*SPLIT EVENT INTO COMPONENT PARTS
CALL MVBITS(CODENO, 17, 10, LPN, 0)
CALL MVBITS(CODENO, 8, 1, MFL, 0)
CALL MVBITS(CODENO, 5, 3, MTY, 0)
CALL MVBITS(CODENO, 0, 5, MEV, 0)
CALL MVBITS(CODENO, 0, 7, EBA, 0)
C *NOW PROCESS THE INDIVIDUAL PARTS
WRITE(RESTR,2000) LPN, MFL, MTY, MEV, EBA
2000 FORMAT(5I4)
C *IF(THIS IS A REGULAR EBA - NOT MILES)
IF(MFL EQ. 0) THEN
C *CHECK FOR EBA VALUE VALIDITY - FORCE IT GOOD
IF(EBA .GT. 9) EBA=0
C *SET GO FLAG ACCORDING TO COMMO AND EVENT FLAGS
IF(EBA .GT. 5) THEN
   GO=(COMMO
   CALL COMST(EBA, LPN, CSTIME, COMMSG)
   DSCR(1 30)=EBAD(EBA)(1 30)
   DSCR(31 50)=COMMSG
ELSE
   IF(EBA .GT. 0) GO=EVENT
C *LOAD EBA DESCRIPTION
   DSCR(1 55)=EBAD(EBA)
ENDIF
ELSE
C *IT'S GOTTA BE A MILES EVENT
   DSCR(1 30)=MILES(MTY)
C *SET GO IF EVENT FLAG SET
   GO=EVENT
C *IF(Weapon fire, Hit, Kill, or Near Miss) get weapon
   IF((MTY .GE. 1) .AND. (MTY .LE. 4)) THEN
      DSCR(31 70)=WC(MEV)
   ENDIF
C *IF(ROUNDS REMAINING MILES MESSAGE)
   IF(MTY .EQ. 6) THEN
      C *IF(BIT 4 SET - MOST SIGNIFICANT DIGIT)
         IF(MEV .GE. 16) RRVAL=10*(MEV-16)
      C *OTHERWISE, LEAST SIGNIFICANT DIGIT
         IF(MEV .LT. 16) RRVAL=MEV
      WRITE(RR,2001) RRVAL
      2001 FORMAT(' Rounds remaining', I3)
   DSCR(31 50)=RR
   ENDIF
ENDIF
RETURN
END
SUBROUTINE COMST(EBA, LPN, CST, RETMSG)

NAME - COMST

AUTHOR - JACK BRISCOE

DATE - 15 APRIL, 1985

DESCRIPTION - COMST checks COMM messages read from the GDE tape

  For a KEY DOWN message:
    The time of the message is saved in the KEY array.
  For a KEY UP message:
    If there is a time saved for this player and net,
      Calculate the time from KEY DOWN to KEY UP
    If there is NO time saved for this player and net,
      Save error message 'No Key Down Message'

INCLUDE FILES - None

CALLING SEQUENCE DESCRIPTION

EBA - Message Code (6: KEY DOWN, Net 1;

       7: KEY UP, Net 1;

       8: KEY DOWN, Net 2; and

       9: KEY UP, Net 2)

LPN - Logical Player Number

CST - Message time in centiseconds since midnight

REIMSG - Return message (either elapsed time or error)
          (20 characters max)

INPUTS - From calling sequence

OUTPUTS - To calling sequence

IMPLICIT INTEGER*4 (A-Z)

CHARACTER*20 RETMSG

DIMENSION KEY(2,500)

*Blank out return message
REIMSG = ' '

*Get message between 1 and 4
CMG5 = EBA - 5

*Set Net to either 1 or 2
INDEX = 1
IF(CMSG GT 2) INDEX = 2

*If KEY UP
IF(MOD(CMSG, 2).NE. 0) THEN
  *If there was a key down time saved
    IF(KEY(INDEX, LPN).NE. 0) THEN
*Compute & convert elapsed time to ASCII
DELTA=CST/100-KEY(INDEX,LPN)
WRITE(RETMSG, 1000) DELTA
1000 FORMAT(' Length =', i7, ' sec')
C *Zero out KEY DOWN time
   KEY(INDEX,LPN)=0
C *Else no time saved for KEY DOWN
ELSE
   *Save an error message
   RETMSG = ' No Key Down Message'
ENDIF
C *Else KEY DOWN message
ELSE
   *Save KEY DOWN time
   KEY(INDEX,LPN)=CST/100
ENDIF
RETURN
END
SUBROUTINE PPROC(SUB, LEN)

NAME - PPROC

AUTHOR - JACK BRISCOE

DATE - 15 APRIL, 1985

DESCRIPTION - PPROC processes all the player type messages encountered in a GDE tape block.

INCLUDE FILES

BLOCK.INC - BLO COMMON. Read buffer, current block no., flags

CALLING SEQUENCE DESCRIPTION

SUB - Subscript of first word of message in array BLOCK
LEN - Length of packed message(s)

INPUTS - Array BLOCK, containing all messages

OUTPUTS - Print of each message

SUBROUTINES CALLED

BREP - Writes player change messages
REPACK - Swaps bytes in each word

IMPLICIT INTEGER*4 (A-Z)

LOGICAL FIRST
COMMON/PLAY/LPN, PB, PT, PV, PBU(500), PTY(500), PVA(500), FIRST

DIMENSION PBUT(500), LBUT(500)

INCLUDE 'BLOCK INC/LIST'

DATA FIRST/ TRUE /

BSUB=SUB
C *MOVE ALL BYTES
DO 100 I=I, LEN
   NSUB=BSUB+I-1
   CALL REPACK(BLOCK(NSUB))
100 CONTINUE

C *SET NEW SUBSCRIPT LOCATION - FOR FIRST SUB-BLOCK
NEWSUB=BSUB+3
C
C *GET MESSAGE COUNT FOR FIRST SUB-BLOCK
CALL MVBITS(BLOCK(NEWSUB), 0, 16, SLEN, 0)
C
C *GET MESSAGE TYPE
CALL MVBITS(BLOCK(NEWSUB), 24, 8, TYPE, 0)

PAGE C-12
PRINT 10010, BLDNO, LEN, BSUB, SLEN, TYPE
10010 FORMAT( 'BLIDNO, LEN, BSUB, SLEN, TYPE ', 315, 2115)

NEWSUB=NEWSUB+1
C
*IF(PLAYER INIT SUBBLOCK)
IF(TYPE EQ 150) THEN

DO 200 RNO=NEWSUB, NEWSUB+SLEN-2, 3
LPN=0
PB=0
PT=0
PV=0

CALL MVBITS(BLOCK(RNO ), 0, 16, LPN, 0)
IF(LPN GT. MAXLPN) MAXLPN=LPN
CALL MVBITS(BLOCK(RNO ), 16, 16, PB, 0)
CALL MVBITS(BLOCK(RNO+1), 24, 8, PT, 0)
CALL MVBITS(BLOCK(RNO+1), 16, 8, PV, 0)
CALL BREP
IF((RNO+2). GT. (BSUB+SLEN-1)) GOTO 200

LPN=0
PB=0
PT=0
PV=0

CALL MVBITS(BLOCK(RNO+2), 16, 16, LPN, 0)
IF(LPN GT. MAXLPN) MAXLPN=LPN
CALL MVBITS(BLOCK(RNO+1), 0, 16, PB, 0)
CALL MVBITS(BLOCK(RNO+2), 8, 8, PT, 0)
CALL MVBITS(BLOCK(RNO+2), 0, 8, PV, 0)
CALL BREP

200 CONTINUE
IF(FIRST) THEN
IF(MAXLPN GE. 500) THEN
DO 300 I=1, MAXLPN, 5
WRITE(9, 2000) I, PBU(I), PBU(I), PTY(I), PVA(I)
1, I+1, PBU(I+1), PBU(I+1), PTY(I+1), PVA(I+1)
2, I+2, PBU(I+2), PBU(I+2), PTY(I+2), PVA(I+2)
3, I+3, PBU(I+3), PBU(I+3), PTY(I+3), PVA(I+3)
4, I+4, PBU(I+4), PBU(I+4), PTY(I+4), PVA(I+4)
2000 FORMAT(5(I8,2X,Z4.4,'/',04.4,213))
300 CONTINUE
DO 320 I=1, MAXLPN
PBUT(I)=PBU(I)
320 CONTINUE
DO 360 II=1, MAXLPN
PBSAV=999999
DO 340 I2=1, MAXLPN
IF(PBUT(I2) LT. PBSAV) THEN
PBSAV=PBUT(I2)
ISAV=I2
340 CONTINUE
ENDIF
360 CONTINUE

PAGE C-13
LI3UT(11)=ISAV
PBUT(ISAV)=999999

360 CONTINUE
C *GO TO TOP OF NEXT PAGE FOR B-UNIT ORDERED LIST
WRITE(9, 2002)
FORMAT(’1’)
C
*PRINT B-UNIT ORDERED LIST
DO 380 J=1, MAXLPN, 5
  I=LBUT(J)
  IP1=LBUT(J+1)
  IP2=LBUT(J+2)
  IP3=LBUT(J+3)
  IP4=LBUT(J+4)
  WRITE(9, 2000) I, PBUT(I), PTY(I), PVA(I)
  IF(.NOT.FIRST) THEN
    WRITE(9, 1000) I, PBUT(I), PTU(I), PT, PVA(I)
    WRITE(9, 1000) I, PBUT(I), PTU(I), PT, PVA(I)
  ENDIF
  CONTINUE
ENDIF
ENDIF
IF(MAXLPN .GE. 500) FIRST=.FALSE.
ENDIF
RETURN
END

SUBROUTINE BREP

LOGICAL FIRST

COMMON/PLAY/LPN, PB, PT, PV, PBU(500), PTY(500), PVA(500), FIRST

IF((FIRST) .OR. (.NOT. FIRST)) .AND.
  ((PBU(LPN) .NE. PB) .OR.
   (PTY(LPN) .NE. PT) .OR.
   (PVA(LPN) .NE. PV))) THEN:
  IF(.NOT. FIRST) THEN
    PRINT 1000, LPN, PBU(LPN), PB, PTY(LPN), PT, PVA(LPN), PV
    WRITE(9, 1000) LPN, PBU(LPN), PB, PTY(LPN), PT, PVA(LPN), PV
    FORMAT(’PL CH FOR LPN ’, I3, 2Z6.4, 419)
  ENDIF
  PBU(LPN)=PB
  PTY(LPN)=PT
  PVA(LPN)=PV
ENDIF
RETURN
END
SUBROUTINE CSCON(WORDIN, DATE, TIMOUT)

NAME - CSCON

AUTHOR - JACK BRISCOE

DATE - 15 APRIL, 1985

DESCRIPTION - CSCON gets the time of date and date from an input containing the number of centiseconds since the beginning of the year.

INCLUDE FILES - None

CALLING SEQUENCE DESCRIPTION

WORDIN - Word containing centiseconds since 12/31 of previous year.

DATE - 9 character date returned (i.e. 10-JUN with 2 blanks leading, 1 following)

TIMOUT - 8 character time of day (i.e. 13:03:35)

INPUTS - From calling sequence

OUTPUTS - To calling sequence

SUBROUTINES CALLED - JDAY - Returns date for julian date
MSCON - Returns time of day for milliseconds since midnight

IMPLICIT INTEGER*4(A-Z)
CHARACTER*9 DATE
CHARACTER*8 TIMOUT
REAL*8 CSTIM, CSVAL

DATA CSVAL/8640000.0/

*CONVERT TIME TO REAL*8 TO SOLVE SIGN PROBLEM
CSTIM=DBLE(ABS(WORDIN))
*If sign bit was set, add the value back in
IF(WORDIN.LT.0) CSTIM=CSTIM+2.0**31
*JD is really julian date-1
JD=INT(CSTIM/CSVAL)
*Get remainder in milliseconds
MS=10*INT(CSTIM-JD*CSVAL)
*If Julian date looks kosher
IF((JD)GE.0).AND. (JD.LE.366) THEN
*Get the date in English
CALL JDAY(JD,DATE)
ENDIF
*Put the time into as ASCII string
CALL MSCON(MS, TIMEOUT)
RETURN
END
SUBROUTINE MCON(MS, TIME)

NAME - MCON

AUTHOR - JACK BRISCOE

DATE - 15 APRIL, 1985

DESCRIPTION - MCON returns an 8-character ASCII string containing the time of day, given an integer that contains the number of milliseconds since midnight.

INCLUDE FILES - None

CALLING SEQUENCE DESCRIPTION
MS - Word containing milliseconds since midnight
TIME - 8 character time of day (i.e. 13:03:35)

INPUTS - From calling sequence

OUTPUTS - To calling sequence

SUBROUTINES CALLED - None

IMPLICIT INTEGER*4 (A-Z)
CHARACTER*8 TIME

*Convert milliseconds, first to seconds, then to
* hours, minutes, and seconds
TSEC=MS/1000
HRS = TSEC/3600
MIN = (TSEC-HRS*3600)/60
SEC = (TSEC-HRS*3600-MIN*60)

*Now put time into an ASCII string HH:MM:SS
WRITE(TIME,2000) HRS,MIN,SEC

2000 FORMAT(I2,':',I2,':',I2)
RETURN
END
SUBROUTINE JDAY(JD, RETSTR)

NAME - JDAY

AUTHOR - JACK BRISCOE

DATE - 15 APRIL, 1985

DESCRIPTION - JDAY returns a 9-character ASCII string containing the date, given an integer that contains the Julian date+1 (number of days since last 12/31).

INCLUDE FILES -- None

CALLING SEQUENCE DESCRIPTION

JD - Word containing days since 12/31 previous year
RETSTR - 9 character date i.e. ##10#JUN#, where # represents a blank character.

INPUTS - From calling sequence

OUTPUTS - To calling sequence

SUBROUTINES CALLED - None

IMPLICIT INTEGER*4 (A-Z)
CHARACTER*9 RETSTR
CHARACTER*3 DST, MON(12)
DIMENSION MONNO(0:12)
DATA MON/'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
        'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'/
DATA MONNO/0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334, 365/

C *Blank out output ASCII string to start
RETSTR='   '
I=0
DO WHILE ((JD.GE.MONNO(I)) .AND. (I.LE.12))
   I=I+1
END DO
IF(I.LE.12)THEN
   *Set 3-character month name into characters 6-8
   RETSTR(6:8)=MON(I)
   *Encode ASCII day of month into characters 3-5
   DDIF=JD-MONNO(I-1)
   WRITE(DST,1001) DDIF
   1001 FORM(12. ' ')
   RETSTR(3:5)=DST
ENDIF
RETURN
END
SUBROUTINE REPACK(WORD)

NAME - REPACK

AUTHOR - JACK BRISCOE

DATE - 15 APRIL, 1985

DESCRIPTION - REPACK takes a 32-bit word and reverses the order of the bytes. If the bytes were originally numbered 1234, they would wind up 4321.

INCLUDED FILES - None

CALLING SEQUENCE DESCRIPTION

WORD - The word in which the bytes are to be reordered.

INPUTS - See calling sequence

OUTPUTS - See calling sequence

SUBROUTINES CALLED

MVBITS - System subroutine to manipulate bits

IMPLICIT INTEGER*4 (A-Z)

CALL MVBITS(WORD, 0, 8, NUWORD, 24)
CALL MVBITS(WORD, 8, 8, NUWORD, 16)
CALL MVBITS(WORD, 16, 8, NUWORD, 8)
CALL MVBITS(WORD, 24, 8, NUWORD, 0)
WORD=NUWORD
RETURN
END