MEASUREMENT OF COMPRESSION OF THE MODIFIED READ CODE II

NOVEMBER 1982

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED
The report describes the work on two basic tasks as listed below:

Task 1 - Measure Modified READ II Compression

It is generally agreed that the Modified READ II code is a prime candidate for use in Group 4 facsimile equipment. The Modified READ II code is the same as the code defined in CCITT Recommendation T.4 with the following exceptions.
- K-Factor 00
- Minimum Scan Line Time 0
- No end of line coded between scan lines
- Two end of line codes at the end of a transmitted page as part of Task 1.

30 computer simulation runs were performed for every combination of the following three parameters.

- Resolution (Lines/inch) - 200, 240, 300, 400, 480
- CCITT Image No. - 1, 5, 7
- Wrap Around - In, Out

Wrap Around refers to the technique where a run length is not necessarily terminated at the end of a scan line. Instead, the coding process is continuous from line to line.

Delta Information Systems measured compression two different ways as listed below:

1. Compression Ratio - Total image pels/Transmitted bits
2. Total transmitted bits

Task 2 - Measure Scan Line Statistics for the Modified READ Code II

Under this task, Delta Information Systems measured the bits/line statistics for the fifteen images processed in Task 1 - 200, 240, 300, 400, and 480 lines/inch; CCITT Images 1, 5, 7. The statistical data includes the minimum bits/line, maximum bits/line, average bits/line, and standard deviation.
FOREWORD

Among the responsibilities assigned to the Office of the Manager, National Communications System, is the management of the Federal Telecommunication Standards Program. Under this program, the NCS, with the assistance of the Federal Telecommunication Standards Committee identifies, develops, and coordinates proposed Federal Standards which either contribute to the interoperability of functionally similar Federal telecommunication systems or to the achievement of a compatible and efficient interface between computer and telecommunication systems. In developing and coordinating these standards a considerable amount of effort is expended in initiating and pursuing joint standards development efforts with appropriate technical committees of the Electronic Industries Association, the American National Standards Institute, the International Organization for Standardization, and the International Telegraph and Telephone Consultative Committee of the International Telecommunication Union. This Technical Information Bulletin presents an overview of an effort which is contributing to the development of compatible Federal, national, and international standards in the area of digital facsimile standards. It has been prepared to inform interested Federal activities of the progress of these efforts. Any comments, inputs or statements of requirements which could assist in the advancement of this work are welcome and should be addressed to:

Office of the Manager
National Communications System
ATTN: NCS-TS
Washington, D.C. 20305
(202) 692-2124
MEASUREMENT OF COMPRESSION
OF THE MODIFIED READ CODE II

FINAL REPORT
November 1, 1982

Modification P0006 to
Contract No. DCA100-80-C-0042

Submitted to:
NATIONAL COMMUNICATIONS SYSTEM
OFFICE OF TECHNOLOGY AND STANDARDS
Washington, D.C. 20305

Contracting Agency:
DEFENSE COMMUNICATIONS AGENCY

Submitted by:
DELTA INFORMATION SYSTEMS, INC.
310 Cottman Street
Jenkintown, Pa. 19046
# Measurement of Compression of the Modified READ Code II

## Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 Introduction</td>
<td>1.0</td>
</tr>
<tr>
<td>2.0 Modified READ Code II Algorithm</td>
<td>2.0</td>
</tr>
<tr>
<td>3.0 Computer Program Overview</td>
<td>3.0</td>
</tr>
<tr>
<td>4.0 Compression Data for the Modified READ Code II</td>
<td>4.0</td>
</tr>
<tr>
<td>5.0 Scan Line Statistics for the Modified READ Code II</td>
<td>5.0</td>
</tr>
</tbody>
</table>

## Appendices

A. Code Listing for the Program to Simulate Modified READ II without Wrap-Around

B. Code Listing for the Program to Simulate Modified READ II with Wrap-Around
1.0 **INTRODUCTION**

This document summarizes work performed by Delta Information Systems, Inc. for the Office of Technology and Standards of the National Communications System, an organization of the U.S. Government, under Modification P00006 to Contract DCA100-80-C-0042. The Office of Technology and Standards, headed by National Communications System Assistant Manager Marshall L. Cain, is responsible for the management of the Federal Telecommunications Standards Program, which develops telecommunication standards whose use is mandatory by all Federal agencies.

Under the basic contract DCA100-80-C-0042 Delta Information Systems has analyzed alternative resolutions for Group 4 facsimile. The final report for this study was issued in August 1982. As part of this investigation four test documents were scanned with five candidate resolutions, and the results of all twenty scans were printed. The test documents and resolutions used in this study are listed below.

<table>
<thead>
<tr>
<th>Test Images</th>
<th>Resolutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCITT Image No. 1</td>
<td>200 lines/inch</td>
</tr>
<tr>
<td>CCITT Image No. 5</td>
<td>240 lines/inch</td>
</tr>
<tr>
<td>CCITT Image No. 7</td>
<td>300 lines/inch</td>
</tr>
<tr>
<td>Legibility Test Chart</td>
<td>400 lines/inch</td>
</tr>
<tr>
<td></td>
<td>480 lines/inch</td>
</tr>
</tbody>
</table>

After these 20 images were printed it was noted that several pages had artifacts and noisy pels around the edges. These artifacts were removed on subsequent projects in preparation for
the subject simulation study.

The subject project has two basic tasks as listed below.

**Task 1 - Measure Modified READ II Compression:**

It is generally agreed that the Modified READ II code is a prime candidate for use in Group 4 facsimile equipment. The Modified READ II code is the same as the code defined in CCITT Recommendation T.4 with the following exceptions:

- K-Factor
- Minimum Scan Line Time
- No end of line codes between scan lines
- Two end of line codes at the end of a transmitted page

As part of Task 1, 30 computer simulation runs were performed for every combination of the following three parameters:

- Resolution (lines/inch) - 200, 240, 300, 400, 480
- CCITT Image No. - 1, 5, 7
- Wrap Around - In, Out

Wrap Around refers to the technique where a run length is not necessarily terminated at the end of a scan line. Instead, the coding process is continuous from line to line.

Delta Information Systems measured compression two different ways as listed below:

1. Compression Ratio - Total image pels/Transmitted bits
2. Total transmitted bits

**Task 2 - Measure Scan Line Statistics for the Modified READ Code II:**

Under this task, Delta Information Systems measured the bits/line statistics for the fifteen images processed in Task 1.
300, 400, and 480 lines/inch, CCITT Images 1, 5, 7. The statistical data includes the minimum bits/line, maximum bits/line, average bits/line, and standard deviation.

The test results for tasks 1 and 2 are included in sections 4.0 and 5.0 respectively. Section 2.0 describes the wrap-around algorithm which was simulated while section 3.0 provides an overview of the computer program used in the simulation. Appendices A and B contain the code listings for the program to simulate the Mod READ II code with and without wrap-around respectively.
2.0 MODIFIED READ CODE II ALGORITHM

No Wrap-Around

The algorithm for the basic Modified READ Code is defined in CCITT Recommendation T.4 entitled "Standardization of Group 3 Facsimile Apparatus for Document Transmission". It is generally agreed that a strong candidate for the Group 4 coding technique is a variation of the Group 3 Modified READ Code. Table 2-1 is a list of the parameters which differ for the two algorithms.

Table 2-1
Comparison of Group 3 Modified READ Code and Potential Group 4 Modified READ II Code

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Group 3 T.4 Recommendation</th>
<th>Modified READ Code II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Scan Line Time</td>
<td>20 MS</td>
<td>0 MS</td>
</tr>
<tr>
<td>K-Factor @ 200 lpi</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>End of Line Code between Scan Lines</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>No. of EOL Codes at End of Page</td>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>

Note that in the MRCII code EOL codes are used to change from the one-dimensional coding to the two-dimensional coding or vice versa in addition to two EOL's at the end of the page. This means that for the infinite K factor the first two code lines are preceded by EOL's.
Wrap-Around

The Wrap-Around option provides for images to be treated as a continuous pel stream in which the rightmost pel on a line is followed immediately by the leftmost pel on the next scan line. Otherwise the algorithm is the same as that for no wrap-around. Both the reference line and the coding line continue into the following line with no artificial transition added at line boundaries. This procedure is repeated until end-of-page is reached. That is, if the reference line is \( n \) and the coding line is \( n+1 \), then the reference line is continued into \( n+1 \) and the coding line is continued into \( n+2 \). One consequence of this procedure is that long runs in horizontal mode are generated for a sequence of all one-color lines.
3.0 COMPUTER PROGRAM OVERVIEW

The computer programs used to measure the compression of the Mod READ II code are described in this section. Two different programs have been written, one for coding without wrap-around and the other with wrap-around. Code listings of these programs appear in appendices.

3.1 Computer Program without Wrap-Around

This program simulates both the encode and decode processes. To initiate the simulation process, the operator must type in a set of input parameters. The insertion of the input parameters is accomplished on an interactive basis with prompting. After the data has been entered and the measurement parameters have been selected, the first step in the simulation process is the "ENCODE" function. This function detects color changes in the input data and constructs the appropriate code word by table look-up or algorithm. The encoded signal is reversed and fed to the decode function. The decoder basically performs the inverse function of the encoder, generating a series of lines of image pels. The simulation process provides a printout of all the computed performance data as well as a summary tabulation of the input parameters.

The following section describes the structure of the computer program written to simulate the MRCII code. A brief description of each of the functions/subroutines follows:
MRC2
The MRC2 program controls the decoding process. The simulation process is "decode driven;" that is, the main program controls the decode process which decodes a buffered line of compressed data. When the contents of the buffer have been used up, a new line of data is encoded. The MRC2 program also controls parameter input via the INIT routines, and reports computed results.

INIT2
The INIT2 subroutine controls parameter input interactively, prints a summary of the input parameters, and initializes variables.

GETL2
The GETL2 subroutine retrieves a number of requested bits from the coded line and delivers the bits packed into a word (right justified). End-of-line codes (EOL) are detected. If the number of coded bits requested by the calling program is not available, the ENCD2 subroutine is called to provide them.

ENCD2
This subroutine supplies a line of compressed data. Color transitions on an input line are detected bit-by-bit. Both one-dimensional and two-dimensional lines are encoded depending on the parameter K. The code word is generated by table look-up, or algorithm, as appropriate, and added to the coded line buffer via CODELN and/or CODENG.

CODELN
The subroutine CODELN is called by ENCD2 to look up the
Modified Huffman Code (MHC) corresponding to a given run length and color, and add the code word to the coded line buffer.

**CODENG**

The subroutine CODENG performs a similar function for the two-dimensional case. Based on a particular feature, the appropriate code word is generated by table look-up or algorithm and added to the coded line buffer. All code tables for both one-dimensional and two-dimensional codes are stored in labelled common which is initialized by a BLOCK DATA subprogram.

**ONED2**

The ONED2 subroutine decodes the MHC. It extracts a set of n bits (n=3 initially) from the coded line and looks for a match with all code words of length n, increasing n until a match is found or the code table is exhausted. When and if a match is found, the indicated bits are constructed on the output line.

**TWOD2**

This subroutine performs the same function as ONED2 for the two-dimensional line.

**M12B and I4B**

The subprograms M12B and I4B are used to pack and unpack a set of bits into (or from) an array of words.
3.2 Computer Program with Wrap-Around

Since the Wrap-Around algorithm requires information from many input lines at the same time, the structure of the program described in the previous section was not suitable for modification to accommodate wrap-around. Also, since the programs are designed to operate in an error free environment, there is no need to inject errors or decode the coded data in order to measure compression. Therefore a new program was written to accommodate wrap-around efficiently, without the decode function. The modules included in this program are described below:

**MRCW**

The main program, MRCW, simply calls the initialization and encoding routines and reports the results.

**ENCDW**

This routine performs the one-dimensional and two-dimensional algorithms as required and builds the resultant code lines. Input to this routine is obtained from a list of black-white or white black transitions stored in a circular list in memory.

**GTRAN**

The subroutine GTRAN scans the input image data for transition and adds the next transition encountered along the scan line to the circular list called TRANS. Transitions are added to the bottom of the list by GTRAN and taken from the top of the list by ENCDW. One transition is added by GTRAN each time it is called. Care must be taken to ensure that the list does not go empty (TOP passes BOTTOM) or that the list overflows (BOTTOM passes TOP).
INIT2

This subroutine is used to enter input parameters interactively and to initialize variables. It performs the initial fill of the transition list.

CODNG & CODLN

These subroutines perform the same functions as for the no wrap-around case, except that CODLN has been modified to accommodate very long runlengths.
4.0 COMPRESSION DATA FOR THE MODIFIED READ CODE II

A computer program was written to measure the number of transmitted bits/page for each of the three CCITT test documents (1, 5, 7) at each of five resolutions (200, 240, 300, 400, 480 lpi). The bits/pg was measured both with and without wrap-around as the algorithms are described in section 2.0. The results of these simulation runs are tabulated in table 4-1. The number of bits/page and the compression ratio is listed for each combination of test document and resolution. The average for the three test CCITT images is also provided. The compression ratio is computed by dividing the number of pels in each test image by the number of compressed bits/pg. The number of pels in each image is listed below.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Pels/line</th>
<th>Lines/image</th>
<th>Pels/image</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>1,728</td>
<td>2,336</td>
<td>4,036,608</td>
</tr>
<tr>
<td>240</td>
<td>2,048</td>
<td>2,800</td>
<td>5,734,400</td>
</tr>
<tr>
<td>300</td>
<td>2,560</td>
<td>3,500</td>
<td>8,960,000</td>
</tr>
<tr>
<td>400</td>
<td>3,456</td>
<td>4,672</td>
<td>16,146,432</td>
</tr>
<tr>
<td>480</td>
<td>4,096</td>
<td>5,600</td>
<td>22,937,600</td>
</tr>
</tbody>
</table>

Figures 4-1 and 4-2 are graphs of the data in table 4-1 (without wrap-around) for the compressed bits/page and the compression ratio respectively. Note that the compression ratio increases linearly as a function of resolution. Also note that the compression ratio approximately doubles when the resolution doubles. This causes the number of compressed bits/pg to also double as the resolution doubles since the number of pels/image...
Table 4-1
COMPRESSION DATA FOR THE MODIFIED READ CODE II

<table>
<thead>
<tr>
<th>CCITT IMAGE</th>
<th>RESOLUTION LPI</th>
<th>NO WRAP AROUND</th>
<th>WITH WRAP AROUND</th>
<th>% DIFF.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NO WRAP AROUND</td>
<td>WITH WRAP AROUND</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BITS</td>
<td>COMP. RATIO</td>
<td>BITS</td>
<td>COMP. RATIO</td>
</tr>
<tr>
<td>200</td>
<td>132,057</td>
<td>30.57</td>
<td>140,215</td>
<td>28.79</td>
</tr>
<tr>
<td>240</td>
<td>156,932</td>
<td>36.54</td>
<td>170,073</td>
<td>33.72</td>
</tr>
<tr>
<td>300</td>
<td>197,170</td>
<td>45.44</td>
<td>218,763</td>
<td>40.96</td>
</tr>
<tr>
<td>400</td>
<td>271,040</td>
<td>59.57</td>
<td>308,496</td>
<td>52.34</td>
</tr>
<tr>
<td>480</td>
<td>326,525</td>
<td>70.25</td>
<td>386,287</td>
<td>59.39</td>
</tr>
<tr>
<td>200</td>
<td>229,256</td>
<td>17.61</td>
<td>230,370</td>
<td>17.52</td>
</tr>
<tr>
<td>240</td>
<td>273,023</td>
<td>21.00</td>
<td>274,976</td>
<td>20.85</td>
</tr>
<tr>
<td>300</td>
<td>345,827</td>
<td>25.91</td>
<td>349,976</td>
<td>25.60</td>
</tr>
<tr>
<td>400</td>
<td>467,330</td>
<td>34.55</td>
<td>476,159</td>
<td>33.91</td>
</tr>
<tr>
<td>480</td>
<td>570,354</td>
<td>40.22</td>
<td>583,568</td>
<td>39.31</td>
</tr>
<tr>
<td>200</td>
<td>531,782</td>
<td>7.59</td>
<td>534,039</td>
<td>7.56</td>
</tr>
<tr>
<td>240</td>
<td>628,491</td>
<td>9.12</td>
<td>631,118</td>
<td>9.09</td>
</tr>
<tr>
<td>300</td>
<td>783,644</td>
<td>11.43</td>
<td>788,697</td>
<td>11.36</td>
</tr>
<tr>
<td>400</td>
<td>1,041,310</td>
<td>15.50</td>
<td>1,050,767</td>
<td>15.37</td>
</tr>
<tr>
<td>480</td>
<td>1,262,786</td>
<td>18.16</td>
<td>1,276,678</td>
<td>17.97</td>
</tr>
<tr>
<td>200</td>
<td>297,769</td>
<td>13.56</td>
<td>301,541</td>
<td>13.39</td>
</tr>
<tr>
<td>240</td>
<td>352,815</td>
<td>16.25</td>
<td>358,722</td>
<td>15.99</td>
</tr>
<tr>
<td>300</td>
<td>442,217</td>
<td>20.26</td>
<td>452,479</td>
<td>19.80</td>
</tr>
<tr>
<td>400</td>
<td>593,226</td>
<td>27.22</td>
<td>611,807</td>
<td>26.39</td>
</tr>
<tr>
<td>480</td>
<td>719,888</td>
<td>31.86</td>
<td>748,844</td>
<td>30.63</td>
</tr>
</tbody>
</table>
increases by 4 to 1 as the resolution doubles.

The Wrap Around coding technique was chosen for analysis with the anticipation that it might provide more compression than the non wrap around technique. However, the data in table 4-1 shows that the compression for Wrap Around is consistently poorer than the conventional Mod READ Code II (MRCII) algorithm. The following paragraph examines the Wrap Around technique to give some indication as to why the compression is reduced rather than increased.

The conventional MRCII algorithm transmits an all white line with one bit; this is very efficient. With Wrap Around the average number of bits for a number of consecutive white lines is increased to eight to nineteen per line depending on the resolution. It is this inefficient consecutive all-white-line coding which is primarily responsible for the poorer performance of Wrap Around.

The last column in Table 4-1 is a list of the percentages by which the wrap-around compressions are reduced relative to the non-wrap-around compressions. Notice how the percentage reductions are far greater for the English Letter than they are for the Kanji image. This is due to the fact that there are more all-white lines in the English letter than there are in the Kanji page. Also notice how the percentage generally increases with resolution. This is due to the fact that the number of all-white lines increase with resolution.

This inefficient white-line coding could be improved by transmitting the number of extended runlengths of 2560 pels instead of repeating the code for 2560. However that would require a change to the MRC code table.
5.0 SCAN LINE STATISTICS FOR THE MODIFIED READ CODE II

The computer program which was written to measure the compression of the Modified READ Code II also measured the statistics for the number of bits per scan line for each of the test documents. The following bits/line statistics were measured for each of the test documents:

- Minimum bits/line
- Maximum bits/line
- Average bits/line
- Standard deviation

Table 5-1 is a tabulation of these statistics for each test document scanned at each of the five resolutions. Note that the minimum bits/line for each simulation is 1. This occurs because an all-white line is transmitted with one bit regardless of the number of pels in the line. Also note that the average number of bits/line and the standard deviation is relatively independent of the resolution. This is due to the fact that the transmitted bits per line is largely based upon the number of transitions per line and the number of transitions per line is independent of resolution.

The maximum number of bits/line varies over a relatively wide range for different resolutions due to the totally different locations of scan lines and sampling pels relative to the image.
<table>
<thead>
<tr>
<th>CCITT IMAGE</th>
<th>RESOLUTION LPI</th>
<th>MINIMUM BITS/LINE</th>
<th>MAXIMUM BITS/LINE</th>
<th>AVERAGE BITS/LINE</th>
<th>STANDARD DEVIATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENGLISH</td>
<td>200</td>
<td>1</td>
<td>641</td>
<td>56.5</td>
<td>110.9</td>
</tr>
<tr>
<td></td>
<td>240</td>
<td>1</td>
<td>594</td>
<td>56.0</td>
<td>110.7</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>1</td>
<td>760</td>
<td>56.3</td>
<td>118.4</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>1</td>
<td>680</td>
<td>58.0</td>
<td>115.3</td>
</tr>
<tr>
<td></td>
<td>480</td>
<td>1</td>
<td>659</td>
<td>58.3</td>
<td>117.3</td>
</tr>
<tr>
<td>FRENCH JOURNAL</td>
<td>200</td>
<td>1</td>
<td>907</td>
<td>98.1</td>
<td>132.5</td>
</tr>
<tr>
<td></td>
<td>240</td>
<td>1</td>
<td>778</td>
<td>97.5</td>
<td>133.1</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>1</td>
<td>755</td>
<td>98.8</td>
<td>127.1</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>1</td>
<td>1124</td>
<td>100.0</td>
<td>138.6</td>
</tr>
<tr>
<td></td>
<td>480</td>
<td>1</td>
<td>1256</td>
<td>101.8</td>
<td>144.0</td>
</tr>
<tr>
<td>KANJI</td>
<td>200</td>
<td>1</td>
<td>507</td>
<td>227.6</td>
<td>123.2</td>
</tr>
<tr>
<td></td>
<td>240</td>
<td>1</td>
<td>514</td>
<td>224.4</td>
<td>123.4</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>1</td>
<td>530</td>
<td>123.9</td>
<td>127.4</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>1</td>
<td>559</td>
<td>222.9</td>
<td>123.4</td>
</tr>
<tr>
<td></td>
<td>480</td>
<td>1</td>
<td>605</td>
<td>225.5</td>
<td>126.2</td>
</tr>
</tbody>
</table>

* NO WRAP AROUND
APPENDIX A
CODE LISTING FOR THE MODIFIED READ
CODE II WITHOUT WRAP-AROUND

<table>
<thead>
<tr>
<th>SUBROUTINE</th>
<th>PAGE NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. MRC2</td>
<td>A-1</td>
</tr>
<tr>
<td>2. INIT2</td>
<td>A-7</td>
</tr>
<tr>
<td>3. ENCD2</td>
<td>A-11</td>
</tr>
<tr>
<td>4. GETL2</td>
<td>A-17</td>
</tr>
<tr>
<td>5. BLOCK DATA</td>
<td>A-19</td>
</tr>
<tr>
<td>6. ONED2</td>
<td>A-25</td>
</tr>
<tr>
<td>7. TWOD2</td>
<td>A-28</td>
</tr>
<tr>
<td>8. CODNG</td>
<td>A-32</td>
</tr>
<tr>
<td>9. CODLN</td>
<td>A-34</td>
</tr>
<tr>
<td>10. INOUT</td>
<td>A-36</td>
</tr>
</tbody>
</table>
PROGRAM MRC2
IMPLICIT INTEGER(A-Z)

ASSUMPTIONS:
MAXIMUM LINE LENGTH=4996
MAXIMUM NUMBER OF LINES=5698
MAXIMUM INPUT RECORD SIZE=256

COMMON VARIABLES - DEFINITION

IMLNO - INPUT LINE NUMBER
OTLNO - OUTPUT LINE NUMBER
OTELW - NO. WORDS IN OUTPUT LINE
INELP - NOT USED
CDELP - CODE LINE ELEMENT POINTER
OTELE - OUTPUT LINE ELEMENT POINTER
CDELW - CODES CONTAINING CODED DATA BITS
CDDATA - NO. OF CODED BITS ON A LINE
CDELC - NO. OF CODED DATA BITS ON A LINE
CINELC - SPECIFIED BITS ON INPUT LINE
CTDATA - TOTAL CODED DATA BITS IN IMAGE
TCDEL - TOTAL CODED BITS IN IMAGE

ERRPNT -
ERRFF -
ERPLM -
ERRCNT - TOTAL ERRORS IN IMAGE
INLNC - NO. OF INPUT LINES PROCESSED
CONSEC - CONSECUTIVE EOL'S READ
ONECNT - NOT USED
LNOHOF - LINE NUMBER BUFFER
KCHT - NOT USED
ZERO - COUNT OF ZEROS IN CODED LINE
NBPV - NO. OF BITS PER WORD
INCOD - CODE LINE POINTER (INPUT)
INREF - REFERENCE LINE POINTER (INPUT)
OTCOD - CODE LINE POINTER (OUTPUT)
OTREF - REFERENCE LINE POINTER (OUTPUT)
STFTB - NO. OF 2-DIMENSIONAL LINES IN IMAGE
ERROR -
BUFIM - INPUT RECORD SIZE DESIRED

**********LABELLED COMMON /G16BT/ ************

DIMENSION MASK(16),COMASK(16),LIBIT(16),LZBIT(16)
COMMON /G16BT/MASK,COMASK,LIBIT,LZBIT,NBPV

**********LABELLED COMMON ARRAYS************

DIMENSION PELREF(256),PELBUF(256),PELCOD(256),PELTREF(256),PELOUTC(256)
EQUIVALENCE (PELREF,PELBUF),(PELCOD,PELBUF(1,2))
EQUIVALENCE (PELREF,PELBUF),(PELCOD,PELBUF(1,2))
COMMON/BUFF/PELBUF(256,2),OTBUF(256,2),
& CDBUF(1924),STFBUF(1924),STAT(1)
COMMON/HUFF/CODE(3,105,2),CORDERD(3,11)
COMMON/ERAV/ERRORS(100)
C**********FILE BUFFERS**************
COMMON/FILES/TERM,LPFL,ICDB7(144),ICDB8(144),ICDB9(144)
INTEGER TERM,LPFL,ICDB7,ICDB8,ICDB9
C IDCB7 - PELFIL
C IDCB8 - OFIL
C IDCB9 - SDFIL
C**********LABELLED COMMON VARIABLES**********
C COMMON/IVAR/PELMAX,VRES,EPHASE,CMPMAX,ERRMOD,LINMAX,K
COMMON/PVAR/INLINO,DLLNHO,OCOLW,CDEL,OTELP,CDELW,CDDATA,
CDELC,INELC,TCDATA,TCDEL,ERRPTN,ERROFF,EROLL,
C ERRCNT,INLCNT,CONSEC,LMNOB,ZERO,
C INCOC,ONREF,OTCOD,OTREF,STFB,ERRCOR,BUFDIM
C COMMON/LOGIC/SEARCH,DIAG,SYNC,WRITE,LEFT,CHCOL,ONE,WHITE,FLEN,
C OUTF
C LOGICAL SEARCH,DIAG,SYNC,WRITE,LEFT,CHCOL,ONE,WHITE,FLEN,
C OUTF
C DOUBLE PRECISION TCDATA,TCDEL
C**********END COMMON**********
C**********LOCAL VARIABLES**********
REAL CF3,CF4,ERRATE
INTEGER STATUS,COLOR,TEMP,BUDFEX,BUMAX
CALL INIT2
C SEARCH MODE; LOOK FOR EOL1 BIT-BY-BIT
999 CONTINUE
CALL GETL2(MODE,MBITS,L)
GO TO (910,999,93#,2#),MODE
997 910 CONTINUE
C EOL NOT FOUND; ADVANCE POINTER AND TRY AGAIN
CDELP=CDELP+1
GO TO 999
CDELP=CDELP+1
STOP 72
C EOL FOUND
C EOL FOUND
C SEARCH=.FALSE.
CDELP=CDELP+1
```
C IF(MODE=2)965,1000,988
965 STOP 765
1000 CONTINUE
C PERFORM ONE-DIMENSIONAL DECODE OF A COMPLETE LINE
C FIRST, SET OUTPUT BUFFER TO WHITE
C (ONLY BLACK RUNS WILL BE INSERTED)
DO 181# I=1,BUFDIM
OTBUF(I,OTCOD)=W
181# CONTINUE
INDEX=3
COLOR=1
OTELP=1
182# CONTINUE
CALL ONED2(INDEX,COLOR,STATUS,L)
GO TO (183#,187#,1879,1835,1849,STATUS)
183# INDEX=3
COLOR=MOD(COLOR+2,2)+1
INDEX=3
GO TO 182#
3000 CONTINUE
C PERFORM TWO-DIMENSIONAL DECODE
184# CONTINUE
C FIRST, SET OUTPUT BUFFER TO WHITE
C (ONLY BLACK RUNS WILL BE INSERTED)
DO 161# I=1,BUFDIM
OTBUF(I,OTCOD)=W
161# CONTINUE
INDEX=3
COLOR=1
OTELP=1
302# CONTINUE
CALL TWD2(INDEX,COLOR,STATUS,L)
GO TO (303#,187#,1879,1835,1849,STATUS)
162# INDEX=3
COLOR=MOD(COLOR+2,2)+1
RUN ADDED; LOOK FOR NEXT RUN
```
ONE=.FALSE.
IF(OTELP-I-PELMAX) 3831,1#32,1#58
3#31 CONTINUE
IF(CHCOL)COLOR=MOD(COLOR+2,2)+1
INDEX=3
GO TO 3#28
C
LINE LENGTH=PELMAX; NO FILL EXPECTED
IF EOL, CONTINUE IN SPECIFIED MODE
IF NO EOL, CONTINUE IN PREVIOUS MODE
1#32 CONTINUE
C
CHECK FOR EOL
CALL GETL2(13,MODE,LBITS,L)
GO TO (1#65,1#66,1#67,1#68),MODE
PREMATURE EOL DETECTED
EOL1 DETECTED
EOL2 DETECTED
1#35 CONTINUE
CDELP=CDELP+1
STATUS=4
IF(OTELP.LE.1) CONSEC=CONSEC+1
IF(CONSEC-2)1#69,1#69,2#69
EOL2 DETECTED
1#49 CONTINUE
CDELP=CDELP+1
STATUS=5
GO TO 1#89
C
PROBLEMS,PROBLEMS
1#59 STOP 1#59
LINE LENGTH CORRECT, WRITE OUTPUT LINE
1#69 CONTINUE
EOL DETECTED PROPERLY
CDELP=CDELP+1
CONSEC=1
IF(ONE) SYNC=TRUE.
ENTRY FOR NO EOL
1#65 CONTINUE
IEOF=1
IF (OUTF) CALL IOUT(OTCOD+2,IEOF)
OTLNO=LNMOBF
TEMP=OTREF
OTREF=OTCOD
OTCOD=TEMP
IF (MODE,EQ.2) GO TO 1800
GO TO 3000
C LINE TOO LONG OR NO MATCH
C
1800 CONTINUE
WRITE(*,FALSE.
DIAG=.TRUE.
C
C LINE SHORT
C
1800 CONTINUE
IF (STATUS,EQ.4) GO TO 1800
SEARCH=.TRUE.
GO TO 900
C
C END OF MESSAGE
C
2000 CONTINUE
WRITE(LPFLI,2010) CONSEC
2010 FORMAT("END OF MESSAGE DETECTED (*.12," EOL'S")")
C
REPORT COMPRESSION FACTOR, ERROR SENSITIVITY FACTOR, BIT ERROR RATE
C
ERRATE=FLOAT(Errcnt)/Tcodel
WRITE(LPFLI,2020) Tcodel,Tcdatal,Tsfbit,Inlnct,Errate
2020 FORMAT("#CODED BITS = ",FB.8/
C
* CODED DATA BITS = ",FB.8/
C
* 2-DIM LINES = ",10/
C
* INPUT LINES PROCESSED = ",10/
C
* BIT ERROR RATE = ",G14.6)
C
CF3=FLOAT(Pelmax)*FLOAT(Inlnct)/Tcodel
CF4=FLOAT(Pelmax)*FLOAT(Inlnct)/Tcdatal
WRITE(LPFLI,2030) CF3,CF4
2030 FORMAT("#(CF3) = ",FB.4/
C
(CF4) = ",FB.4)
C
WRITE EOFS INDICATOR ON STAT FILE & CLOSE
C
STAT(1)=1
CALL WRITF(IDC89,IERR,STAT)
IF (IERR.LT.1) STOP 293
CALL CLOSE(IDC89)
IF (OUTF) CALL ERMS
C
CALL FTIME(Pelref)
WRITE(LPFLI,4000) (pelref(I),I=1,15)
**---FORTRAN---**

```fortran
C SUBROUTINE INIT2
C
C IMPLICIT INTEGER(A-Z)
C
C ***************LABELLED COMMON /G16BT/ **************
C
C DIMENSION MASK(16),COMASK(16),LIBIT(16),LZBIT(16)
C COMMON /G16BT/MASK,COMASK,LIBIT,LZBIT,NBPW
C
C ***************LABELLED COMMON ARRAYS***************
C
C DIMENSION PELREF(256),PELCOD(256),OUTREF(256),OUTCOD(1256)
C EQUIVALENCE (PELREF,PELBUF),(PELCOD,PELBUF(1,2))
C EQUIVALENCE (OUTREF,OTBUF),(OUTCOD,OTBUF(1,2))
C COMMON/BUFF/PHELBUF(256,2),OTBUF(256,2),
C CDBUF(1924),STFBUF(1924),STAT(1)
C COMMON/HUFF/CCODE(3,195,2),CODERD(3,11)
C COMMON/ERAY/ERRORS(199)
C
C ***********FILE BUFFERS*****************
C COMMON/FILES/TERM,LPFL,ICB7(1944),ICB8(1944),ICB9(1944)
C INTEGER TERM,LPFL,ICB7,ICB8,ICB9
C IDC87 - PELFL
C IDC88 - OTFL
C IDC89 - STATFL
C
C ***************LABELLED COMMON VARIABLES************
C
C COMMON/IVAR/PELMX,MPHASE,CHPMAX,ERRMOD,LINMAX,K
C COMMON/FVVAR/INLNO,OUTLNO,OTELW,CDLNP,OTELP,CDLNV,CCDATA,
C * CDELCT,INELCT,TCDATA,TCDEL,ERRPNT,ERRORF,ERRLIM,
C * ERRCRT,INELCT,CONSEC,LNONF,ZERD
C ICOD,INREF,OUTCOD,OUTREF,STFBIT,ERRORC,BUFNIM
C COMMON/LOGIC/SEARCH,DIAG,SYNC,WRITE,LEFT,CHCOL,ONE,WRITE,FILEND,
C OUTF
C
C LOGICAL SEARCH,DIAG,SYNC,WRITE,LEFT,CHCOL,ONE,WRITE,FILEND,
C OUTF
C
C DOUBLE PRECISION TCDATA,TCDEL
C
C ************END COMMON*****************************
C
C ************FILE PARAMETERS***********************
C
C DIMENSION LINE(80),IBUF(10),ITBUF(15),MBUF(3)
C EQUIVALENCE(IBUF,FNAM),(IBUF(5),ISECU),(IBUF(6),ICR)
C
C DATA LINE/CCM/LENN/LEN/ISTRCL/
C
C************BEGIN PROGRAM**************************
C```
GET INPUT IMAGE FILE NAME AND OPEN
CALL GETST(LINE,LEN,IDUM)
CALL NAMR(IBUF,LINE,2*IDUM,ISTRC)
CALL OPEN(IDC87,IERR,FNAM,SIZE,ISECU,ICR)
IF(IERR.LT.0) STOP
GET OUTPUT IMAGE FILE NAME AND OPEN
CALL NAMR(IBUF,LINE,2*IDUM,ISTRC)
CALL OPEN(IDC88,IERR,FNAM,SIZE,ISECU,ICR)
IF(IERR.GE.0) GO TO 5
IF NO OUTPUT FILE
THEN NO ERROR INSERTION
AND
NO ERROR COMPARISON (ERRMS)
AND
NO INPUT LINE NUMBER OR PEL COUNT USED/REQUIRED
ELSE ERROR INSERTION OPTIONAL

NO OUTPUT FILE
OUTF=.FALSE.
WRITE(TERM,6)
& FORMAT("NO OUTPUT FILE SPECIFIED.")
GET SCRATCH FILE NAME & OPEN
CALL NAMR(IBUF,LINE,2*IDUM,ISTRC)
CALL OPEN(IDC89,IERR,FNAM,SIZE,ISECU,ICR)
IF(IERR.LT.0) STOP
READ INPUT RECORD SIZE
2# WRITE(TERM,2#)
3# FORMAT("ENTER INPUT RECORD SIZE: ")
READTERM,* BUFDIM
IF(BUFDIM.GE.0.AND.BUFDIM.LE.256) GO TO 114
WRITE(TERM,15#) BUFDIM
GO TO 2#
READ DIAGNOSTIC SWITCH
114 WRITE(TERM,115)
115 FORMAT("DIAGNOSTIC PRINTOUT? (Y OR N): ")
READTERM,11# INSV
11# FORMAT(A1)
IF(INSV.EQ.2NY) GO TO 116
IF(INSV.EQ.2HN) GO TO 120
GO TO 114
116 CONTINUE
117 C
118 C READ MAXIMUM NUMBER OF PELS PER LINE
119 C
120 CONTINUE
121 WRITE(TERM,130)
122 130 FORMAT("ENTER MAXIMUM NUMBER OF PELS PER LINE; ")
123 READ(TERM,"") PELMAX
124 140 FORMAT(14)
125 IF(PELMAX.GE.1.AND.PELMAX.LE.4996) GO TO 160
126 WRITE(TERM,150) PELMAX
127 150 FORMAT("NUMBER OUT OF RANGE ("",16,""")
128 GO TO 120
129 C READ VERTICAL SAMPLING
130 C
131 CONTINUE
132 WRITE(TERM,170)
133 170 FORMAT("ENTER VERTICAL SAMPLING; ")
134 READ(TERM,"") VRES
135 IF(VRES.GE.1.AND.VRES.LE.10) GO TO 190
136 WRITE(TERM,180) VRES
137 GO TO 160
138 C READ PARAMETER K
139 C
140 CONTINUE
141 WRITE(TERM,192)
142 192 FORMAT("ENTER PARAMETER K; ")
143 READ(TERM,"") K
144 IF(K.GE.1.AND.K.LE.5600) GO TO 320
145 WRITE(TERM,190) K
146 GO TO 190
147 C READ NUMBER OF SCAN LINES TO BE PROCESSED
148 C
149 CONTINUE
150 WRITE(TERM,330)
151 330 FORMAT("NUMBER OF SCAN LINES TO BE PROCESSED=7 ")
152 READ(TERM,"") LINMAX
153 IF(LINMAX.GE.1.AND.LINMAX.LE.5600) GO TO 350
154 WRITE(TERM,150) LINMAX
155 GO TO 320
156 350 CONTINUE
157 C READ INPUT IMAGE NAME
158 C
159 WRITE(TERM,360)
160 360 FORMAT(" ENTER INPUT IMAGE NAME; ")
161 READ(TERM,360) NMBUF
162 C WRITE INPUT PARAMETERS
163 CALL FTIME(1TBUF)
164 WRITE(LPFFIL,370) 1TBUF
165 370 FORMAT(1H#,16A2)
**NO WARNINGS**  **NO ERRORS**  **PROGRAM = 8841**  **COMMON = 88888**
!FTN4.L,T,C
SUBROUTINE ENCD2
C     IMPLICIT INTEGER(A-Z)
C     **********LABELLED COMMON /G16BT/ ************
C     DIMENSION MASK(16),COMASK(16),LIBIT(16),LZBIT(16)
C     COMMON /G16BT/MASK,COMASK,LIBIT,LZBIT,MBFW
C     **********LABELLED COMMON ARRAYS**************
C     DIMENSION PELREF(258),PELCOD(258),OUTREF(258),OUTCOD(258)
C     EQUIVALENCE (PELREF,PELBUF),(PELCOD,PELBUF(1,2))
C     EQUIVALENCE (OUTREF,OTBUF),(OUTCOD,OTBUF(1,2))
C     COMMON/BUFSIZE,PELBUF(258,2),OTBUF(258,2),
C     COMMON/HUFF/CODE(3,185,2),CODERD(3,11)
C     COMMON/ERAY/ERRORS(188)
C     **********FILE BUFFERS***********************
C     COMMON/FILES/TERM,LPFIL,ICB7(144),ICB8(144),ICB9(144)
C     COMMON/FILES/TERM,LPFIL,ICB7,ICB8,ICB9
C     IDC87 - PELFIL
C     IDC88 - OTFIL
C     IDC89 - STATFIL
C     **********LABELLED COMMON VARIABLES**********
C     COMMON/IVAR/PELMAX,VRES,EPSHASE,CMAXMAX,ERRMOD,LINMAX,K
C     COMMON/PVAR/INLNNO,OTLNNO,OTELP,OTELP,CDELV,CDDATA,
C     * CDELCT,INELCT,TCDATA,TCDEL,ERRPNT,ERROFF,ERRLIM,
C     * ERRCNT,INELCT,CONSEC,LNNOFB,ZERO,
C     * INCOD,INREF,OTCOD,OTREF,STFBIT,ERRCOR,BUFIDIM
C     COMMON/LOGIC/SEARCH,DIAG,SYNC,WRITE,LEFT,CHCOL,ONE,WHITE,FILEND,
C     * OUTF
C     * LOGICAL/SEARCH,DIAG,SYNC,WRITE,LEFT,CHCOL,ONE,WHITE,FILEND,
C     OUTF
C     DOUBLE PRECISION TCDATA,TCDEL
C     **********************************************END COMMON*********************************************
C     INTEGER A0,A1,A2,BIT,B1,B2,ERRBIT,FILE,FILL,1,HAB,PEL,PELM1
C     INTEGER POL,POLAR,RUN,TEMP
C     ********************************************** BEGIN PROGRAM ****************************************
C     ********** INITIALIZE VARIABLES ***************
C     CDELCT-MBFW
C     CDDATA-N
C     DO 50 I=2,BUFDIM*4
C     CDBUF(1)-N
STFBUF(1)=0
60 CONTINUE
C READ INPUT PICTURE FILE
100 CONTINUE
IF(FIEN)D) GO TO 120
IF(INLNCT.GE.LINMAX) GO TO 120
CALL INOUT(INCOD, IEOF)
IF(EOF.EQ.-1) GO TO 120
IF(120) NOT(DUF) INLNNO=INLNNO+1
IF(MOD(INLNNO-1, VRES).NE.0) GO TO 100
IF(INLCr.LT. PELMAX) STOP 2222
INLNCT=INLNCT+1
LOAD OUTPUT LINE NUMBER BUFFER
LNNOBF=INLNNO
IF(SEARCH) THEN
LNNOB=INLNNO
IF(LNNOF.LE.LINMAX) GO TO 140
WRITE SIX EOL'S
120 CONTINUE
FINEF=.TRUE.
DO 130 I=1,6
CALL CONG(18, 5, 5, 5)
130 CONTINUE
DO 135 I=1,6
STFBUF(1)=CBUF(1)
135 CONTINUE
GO TO 390
1 FIRST OF K LINES
140 CONTINUE
IF(MR(INLNCT-1, K).NE.0) GO TO 600
ONE-DIMENSIONAL CODING
WRITE ONE EOL
CALL CONG(18, 5, 5, 5)
POLAR=1
TEST COLOR OF FIRST ELEMENT
IF(148(PELBUF(1, INCOD), 1, 1).EQ.0) GO TO 150
FIRST ELEMENT BLACK; ENCODE #-LENGTH WHITE RUN
CALL CODLM(1, POLAR=2
C CALCULATE RUN LENGTH AND ENCODE

150 CONTINUE
RUN=0
DO 200 I=1,PELMAX
PEL=14B(PELBUF(1,INCOD),1,1)+1
IF(PEL.EQ.POLAR) GO TO 160
CALL CODLM(RUN,POLAR)
IF(NOT.DIG) GO TO 170
WRITE(TERM,160) RUN,POLAR,CDELCU,CDATA
160 FORMAT(410)
170 CONTINUE
RUN=1
POLAR=MOD(POLAR+2,2)+1
GO TO 200
200 CONTINUE
CALL CODLM(RUN,POLAR)
IF(NOT.DIG) GO TO 210
WRITE(TERM,160) RUN,POLAR,CDELCU,CDATA
GO TO 210

C TWO-DIMENSIONAL CODING

680 CONTINUE
STFBIT=STFBIT+1
C IF PREVIOUS LINE IS ONE-DIMENSIONAL, WRITE ONE EOL2
C IF(MOD(INLCU-2,K).EQ.0) CALL CODING(11,0,0,0,0)
C SET A# TO LEFT EDGE-1 AND POLARITY=WHITE
C
AB=0
POL=0
LEFT=.TRUE.

C DETECT AI

620 CONTINUE
I=AB+1
IF(I.GT.PELMAX) GO TO 640
630 CONTINUE
PEL=14B(PELBUF(1,INCOD),1,1)
IF(PEL.NE.POL) GO TO 640
I=I+1
IF(I.LE.PELMAX) GO TO 630
640 CONTINUE
AI=I

C DETECT BI

I=AB+1
IF (I.GT.PELMAX) GO TO 665
IF (LEFT) GO TO 645
PELM1=14B(PELBUF(1,INREF),A#1)
GO TO 659
PELM1=0
CONTINUE
PEL=14B(PELBUF(1,INREF),1,1)
IF (PEL NE PELM1) GO TO 670
CONTINUE
PELM1=PEL
I=I+1
IF (I.LT.PELMAX) GO TO 659
CONTINUE
B1=1
GO TO 710
670 CONTINUE
IF (PEL NE POL) GO TO 690
GO TO 669
690 CONTINUE
B1=I
POL=PEL
C DETECT B2
1=81+1
IF (I.GT.PELMAX) GO TO 710
700 CONTINUE
PEL=14B(PELBUF(1,INREF),I,1)
IF (PEL NE POL) GO TO 720
I=I+1
IF (I.LT.PELMAX) GO TO 700
710 CONTINUE
B2=I
GO TO 730
720 CONTINUE
B2=I
POL=PEL
730 CONTINUE
IF (.NOT.LEFT) POLAR=14B(PELBUF(1,INCOD),A#,1)+1
IF (.NOT.LEFT) GO TO 740
POLAR=I
A#1=I
LEFT=.FALSE.
740 CONTINUE
C TEST FOR PASS MODE
IF (B2.GE.A1) GO TO 760
C PASS MODE CODING (CAN'T END AT LINE IN PASS MODE; NEW A# MUST HAVE
C SAME POLARITY AS B2)
CALL CODING(1,#,#,#,#)
A#2=0
GO TO 620
750 CONTINUE
752 C
753 MAB=IABS(A1-B1)
754 IF(MAB-3) 835,835,799
756 C CODE BY HORIZONTAL MODE; FIRST DETECT A2
757 C
758 799 CONTINUE
759 1=A1+1
760 IF(1.GT.PELMAX) GO TO 810
761 C CALCULATE POLARITY OF A1
762 C
763 POL=14B(PELBUF(1,INCOD),A1,1)
766 888 CONTINUE
767 PEL=14B(PELBUF(1,INCOD),1,1)
768 IF(PEL.NE.POL) GO TO 828
769 1=1+1
770 IF(1.LE.PELMAX) GO TO 888
772 810 A2=PELMAX+1
773 GO TO 630
774 828 CONTINUE
775 A2=1
776 830 CONTINUE
777 CALL CODNG(2,POLAR,A$,A1,A2)
778 A$=A2
779 GO TO 960
780 C CODE BY VERTICAL MODE
782 C
783 835 CONTINUE
784 IF(A1-B1) 850,840,840
786 C
787 840 CALL CODNG(A1-B1+3,$,$,$,$)
788 GO TO 950
789 850 CONTINUE
790 CALL CODNG(B1-A1+6,$,$,$,$)
792 950 CONTINUE
793 A$=A1
794 C TEST FOR END OF LINE
796 C
797 960 CONTINUE
798 IF(AW.GT.PELMAX) GO TO 210
799 POL=14B(PELBUF(1,INCOD),A$,1)
800 GO TO 628
801 210 CONTINUE
802 C SWITCH CODE & REFERENCE LINES
803 C
804 TEMP=INREF
805 INREF=INCOD
806 INCOD=TEMP
807 C
808 CDELW=(CDELCT+NBPW-1)/NBPW
DO 300 I=2,CDELW
STFBUF(I)=CDBUF(I)
300 CONTINUE
C SAVE LINE LENGTH (DATA BITS ONLY)
CALL WRITF(IDC89,IERR,STAT)
IF(IERR.LT.0)STOP 300
C COMPUTE STATISTICS
390 CONTINUE
TCDAT=TCDAT+CDELC-NSCW
TCDAT=TCDAT+CDDATA
IF(DIAG) WRITE(TERM,160) INLINC, CDDATA
IF (.NOT.DIAG) GO TO 460
CDELW=(CDELC+NSCW-1)/NSCW
WRITE(LPFL,450) (STFBUF(I),I=1,CDELW)
WRITE(LPFL,450) (STFBUF(I),I=1,CDELW)
FORMAT(BO12)
460 CONTINUE
RETURN
C
END

FTM4 COMPILER: HP9286-16992 REV. 2B26 (888423)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 88842 COMMON = 88888
SUBROUTINE GETLZ(LBITS,MODE,WRD,L)

IMPLICIT INTEGER(A-Z)

C       LBITS - NO. OF BITS REQUESTED
C       MODE -
C       1 - NORMAL RETURN
C       2 - EOL1 DETECTED
C       3 - EOL2 DETECTED
C       4 - NOT USED
C       WRD - CONTAINS BITS RETURNED
C       L - NO. OF BITS RETURNED

********LABELLED COMMON /G16BT/ **************

C       DIMENSION MASK(16),COMASK(16),LIBIT(16),LZBIT(16)
C       COMMON /G16BT/MASK,COMASK,LIBIT,LZBIT,MBFW

********LABELLED COMMON ARRAYS***************

C       DIMENSION PELREF(256),PELCOD(256),OUTREF(256),OUTCOD(256)
C       EQUIVALENCE (PELREF,PELBUF,(PELCOD,PELBUF(1,2))
C       EQUIVALENCE (OUTREF,OTBUF),(OUTCOD,OTBUF(1,2))
C       COMMON/BUF/PELBUF(256,2),OTBUF(256,2),
C       & CDBUF(1824),STBUF(1824),STAT(1)
C       COMMON/HU/CHDRE(3,165,21),CODERD(3,11)
C       COMMON/ERAY/ERRORS(188)

********LABELLED COMMON VARIABLES************

C       COMMON/IVAR/PELMAX,RES,EPHAS,CMPMAX,ERRMOD,LMNMAX,K
C       COMMON/VAR/INLMNO,OTLMNO,OTLW,CDNL,DEL,CLNL,CDLW,CDLLA,
C       CDLCT,IMLCT,TCODATA,TCDEL,ERRPMT,ERROFF,EROLL,
C       ERCNT,INLCT,CONSEC,LMNDFZ,ZERO,
C       IMCOD,INREF,OTCOD,OTREF,STBRT,ERRCOR,BUFFIM
C       COMMON/LOG/SEARCH,DIA,SYNC,WRIT,LEFT,CHCOL,ONE,WHITE,FILED,
C       OUTF
C       LOGICAL SEARCH,DIA,SYNC,WRIT,LEFT,CHCOL,ONE,WHITE,FILED,
C       OUTF

C       DOUBLE PRECISION TCDATA,TCDEL

***********END COMMON**********************

INTEGER WRD

*********** BEGIN PROGRAM ******************

RETRIEVE NEXT BIT FROM CDBUF
ENCODE A NEW LINE IF NECESSARY
IF(LBITS+CDEL-1.LE.CDELCT) GO TO 200
IF(CDELCT-CDELP+1) 178,190,188
178 STOP 178
188 CONTINUE
STFBUF(1)=148(STFBUF,CDELP,CDELCT-CDELP+1)
CONTINUE
CDELP=NBPV-(CDELCT-CDELP)
CALL ENCD2
CONTINUE
CALL ENCD2
CONTINUE
WRD=148(STFBUF,CDELP,LBITS)
L=LBITS
IF(L.LT.13) GO TO 258
IF(L.EQ.13.AND.WRD.EQ.CODERD(3,10)) GO TO 388
IF(L.EQ.13.AND.WRD.EQ.CODERD(3,11)) GO TO 488
CONTINUE
MODE=1
RETURN
CONTINUE
MODE=2
RETURN
CONTINUE
MODE=3
RETURN
END

FTN4 COMPILER: HP92868-1692 REV. 2826 (080423)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 55126 COMMON = 55555
**FTM4.L.T.C**

**Block Data**

```fortran
  BLOCK DATA MBLK2
```

**Implicit Declaration**

```fortran
  IMPLICIT INTEGER(A-Z)
```

**Labelled Common Arrays**

```fortran
  C************************LABELLED COMMON ARRAYS************************
  C
  DIMENSION PELREF(256), PELCOD(256), OUTREF(256), OUTCOD(256)
  EQUIVALENCE (PELREF, PELBUF), (PELCOD, PELBUF(1,2))
  EQUIVALENCE (OUTREF, OTBUF), (OUTCOD, OTBUF(1,2))
  COMMON/BUFF/PELBUF(256,2), OTBUF(256,2),
  & CDBUF(1024), STFBUF(1024), STAT(1)
  COMMON/HUFF/ CODE(3,125,2), CORDER(3,11)
  COMMON/ERAY/ERRORS(160)
  C************************FILE BUFFERS*******************************
  C
  COMMON/FILES/TERM, LPFIL, IDC87(144), IDC88(144), IDC89(144)
  INTEGER TERM, LPFIL, IDC87, IDC88, IDC89
  C
  IDC87 - PEFLFIL
  IDC88 - OTFIL
  IDC89 - STATFL
  C************************LABELLED COMMON VARIABLES****************
  C
  COMMON/IVAR/PELMAX, VRES, EPHTUE, CMPMAX, ERRMOD, LINMAX, K
  COMMON/PVAR/INLNO, OTLNO, OTELV, CDELP, CDELY, CDELY, CDDDATA,
  & CDLCT, IME='CT, TCDATA, TCDEL, ERRPT, ERROFF, ERRSTIM,
  & ERRNT, LNLCNT, CONSEC, LHNOF, ZERO,
  & INCOD, INREF, FTPCOD, TREF, STB87, ERROR, BUFDM
  COMMON/LOGIC/SEARCH, DIAG, SYNC, WRITE, LEFT, CHCOL, ONE, WHITE, FILEND,
  & QUTF
  LOGICAL SEARCH, DIAG, SYNC, WRITE, LEFT, CHCOL, ONE, WHITE, FILEND,
  & QUTF
  C
  DOUBLE PRECISION TCDATA, TCDEL
  C
  C****************************END COMMON****************************
  C
  DATA TERM/1/
  DATA LPFIL/6/
  DATA DIAG/.FALSE./
  DATA WRITE/.FALSE./
  DATA SEARCH/.TRUE./
  DATA SYNC/.FALSE./
  DATA FILEND/.FALSE./
  DATA QUTF/.TRUE./
  DATA ERRNT/0/
  DATA INREF/1/
  DATA IMCOD/2/
  DATA TREF/1/
  DATA OTCOD/2/
  DATA INLNO/0/
  DATA INLCT/0/
```

---

**Note:** The code snippet appears to be a part of a Fortran program, likely related to file and common variable management. The code includes declarations for arrays, variables, and common blocks, along with various logical and integer data. It seems to be a part of a larger program, possibly involved in file I/O or text processing based on the context of the variables and arrays.
FTN4, L.T.C

SUBROUTINE ONEED2(INDEX, COLOR, STATUS, L)

C IMPLICIT INTEGER(A-Z)

C ********* LABELLED COMMON ARRAYS ***********

DIMENSION PELREF(258), PELCOD(258), OUTREF(258), OUTCOD(258)

EQUIVALENCE (PELREF, PELBUF), (PELCOD, PELBUF(1,2))

EQUIVALENCE (OUTREF, OTBUF), (OUTCOD, OTBUF(1,2))

COMMON/BUFF/PPELBUF(258,2), OPELBUF(258,2),
CDBUF(1#24), STBUF(1#24), STAT(1)

COMMON/HUFF/Code(3,1#5,2, CODERD(3,1,1)

COMMON/ERAY/ERRORS(1#0)

C ********* FILE BUFFERS ***********

COMMON/FILES/TERM, LPFIL, IDC87(144), IDC88(144), IDC89(144)

INTEGER TERM, LPFIL, IDC87, IDC88, IDC89

C IDC87 - PPELBUF

C IDC88 - OTBUF

C IDC89 - STATFL

C ********* LABELLED COMMON VARIABLES **********

COMMON/IVAR/PELMAX, VRES, EPHEME, CNPMAX, ERRMOD, LINMAX, K

COMMON/PVAR/INLNO, OUTLNO, OTELV, CDELP, OTELV, CDELV, CDDATA,
* CDELCT, INELCT, TCDATA, TCDEL, ERRPMT, ERROFF, ERRLIN,
* ERRCNT, INLCT, CONSEC, LHMIF, ZERO,
* INCOD, INREF, OTCOD, OTREF, STBIT, ERRCOR, BUFDIM

COMMON/LOGIC/SEARCH, DIAG, SYNC, WRITE, LEFT, CHCOL, ONE, WHITE, FILEND,
* OUTF

LOGICAL SEARCH, DIAG, SYNC, WRITE, LEFT, CHCOL, ONE, WHITE, FILEND,

OUTF

C DOUBLE PRECISION TCDATA, TCDEL

C ********* END COMMON *********************

C INTEGER COLOR, I, INDEX, L, LBITS, LEND, MODE, RUNLEN, STATUS

C ********* BEGIN PROGRAM *****************

C BEGIN DECODE LOOP; RETRIEVE NEXT CODE WORD LENGTH (L)

1#99 CONTINUE

1#92 LEND-CODE(1, INDEX, COLOR)

CALL GETL2(LEND, MODE, LBITS, L)

IF(DIAG) WRITE(TERM, 1#83) LEND, MODE, LBITS, L

1#83 FORMAT(216, 08, 16)

GO TO (1#84, 1#88, 1#85, 1#89), MODE

1#84 CONTINUE

1#88 CONTINUE

IF(LBITS.EQ.CODE(3, INDEX, COLOR)) GO TO 1#88

1#85 C
C NO MATCH; ADVANCE CODE WORD INDEX VIA DECODE THREAD
C INDEX-CODE(2,INDEX,COLOR)
C IF(INDEX.GE.186) GO TO 119#
C IF(CODE(1,INDEX,COLOR).EQ.LENBIT) GO TO 184#
C CODE WORD LONGER; FROM THE TOP
C GO TO 184#
C MATCH FOUND
C 110# CONTINUE
C CDELP=CDELP+1
C NOT AN EOL
C TEST FOR MAKE UP OR TERMINATING CODE
C RUNLEN=INDEX-1
C IF(INDEX.GE.65) RUNLE=+(INDEX-64)*64
C IF(RUNLEN.EQ.0) GO TO 116#
C IF(COLOR.EQ.1) GO TO 1155
C IF(RUNLEN.LT.0) STOP 110#
C ADD BLACK RUN TO OUTPUT BUFFER
C GO 116# I=1,RUNLEN
C CALL M12B(COLOR-1,OTBUF(1,OTCOD),OTELP,1)
C OTELPE=OTELP+1
C IF(OTELP-1.GT.PELMAX) GO TO 110#
C 115# CONTINUE
C GO TO 116#
C ADD WHITE RUN TO OUTPUT BUFFER (BY DEFAULT)
C 1155 CONTINUE
C OTELPE=OTELP+RUNLEN
C IF(OTELP-1.GT.PELMAX) GO TO 110#
C OUTPUT LINE LESS THAN OR EQUAL TO MAX SPECIFIED
C 116# CONTINUE
C IF(INDEX.LT.65) GO TO 117#
C INDEX=3
C GO TO 118#
C RUN ADDED TO OUTPUT LINE; LENGTH LESS THAN OR EQUAL TO PELMAX (1)
C 117# CONTINUE
C CHCOL=.TRUE.
C STATUS=1
C RETURN
C
C RUN ADDED UNTIL PELMAX EXCEEDED; LINE TOO LONG (2)
C
110# CONTINUE
114 IF(DIAG) WRITE(TERM,1105) (OTBUF(I,OTCOD),I=1,BUFDIM)
115 1105 FORMAT(150B)
116 STATUS=2
117 RETURN
118 C NO MATCH FOUND IN CODE TABLE (3)
119 C
120# CONTINUE
121 STATUS=3
122 RETURN
124 C EOL1 DETECTED (4)
125 C
126# CONTINUE
127 STATUS=4
128 RETURN
129 C EOL2 DETECTED (5)
130 C
131# CONTINUE
132 STATUS=5
133 RETURN
134 E N D

FTM4 COMPILER: HP92068-1692 REV. 2026 (980423)

** NO WARNINGS ** NO ERRORS ** PROGRAM = S271 COMMON = S271

A-27
Subroutine TWD2 (Index, Color, Status, L)

Implicit Integer (A-Z)

Dimension Pelref(258), Pelcod(258), Outref(258), Outcod(258)
Equivalence (Pelref, Pelbuf), (Pelcod, Pelbuf(1, 2))
Equivalence (Outref, Outbuf), (Outcod, Outbuf(1, 2))
Common/Buffer/Pelbuf(258, 1), Outbuf(258, 2),
& CDbuf(1524), Stbuf(1524), Stat(1)
Common/Huff/Code(3, 152, 2), Coderd(3, 11)
Common/Error/Errors(152)

File Buffers

Common/Files/Term, Lpfil, Idcb7(144), Idcb8(144), Idcb9(144)
Integer Term, Lpfil, Idcb7, Idcb8, Idcb9

Idcb7 - Pelfil
Idcb8 - Otfil
Idcb9 - Statfl

Labelled Common Variables

Common/Ivar/Pelmax, Vreq, Ephase, Cmmax, Errmod, Linmax, K
Common/Pvar/Inlno, Otlno, Otelw, Cotelw, Cigelw, Cgdata,
* Cdect, Inelct, Tcdatal, Tcdel, Erpmt, Erroff, Errelm,
* Errnt, Inlcnt, Consec, Lmnbf, Zero,
* Tncod, Inref, Otcod, Otref, Stbuf, Earror, Bufdim
Common/Logic/Search, Diag, Sync, Write, Left, Chcol, One, White, Filend,
Out
Logical Search, Diag, Sync, Write, Left, Chcol, One, White, Filend,
Out

Double Precision Tcdatal, Tcdel

End Common

Integer Aa, B1, B2, Color, Entry, I, Index, L, Lbits, Lenbit, Mode
Integer Fbl, Pelmi, Pol, Runlen, State, Status

Begin Program

Begin Decode Loop; Retrieve Next Code Word Length (L)

Continue

1#5 Continue
1#6 Lbits = Coderd(1, Index)
Call Get2(Lbits, Mode, Lbits, L)
If (Diag) Write(Term, 1#3) Lbits, Mode, Lbits, L
Format(216, 0, 16)
Go To (1#48, 1205, 1205, 1190), Mode
Continue
If (Lbits = Coderd(3, Index)) Go To 1#8
CL
CC I.
I
I
O
U
N
4
W
+ 8
U
W
-
Z
...
E
X
AO
0
;
DO
OC
A.
L
A
-
U
0
Z
6
1
U
W
4
J
4
J
I
U
Q
W
Q
W
W
W
*
A
9
6
U
N
Z
2
A
C
C
FIND B1 AND B2
C
A8=OTELP
C
IF(OTELP.EQ.1) A8=8
C
POL-COLOR-1
B
C
DETECT B1
A
I=A8+1
A
IF(I.GT.PELMAX) GO TO 65
B
PELM1=8
B
IF(A8.EQ.0) GO TO 58
A
PELM1=14B(OTBUF1,OTREF),A8,1)
B
CONTINUE
B
PEL=14B(OTBUF1,OTREF),I,1)
A
IF(PEL.NE.PELMI) GO TO 78
B
68 CONTINUE
A
PELM1=PEL
A
I=1+1
A
IF(I.LE.PELMAX) GO TO 58
A
65 CONTINUE
A
B1=I
A
GO TO 92
B
78 CONTINUE
A
IF(PEL.NE.POL) GO TO 98
A
GO TO 68
A
98 CONTINUE
A
B1=I
A
POL=PEL
A
C
DETECT B2
B
I=B1+1
B
IF(I.GT.PELMAX) GO TO 92
B
91 CONTINUE
A
PEL=14B(OTBUF1,OTREF),I,1)
IF(PEL.ME.POL) GO TO 92
I=I+1
IF(I.LE.PELMAX) GO TO 91
92 CONTINUE
B2=1
GO TO (180,280,380,480,480,480,680,680,680),INDEX
C PASS MODE
C HORIZONTAL MODE
180 CONTINUE
RULLEN=B2-OTELP
CHCOL=.FALSE.
GO TO (1155,1145),COLOR
C HORIZONTAL MODE
280 CONTINUE
ENTRY=3
CALL ONE2D(ENTRY,COLOR,STATE,L)
GO TO (210,1180,1190,1200,1205),STATE
210 CONTINUE
COLOR=MOD(COLOR+2,2)+1
ENTRY=3
CALL ONE2D(ENTRY,COLOR,STATE,L)
GO TO (220,1180,1190,1200,1205),STATE
220 CONTINUE
CHCOL=.TRUE.
GO TO 1160
C VERTICAL MODE A181-8
380 CONTINUE
RULLEN=B1-OTELP
CHCOL=.TRUE.
GO TO (1155,1145),COLOR
C VERTICAL MODE VR1 A181-1,2,3
480 CONTINUE
RULLEN=B1-OTELP+INDEX-3
CHCOL=.TRUE.
GO TO (1155,1145),COLOR
C VERTICAL MODE LEFT V1 A181-1,2,3
680 CONTINUE
RULLEN=B1-OTELP-(INDEX-6)
CHCOL=.TRUE.
GO TO (1155,1145),COLOR
C ADD BLACK RUN TO OUTPUT BUFFER
1145 CONTINUE
IF(RULLEN) 1190,1160,1147
1147 CONTINUE
  DO 1151 I=1,NUMLEN
  CALL M12B(COLOR-I,OTBUF(I,OCTOD),OTEIP,1)
  OTELP=OTEIP+I
  IF(OTELP-1.GT.PELMAX) GO TO 1189
1171 1151 CONTINUE
  GO TO 1169
1173 C  ADD WHITE RUN TO OUTPUT BUFFER (BY DEFAULT)
1174 C  IF(RUHLEN.LT.0) GO TO 1199
1175 C  OTELP=OTEIP+RUHLEN
1176 C  IF(OTELP-1.GT.PELMAX) GO TO 1189
1178 C  RUN ADDED TO OUTPUT LINE; LENGTH LESS THAN OR EQUAL TO PELMAX (1)
1181 C 1169 CONTINUE
1184 C  STATUS=1
1185 C  RETURN
1186 C  RUN ADDED UNTIL PELMAX EXCEEDED; LINE TOO LONG (2)
1188 C 1189 CONTINUE
1190 C  IF(DIAG) WRITE(TERM,1185) (OTBUF(I,OCTCOD),I=1,BUFDIM)
1191 C  1185 FORMAT(150B8)
1192 C  STATUS=2
1193 C  RETURN
1194 C  NO MATCH FOUND IN CODE TABLE (3)
1195 C 1199 CONTINUE
1199 C  STATUS=3
1200 C  RETURN
1201 C  EOL1 DETECTED (4)
1202 C 1219 CONTINUE
1204 C  STATUS=4
1205 C  RETURN
1206 C 1229 CONTINUE
1207 C  EOL2 DETECTED (5)
1208 C 1239 CONTINUE
1210 C  STATUS=5
1211 C  RETURN
1212 C  END

FTM4 COMPILER: HP92060-1692 REV. 2026 (B88423)

** NO WARNINGS ** NO ERRORS ** PROGRAM = B8476 COMMON = B8888

A-3L
SUBROUTINE CODNG(MODE, POLAR, A, B, C)

IMPLICIT INTEGER(A-Z)

C**********Labeled COMMON Arrays***********

DIMENSION PELREF(256), PELCOD(256), OUTREF(256), OUTCOD(256)

EQUIVALENCE (PELREF, PELBUF), (PELCOD, PELBUFI, 2)

EQUIVALENCE (OUTREF, OTBUF), (OUTCOD, OTBUFI(1, 2))

COMMON/BUFF/PBREF(256, 2), OTBUFI(256, 2),
& CDBUF(1024), STFBUF(1024), STAT(1)

COMMON/HUFF/CODE(3, 185, 2), CORDERD(3, 11)

COMMON/ERAY/ERRORS(188)

C**********Labeled COMMON Variables**********

COMMON/IVAR/PELMAX, VRES, EPHAS, CMPMAX, ERRMOD, LINMAX, K

COMMON/PVAR/INLNNO, OTLNNO, OTDEP, OTELW, CDELP, CDELW, CDDATA,

* CDELT, INELCT, TCDATA, TCDEL, ERRPMT, EROFF, ERLIM,

* ERRCT, INLCT, CONSEC, LTNOF, ZERO,

* INCOD, INREF, OTCOD, OTRF, STFBIT, ERRCOR, BUFIM

COMMON/LOGIC/SEARCH, DIAG, SYNC, WRITE, LEFT, CHCOL, ONE, WHITE, FILEND,

* OUTF

LOGICAL SEARCH, DIAG, SYNC, WRITE, LEFT, CHCOL, ONE, WHITE, FILEND,

OUTF

DOUBLE PRECISION TCDATA, TCDEL

C******************End COMMON************************

INTEGER A, B, C, MODE, NEVPOL, POLAR

C********** Begin Program **********

CALL MI28(CORDERD(3, MODE), CDBUF, CDELT+1, CORDERD(1, MODE))

CDELT=CDELT+CORDERD(1, MODE)

GO TO (188, 288, 180, 180, 180, 180, 180, 180, 180, 180, 180, 180)

MODE

C MODE 1 2 3 4 5 6 7 8 9 10 11

PASS MODE(1), VERTICAL MODE: A1B1=0(3), A1B1=1(4, 7), =2(5, 8), =3(6, 9)

188 CONTINUE

CDDATA=CDDATA+CORDERD(1, MODE)

RETURN

C HORIZONTAL MODE(2)

288 CONTINUE

CDDATA=CDDATA+CORDERD(1, MODE)

CALL CODH(18-A, POLAR)

NEVPOL=MOD(POLAR+2, 2)+1

CALL CODVNC-E, NEVPOL
FTN4,L.T.C
SUBROUTINE CODLN(LENGTH,POLAR)
C     IMPLICIT INTEGER(A-Z)
C     *****************************************LABELLED COMMON ARRAYS**************
C     DIMENSION PELREF(258),PELCOD(258),OUTREF(258),OUTCOD(258)
C     EQUIVALENCE (PELREF,PELBUF),(PELCOD,PELBUF(1,2))
C     EQUIVALENCE (OUTREF,OTBUF),(OUTCOD,OTBUF(1,2))
C     COMMON/BUF/OTBUF(258,2),OTBUF(258,2),LDBUF(124),STFBUF(124),STAT(1)
C     COMMON/HUFF/CODE(3,1#5,2),CODERD(3,11)
C     COMMON/ERAY/ERRORST(100)
C     *****************************************FILE BUFFERS******************************
C     COMMON/FILES/TERM,LPFIL,IDCB7(144),IDCB8(144),IDCB9(144)
C     INTEGER TERM,LPFIL,IDCB7,IDCB8,IDCB9
C     IDCB7 = PELFIL
C     IDCB8 = OTFIL
C     IDCB9 = STATFL
C     *****************************************LABELLED COMMON VARIABLES***********
C     COMMON/IVAR/PELMAX,VRES,EPHASE,CMFPAX,ERNMOD,LIMAX,K
C     COMMON/PVAR/INLNNO,OTLNNO,OTELW,CDELW,CDELW,CDDATA,
C     CDELT,IMECT,TCDATA,TCDL,ERRPMT,ERROFF,ERRLIM,
C     ERCMNT,INCMCT,CONSEC,LNNOF,ZERO,
C     INCOD,INREF,OTCOD,OTREF,STFBIT,ERRCOR,BUFDIM
C     COMMON/LOGIC/SEARCH,DIAG,SYNC,WRITE,LEFT,CHCOL,ONE,WHITE,FilenD,
C     OUTF
C     LOGICAL SEARCH,DIAG,SYNC,WRITE,LEFT,CHCOL,ONE,WHITE,FilenD,
C     OUTF
C     DOUBLE PRECISION TCDATA,TCDL
C     *****************************************END COMMON****************************
C     INTEGER LENGTH,POLAR,INDEX,DEX,CODESW,TCODE,TLEN
C     ***************************************** BEGIN PROGRAM ******************
C     CHECK INPUTS
C     IF(POLAR.LT.1.OR.POLAR.GT.2) STOP 3333
C     IF(LENGTH.LT.6.OR.LENGTH.GT.PELMAX) STOP 4444
C     IF(LENGTH.LE.63) GO TO 48
C     CALCULATE MAKE UP CODE INDEX
C     INDEX=LENGTH/64+64
C     # IF(INDEX.LE.1#4) GO TO 38
INDEX = INDEX - 40
DEX = 154
ASSIGN 2# TO CODESW
D5 IF(DIAG) WRITE(ITEM, 25) CODESW, DEX, POLAR
D25 FORMAT(* 2# CODESW = "316")
D51 GO TO 1000
D52 3# DEX = INDEX
D53 ASSIGN 4# TO CODESW
D54 IF(DIAG) WRITE(ITEM, 35) CODESW, DEX, POLAR
D55 D35 FORMAT(* 3# CODESW = "316")
D56 GO TO 1000
D57 C CALCULATE TERMINATING CODE INDEX
D58 C
D70 4# DEX = MOD(LENGTH, 64) + 1
D71 ASSIGN 6# TO CODESW
D72 D IF(DIAG) WRITE(ITEM, 45) CODESW, DEX, POLAR
D73 D45 FORMAT(* 4# CODESW = "316")
D74 GO TO 1000
D75 6# RETURN
D76 C
D77 C CODE LOOK-UP AND INSERTION ROUTINE
D78 C
D79 1000 CONTINUE
D80 TCODE = CODE(3, DEX, POLAR)
D81 TLEN = CODE(1, DEX, POLAR)
D82 D IF(DIAG) WRITE(ITEM, 1005) CODE(1, DEX, POLAR)
D83 D1005 FORMAT(* WORD LENGTH = "16")
D84 CALL M12B(TCODE, CBUFS, CDELCT+1, TLEN)
D85 CDELCT = CDELCT + TLEN
D86 CDATA = CDATA + TLEN
D87 C
D98 GO TO CODESW, (28, 48, 60)
D99 END

FTM4 COMPILER: HP9266–16992 REV. 2926 (B89423)

** NO WARNINGS ** NO ERRORS ** PROGRAM = B8135 COMMON = B8888
SUBROUTINE INOUT(ICLT, IEOF)

IMPLICIT INTEGER(A-Z)

THIS ROUTINE PERFORMS ALL DISC I/O FOR IMAGE FILES

A RECORD CONSISTS OF:
- LINE NUMBER - 1 WORD
- MAX LINE SIZE - 1 WORD (NEGATIVE FOR A CORRUPTED LINE)
- DATA FOR ONE SCAN LINE - UP TO 256 WORDS

THE INPUT/OUTPUT BUFFERS ARE CONNECTED TO THE TWO-DIMENSIONAL
PROGRAM ARRAYS BY EQUIVALENCE. LINE NUMBER AND SIZE MUST BE
LOADED/EXTRACTED BEFORE/AFTER WRITING/READING.

ENTRY PARAMETERS:
ICLT - I/O CONTROL WORD

1 - READ INTO REFERENCE BUFFER (PELREF)
2 - READ INTO CODE BUFFER (PELCOD)
3 - WRITE FROM REFERENCE BUFFER (OUTREF)
4 - WRITE FROM CODE BUFFER (OUTCOD)
5 - READ INTO REFERENCE BUFFER (OUTREF)
6 - READ INTO CODE BUFFER (OUTCOD)
99 - CLOSE FILES AND STOP

IEOF - INDICATES POS OR NEG LINE SIZE TO BE WRITTEN

+1 - POS
-1 - NEG

EXIT PARAMETERS:
IEOF - CONTAINS END OF FILE INDICATION

*******LABELLED COMMON ARRAYS**************

DIMENSION PELREF(256), PELCOD(256), OUTREF(256), OUTCOD(256)

EQUIVALENCE (PELREF,PELBUFF,(PELCOD,PELBUFF(1,2))

EQUIVALENCE (OUTREF,OUTBUFF),(OUTCOD,OUTBUFF(1,2))

COMMON/BUFF/PELBUFF(256,2),OUTBUFF(256,2),

& COMMON/ERAY/ERRORS(100)

COMMON/HUFF/OUTBUFF(128),STAT(1)

COMMON/ERAY/ERRORS(100)

*******FILE BUFFERS***************

COMMON/FILES/TERM,LPFIL, IDC81(144), IDC80(144), IDC9(144)

INTEGER TERM,LPFIL, IDC81, IDC80, IDC9
**C************LABELLED COMMON VARIABLES************

**COMMON/IVAR/PELMAX,VRES,EPMAX,CMMAX,ERRMOD,LLMAX,K**

**COMMON/PVAR/INLNO,OTLNO,OTELW,CDLWP,OTELP,CDLW,CDDATA,**

**CDELC,INELCT,TCDATA,TCDEL,ERRPNT,ERROFF,ERLIM,**

**ERRCT,INLCT,CONSEC,LNHOFF,ZERO,**

**INCOD,INREF,OTCOD,OTREF,STFBIT,ERRCO,BUFDIM**

**COMMON/LOGIC/SEARCH,DIAG,SYNC,WRITE,LEFT,CHCOL,ONE,WHITE,FILENO,**

**OUTF**

**LOGICAL SEARCH,DIAG,SYNC,WRITE,LEFT,CHCOL,ONE,WHITE,FILENO,**

**OUTF**

**DOUBLE PRECISION TCDATA,TCDEL**

**C************END COMMON**********************

**C************BEGIN PROGRAM**********************

**GO TO(100,200,300,400,500,600,9000),ICTL**

**CREAD INTO REFERENCE BUFFER**

**CALL READF(IDC87,IERR,PELREF,BUFDIM+2,IEOF)**

**IF(.NOT.OUTF) GO TO 9320**

**INLNO=PELREF(BUFDIM+1)**

**INELCT=PELREF(BUFDIM+2)**

**GO TO 9360**

**CREAD INTO CODE BUFFER**

**CALL READF(IDC87,IERR,PELCOD,BUFDIM+2,IEOF)**

**IF(.NOT.OUTF) GO TO 9320**

**INLNO=PELCOD(BUFDIM+1)**

**INELCT=PELCOD(BUFDIM+2)**

**GO TO 9360**

**CWRITE FROM REFERENCE BUFFER**

**OUTREF(BUFDIM+1)=OTLNO**

**OUTREF(BUFDIM+2)=IEOF*PELMAX**

**CALL WRTF(IDC88,IERR,OUTREF)**

**GO TO 1500**

**CWRITE FROM CODE BUFFER**

**OUTCODE(BUFDIM+1)=OTLNO**

**OUTCODE(BUFDIM+2)=IEOF*PELMAX**

**CALL WRTF(IDC88,IERR,OUTCODE)**

**GO TO 1500**

**CREAD FROM OUTPUT REFERENCE BUFFER**

**CALL READF(IDC88,IERR,OUTREF,BUFDIM+2,IEOF)**

**OTLNO-OUTREF(BUFDIM+1)**

**LOCAL -OUTREF(BUFDIM+2)**
GO TO 9000
C READ FROM OUTPUT CODE BUFFER
CALL READF(IDCBB, IERR, OUTCOD, BUFDM+2, IEOF)
QTLNM=OUTCOD(BUFDM+1)
LOCAL=OUTCOD(BUFDM+2)
GO TO 9000
C TEST FOR ERROR
1500 CONTINUE
IEOF=0
IF(IERR.EQ.-12) IEOF=-1
IF(IEOF.EQ.-1) RETURN
IF(IERR.EQ.0) RETURN
WRITE(TERM,9100) IERR
9100 FORMAT("FILE ERROR"*"ABORT")
9900 CALL CLOSE(IDCBB, IERR)
IF(.NOT.OUTF) GO TO 9940
CALL LOCDF(IDCBB, IERR, IRR, 1, JSEC)
ISTR=JSEC/2-IRB-1
ISTR=JSEC/2-IRB-1
CALL CLOSE(IDCBB, IERR, ITRUN)
IF(IERR.LT.0)STOP 1500
9940 CONTINUE
CALL CLOSE(IDCBB, IERR)
WRITE(TERM,9950)
9950 FORMAT("RUN COMPLETE")
STOP
END

FTN4 COMPILER: HP92068-1692 REV. 2026 (888423)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 88265 COMMON = 88888
## APPENDIX B

CODE LISTING FOR THE MODIFIED READ CODE II WITH WRAP-AROUND

<table>
<thead>
<tr>
<th>SUBROUTINE</th>
<th>PAGE NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. MRCW</td>
<td>B-1</td>
</tr>
<tr>
<td>2. INITW</td>
<td>B-3</td>
</tr>
<tr>
<td>3. ENCDW</td>
<td>B-7</td>
</tr>
<tr>
<td>4. GTRAN</td>
<td>B-12</td>
</tr>
<tr>
<td>5. CODNG</td>
<td>B-15</td>
</tr>
<tr>
<td>6. CODLN</td>
<td>B-17</td>
</tr>
</tbody>
</table>
FTN4.L.T.C

PROGRAM MRCV

C IMPLICIT INTEGER(A-Z)

C ASSUMPTIONS:
C MAXIMUM LINE LENGTH=496
C MAXIMUM NUMBER OF LINES=5600
C MAXIMUM INPUT RECORD SIZE=256

C***************LABELLED COMMON ARRAYS***************
C
C COMMON/BUFF/PELBUF(256),CDBUF(1024),STAT(1),TRANS(3,1#24)
C COMMON/HUFF/CODE(3,1#5,2),CODERD(3,11)

C***********FILE BUFFERS**********************
C
C COMMON/FILES/TERM,LPFIL,IDC87(144),IDCB9(144),IDCB9(144)
C INTEGER TERM,LPFIL,IDC87,IDCB8,IDCB9

C IDC87 - PELFIL
C IDC88 - OTFIL
C IDC90 - STAFIL

C***************LABELLED COMMON /G16B/ ************
C
C DIMENSION MASK(16),COMASK(16),LIBIT(16),LZBIT(16)
C COMMON /G16B/MASK,COMASK,LIBIT,LZBIT,NBPW

C***************LABELLED COMMON VARIABLES**************
C
C COMMON/IWAR/PBINMAX,LINMAX,K
C COMMON/PVAR/INLNN0,INMCT,CDELCT,TCDATA,TCDRL,CDELW,CDATA,
C = SFBIT,BUFDIM,TOP,BOF,TOPF,COLOR,INP,PMXD64
C COMMON/LOGIC/DIAG,FILEND,OUTF,DONE,WRAP
C LOGICAL DIAG,FILEND,OUTF,DONE,WRAP
C
C DOUBLE PRECISION TCDATA,TCDRL
C
C************END COMMON************
C
C**********LOCAL VARIABLES**********
C
C REAL CF3,CF4
C
C CALL INITW

C 1BP CONTINUE
C IF(INLNN0.GT.LINMAX) GO TO 2000
C CALL EMCDW
C INLNN0=INLNN0+1
C GO TO 100
C
C 2000 CONTINUE
C
C REPORT COMPRESSION FACTOR, ERROR SENSITIVITY FACTOR
ERRATE=FLOAT(ERRCNT)/TCDEL

CALL FTIME(PELBUF)
WRITE(LPFILE,4000) (PELBUF(I),I=1,16)
WRITE(LPFILE,2020) TCDEL,TCDATA
2020 FORMAT("#CODED BITS = ",F8.0/
" CODED DATA BITS = ",F8.0)

C CF3=FLOAT(PELMAX)*FLOAT(INLNT)/TCDEL
CF4=FLOAT(PELMAX)*FLOAT(INLNT)/TCDATA

WRITE(LPFILE,2030) CF3,CF4
2030 FORMAT("#(CF3) = ",F8.4/
" (CF4) = ",F8.4)

WRITE EOF INDICATOR ON STAT FILE & CLOSE

STAT(1)=-1
CALL WRITF(IDC99,IERR,STAT)
IF(IERR.LT.1)STOP 203
CALL CLOSE(IDC99)

4000 FORMAT(1HB,16A2)
CALL CLOSE(IDC77,IERR)
WRITE(TERM,5000)
5000 FORMAT(" RUN COMPLETE")
STOP
END

FTM4 COMPILER: HP9068-1692 REV. 2026 (B3423)

** NO WARNINGS ** NO ERRORS ** PROGRAM = #8235 COMMON = BBBBB
I haven't been able to scan the image, but I can see that it is a page from a printed document written in Fortran. The code appears to be for a subroutine named INITW, which is used to initialize common variables and arrays. The subroutine is written in the Fortran language and includes declarations for variables, dimension statements, and a main program block. The code snippet includes calls to other routines for file handling and image processing, and it appears to be part of a larger program for image processing or manipulation.
C
CALL NAMR(IBM.LINE,2*IDUM,ISTRC)
C
CALL OPEN(IDC89,IERR,FNAM,8,ISECU,ICR)
C
IF(IERR.GE.9) GO TO 5
C
C***********************************************************************
C
C IF NO OUTPUT FILE
C THEN NO ERROR INSERTION
C AND
C NO ERROR COMPARISON (ERRH)
C AND
C NO INPUT LINE NUMBER OR PEL COUNT USED/REQUIRED
C ELSE ERROR INSERTION OPTIONAL
C
C***********************************************************************
C
C
C NO OUTPUT FILE
C
OUTF=.FALSE.
C
WRITE(TERM,6)
C
WRITE(TERM,6)
C
FORMAT("NO OUTPUT FILE SPECIFIED.")
C
C
C GET SCRATCH FILE NAME & OPEN
C
C
C CALL NAMR(IBM.LINE,2*IDUM,ISTRC)
C
CALL OPEN(IDC89,IERR,FNAM,8,ISECU,ICR)
C
IF(IERR.LT.150) STOP 888
C
C READ INPUT RECORD SIZE
C
C
C WRITE(TERM,3)
C
C FORMAT("ENTER INPUT RECORD SIZE: ")
C
C READ(TERM,*) BUFDIM
C
C IF(BUFDIM.GE.9.AND.BUFDIM.LE.256) GO TO 114
C
C WRITE(TERM,15) BUFDIM
C
C GO TO 20
C
C
C READ DIAGNOSTIC SWITCH
C
C
C WRITE(TERM,115)
C
C FORMAT("DIAGNOSTIC PRINTOUT? (Y OR N): ")
C
C READ(TERM,110) INSW
C
C FORMAT(A)
C
C IF(INSW.EQ.21H) GO TO 116
C
C IF(INSW.EQ.22H) GO TO 128
C
C GO TO 114
C
C CONTINUE
C
C DIAG=.TRUE.
C
C
C READ MAXIMUM NUMBER OF PELS PER LINE
C
C
C WRITE(TERM,13)
C
C FORMAT("ENTER MAXIMUM NUMBER OF PELS PER LINE: ")
C
C READ(TERM,*) PELMAX
```fortran
#111 140 FORMAT(14)
#112 IF(MODIPELMAX,64).NE.0) GO TO 145
#113 IF(PELMAX,GE.1.AND.PELMAX.LE.4096) GO TO 320
#114 145 WRITE(TERM,350) PELMAX
#115 150 FORMAT("NUMBER OUT OF RANGE (\$,16,\")")
#116 GO TO 120
#117 C READ NUMBER OF SCAN LINES TO BE PROCESSED
#118 C
#119 C 320 CONTINUE
#120 C WRITE(TERM,330)
#121 C 330 FORMAT("NUMBER OF SCAN LINES TO BE PROCESSED=\")
#122 C READ(TERM,\") LINMAX
#123 C IF(LINMAX,GE.1.AND.LINMAX.LE.56688) GO TO 350
#124 C WRITE(TERM,150) LINMAX
#125 C GO TO 320
#126 C 350 CONTINUE
#127 C
#128 C READ INPUT IMAGE NAME
#129 C
#130 C 360 FORMAT("ENTER INPUT IMAGE NAME:")
#131 C READ(TERM,365) NMBUF
#132 C 365 FORMAT(3A2)
#133 C C WRITE INPUT PARAMETERS
#134 C CALL FTIME(ITBUF)
#135 C WRITE(LPFIL,370) ITBUF
#136 C 370 FORMAT(16A2)
#137 C WRITE(LPFIL,380) NMBUF
#138 C 380 FORMAT("IMAGE NAME=*3A2")
#139 C
#140 C WRITE(LPFIL,400) PELMAX,LINMAX,BUFDIM
#141 C 400 FORMAT("INPUT PARAMETERS;="/)
#142 C "MAXIMUM NUMBER OF PELS PER LINE=",16/
#143 C "NUMBER OF SCAN LINES TO BE PROCESSED =",16/
#144 C "RECORD SIZE ="14)
#145 C WRITE(LPFIL,410)
#146 C 410 FORMAT("NO ERRORS INSERTED")
#147 C
#148 C END PROGRAM -----------------------------------------------
#149 C
#150 C INITIALIZE
#151 C
#152 C WRAP=.TRUE.
#153 C INLMNO=1
#154 C INLMNO=PELMAX/64
#155 C INELCT=PELMAX
#156 C CDLCT=NMBP
#157 C CDELP=NBFP/1
#158 C DG #1 =1,BUFDIM*4
#159 C CDBUF(I)=&
#160 C #161 C CONTINUE
#162 C DO 850 I=1,BUFDIM
#163 C PELBUF(I)=&
#164 C 850 CONTINUE
#165 C
```
C FILL TRANSITIONS LIST
DONE=.FALSE.
TOP=1
BOT=1
INF=0
DO 1888 I=1,1#29
IF(DONE) GO TO 1818
CALL GTRAN
IF(BOT.EQ.TOP) STOP 1888
1888 CONTINUE
1818 CONTINUE
RETURN
END
FTM4,L,T,C
SUBROUTINE ENSDW
C IMPLICIT INTEGER(A-Z)
C**********LABELLED COMMON ARRAYS************
C COMMON/BUFF/PELBUF(256),CDBUF(124),STAT(1),TRANS(3,1624)
C COMMON/HUFF/CDR(C,1,83,2),CODERD(3,11)
C**********FILE BUFFERS***************
C COMMON/FILE/TERM,LPFUL,IDCB7(144),IDCB8(144),IDCB9(144)
INTEGER TERM,LPFUL,IDCB7,IDCB8,IDCB9
C IDC87 - PELFIL
C IDC88 - OLFIL
C IDC89 - STATFL
C**********LABELLED COMMON /G1GBT/ **********
C DIMENSION MASK(16),COMASK(16),LBIT(16),LZBIT(16)
C COMMON /G1GBT/MASK,COMASK,LBIT,LZBIT,NBPW
C**********LABELLED COMMON VARIABLES********
C COMMON/IVAR/PELMAX,LINEAX,K
C COMMON/PVAR/INLNO,INLCT,CDELCT,TCDATA,TCDEL,CDELW,CDATA,
C STBIT,BUFDS,TOP,BOT,TOPO,E,Y,COLOR,INP,PMXDO
C COMMON/LOGIC/DIAG,FILEND,OUTF,DONE,WRAP
C LOGICAL DIAG,FILEND,OUTF,DONE,WRAP
C DOUBLE PRECISION TCDATA,TCDEL
C**********END COMMON*****************************************
C******************************************************
C INCMIX(IX)=MOD(IX+124,124)+1
C DIFF(IA,IB,ILA,ILB)=(ILA-ILB-1)*PELMAX+IA-IB
C DIFF(IA,IB,ILA,ILB)=(ILA-ILB+1)*PELMAX+IA-IB
C INITIALIZ VARIABLES
C CDELCT=NBPW
C CDDATA=1
C DO 6# I=2,BUFDS=4
C CDBUF(I)=1
C CONTINUE
C IF(INLNO .NE. 1) GO TO 600
C ONE-DIMENSIONAL CODING
C WRITE ONE EO1
CALL CODNG(10,0)
TB1=TOP
POLAR=1
A=1
LA=1
LB=1
C CALCULATE RUNLENGTH AND ENCODE
150 CONTINUE
B=TRANS(2,TOP)
C TEST FOR END OF LINE
IF(TRANS(1,TOP).GT.INLMNO) B=PELMAX+1
CALL CODLN(B,A,LA-LA,POLAR)
RUN=B-A
IF(DIAG)WRITE(TERM,160) RUN,POLAR,CDELCT,CDDATA
160 FORMAT(18)
C UPDATE OLD RUN END AND POLARITY
A=B
POLAR=TRANS(3,TOP)+1
C TEST FOR END OF LINE
IF(A.GT.PELMAX) GO TO 1800
C INCREMENT TOP
TOP=INCMD(TOP)
IF(TOP.EQ.BOT) STOP 168
IF(.NOT.DONE.AND.INCMD(BOT).NE.TB1) GTRAN
GO TO 150
C TWO-DIMENSIONAL CODING
600 CONTINUE
STFBIT=STFBIT+1
C IF PREVIOUS LINE IS ONE-DIMENSIONAL, WRITE ONE EOL2
C CALL CODNG(11,0)
C SET AD TO LEFT EDGE-1
AD=0
LAB=2
TAI=TOP
POLAR=1
GO TO 620
C INITIALIZE CODE LINE POINTERS

615 CONTINUE
CDELECT=NPW
CDDATA=$
DO 615 I=2,BUF(I)
615 CONTINUE

C DETECT A1
620 CONTINUE
A1=TRANS(2,TA1)
LA1=TRANS(1,TA1)
IF(LA1.GT.LINMAX) LA1=LINMAX

C DETECT B1
655 CONTINUE
B1=TRANS(2,TB1)
LB1=TRANS(1,TB1)
IF(LB1.GT.PELMAX) GO TO 700
IF(LB1.GE.LA1) GO TO 655
IF(LB1+1.EQ.LA1.AND.B1.GT.AB) GO TO 655
GO TO 66#

C B1 TO RIGHT OF A# TEST FOR OPPOSITE POLARITY
670 C
655 IF(TRANS(3,TB1)+1.NE.POLAR) GO TO 67#

C SAME COLOR LOOK AT NEXT TRANSITION
660 TB1=INCMD(TB1)
GO TO 65#

C HAVE B1
670 CONTINUE

C DETECT B2
680 TB2=INCMD(TB2)
B2=TRANS(2,TB2)
LB2=TRANS(1,TB2)
IF(B2.GT.PELMAX) GO TO 710
GO TO 740
700 B1=PELMAX+1
710 B2=PELMAX+1
740 CONTINUE

C TEST FOR PASS MODE
690 IF(LB2.GE.LA1) GO TO 750
653 IF(LB2+1.EQ.LA1.AND.B2.GE.A1) GO TO 750

C

C PASS MODE CODING (CANT END A LINE IN PASS MODE; NEW AS MUST HAVE
SAME POLARITY AS B2)

C

CALL CODNG(1,0)
A#-B2
LA#=LB2+1
TB1=TB2
GO TO 62#

75# CONTINUE

IF(IABS(LA#-(LB1+1)).GT.1) GO TO 799
MAB=IABS(DIFF(A1,B1,LA1,LB1))
IF(DIFG)WRITE(LFIL,16#) A1,B1,LA1,LB1,MAB
IF(MAB=3) 835,835,799

C

CODE BY HORIZONTAL MODE; FIRST DETECT A2

C

799 CONTINUE
IF(A1.GT.PELMAX) GO TO 88#
TA2=INCMD(TA1)
A2=TRANS(2,TA2)
LA2=TRANS(I,TA2)
IF(LA2.GT.LINMAX) LA2=LINMAX
GO TO 818

88# CONTINUE

A2=PELMAX+1
LA2=LA1
TA2=TA1

818 CONTINUE
IF(INLNNO.EQ.2.AND.A8.EQ.0) A#=1
CALL CODNG(2,POLAR)
IF(DIFG)WRITE(LFIL,16#) A8,A1,A2,LA8,LA1,LA2,POLAR
CALL CODLN(A8,LA8,LA1,LA2,POLAR)
NEWPOL=MOD(POLAR+2,2)+1
CALL CODLN(A8,A1,LA2-LA1,NEWPOL)

A8=A2
LA8=LA2
TOP=TA2
IF (.NOT.DONE.AND.INCMD(BOT).NE.TB1) CALL GTRAN
IF (.NOT.DONE.AND.INCMD(BOT).NE.TB1) CALL GTRAN
GO TO 96#

C

CODE BY VERTICAL MODE

C

835 CONTINUE
AIMBI=DIFF(A1,B1,LA1,LB1)
IF(AIMBI) 858,84#,84#

85# CALL CODNG(AIMBI+3,0)
GO TO 95#

85# CONTINUE
BIMAI=DIFF(B1,A1,LB1,LA1)
CALL CODNG(BIMAI+6,0)

95# CONTINUE
A#-A1
IZZ 2I0 10 416 0 .n S I- b IM woo U-6-U S Mt 0 .PIw = no 39 0 K 44 amW Z CI-M j I-- IA Zw a 0 U p-U (Au0 a w - a I- u I-- 36 CI~ $~~ CI- n0 3i UW wu1 Ut 267* u u. uw 485 a 191x167 * I aU $~~ N W N £ 4 -

PAGE #95 ENCDW 4:56 PM SUN., 31 OCT., 1982

#221  LAY=LAI
#222  TOP=TAI
#223  IF(.NOT.DONE.AND.INCM(80T).NE.TBI) CALL GTRAN
#224  C
#225  C TEST FOR END OF LINE
#226  C
#227  96# CONTINUE
#228  POLAR=TRANS(3, TOP)+1
#229  TAI=INCM(TOP)
#230  IF(TRANS(1, TOP).LE.INLNO) GO TO 62#
#231  C
#232  C LINE FINISHED
#233  C
#234  INLNO=TRANS(1, TOP)
#235  1## CONTINUE
#236  C SAVE LINE LENGTH (DATA BITS ONLY)
#237  C
#238  STAT(I)=CDATA
#239  CALL WRITF(IDC9, IERR, STAT)
#240  IF(IERR.LT.0) STOP 38#
#241  C
#242  C COMPUTE STATISTICS
#243  C
#244  39# CONTINUE
#245  TCDEL=TCDEL+CDELCT-NBPW
#246  TCDATA=TCDATA+CDATA
#247  C
#248  IF(.NOT.DIAG) GO TO 468
#249  CDELW=(CDELCT+NBPW-1)/NBPW
#250  WRITE(LPFL, 450) (CDBUF(1), I=1, CDELW)
#251  45# FORMAT(BO12)
#252  468 CONTINUE
#253  C
#254  C TEST FOR END OF FILE
#255  C
#256  IF(INLNO.GT.1.AND.INLNO.LT.32888) GO TO 618
#257  RETURN
#258  C
#259  C
#260  C E N D

FTR4 COMPILER: HF92#68-16#92 REV. 2826 (88423)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 8848 COMMON = 88888
FIN4, L.T.C
SUBROUTINE GTRAN
C RETURNS ONE TRANSITION AT EACH CALL
C IMPLICIT INTEGER(A-Z)
C**********LABELLED COMMON ARRAYS************
C
C COMMON/BUFF/PELBUF(256), CDBUF(1624), STAT(1), TRANS(3,1624)
C COMMON/HUFF/COKE(3,165,2), CODERD(3,11)
C**********FILE BUFFERS***************
C
C COMMON/FILES/TERM,LPFIL, IDC87(144), IDC88(144), IDC9(144)
C INTEGER TERM, LPFIL, IDC87, IDC88, IDC9
C
C IDC87 - PELFIL
C IDC88 - OTFIL
C IDC99 - STATFL
C**********LABELLED COMMON /GI6BT/ ************
C
C DIMENSION MASK(16),COMASK(16),LIBIT(16), LZBIT(16)
C COMMON /GI6BT/MASK,COMASK,LIBIT,LZBIT,NBPW
C**********LABELLED COMMON VARIABLES***********
C
C COMMON/VAR/PELMAX, LINMAX, K
C COMMON/IVAR/INLNO, INLNC, CDECT, TCDATA, TCDEL, CDELW, CDATA,
C * STFBMT, BUFDIM, TOP, BOT, TOPREF, COLOR, INP, PNX64
C COMMON/LGC/LCLG, diag, FILEND, OUTF, DONE, WRAP
C LOGICAL DIAG, FILEND, OUTF, DONE, WRAP
C
C DOUBLE PRECISION TCDATA, TCDEL
C
C**********END COMMON*************************
C
C INCMDIX=MOD(IX+1624,1624)+1
C
C********** BEGIN PROGRAM *******************
C
LINE AVAILABLE?
C
110 CONTINUE
C
IF (IM) 118,288,388
110 STOP 118
C
C NO READ NEXT SCAN LINE
C
288 CONTINUE
C
CALL READF(IDC87, IERR, PELBUF, BUFDIM, LEN)
C IF (IERR.EQ. -12 OR LEN.EQ. -1) GO TO 288
C IF (IERR.LT.0) STOP 288
C IF (INLNC.GE.LINMAX) GO TO 288
INLNCT=INLNCT+1
IF(WRAP) GO TO 260
STORE REFERENCE TRANSITION
TRANS(1,BOT)=INLNCT
TRANS(2,BOT)=
TRANS(3,BOT)=
IF(DIAG)WRITE(LPFIL,500)TRANS(1,BOT),TRANS(2,BOT),TRANS(3,BOT),BOT
BOT=INCMO(BOT)
COLOR=#
INP=1
RETURN
EOF
STORE FINAL TRANSITION WITH MAX LINE NO. AT 0 AND WHITE
SET DONE FLAG
CONTINUE
DONE=.TRUE.
TRANS(1,BOT)=32000
TRANS(2,BOT)=PELMAX+1
TRANS(3,BOT)=
IF(DIAG)WRITE(LPFIL,500)TRANS(1,BOT),TRANS(2,BOT),TRANS(3,BOT),BOT
BOT=INCMO(BOT)
RETURN
260 INP=1
LINE AVAILABLE TEST FOR BEGINNING OF WORD
300 CONTINUE
IF(INP.GT.PELMAX) GO TO 780
IF(MOD(INP,40)) INP,350,400
AT BEGINNING OF WORD TEST FOR COLOR CHANGE
CONTINUE
PEL=14B(PELBUF,INP,1)
IF(PEL.NE.COLOR) GO TO 420
WORD=(INP+NPW-1)/NPW
GO TO (369,379).COLOR+1
369 IF(PELBUF(WORD)) 480,380,480
379 IF(PELBUF(WORD).NE.-1) GO TO 480
WORD ALL BLACK OR ALL WHITE
CONTINUE
380 CONTINUE
IMP=IMP+16
IF(INP.LE.PELMAX) GO TO 350
GO TO 780
NOT AT BEGINNING OF WORD OR NOT ALL ONE COLOR
CONTINUE
480 CONTINUE
PEL=14B(PELBUF,INP,1)
IF(PEL.NE.COLOR) GO TO 420
IMP=IMP+1
```
IF(INP.LE.PELMAX) GO TO 330
GO TO 700
C HAVE A TRANSITION
420 CONTINUE
COLOR=PEL
TRANS(1,BOT)=INLNCT
TRANS(2,BOT)=INF
TRANS(3,BOT)=COLOR
IF(DIAG)WRITE(LPFL,5#)TRANS(1,BOT),TRANS(2,BOT),TRANS(3,BOT),BOT
500 FORMAT(4(1B)
C INCREMENT BOT POINTER
INCP=INCP+1
RETURN
C ADD END-OF-LINE TRANSITION
700 CONTINUE
IF(WRAP) GO TO 200
TRANS(1,BOT)=INLNCT
TRANS(2,BOT)=PELMAX+1
TRANS(3,BOT)=9
INF=9
IF(DIAG)WRITE(LPFL,5#)TRANS(1,BOT),TRANS(2,BOT),TRANS(3,BOT),BOT
BOT=INCM(BOT)
RETURN
END
```

**SUBROUTINE CODNG(MODE,POLAR)**

**C** IMPLICIT INTEGER(A-Z)**

**C*************LABELLED COMMON ARRAYS***************

**C** COMMON/BUFF/PELBUFF(256),CDUFFL(1824),STAT(1),TRANS(3,1824)

**C** COMMON/HUFF/CODE(3,1825),CODERD(3,11)

**C*************FILE BUFFERS***************

**C** COMMON/FILES/TERM,LPFIL,IDCB7(144),IDCB8(144),IDCB9(144)

**C** INTEGER TERM,LPFIL,IDCB7,IDCB8,IDCB9

**C** IDC7 - PELFIL

**C** IDC8 - OPEIL

**C** IDC9 - STATFL

**C*************LABELLED COMMON /G16BT/ ***************

**C** DIMENSION MASK(16),COMASK(16),LIBIT(16),LZBIT(16)

**C** COMMON /G16BT/MASK,COMASK,LIBIT,LZBIT,NBPM

**C*************LABELLED COMMON VARIABLES***********

**C** COMMON/IVAR/PELMAX,LINMAX,K

**C** COMMON/IVAR/INLNR4,INLCT,CDELCT,TCDATA,TCDIM,CDLW,CDDATA,

**C** STFBIT,BUF,DIN,TOP,BOT,TOPBP,COLOUR,INP,PMX64

**C** COMMON/LOGIC/DIAG,FILEND,OUTF,DONE,WRAP

**C** LOGICAL DIAG,FILEND,OUTF,DONE,WRAP

**C** DOUBLE PRECISION TCDATA,TCDIM

**C*************END COMMON*****************************

**C** INTEGER A,B,C,MODE,NEWPOL,POLAR

**C************* BEGIN PROGRAM ***********************

**C** CALL M121BICODERD(3,MODE,CDBUF,CDLCT,1,CODERD1,MODE))

**C** CDELCT=CDELCT+CODERD1,MODE)

**C** GO TO (180,280,180,180,180,180,180,180,180,180,888,888),MODE

**C** MODE 1 2 3 4 5 6 7 8 9 10 11

**C** PASS MODE(1),VERTICAL MODE: A1B1=0(3), A1B1=1(4,7),=2(5,8),=3(6,9)

**C** 180 CONTINUE

**C** CDATA=CDATA+CODERD1,MODE)

**C** RETURN

**C** HORIZONTAL MODE(2)

**C** 280 CONTINUE
SUBROUTINE COLD (AMB, LAMB, POLAR)

IMPLICIT INTEGER(A-Z)

C**********LABELLED COMMON ARRAYS**************
C
COMMON/BUFF/PELBUF(258),CDBUF(1824),STAT(1),TRANS(3,1024)
COMMON/NUFF/CODE(3,1#5,2),CODERD(3,11)
C**********FILE BUFFERS*************************
C
COMMON/FILES/TERM,LPFIL,IDCB7(144),IDCB8(144),IDCB9(144)
INTEGER TERM,LPFIL,IDCB7,IDCB8,IDCB9
C
IDCB7 - PELFIL
IDCB8 - OTFIL
IDCB9 - STATFL
C**********LABELLED COMMON /G16BT/ **************
C
DIMENSION MASK(16),COMASK(16),LIBIT(16),LZBIT(16)
COMMON /G16BT/MASK,COMASK,LIBIT,LZBIT,NBPW
C**********LABELLED COMMON VARIABLES**********
C
COMMON/IVAR,PELMAX,LINMAX,K
COMMON/PVAR/INLNO,INLCT,TCDEL,TCDEL,CDELW,CDDATA,
STBIT,BUFIDN,TOP,BOT,TOREF,COLOR,INF,PMXD64
COMMON/LOGIC/DIAF,FILED,OFT,DONE,WRAP
LOGICAL DIAF,FILED,OFT,DONE,WRAP
C
DOUBLE PRECISION TCDEL,TCDATA
C********************END COMMON*********************
C
INTEGER LENGTH,POLAR,INDEX,DEX,CODESW,TCODE,TLENG
INTEGER AMB,LAMB,PMXD64
C*************** BEGIN PROGRAM *********************
C
C CHECK INPUTS
C
IF(POLAR.LT.1.OR.POLAR.GT.2) STOP 3333
LENGTH=64
IF(LAMB.LT.2) LENGTH=LAMB*PELMAX+AMB
C
IF(LENGTH.LE.63) GO TO 48
C
ATTACK LONG RUN IN PIECES
C
IND256=#
6 IF(LAMB.LE.256) GO TO 15
LAMB=LAMB-256
DEX=1#4
IND256=256*PMXD64+IND256
PAGE 8884 COILM 12:26 PM WED., 3 NOV., 1982

10 IF(INDEX.LE.40) GO TO 5
110 INDEX=INDEX-1
1110 INDEX=INDEX-1
1120 IF(INDEX.LE.100) GO TO 30
1130 INDEX=INDEX+4
1140 DEX=INDEX
1150 ASSIGN 29 TO CODESW
1160 D IF(DIAG)WRITE(TERM,25)CODESW,DEX,POLAR
1170 DATA=CODESW
1180 DATA=CODESW
1190 DATA=CODESW
1200 DATA=CODESW
1210 DATA=CODESW
1220 DATA=CODESW
1230 DATA=CODESW
1240 DATA=CODESW
1250 DATA=CODESW
1260 DATA=CODESW
1270 DATA=CODESW
1280 DATA=CODESW
1290 DATA=CODESW
1300 DATA=CODESW
1310 DATA=CODESW
1320 DATA=CODESW
1330 DATA=CODESW
1340 DATA=CODESW
1350 DATA=CODESW
1360 DATA=CODESW
1370 DATA=CODESW
1380 DATA=CODESW
1390 DATA=CODESW
1400 DATA=CODESW
1410 DATA=CODESW
1420 DATA=CODESW
1430 DATA=CODESW
1440 DATA=CODESW
1450 DATA=CODESW
1460 DATA=CODESW
1470 DATA=CODESW
1480 DATA=CODESW
1490 DATA=CODESW
1500 DATA=CODESW
1510 DATA=CODESW
1520 DATA=CODESW
1530 DATA=CODESW
1540 DATA=CODESW
1550 DATA=CODESW
1560 DATA=CODESW
1570 DATA=CODESW
1580 DATA=CODESW
1590 DATA=CODESW
1600 DATA=CODESW
1610 DATA=CODESW
1620 DATA=CODESW

FTN4 COMPILER: HP92068-16092 REV. 2026 (888423)

** NO WARNINGS ** NO ERRORS ** PROGRAM = 88106 COMMON = 88000