NCS TIB 79-10

NATIONAL COMMUNICATIONS SYSTEM

TECHNICAL INFORMATION BULLETIN
79-10

MEASUREMENT OF COMPRESSION FACTOR AND ERROR SENSITIVITY FACTOR OF FACSIMILE CODING TECHNIQUES SUBMITTED TO THE CCITT BY GREAT BRITAIN AND THE FEDERAL REPUBLIC OF GERMANY

OCTOBER 1979

APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED

DISCLAIMER NOTICE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NCS TIB-79-10	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Measurement of Compression Factor and Error Sensitivity Factor of Facsimile Coding Techniques Submitted to the CCITT By Great Britain and the Federal Republic of Germany		5. TYPE OF REPORT & PERIOD COVERED Final
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Neil Randall, Richard Schaphorst, Steve Urban		8. CONTRACT OR GRANT NUMBER(s) DCA100-79-M-0209
9. PERFORMING ORGANIZATION NAME AND ADDRESS Delta Information Systems, Inc., 259 Wyncote Road, Jenkintown, PA 19046		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS National Communications System, Office of Technology and Standards (NCS-TS), Washington, D.C. 20305		12. REPORT DATE October 1979
		13. NUMBER OF PAGES 141
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution unlimited; approved for public release

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Image Coding	Error Sensitivity	Image Statistics
Digital Facsimile	Two-dimensional Coding	CCITT Standards
Facsimile Coding	Coding Algorithms	
Compression Factor	Computer Simulation	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This Technical Information Bulletin (TIB) describes the measurement of Compression factor and error sensitivity factor of two facsimile coding techniques submitted to the CCITT for adoption as an international standard by Great Britain and the Federal Republic of Germany. The TIB contains detailed flow charts and code listings for each algorithm. Compression factor, error sensitivity factor and statistical data have been tabulated. This TIB is a companion document to NCS TIB 79-9.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

NCS TECHNICAL INFORMATION BULLETIN 79-10

MEASUREMENT OF COMPRESSION FACTOR AND ERROR SENSITIVITY FACTOR OF FACSIMILE CODING TECHNIQUES SUBMITTED TO THE CCITT BY GREAT BRITAIN AND GERMANY

October 1979

PREPARED BY:

DENNIS BODSON
Senior Electronics Engineer
Office of NCS Technology
and Standards

APPROVED FOR PUBLICATION:

MARSHALL L. CAIN
Assistant Manager
Office of NCS Technology
and Standards

FOREWORD

Among the responsibilities assigned to the Office of the Manager, National Communications System, is the management of the Federal Telecommunication Standards Program which is an element of the overall GSA Federal Standardization 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 FACTOR AND ERROR SENSITIVITY FACTOR OF FACSIMILE CODING TECHNIQUES SUBMITTED TO THE CCITT BY GREAT BRITAIN AND GERMANY

October, 1979

FINAL REPORT

Submitted to:

NATIONAL COMMUNICATIONS SYSTEMS
8th & S. COURTHOUSE RD.
ARLINGTON, VIRGINIA 2204

CONTRACTING AGENCY:

DEFENSE COMMUNICATIONS AGENCY

Purchase Order: DCA 100-79-M-0209

Submitted by:

DELTA INFORMATION SYSTEMS, INC.
259 WYNCOTE ROAD
JENKINTOWN, PENNA. 19046

TABLE OF CONTENTS

APPENDICES

1.0 INTRODUCTION

Several organizations have submitted contributions to the CCITT (see Appendices A, B, and References 4, 5, 9, 10, 11) describing two-dimensional coding techniques for selection of a standard compression algorithm for advanced digital facsimile systems. At the December 1978 meeting in Geneva, a working party of CCITT Study Group XIV adopted specific procedures to measure compression and error sensitivity so that candidate coding techniques may be compared on a meaningful basis. These definitions and procedures are outlined in references 1 and 2. The National Communications System of the U. S. Government has issued three contracts to Delta Information Systems, Inc. to evaluate seven candidate two-dimensional coding techniques using the criteria recommended by the CCITT.

In the first contract (Purchase Order DCA-79-M-0105), a basic computer program was developed to measure the compression and error sensitivity of digital facsimile coding techniques. To validate this program, the Modified-Huffman code, recommended as the one-dimensional standard for Group 3 machines, was tested and simulated on the model. The computer program and work accomplished on this initial contract is described in a Final Report issued August 10, 1979 (see Reference 3).

In the second contract, the validated computer model was used to measure the compression and error sensitivity of five two-dimensional coding techniques. The five coding algorithms which were selected for simulation were proposed by Japan, 3M, IBM, XEROX, and AT&T. These coding techniques were selected simply because no other contributions had been submitted to the CCITT when this NCS measurement contract was initiated. Contributions were subsequently submitted to the CCITT by the Federal

Republic of Germany and the British Post Office. The NCS organization issued a third contract (Purchase Order DCA 100-79-M-0209) to Delta Information Systems to measure the compression and error sensitivity of these latter two coding techniques and the results of this investigation are included in this document.

The measurement parameters which were involved in this program are summarized in Section 2.0 of this report. Section 3.0 describes the hierarchy and interrelationship of computer programs which are used in the measurement process. In many instances, the proposed operation of the coding algorithm was not totally defined when a transmission error was encountered. Section 4.0 describes the generalized error detection and correction procedure which was employed. As the computer programs were prepared for each algorithm, certain assumptions were made for each coding technique, particularly in the area of error detection and correction. These assumptions made for each individual coding technique are documented in Section 5.0.

Five separate computer runs were implemented for each algorithm at different combinations of test document, vertical resolution and K-factor. Section 6.0 summarizes the results of these measurements in terms of compression data, error sensitivity data, and coded line length statistics. Section 7.0 contains a list of reference documents related to the contract.

The CCITT contributions describing the two coding algorithms have been included in Appendices A and B for reference purposes. Appendix C contains the program code listings for those subroutines which are common to all algorithms, e.g. data packing, data unpacking, error measurement, etc. Appendices D, E, F, and G contain the flow charts and the listing of the code for the computer program for the two algorithms which were investigated.

Delta Information Systems wishes to acknowledge the Contracting Officer's Technical Representative, Dennis Bodson, for the extraordinary level of support he has provided during the course of this contract. The assistance of Marla Thomas, from the DCEC computer facility, is also greatly appreciated.

2.0 MEASUREMENT PARAMETERS

In this section, the various parameters involved in the measurement of compression and error sensitivity will be summarized. In general, Study Group XIV of the CCITT agreed upon these measurement parameters at the general meeting held in Geneva in December 1978 (see Reference 2).

2.1 Test Documents

The test documents were chosen from the eight CCITT test documents (see Figure 2-1) since they have been widely used by data compression experimenters in the past. Documents numbered 1, 4, 5, and 7 (see Figures 2-2, 2-3, 2-4, and 2-5 respectively) were selected as the standard test images since these were considered most representative of documents to be transmitted.

The French PTT Administration has scanned the eight CCITT documents at the high resolution specified for Group 3 machines--7.7 lines/mm. They have also quantized each pel to be either black or white and stored the resultant image on magnetic tape. This tape was used as the source of input documents in this simulation program. Appendix B of Reference 3 describes the format of the test document magnetic tape supplied by the French PTT.

2.2 Resolution

It was agreed that measurements would be performed at both standard resolution (3.85 lines/mm.) and high resolution (7.7 lines/mm.). In the high resolution case, all lines on the input test documents shall be used. In standard resolution tests, every odd scan line

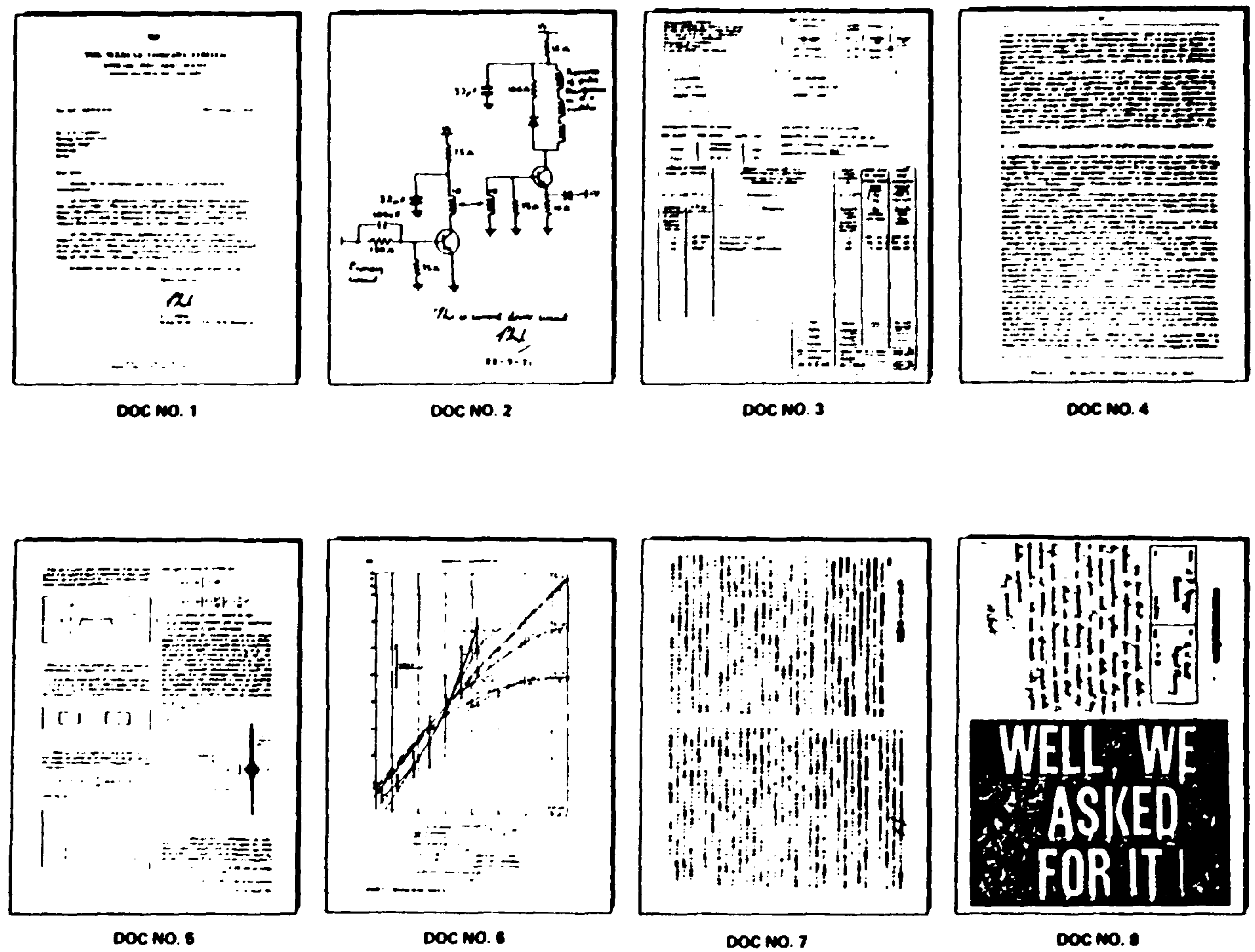


Figure 2-1 CCITT Standard Test Documents

SAPORS LANE · BOOLE · DORSET · BH 25 8 ER

TELEPHONE BOOLE (945 13) 51617 · TELEX 123456

Our Ref. 350/PJC/EAC 18th January, 1972.

Dr. P.N. Cundall,
Mining Surveys Ltd.,
Holroyd Road,
Reading,
Berks.

Dear Pete,

Permit me to introduce you to the facility of facsimile transmission.

In facsimile a photocell is caused to perform a raster scan over the subject copy. The variations of print density on the document cause the photocell to generate an analogous electrical video signal. This signal is used to modulate a carrier, which is transmitted to a remote destination over a radio or cable communications link.

At the remote terminal, demodulation reconstructs the video signal, which is used to modulate the density of print produced by a printing device. This device is scanning in a raster scan synchronised with that at the transmitting terminal. As a result, a facsimile copy of the subject document is produced.

Probably you have uses for this facility in your organisation.

Yours sincerely,

Phil.

P.J. CROSS
Group Leader - Facsimile Research

Figure 2-2 CCITT Test Document No. 1

Registered in England: No. 2038
Registered Office: 60 Vicars Lane, Ilford, Essex.

L'ordre de lancement et de réalisation des applications fait l'objet de décisions au plus haut niveau de la Direction Générale des Télécommunications. Il n'est certes pas question de construire ce système intégré "en bloc" mais bien au contraire de procéder par étapes, par paliers successifs. Certaines applications, dont la rentabilité ne pourra être assurée, ne seront pas entreprises. Actuellement, sur trente applications qui ont pu être globalement définies, six en sont au stade de l'exploitation, six autres se sont vu donner la priorité pour leur réalisation.
Chaque application est confiée à un "chef de projet", responsable successivement de sa conception, de son analyse-programmation et de sa mise en oeuvre dans une région-pilote. La généralisation ultérieure de l'application réalisée dans cette région-pilote dépend des résultats obtenus et fait l'objet d'une décision de la Direction Générale. Néanmoins, le chef de projet doit dès le départ considérer que son activité a une vocation nationale donc refuser tout particularisme régional. Il est aidé d'une équipe d'analystes-programmeurs et entouré d'un "groupe de conception" chargé de rédiger le document de "définition des objectifs globaux" puis le "cahier des charges" de l'application, qui sont adressés pour avis à tous les services utilisateurs potentiels et aux chefs de projet des autres applications. Le groupe de conception comprend 6 à 10 personnes représentant les services les plus divers concernés par le projet,et comporte obligatoirement un bon analyste attaché à l'application.

II - L'IMPLANTATION GEOGRAPHIQUE D'UN RESEAU INFORMATIQUE PERFORMANT

L'organisation de l'entreprise française des télécommunications repose sur l'existence de 20 régions. Des calculateurs ont été implantés dans le passé au moins dans toutes les plus importantes. On trouve ainsi des machines Bull Gamma 30 à Lyon et Marseille, des GE 425 à Lille, Bordeaux, Toulouse et Montpellier, un GE 437 à Massy, enfin quelques machines Bull 300 TI à programmes câblés étaient récemment ou sont encore en service dans les régions de Nancy, Nantes, Limoges, Poitiers et Rouen ; ce parc est essentiellement utilisé pour la comptabilité téléphonique.
A l'avenir, si la plupart des fichiers nécessaires aux applications décrites plus haut peuvent être gérés en temps différé, un certain nombre d'entre eux devront nécessairement être accessibles, voire mis à jour en temps réel : parmi ces derniers le fichier commercial des abonnés, le fichier des renseignements, le fichier des circuits, le fichier technique des abonnés contiendront des quantités considérables d'informations.
Le volume total de caractères à gérer en phase finale sur un ordinateur ayant en charge quelques 500 000 abonnés a été estimé à un milliard de caractères au moins. Au moins le tiers des données seront concernées par des traitements en temps réel.
Aucun des calculateurs énumérés plus haut ne permettait d'envisager de tels traitements.
L'intégration progressive de toutes les applications suppose la création d'un support commun pour toutes les informations, une véritable "Banque de données", répartie sur des moyens de traitement nationaux et régionaux, et qui devra rester alimentée, mise à jour en permanence, à partir de la base de l'entreprise, c'est-à-dire les chantiers, les magasins, les guichets des services d'abonnement, les services de personnel etc.
L'étude des différents fichiers à constituer a donc permis de définir les principales caractéristiques du réseau d'ordinateurs nouveaux à mettre en place pour aborder la réalisation du système informatif. L'obligation de faire appel à des ordinateurs de troisième génération, très puissants et dotés de volumineuses mémoires de masse, a conduit à en réduire substantiellement le nombre.
L'implantation de sept centres de calcul interrégionaux constituera un compromis entre : d'une part le désir de réduire le coût économique de l'ensemble, de faciliter la coordination des équipes d'informaticiens; et d'autre part le refus de créer des centres trop importants difficiles à gérer et à diriger,et posant des problèmes délicats de sécurité. Le regroupement des traitements relatifs à plusieurs régions sur chacun de ces sept centres permettra de leur donner une taille relativement homogène. Chaque centre "gèrera" environ un million d'abonnés à la fin du VIème Plan.
La mise en place de ces centres a débuté au début de l'année 1971 : un ordinateur IRIS 50 de la Compagnie Internationale pour l'Informatique a été installé à Toulouse en février ; la même machine vient d'être mise en service au centre de calcul interrégional de Bordeaux.

Figure 2-3 CCITT Test Document No. 4

Photo n° 1 - Document très dense lettre 1,5mm de haut -

Restitution photo n° 9

Cela est d'autant plus valable que $T\Delta f$ est plus grand. A cet égard la figure 2 représente la vraie courbe donnant $|\phi(f)|$ en fonction de f pour les valeurs numériques indiquées page précédente.

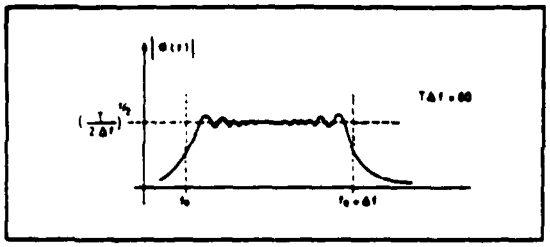


FIG. 2

Dans ce cas, le filtre adapté pourra être constitué, conformément à la figure 3, par la cascade :

— d'un filtre passe-bande de transfert unité pour $f_0 \leq f \leq f_0 + \Delta f$ et de transfert quasi nul pour $f < f_0$ et $f > f_0 + \Delta f$, filtre ne modifiant pas la phase des composants le traversant ;

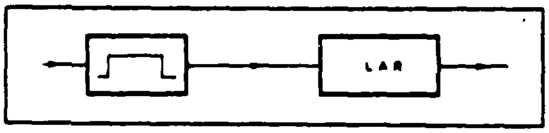


FIG. 3

— filtre suivi d'une ligne à retard (LAR) dispersive ayant un temps de propagation de groupe T_R décroissant linéairement avec la fréquence f suivant l'expression :

$$T_R = T_0 + (f_0 - f)\frac{T}{\Delta f} \quad (\text{avec } T_0 > T)$$

(voir fig. 4).

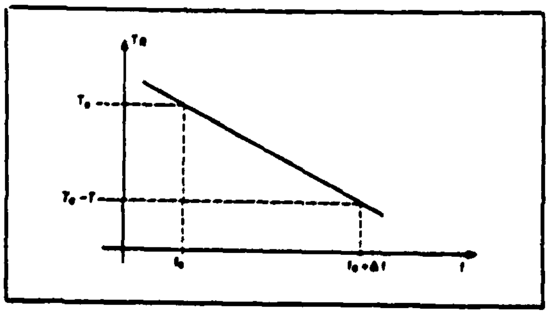


FIG. 4

telle ligne à retard est donnée par :

$$\varphi = -2\pi \int_0^f T_R \, df$$

$$\varphi = -2\pi \left[T_0 + \frac{f_0 T}{\Delta f} \right] f + \pi \frac{T}{\Delta f} f^2$$

Et cette phase est bien l'opposé de $/\phi(f)$, à un déphasage constant près (sans importance) et à un retard T_0 près (inévitable).

Un signal utile $S(t)$ traversant un tel filtre adapté donne à la sortie (à un retard T_0 près et à un déphasage près de la porteuse) un signal dont la transformée de Fourier est réelle, constante entre f_0 et $f_0 + \Delta f$, et nulle de part et d'autre de f_0 et de $f_0 + \Delta f$, c'est-à-dire un signal de fréquence porteuse $f_0 + \Delta f/2$ et dont l'enveloppe a la forme indiquée à la figure 5, où l'on a représenté simultanément le signal $S(t)$ et le signal $S_1(t)$ correspondant obtenu à la sortie du filtre adapté. On comprend le nom de récepteur à compression d'impulsion donné à ce genre de filtre adapté : la « largeur » (à 3 dB) du signal comprimé étant égale à $1/\Delta f$, le rapport de compression est de $\frac{T}{1/\Delta f} = T\Delta f$

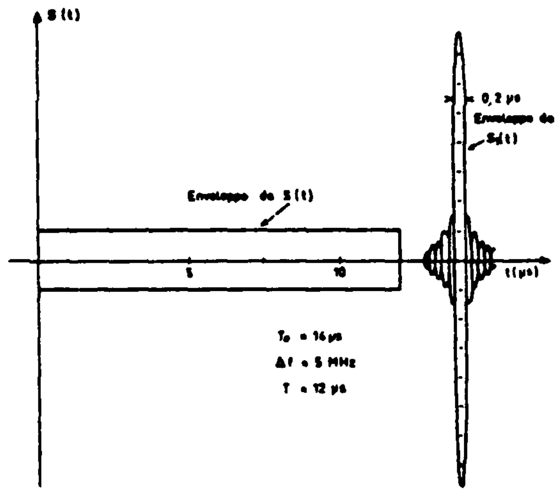


FIG. 5

On saisit physiquement le phénomène de compression en réalisant que lorsque le signal $S(t)$ entre *dans la ligne à retard* (LAR) *la fréquence qui entre* la première à l'instant 0 est la fréquence basse f_0, qui met un temps T_0 pour traverser. La fréquence f entre à l'instant $t = (f - f_0)\frac{T}{\Delta f}$ et elle met un temps $T_0 - (f - f_0)\frac{T}{\Delta f}$ pour traverser, ce qui la fait ressortir à l'instant T_0 également. Ainsi donc, le signal $S(t)$

Figure 2-4 CCITT Test Document No. 5

CCITTの概要

沿革

CCITTは、国際電気通信連合（ITU）の四つの常設機関（事務総局、国際周波数登録委員会、CCIR、CCITT）の一つとして、ITUの中でも、世界の国際通信上の諸問題を真先に取上げ、その解決方法を見出して行く重要な機関である。日本名は、国際電信電話諮問委員会と称する。

CCITTの前身は、CCIF（国際電話諮問委員会）とCCIT（国際電信諮問委員会）である。CCIFは、1924年にヨーロッパに「国際長距離電話通信諮問委員会」が設置され、これが1925年のパリ電信電話会議のとき、正式に、「国際電話諮問委員会」として万国電信連合の公式機関となったものである。CCITは、同じく1925年の会議のとき、CCIFと併立するものとして設置された。

そして、CCIFは、1956年の12月に第18回総会が開催されたのち、CCITは、同年同月に第8回総会が開催されたのち、併合されて現在のCCITTとなった。このCCITTは、CCIFとCCITが解散した直後、第1回総会を開催し、第2回総会は、1960年にニューデリーで、第3回総会は、1964年、ジュネーブで、第4回総会は、1968年、アルゼンチンで開催された。

CCIFとCCITが合併したのは、有線電気通信の分野、とくに伝送路について電信回線と電話回線とを技術的に分ける意味がなくなってきたこと、各国とも大体において、電信部門と電話部門は同一組織内にあること、CCIFの事務局とCCITの事務局の合併による能率増進等がおもな理由であった。

CCITTは、上述のように、ヨーロッパ内の国ぐにによって、ヨーロッパ内の電信・電話の技術・運用・料金の基準を定め、あるいは統一をはかってきたので、現在でも、その影響を受け、会合参加国は、ヨーロッパの国が多く、ヨーロッパで生起する問題の研究が多い。たとえば、1960年のCCITT勧告の中で、技術上配慮する距離は約2,500kmであったが、これはヨーロッパ内領域を想定したものである。

しかしながら、1956年9月に敷設された大西洋横断電話ケーブルは、大陸間電話通信の自動化および半自動化への技術的可能性を与え、CCITTがこの問題を取り上げるに及び、CCITTの性格は漸次、汎世界的色彩を実質的に帯びるに至った。この汎世界的性格は第2次世界大戦後目ざましくなったアジア・アフリカ植民地の独立に伴ってITUの構成員の中にこれらの国が加わり、ITUの中に新しい意見が導入されたことにも起因して、技術面、政治面の双方から導入されてきた。CCITTの汎世界化は、1960年の第2回総会がニューデリーで開催されたことにもあらわれている。この総会までは、CCIT、CCIFのいずれにしろ、アメリカやアジアで総会が開催されたことがなく、CCITT委員長も、ニューデリー総会の準備文書で、この点には注目すべきであるとのべている。

任務

ITUは、全権委員会議、主管庁会議を始めとして、七つの機関をもち、それぞれの機関の権限と任務は国際電気通信条約に明記されている。そこで条約を参照してみるならば、CCITTの任務は、つぎのとおりとなっている。

「国際電信電話諮問委員会(CCITT)は、電信および電話に関する技術、運用および料金の問題について研究し、および意見を表明することを任務とする。」（1965年モントルー条約第187号）

「各国際諮問委員会は、その任務の遂行に当たって、新しい国または発展の途上にある国における地域的および国際的分野にわたる電気通信の創設、発達および改善に直接関連のある問題について研究し、および意見を作成するように妥当な注意を払わなければならない。」（同第188号）

「各国際諮問委員会は、また、関係国の要請に基づき、その国内電気通信の問題について研究し、かつ、勧告を行なうことができる。」（同第189号）

上記第187号と第188号にいわれる「意見」とは、フランス語の Avis から訳したもので、英語では、「勧告(Recommendation)」となっている。CCITTの表明する意見は、国際法的には強制力をもたないものであって、この点が、条約、電信規則、電話規則等各国を拘束する力をもっているものと異なる。もっとも意見とは称しても、技術的分野では、電信規則のごとき、各国政府が承認してその内容を実施する強制規則をもたないので、実際にある機器の仕様を定める場合には、多くの国の意見が統一されたこの「意見」に従わなければ、円滑な国際通信を行なうことができない場合が多い。この意見（または勧告）は、国際通信を行なう場合各国が直面する問題について、具体的意見を表明するもので、たとえば、大陸間ケーブルで大陸間通話を半自動化しようとする場合、その信号方式や取り扱う通話の種類および料金は、どのようにするかを研究して意見を表明する。したがって、CCITTの活動は、つねに時代の最先端を行くもので、CCITTの活動方向は、そのまま世界の国際通信の活動方向であるともいえる。

この意見は、また、電信規則以下のその他の規則のごとく、数年以上の間隔をもって開催される主管庁会議というような大会議の決定をまたなくても表明することができ、また、その改正も容易であるので、現在のように進歩の早い国際通信界では、関係国の意見を統一した国際的見解としては非常に便利である。

Figure 2-5 CCITT Test Document No. 7

should be used. Figure 2-6 is a copy of the French PTT Test Document No. 4 scanned with 7.7 lines/mm. resolution. Figure 2-7 is a copy of the same document where the even scan lines have been replaced with the line above. Therefore, this represents a document in which the vertical resolution is 3.85 lines/mm.

2.3 Minimum Scan Line Time (MSLT)

The standard MSLT to be used in the measurement program will be 5, 10, and 20 ms. with EOL-code and 0 ms. without EOL-code. It was later clarified in a memo from the chairman of the Working Committee (see Reference 7) that if, for reasons of test economy, only one value of MSLT can be used in the test program, that value shall be 20 ms.

2.4 Transmission Bit Rate

The standard transmission bit rate is 4800 bits/sec.

2.5 Measurement of Compression

Two standard measures of compression have been established-- (1) number of coded bits (2) Compression Factor. The number of coded bits is the number of bits required to transmit a document, including all overhead bits such as End of Line (EOL) and Fill bits. The Compression Factor is computed by dividing the total number of picture elements (pels) per test document by the number of coded bits. It was further agreed that the Compression Factor and coded bits should be computed for two different conditions--with overhead and without overhead. The measurement with overhead applies to the

L'ordre de lancement et de réalisation des applications fait l'objet de décisions au plus haut niveau de la Direction Générale des Télécommunications. Il n'est certes pas question de construire ce système intégré "en bloc" mais bien au contraire de procéder par étapes, par paliers successifs. Certaines applications, dont la rentabilité ne pourra être assurée, ne seront pas entreprises. Actuellement, sur trente applications qui ont pu être globalement définies, six en sont au stade de l'exploitation, six autres se sont vu donner la priorité pour leur réalisation.
Chaque application est confiée à un "chef de projet", responsable successivement de sa conception, de son analyse-programmation et de sa mise en oeuvre dans une région-pilote. La généralisation ultérieure de l'application réalisée dans cette région-pilote dépend des résultats obtenus et fait l'objet d'une décision de la Direction Générale. Néanmoins, le chef de projet doit dès le départ considérer que son activité a une vocation nationale donc refuser tout particularisme régional. Il est aidé d'une équipe d'analystes-programmeurs et entouré d'un "groupe de conception" chargé de rédiger le document de "définition des objectifs globaux" puis le "cahier des charges" de l'application, qui sont adressés pour avis à tous les services utilisateurs potentiels et aux chefs de projet des autres applications. Le groupe de conception comprend 6 à 10 personnes représentant les services les plus divers concernés par le projet, et comporte obligatoirement un bon analyste attaché à l'application.

II - L'IMPLANTATION GEOGRAPHIQUE D'UN RESEAU INFORMATIQUE PERFORMANT

L'organisation de l'entreprise française des télécommunications repose sur l'existence de 20 régions. Des calculateurs ont été implantés dans le passé au moins dans toutes les plus importantes. On trouve ainsi des machines Bull Gamma 30 à Lyon et Marseille, des GE 425 à Lille, Bordeaux, Toulouse et Montpellier, un GE 437 à Massy, enfin quelques machines Bull 300 TI à programmes câblés étaient récemment ou sont encore en service dans les régions de Nancy, Nantes, Limoges, Poitiers et Rouen ; ce parc est essentiellement utilisé pour la comptabilité téléphonique.
A l'avenir, si la plupart des fichiers nécessaires aux applications décrites plus haut peuvent être gérés en temps différé, un certain nombre d'entre eux devront nécessairement être accessibles, voire mis à jour en temps réel : parmi ces derniers le fichier commercial des abonnés, le fichier des renseignements, le fichier des circuits, le fichier technique des abonnés contiendront des quantités considérables d'informations.
Le volume total de caractères à gérer en phase finale sur un ordinateur ayant en charge quelques 500 000 abonnés a été estimé à un milliard de caractères au moins. Au moins le tiers des données seront concernées par des traitements en temps réel.
Aucun des calculateurs énumérés plus haut ne permettait d'envisager de tels traitements.
L'intégration progressive de toutes les applications suppose la création d'un support commun pour toutes les informations, une véritable "Banque de données", répartie sur des moyens de traitement nationaux et régionaux, et qui devra rester alimentée, mise à jour en permanence, à partir de la base de l'entreprise, c'est-à-dire les chantiers, les magasins, les guichets des services d'abonnement, les services de personnel etc.
L'étude des différents fichiers à constituer a donc permis de définir les principales caractéristiques du réseau d'ordinateurs nouveaux à mettre en place pour aborder la réalisation du système informatif. L'obligation de faire appel à des ordinateurs de troisième génération, très puissants et dotés de volumineuses mémoires de masse, a conduit à en réduire substantiellement le nombre.
L'implantation de sept centres de calcul interrégionaux constituera un compromis entre : d'une part le désir de réduire le coût économique de l'ensemble, de faciliter la coordination des équipes d'informaticiens; et d'autre part le refus de créer des centres trop importants difficiles à gérer et à diriger, et posant des problèmes délicats de sécurité. Le regroupement des traitements relatifs à plusieurs régions sur chacun de ces sept centres permettra de leur donner une taille relativement homogène. Chaque centre "gèrera" environ un million d'abonnés à la fin du VIème Plan.
La mise en place de ces centres a débuté au début de l'année 1971 : un ordinateur IRIS 50 de la Compagnie Internationale pour l'Informatique a été installé à Toulouse en février ; la même machine vient d'être mise en service au centre de calcul interrégional de Bordeaux.

Figure 2-6 Test Document Scanned/Printed 7.7 lines/mm.

Photo n° 1 - Document très dense lettre 1,5mm de haut -

Restitution photo n° 9

L'ordre de lancement et de réalisation des applications fait l'objet de décisions au plus haut niveau de la Direction Générale des Télécommunications. Il n'est certes pas question de construire ce système intégré "en bloc" mais bien au contraire de procéder par étapes, par paliers successifs. Certaines applications, dont la rentabilité ne pourra être assurée, ne seront pas entreprises. Actuellement, sur trente applications qui ont pu être globalement définies, six en sont au stade de l'exploitation, six autres se sont vu donner la priorité pour leur réalisation.
Chaque application est confiée à un "chef de projet", responsable successivement de sa conception, de son analyse-programmation et de sa mise en oeuvre dans une région-pilote. La généralisation ultérieure de l'application réalisée dans cette région-pilote dépend des résultats obtenus et fait l'objet d'une décision de la Direction Générale. Néanmoins, le chef de projet doit dès le départ considérer que son activité a une vocation nationale donc refuser tout particularisme régional. Il est aidé d'une équipe d'analystes-programmeurs et entouré d'un "groupe de conception" chargé de rédiger le document de "définition des objectifs globaux" puis le "cahier des charges" de l'application, qui sont adressés pour avis à tous les services utilisateurs potentiels et aux chefs de projet des autres applications. Le groupe de conception comprend 6 à 10 personnes représentant les services les plus divers concernés par le projet,et comporte obligatoirement un bon analyste attaché à l'application.

II - L'IMPLANTATION GEOGRAPHIQUE D'UN RESEAU INFORMATIQUE PERFORMANT

L'organisation de l'entreprise française des télécommunications repose sur l'existence de 20 régions. Des calculateurs ont été implantés dans le passé au moins dans toutes les plus importantes. On trouve ainsi des machines Bull Gamma 30 à Lyon et Marseille, des GE 425 à Lille, Bordeaux, Toulouse et Montpellier, un GE 437 à Massy, enfin quelques machines Bull 300 TI à programmes câblés étaient récemment ou sont encore en service dans les régions de Nancy, Nantes, Limoges, Poitiers et Rouen ; ce parc est essentiellement utilisé pour la comptabilité téléphonique.
A l'avenir, si la plupart des fichiers nécessaires aux applications décrites plus haut peuvent être gérés en temps différé, un certain nombre d'entre eux devront nécessairement être accessibles, voire mis à jour en temps réel : parmi ces derniers le fichier commercial des abonnés, le fichier des renseignements, le fichier des circuits, le fichier technique des abonnés contiendront des quantités considérables d'informations.
Le volume total de caractères à gérer en phase finale sur un ordinateur ayant en charge quelques 500 000 abonnés a été estimé à un milliard de caractères au moins. Au moins le tiers des données seront concernées par des traitements en temps réel.
Aucun des calculateurs énumérés plus haut ne permettait d'envisager de tels traitements.
L'intégration progressive de toutes les applications suppose la création d'un support commun pour toutes les informations, une véritable "Banque de données", répartie sur des moyens de traitement nationaux et régionaux, et qui devra rester alimentée, mise à jour en permanence, à partir de la base de l'entreprise, c'est-à-dire les chantiers, les magasins, les guichets des services d'abonnement, les services de personnel etc.
L'étude des différents fichiers à constituer a donc permis de définir les principales caractéristiques du réseau d'ordinateurs nouveaux à mettre en place pour aborder la réalisation du système informatif. L'obligation de faire appel à des ordinateurs de troisième génération, très puissants et dotés de volumineuses mémoires de masse, a conduit à en réduire substantiellement le nombre.
L'implantation de sept centres de calcul interrégionaux constituera un compromis entre : d'une part le désir de réduire le coût économique de l'ensemble, de faciliter la coordination des équipes d'informaticiens; et d'autre part le refus de créer des centres trop importants difficiles à gérer et à diriger,et posant des problèmes délicats de sécurité. Le regroupement des traitements relatifs à plusieurs régions sur chacun de ces sept centres permettra de leur donner une taille relativement homogène. Chaque centre "gèrera" environ un million d'abonnés à la fin du VIème Plan.
La mise en place de ces centres a débuté au début de l'année 1971 : un ordinateur IRIS 50 de la Compagnie Internationale pour l'Informatique a été installé à Toulouse en février ; la même machine vient d'être mise en service au centre de calcul interrégional de Bordeaux.

Figure 2-7 Test Document Scanned 7 lines/mm. Printed 3.85 lines/mm.

Photo n° 1 - Document très dense lettre 1,5mm de haut -

Restitution photo n° 9

Group 3 situation while the measurement without overhead applies to the Group 4 case.

2.6 Measurement of Error Sensitivity

An objective measure of error sensitivity is obtained by encoding the test documents with the proposed techniques (all overhead bits must be included), subjecting the resulting bit stream to transmission errors, decoding the transmission to obtain the received image, and comparing the original image with the received image to determine the number of pels in error. The Error Sensitivity Factor (ESF) is calculated as the total number of document pels in error divided by the total number of transmission bits that are in error. In this way, the ESF represents the average disturbance to the output image caused by a single transmission error.

2.6.1 Transmission Error Pattern

It was agreed that a record of actual bit errors incurred over telephone lines will be used in the error sensitivity test. The Federal Republic of Germany (see Reference 8) has obtained a record of such errors by transmitting a known psuedo-random sequence at 4800 bits/sec. using a V27 ter modem over a switched telephone network. The resultant error pattern has been recorded on magnetic tape and made available to experimenters. Appendix C of Reference 3 describes the format of the transmission error magnetic tape. This tape was used in the measurement of error sensitivity described in this report.

2.6.2 Error Phases

One concern with the ESF measurement is the high degree of sensitivity to those few errors which may affect the end of line code and can cause an inordinate number of incorrect pels. If the error pattern happened to fall in an unfortunate phase relative to the encoded bits, a large number of pels could be affected. On the other hand, the error pattern could fall fortuitously and affect a relatively few number of pels. To insure experimenters can achieve an adequate level of statistical validity, the concept of error phases has been introduced. In the basic zero phase, the first bit of the error record is aligned with the first bit of the encoded transmission. In the case of Phase 2, the transmitted bit information is delayed by 1,024 bits relative to the previous run. The transmission bit information is delayed by 2,048 bits for Phase 2. Experimenters would have a higher confidence level in the average of the three phases compared to any one ESF taken alone.

2.6.3 Error Correction

In order to precisely measure the error sensitivity, both the encoding technique and the decoding algorithm must be completely defined. If more than one decoding algorithm is proposed (for example, to achieve differing levels of error control), each must be tested separately. Collective Letter No. 87 from the CCITT (see Reference 7) outlines an error correction procedure to be used for simulating two-dimensional algorithms where an error correction procedure has not been otherwise specified. In this procedure, the erroneous line is replaced by the previous line and following lines are replaced by white lines until a one-dimensional coding line is correctly decoded.

3.0 COMPUTER PROGRAM OVERVIEW

This section contains a general overview of the computer program architecture used under this contract. The description is divided into two parts. Section 3.1 focuses on the overall simulation process from a flow perspective with particular emphasis on the simulation inputs and outputs. Section 3.2 presents the hierarchical structure of the programs illustrating how the programs are organized for each of the 7 different algorithms. For convenience of the reader, a detailed flow chart, and the actual program code listing, has been included in the Appendices for each algorithm (Appendices D through G). All computer programs have been written in conventional Fortran IV language.

3.1 The Simulation Process

Figure 3-1 illustrates the interrelationship between the major functions of each simulation program developed on the subject contract. There are two input data sets to each simulation which originate on magnetic tape. One tape, supplied by the French PTT Administration, contains all eight of the CCITT test documents. The format of this input image tape is described in Appendix B of Reference 3. The other tape, supplied by the Federal Republic of Germany, contains transmission error data from actual switched telephone circuits. The format of this input tape is described in Appendix C of Reference 3. A program called "REDTAP" was prepared to read the data from the input document tape while the error tape is read in directly. Data from the two input tapes are placed on disc in the computer system to be accessed during the simulation process. A separate file is established for each of the

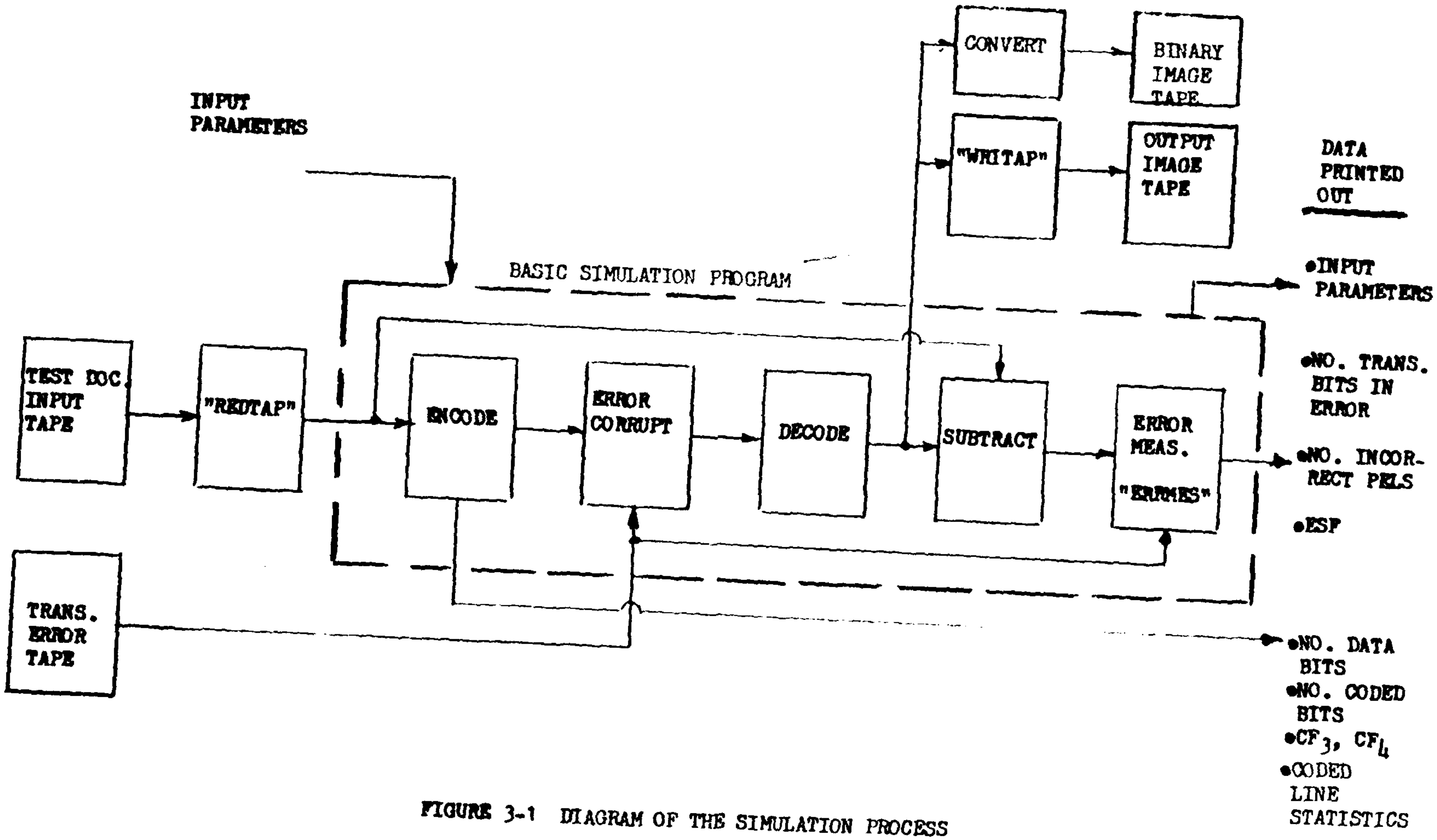


FIGURE 3-1 DIAGRAM OF THE SIMULATION PROCESS

test documents. The transmission error tape is divided into four files, one for each of four different circuit error conditions.

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. A typical interactive sequence with responses is listed below.

1. PARAMETERS: INPUT (=I), OR DEFAULT (=D)? I
2. DIAGNOSTIC PRINTOUT? (Y OR N). N
3. ENTER MAXIMUM NUMBER OF PELS PER LINE: 1728
4. ENTER VERTICAL SAMPLING: 1
5. ENTER PARAMETER K: 4
6. ENTER ERROR PATTERN PHASE: 0
7. ENTER MINIMUM COMPRESSED LINE LENGTH: 96
8. NUMBER OF SCAN LINES TO BE PROCESSED = ? 10
9. ERROR MODE = ? (M=MANUAL, T=TAPE, N=NO ERRORS) N

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 actual code is fed to the error corrupt unit, while the number of code bits is accumulated with fill and EOL codes to provide the output total number of data bits, to compute the Compression Factors, CF_3 and CF_4.

The error corruption step combines the transmission error data with the encoded data. At each point in the image where an error occurs, the corresponding bit in 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. There are two parts of the decoding function which are not obvious and require clarification: (1) what the decoder does when an error occurs (2) what the decoder does when a line is missing. The operation of the decoder under these two conditions is described in Section 4.

The output of the Decode function feeds the "WRITAP" or "CONVERT" functions for writing the error corrupted image on magnetic tape. It is also fed to a subtraction function which compares the decoded image with the original image. Pels which are in error are fed to the "ERRMES" subroutine which counts all the pels in the image which are in error. This subroutine also counts the number of transmission error bits which corrupted the encode signal. Finally, the "ERRMES" subroutine computes the ESF by dividing the number of incorrect pels by the number of transmitted bits in error.

Figure 3-1 shows that the simulation process provides a printout of all the computed performance data as well as a summary tabulation of the input parameters.

For more details on the computer programs, refer to Section 3.2 for a description of the program structure and to the Appendices for flow charts and program listings.

The reader should note that most of the software prepared under this contract is suitable for simulating any compression algorithm. The only subroutines which must be written specifically for a particular coding technique are the encode and decode subroutines.

3.2 Program Structure

The following section describes the structure of the computer program written to simulate the various algorithms. In addition, a brief description of each of the subroutines is given.

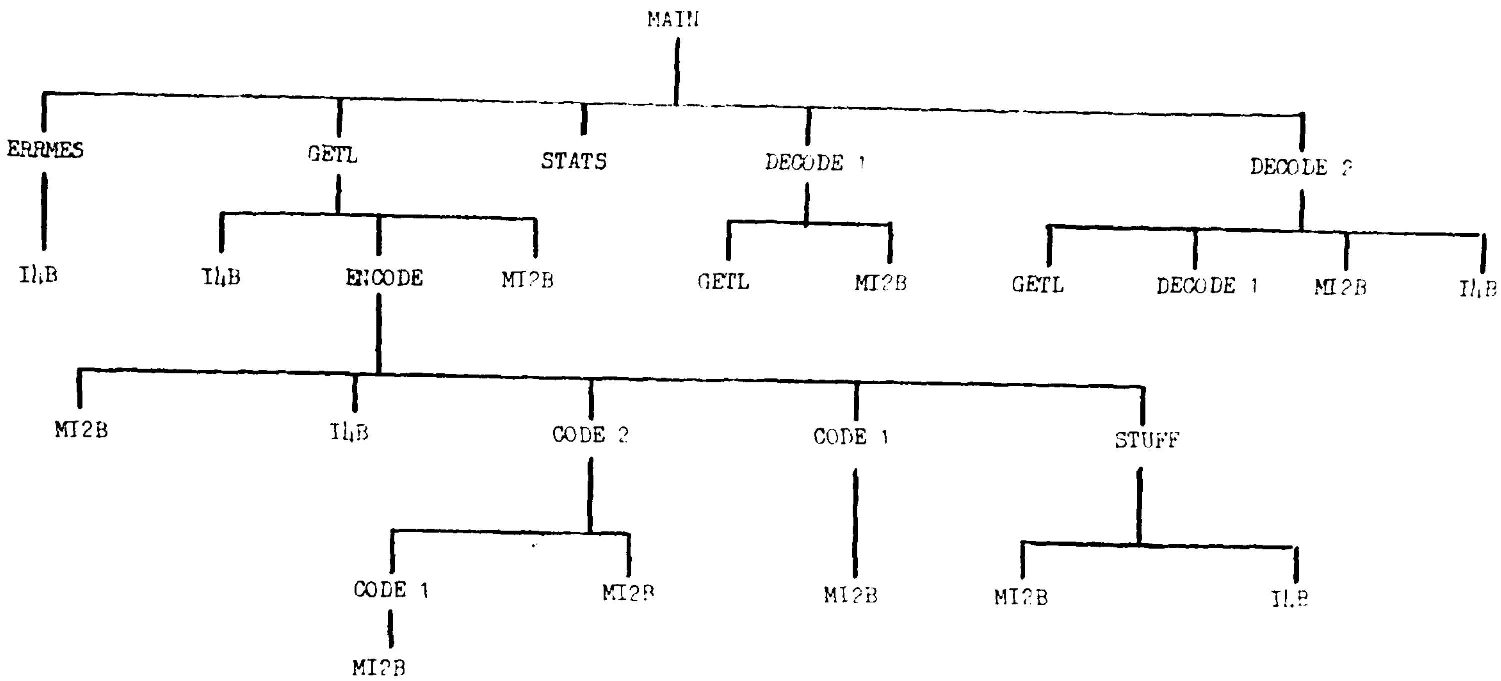
Each of the computer programs written to simulate the seven compression algorithms conforms to the general structure shown in Figure 3-2. The chart given in this figure shows the hierarchy of the functions that make up each simulation program. Some of the functions on the chart are named generically: the table in Figure 3-2 shows how these generic function names are keyed to the actual subroutine names used by each compression algorithm. The names on the hierarchical chart that do not appear in the table are subroutines that are used by all compression algorithms. A brief description of each of the functions/subroutines follows:

MAIN

The MAIN program controls the decoding process and the error recovery procedure for getting back in sync when an error is detected. As can be seen from Figure 3-2, 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 MAIN program also controls parameter input, measurement of errors, and reports computed results.

GETL

The GETL subroutine retrieves a number of requested bits from



FUNCTION	SUBROUTINE NAMES					
	READ	3M	IBM	XEROX	AT&T	FRG	BRG
MAIN	JPREAD	THREEM	IBM	XEROX	BTL	GERMAN	ENGLISH
GETL	GETL	GETL 3	GETLI	GETLX	GETLB	GETLG	GETLE
ENCODE	ENCODR	ENCOD3	ENCODI	ENCODX	ENCODB	ENCODG	ENCDE
CODE 1	CODELN	CODELN	CODEIN	XCODLR	CODELN	GCODLR	CODELN
CODE 2	CODEVH	CODE3M	CODIBM	CODEX	CODBTL	CODEG	CODENG
DECODE 1	ONEDIM	ONED3	ONEIBM	ONEXOX	ONEBTL	ONEGER	ONEENG
DECODE 2	TWODIM	TWOD3	TWOIBM	TWOXOX	TWOBTL	TWOGER	TWOENG

FIGURE 3-2 SUBROUTINE HIERARCHY

the coded line and delivers the bits packed into a word (right justified). If stuffing bits have been used, i.e. in the German code, they are removed. End-of-line codes (EOL) or line synchronization signals (LSS) are detected. If the number of coded bits requested by the calling program is not available, the ENCODE subroutine is called to provide them.

ENCODE

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 CODE 1 and/or CODE 2.

CODE 1

The subroutine CODE 1 is called by ENCODE 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.

CODE 2

The subroutine CODE 2 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.

STUFF

The STUFF subroutine is used by the READ and German algorithms to insert 0's or 1's in the coded data stream in order to avoid ambiguities with the line synchronization signal. A '1' is inserted after every occurence of ten consecutive zeroes in the coded steam for the German algorithm.

DECODE 1

The DECODE 1 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. Any errors detected in the decoding process, such as no match to code table, or line too long, are flagged.

DECODE 2

This subroutine performs the same function as DECODE 1 for the two-dimensional line.

MI2B and I4B

The subprograms MI2B and I4B are used to pack and unpack a set of bits into (or from) an array of words.

4.0 Error Detection/Correction Procedure

In Reference 7.0, the following error checking and processing procedure was specified by the CCITT for testing the proposed two-dimensional coding techniques:

1) Error checking - If decoded signals are not exactly 1728 pels/line, the line is recognized as an erroneous line.

2) Error processing - The erroneous line is replaced by the previous line and following lines are replaced by white lines until one-dimensional coding line is correctly decoded.

The error detection and correction procedures used in this simulation follow the spirit, if not the letter, of this directive.

Not all of the proposed algorithms produce a line pel count that can be checked against the correct 1728 pels per line. The error checking was expanded to include the detection of any condition that could not possibly occur in a correctly received transmission. Some examples of possible error conditions are:

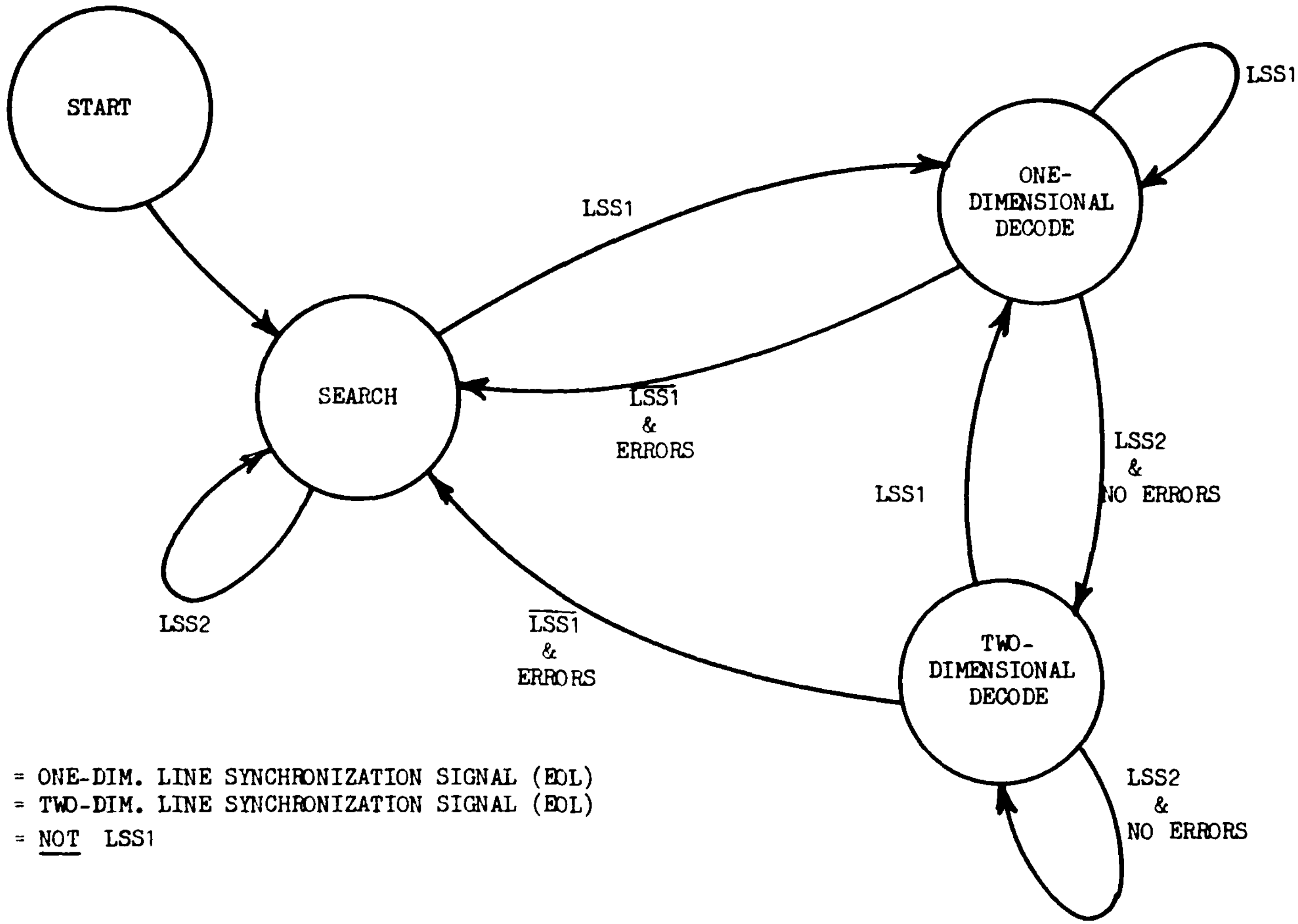
- EOL occurs before 1728 pels have been written
- More than 1728 pels have been written before EOL is received
- No word in applicable code table matches received bit pattern
- Current line decoding references a run that does not exist in the previous line
- Pels are written to the left of the first pel on the line

Conditions that are only improbable, such as a line of pels that differs radically from the previous line, are not considered error conditions. Error conditions specific to each coding algorithm are discussed in Section 5.0.

The AT&T algorithm does not, strictly speaking, have a "one-dimensional coding line." Therefore, the error processing was extended, for this algorithm, to consider any line that can be decoded without an error condition as a correct line. In decoding lines that reference previous lines, the last correctly decoded line is used as the reference line, regardless of whether or not there are intervening error lines. It is believed that the chance of correctly decoding a line, following an error line that references a previous line, is extremely small.

Upon detection of an error condition, the decoder attempts to resynchronize by searching for the next unique Line Synchronization Signal (LSS). All but the AT&T algorithm have different codes for one-dimensional and two-dimensional lines. The state diagram for error recovery for these algorithms is shown in Figure 4-1. For the AT&T algorithm, the One-Dimensional Decode and the Two-Dimensional Decode states are identical, and detection of an EOL in the Search state causes a change to the Decode state, rather than staying in Search.

Following Reference 7, when an error condition is detected, the error line is replaced by the previous correct line, while successive error lines are replaced by all-white lines, until a line is decoded correctly. It should be pointed out that this procedure



LSS1 = ONE-DIM. LINE SYNCHRONIZATION SIGNAL (EOL)
LSS2 = TWO-DIM. LINE SYNCHRONIZATION SIGNAL (EOL)
$\overline{LSS1}$ = NOT LSS1

Figure 4-1 DECODE STATE DIAGRAM

may not be optimum. Repeating the last correct line until the next correct line is received may produce better results from a subjective and objective point of view.

Because of transmission errors, some of the original image lines may be missing in the output, or additional lines may be in the output that were not in the original image. In order that a missing or extra line not have an undue influence on the ESF, it is important that the original and received images not get permanently out of line alignment when they are compared to determine the number of pel errors. To this end, each of the lines in the original image is assigned a serial line number, and this number continues to be associated with the same line in the received image. If a transmitted line is dropped, due to the loss of an EOL, then its line number will be missing in the output. On the other hand, if a line is broken into two or more lines in the received image, due to false EOL's, then its line number will appear more than once in the output.

If no lines are dropped or added, the line numbers of the original and received lines that are compared to detect pel errors will be equal. When a line is added or deleted, the line numbers of the compared lines will become unequal. When this occurs for the first time, the two lines with different line numbers are compared to determine the number of pel errors, which is added to the pel error total. Then, instead of proceeding to the next line in both the original and received images, the next line is used in only one of the images, with the previous line being used in the other image. The line is advanced only in that image that has the smaller line number, so as to tend to make the line numbers of the two images more equal. This continues until the line numbers are equal, after which the next line is used in both images, until another inequality is detected.

This procedure provides a proper penalty for a missing or added line, but prevents this type of error from causing pel errors over the entire image below the place where it occurred.

5.0 ASSUMPTIONS RELATED TO INDIVIDUAL ALGORITHMS

5.1 British Post Office

No modifications or assumptions were required to simulate the British algorithm. Two optional procedures, as defined in the British contribution, were simulated to enable best performance: The resettable K procedure was used to reset K when an all-white line followed a non-all-white line or a non-all-white line followed an all-white line. The optional step 3 in the two-dimensional coding procedure was used in place of step 2ii.

5.2 Federal Republic of Germany

The German compression algorithm encodes run lengths of correct predictions followed by an incorrect prediction. The runs between incorrect predictions are encoded separately for each source state. However, the contribution was not clear on the coding procedure if the last run of predictions on a line for a given source state did not end with a prediction error. Therefore, a hypothetical prediction error was added to each run of correct predictions for a given state if that run did not terminate naturally with a prediction error. This hypothetical error was automatically ignored on decoding. This procedure required a slight modification to the code table for state 0 (S_0). The code word 110 was used for a run length of 1729 instead of 1728 and the code word 10 for the prefix was used for lengths of 65-1728 instead of 65-1727.

Note that it is not possible with the German algorithm to detect errors by checking the decoded line length, since it is always 1728. Errors were detected by checking the residue of run lengths for each state after a complete line was decoded. For an error-free line, the residue must be 0 or 1 corresponding to runs ending with a prediction error and runs ending with a hypothetical prediction error, respectively. Any runs of length greater than 1 "left over" after a line is decoded indicate an error. Taking the above approach, it should be noted that in the example of Figure 2 of Appendix B, the S_0 run should be 6 instead of 5, since the first prediction error for state S_0 occurs the sixth time state zero is present.

6.0 MEASUREMENT RESULTS

During the course of this contract, Delta Information Systems prepared computer programs to simulate the two-dimensional coding algorithms proposed by the British Post Office and the Federal Republic of Germany. These two programs were then run on the Hybrid Computer Facility at the Defense Communications Engineering Center in Reston, Virginia. Two different types of simulation were performed. The first measured the compression and error sensitivity of the two algorithms at five different test conditions (the four CCITT documents at standard resolution plus document number 4 at high resolution). In the second simulation, all seven proposed algorithms (Japan, 3M, IBM, Xerox, AT&T, BPO, FGR) were tested at an infinite K-factor for document number 4 at standard and high resolution. The results of these two simulation tests are summarized in the paragraphs below.

COMPRESSION AND ERROR SENSITIVITY

As explained above, five computer runs were performed for both the British and German algorithms. The following test conditions were held constant during these tests: error phase - 0; error file - 1; minimum scan line time - 20 ms. All four test documents (Documents 1, 4, 5, and 7) were run at standard resolution and a K-factor of 2. For the fifth run, test document number 4 was run at high resolution with a K-factor of 4.

The results of the ten test runs are tabulated in Tables 6-1 and 6-2. To aid in the evaluation process, the corresponding test data for the other five algorithms are also included in these tables. The definitions of measurement parameters included in these tables are reviewed below.

TABLE 6-1 COMPRESSION AND ERROR SENSITIVITY TEST RESULTS*

DOC. NO. VERT. RESOL. K FACTOR	ALGORITHM	NO. CODED BITS	NO. BITS IN ERROR XMTD	BER X 10^{-3}	NO. INCORRECT PELS	NO. CODED DATA BITS	ESF	CF_3	CF_4
	JAPAN	442,434	362	.82	21,030	390,927	58.093	4.6399	5.2513
	3M	441,104	362	.82	12,255	397,549	33.8536	4.6539	5.1638
DOC. NO. 4	IBM	430,215	346	.80	16,013	383,562	46.2803	4.7717	5.3521
	XEROX	468,341	374	.798	15,642	430,660	41.8235	4.3833	4.7668
STANDARD	AT & T	466,613	374	.80	19,378	415,034	51.8128	4.3995	4.9463
RESOL.	BPO	442,129	362	.819	15,250	395,132	42.1271	4.6431	5.1954
K = 2	FRG	430,335	346	.804	15,674	385,149	45.3006	4.7704	5.3301
	JAPAN	727,418	564	.775	38,283	620,671	67.877	5.6442	6.6150
	3M	757,869	564	.74	38,682	668,555	68.5851	5.4175	6.1412
DOC. NO. 4	IBM	727,740	564	.77	30,600	627,122	54.2553	5.6418	6.5469
	XEROX	822,790	564	.685	25,464	748,406	45.1489	4.9900	5.4860
HIGH	AT & T	763,481	564	.73	33,756	655,807	59.8511	5.3776	6.2606
RESOL.	BPO	731,769	564	.771	39,365	628,963	69.7961	5.6107	6.5278
K = 4	FRG	727,121	564	.776	33,293	631,072	59.0301	5.6466	6.5060
	JAPAN	188,070	120	.638	3,538	113,956	29.48	10.915	18.0145
	3M	192,484	132	.68	1,160	126,122	8.7879	10.6651	16.2768
DOC. NO. 1	IBM	187,619	120	.63	3,034	115,011	25.2833	10.9417	17.8493
	XEROX	198,749	132	.664	2,571	133,050	19.4773	10.3289	15.4293
STANDARD	AT & T	193,573	132	.68	1,236	112,546	9.3636	10.6051	18.2402
RESOL.	BPO	189,285	120	.634	3,091	115,540	25.7583	10.8454	17.7675
K = 2	FRG	189,938	120	.632	3,056	118,809	25.4667	10.8081	17.2787

* ERROR PHASE - 0; ERROR FILE - 1; MIN. SCAN LINE TIME - 20 ms.

TABLE 6-2 COMPRESSION AND ERROR SENSITIVITY TEST RESULTS (cont'd)

DOC.NO. & VERT.RESOL.	ALGORITHM	NO.CODED BITS	NO.BITS IN ERROR XMTD	BER X 10^{-3}	NO. INCORRECT PELS	NO. CODED DATA BITS	ESF	CF_3	CF_4
DOC.NO. 5 STANDARD RESOL. K = 2	JAPAN	253,989	216	.850	7,549	210,040	34.94	8.082	9.7737
	3M	264,163	216	.81	7,386	226,815	34.1944	7.7712	9.0508
	IBM	254,459	216	.84	8,211	210,809	38.0139	8.0676	9.7380
	XEROX	269,544	220	.816	3,041	236,284	13.8227	7.6161	8.6881
	AT & T	267,503	220	.82	5,570	220,429	25.3182	7.6742	9.3130
	BPO	255,470	216	.845	8,483	210,971	39.2731	8.0356	9.7305
	FRG	258,815	216	.834	4,332	220,118	20.0555	7.9318	9.3262
DOC.NO. 7 STANDARD RESOL. K = 2	JAPAN	423,040	290	.685	9,361	385,871	32.27	4.852	5.320
	3M	431,481	356	.82	8,485	399,497	23.8343	4.7577	5.1386
	IBM	413,042	272	.65	6,056	379,460	22.2647	4.9701	5.4100
	XEROX	448,809	362	.807	9,017	421,857	24.9088	4.5740	4.8663
	AT & T	451,171	362	.80	9,463	415,929	26.1409	4.5501	4.9356
	BPO	422,007	290	.687	7,673	388,535	26.4586	4.8645	5.2836
	FRG	422,096	290	.687	8,366	389,068	28.8483	4.8635	5.2764

- Coded Data Bits - Total compressed bits required to transmit the document excluding all overhead bits - EOL, fill, etc.
- Coded Bits - Total compressed bits required to transmit the document including all overhead such as EOL, fill, etc.
- CF_4 - Number of document pels* divided by the number of coded data bits
- CF_3 - Number of document pels* divided by the number of coded bits
- BER - Transmitted bits in error divided by the number of coded bits
- ESF - Number of incorrect pels divided by the number of transmitted bits in error

CODED LINE LENGTH STATISTICS

The CCITT suggested that experimenters should measure the statistics related to the number of bits required to define the individual scan lines. Statistics which were measured are minimum bits/line, maximum bits/line, average bits/line, and standard deviation. Statistics were measured for each of the two algorithms and for each of the five test conditions. Table 6-3 is a tabulation of the test results for a minimum scan line time of 20 ms.

INFINITE K-FACTOR TEST RESULTS

The primary objective of this overall measurement program is to contribute to the selection of a standard two-dimensional coding technique for the Group 3 application. For this reason, attention has been focused

*High Resolution - 2,376 lines X 1728 pels/line = 4,105,728 pels
Standard Resolution - 1,188 lines X 1728 pels/line = 2,052,864 pels

TABLE 6-3 CODED LINE LENGTH STATISTICS*

TEST DOCUMENT NO.	VERTICAL RESOLUTION	BPO				FRG			
		MINIMUM BITS/LINE	MAXIMUM BITS/LINE	AVERAGE BITS/LINE	STANDARD DEVIATION	MINIMUM BITS/LINE	MAXIMUM BITS/LINE	AVERAGE BITS/LINE	STANDARD DEVIATION
4	3.85/pm	96	1231	372.10	335.72	96	1089	362.10	321.32
4	7.7/pm	96	1072	307.95	272.77	96	1089	305.88	264.81
1	3.85/pm	96	797	159.27	149.69	96	797	159.80	146.09
5	3.85/pm	96	1045	214.98	183.33	96	1063	217.76	181.41
7	3.85/pm	96	718	355.16	178.37	96	718	355.18	176.32

* MINIMUM SCAN LINE TIME - 20 ms.

on a low K-factor to permit satisfactory operation over noisy transmission channels. It is also anticipated that two-dimensional coding techniques will be employed in the future Group 4 situation where the communication error rate will be very low. In fact, the compression parameter CF_4 was chosen to give some indication of performance in a Group 4 application. However, if the test results are to be truly representative for Group 4 operation, the K-factor should be increased. To provide data for this application, all seven candidate algorithms were tested for an infinite K-factor. Each algorithm was tested at both the standard and high resolution case. Table 6-4 is a tabulation of the test results.

The reader will note that four of the coding techniques (Japan, IBM, Xerox, FRG) exhibit a very large error sensitivity factor, while it is much lower for the others. All those algorithms exhibiting a large ESF cause the input image to turn all white when the first error occurs and it remains so to the bottom of the page. The other three techniques have some degree of automatic self correction for transmission errors. As a result the error sensitivity for these three algorithms is reduced.

NOMENCLATURE OF PRINTED ERROR-CONTAMINATED IMAGES

Independent of this contract, the National Communication System is printing the error-contaminated images which were simulated and listed in Tables 6-1, 6-2, and 6-4. Each of these printed images is labelled in accordance with a particular nomenclature. Table 6-5 is a list of the test parameters and corresponding image nomenclature for the FRG and BPO algorithms. This table is included to assist those readers who may wish to correlate the test results included herein with the NCS images.

TABLE 6-4 TEST RESULTS FOR INFINITE K-FACTOR*

VERT. RESOLUTION	ALGORITHM	NO. CODED BITS	NO. BITS IN ERROR XMTD	BER X 10^{-3}	NO. INCORRECT PELS	NO. CODED DATA BITS	ESF	CF_3	CF_4
STANDARD RESOL. 3.85 /pm	JAPAN	421,115	290	.589	249,247	363,284	859.47	4.8748	5.6509
	3M	425,179	290	.682	16,652	381,510	57.42	4.8282	5.3809
	IBM	399,045	220	.551	245,062	349,188	1,113.9	5.1444	5.8790
	XEROX								
	AT & T	402,686	238	.591	31,493	350,103	132.3	5.0979	5.8636
	BPO	416,057	272	.654	35,666	365,761	131.1	4.9341	5.6126
	FRG	399,140	220	.551	245,092	352,379	1,114.05	5.1432	5.8257
HIGH RESOL. 7.7 /pm	JAPAN	663,182	564	.850	504,457	550,527	894.4	6.1910	7.4578
	3M	703,756	564	.801	96,869	613,946	171.75	5.8340	6.6874
	IBM	664,554	564	.848	501,443	569,271	889.1	6.1782	7.3412
	XEROX								
	AT & T	666,296	564	.846	99,838	556,114	177.0	6.1620	7.3829
	BPO	661,948	564	.852	103,623	554,167	183.7	6.2025	7.4088
	FRG	663,011	564	.851	501,407	563,965	889.02	6.1925	7.2801

*DOCUMENT NO. - 4; ERROR PHASE - 0; ERROR FILE - 1; MIN. SCAN LINE TIME - 20 ms.

Table 6-5 Nomenclature of Printed Error Contaminated Images

Image Nomenclature *	CCITT Document Number	K-Factor	Vertical Resolution
188A	1	2	3.58
488A	4	2	3.58
488B	4	4	7.7
588B	5	2	3.58
788A	7	2	3.58
4881	4	infinite	3.58
4882	4	infinite	7.7

*The nomenclature has a BPO prefix for the British Post Office algorithm and a GRR prefix for the Federal Republic of Germany algorithm.

7.0 REFERENCES

1. CCITT Contribution No. 66, "Criteria for the Evaluation of Two-Dimensional Coding Techniques for use in Digital Facsimile Terminals" Source: United States of America; Date: January 1979.
2. CCITT Contribution COM XIV - No. 70, "Report of the Meeting Held in Geneva," 11-15 Dec. 1978, Annex No. 2, Section III.
3. National Communications System Report, "Development of a Computer Program for Measuring the Compression and Error Sensitivity of Facsimile Coding Techniques," August 10, 1979.
4. CCITT Contribution COM XIV - No. 42, Japan Algorithm.
5. CCITT Contribution COM XIV - No. 74, 3M Algorithm.
6. National Communications System Report, "Measurement of Compression Factor and Error Sensitivity Factor of Five Selected Two-Dimensional Facsimile Coding Techniques," October 1979.
7. Collective Letter No. 87 from the CCITT to Members of Study Group XIV COM/TO dated 21 May 1979, page 5, section 4.0.
8. Federal Republic of Germany, "Sensibility of Redundancy Reducing Codes to Transmission Bit Errors," CCITT Study Group XIV - Contribution No. 5, February 1977.
9. CCITT Contribution COM XIV - No. 64, IBM Algorithm.
10. CCITT Contribution COM XIV - No. 84, XEROX Algorithm.
11. CCITT Contribution COM XIV - No. 81, AT&T Algorithm.

APPENDIX A

CCITT STUDY GROUP XIV

Contribution No. 77

Source: British Post Office

International Telegraph and Telephone
Consultative Committee
(CCITT)

Period 1977-1980

COM XIV-No. 77-E

Original : English

Question : 2/XIV

Date : March 1979

STUDY GROUP XIV - CONTRIBUTION No. 77

SOURCE : BRITISH POST OFFICE

TITLE : PROPOSAL FOR OPTIONAL TWO-DIMENSIONAL CODING SCHEME FOR GROUP 3 FACSIMILE APPARATUS

1. Introduction

In Draft Recommendation T.4 (COM XIV, No 25, Annex 3, Dec 1977) which refers to Group 3 facsimile apparatus, paragraph 4.2 notes that the one-dimensional coding scheme may be extended as an option to a two-dimension coding scheme. This contribution proposes such a two-dimensional coding scheme called the R2 code, which is based upon the one-dimensional coding scheme given in Draft Recommendation T.4.

The R2 code uses a similar coding procedure to that of the READ code proposed by Japan (COM XIV, No 42, Nov 1978) but uses a different code table. Compared with the READ code, the R2 provides higher compression factors, is easier to implement and is expected to have a better performance in the presence of transmission errors.

2. Design of the R2 code

One of the most important factors concerning the choice of a 2-dimensional coding scheme is its sensibility to transmission errors. The one-dimensional coding scheme using a modified Huffman code includes a unique end-of-line (EOL) codeword '000000000001'. This codeword contains a number of redundant bits which ensures that this sequence of digits cannot occur naturally in the coded data stream. Therefore, an error occurring in a coded scan line cannot prevent the detection of the EOL codeword associated with that scan line. This restricts the damage caused by an error to a single line. Also, an error which corrupts one or more digits of the EOL codeword itself may not necessarily prevent that EOL from being detected. This protection is achieved by decoding '0000000' as the end of a scan line. The subsequent coded scan line is then deemed to begin immediately following the next '1' in the data stream. For machines accommodating large paper widths and having upto 2560 picture elements per line, the end of a scan line is recognized by detecting '00000000'.

The R2 two-dimensional coding scheme is designed to provide the same protection against the effects of errors. This is achieved by constructing the R2 code table so that it contains the codeword '0000000'. The remaining codewords are then added by considering the statistics of the various coding elements or modes. The complete table has the prefix property and is exhaustive (ie it is a Huffman code). Redundant bits are then added to the codeword '0000000' to form the required EOL codeword 11 x '0' + '1'. (A similar method was used in the design of the modified Huffman code tables specified for Group 3 machines). The R2 code table and corresponding code tree for the R2 code are shown in Table 1 and Figure 3 respectively.

There are a number of other differences between the R2 and READ codes. Computer simulation tests on the READ code (Section 4) indicate that the vertical coding elements $V_L(n)$ and $V_R(n)$, where n is greater than 3, occur infrequently compared with the other coding elements. Unlike the READ code, the R2 code uses horizontal mode coding in these cases. Hence the R2 code has a range of vertical mode coding of up to plus or minus 3 picture elements and the R2 code table contains specific codewords to represent the vertical coding elements $V_L(2)$, $V_R(2)$, $V_L(3)$ and $V_R(3)$. The R2 code does not include codewords equivalent to the READ code D(n) codewords.

The flow diagram for the R2 code is similar to that for the READ code except that an extra decision box (is $|a_1b_1| > 3$?) is inserted immediately before the decision box (is $\lfloor a_0a_1 \rfloor > \lfloor a_1a_2 \rfloor$?). The latter decision box is an adaptive coding procedure which ensures that certain changing elements on the coding line are coded by the most effecient means. The decision box is not essential in the R2 coding procedure and is therefore included as an optional procedure in the R2 flow diagram.

Table 1 shows that the EOL codeword is followed by a '1' or a '0' flag bit to indicate whether the next scan line is to be coded by one- or two-dimensional coding respectively. This allows the K parameter to be used in a flexible manner, called 'resettable K', as described in Annex 1, Paragraph 4.2.1b. (Note that the resettable K procedure can also be used with the READ code).

A formal description of the R2 coding procedure is given in the Annex in a format capable of being inserted in Paragraph 4.2 of Draft Recommendation T.4.

3 Comments concerning the R2 and read coding schemes

1 There is no redundancy in the end-of-line codewords LSS1 and LSS2. Thus a transmission error which corrupts any of the digits LSS1 or LSS2 will prevent detection of that end-of-line codeword.

2 The need to add a '0' ("stuffing" bit) after the occurrence of five consecutive '1's in the coded data stream obtained using the READ code increases the complexity of the coding and decoding processes. It also increases the transmission time for documents 1, 4, 5 and 7 by an average of 2.5%. On the other hand, stuffing bits are not required when the R2 code is used, since the end-of-line (EOL) codeword is unique.

3 The R2 coding procedure is simpler than the READ coding procedure since the number of vertical coding elements is limited to seven. Hence, in the R2 code, codewords of the form D(n) are not required and the coder needs to consider only a small number of picture elements on the reference line when coding a changing element on the coding line.

4 Step 3 is an optional step in the R2 code which does not affect the compatibility between machines. This step is an adaptive coding procedure represented by the decision box [is $[a_0a_1] > [a_1b_1]$?] in the flow diagram. By omitting it, the R2 coding procedure is simplified since it is not necessary to code each changing element along the coding line by two different methods. The results show that the compression factors are not changed significantly if this decision box is omitted.

(0581)

5 The existence of an error on a scan line of coded data transmitted using the 1-dimensional modified Huffman coding method can usually be detected since each decoded scan line between successive EOL codewords should consist of 1728 picture elements. This 1728 check can be used on every decoded scan line, whether or not "fill" bits have been transmitted. If an error is detected, then it is optionally possible for the receiver to apply some form of corrective action. For example the receiver may attempt to conceal the error by printing an all white line or the previous scan line.

However, the READ code does not always allow this 1728 check to be used to determine the occurrence of an error even when LSS1 or LSS2 has been correctly decoded. The problem is that "fill" bits can sometimes be erroneously decoded, for example, as a sequence of the codeword V(0), (see Table 3, Com XIV, No 42). In this case, the presence of an error may not be detected and error concealment could not be applied. The R2 coding algorithm avoids this problem by using a code table which will not allow "fill" bits to be decoded as valid data.

4 Results

The READ coding scheme, as described in ref 1, but with the addition of the resettable K procedure (see Annex, Section 4.2.1b), was simulated by computer program. This enabled the number of coded bits, both with and without bit stuffing and the frequencies of the various coding elements to be measured for four of the CCITT reference documents. Corresponding measurements were then obtained for the R2 coding scheme. Two sets of measurements were obtained, one which included the adaptive coding step 3 and one which omitted this step. Note that stuffing bits were not added to the coded data obtained using the R2 code.

The measurements were obtained for minimum line periods of 0, 5, 10 and 20 msecs, which correspond to minimum numbers of bits per line of 0, 24, 48 and 96 bits respectively when transmission takes place at 4.8 kbits/sec. The resettable K procedure used to obtain these results was slightly different to that proposed as an option in the R2 code. For these results, each all white scan line was one-dimensionally coded.

A useful comparison between the two codes can be obtained by considering the compression factors for the four documents measured with a minimum line period of 0 msecs and including the appropriate LSS1/2 or EOL codewords. This shows that the addition of stuffing bits to the READ code increases the number of coded bits required by 2.5% on average (cf Tables 2 and 3). Tables 4 and 5 indicate that the omission of the adaptive coding step 3 has very little effect upon the number of coded bits; the number of coded bits is slightly higher for documents 1 and 4 and slightly lower for the other two documents when step 3 is omitted. When compared to the READ code with bit stuffing, the R2 coding procedure requires, on average, 2.6% fewer coded bits (cf Table 3 with Table 4 or 5).

The frequencies of the coding elements for the READ code are listed in Table 6. Table 7 shows some of the frequencies of the coding elements $V_R(n)$ and $V_L(n)$ for n equal to or greater than 2. This indicates that the number of elements where n is greater than 4 is small relative to the frequencies of other coding elements. For comparison purposes, the frequencies of the coding elements obtained for the R2 code (including step 3) are shown in Table 8. It was found that the omission of step 3 had very little effect upon these statistics.

The results relate to documents recorded on a magnetic tape made available by the French Administration. Subsequent testing of the R2 and other codes will be performed according to the agreed test criteria using a new magnetic tape which has been provided recently by the French Administration.

5. Conclusions

This contribution proposes that, by making a number of changes to the READ coding scheme, the performance of the scheme can be improved. These changes allow higher compression factors to be obtained, simplify the coding and decoding processes and may offer an improved performance in the presence of transmission errors. Further measurements are needed to determine the usefulness of the options, ie step 3 and resettable K, described in the R2 code.

Annex : 1

ANNEX

THE R2 CODING SCHEME

4.2 Two-dimensional coding scheme

The two-dimensional coding scheme is an extension of the one-dimensional coding scheme specified in Paragraph 4.1.

4.2.1 One-dimensional coding

a. Fixed K Parameter

The first scan line is transmitted by one-dimensional coding. Also every Kth line following the first line is transmitted by one-dimensional coding to limit the vertical spread of damage caused by transmission errors. The following K-1 lines are coded by two-dimensional coding.

The transmitter determines which lines are transmitted by one- or two-dimensional coding by adding a single flag bit after the EOL codeword as shown in Paragraph 4.2.2e.

b. Resettable K Parameter

This is an optional procedure which may be used to enable higher compression values to be obtained.

If one of the K-1 lines following the Kth line complies with either of the following conditions, then that line is transmitted by 1-dimensional coding and the value of K is again set equal to the K parameter.

i. A scan line which is not all white but which follows an all white scan line.

ii. An all white scan line which follows a scan line which is not all white.

c. Value of the K Parameter

The value of the K parameter should be set as follows.

Normal resolution standard : K = 2
Higher resolution standard : K = 4

d. One-dimensional coding method

This conforms with the description in Paragraph 4.1.

4.2.2 Two-Dimensional Coding

This is a line-by-line coding method in which the position of each changing picture element on the current or coding line is coded with respect to the position of a corresponding reference element situated on either the coding line or the reference line which lies immediately above the coding line. After the coding line has been coded it becomes the reference line for the next coding line.

a. Definition of changing picture elements

A changing element is defined as an element whose "colour" (ie black or white) is different from that of the previous element along the same scan line.

a_o The reference or starting changing element on the coding line. At the start of the coding line a_o is set on an imaginary white changing element situated just before the first element on the line. During the coding of the coding line, the position of a_o is defined by the previous coding mode (see Paragraph 4.2.2b).

a_1 The next changing element to the right of a_o on the coding line. This is the next element to be coded.

a_2 The next changing element to the right of a_1 on the coding line.

b_1 The first changing element on the reference line to the right of a_o and of opposite colour to a_o.

b_2 The next changing element to the right of b_1 on the reference line.

b. Coding Modes

One of three coding modes are chosen according to the coding procedure described in Paragraph 4.2.2c to code the position of each changing element along the coding line. Examples of the three coding modes are given in Figure 2.

i. Pass mode

This mode is identified when the position of b_2 lies to the left of a_1. If the position of b_2 lies directly above a_1, then this does not constitute a pass mode. When this mode has been coded, a_o is set on the element of the coding line below b_2 in preparation for the next coding.

ii. Vertical mode

When this mode is identified, the position of a_1 is coded relative to the position of b_1. The relative distance a_1b_1 can take on one of seven values V(0), $V_R(1)$, $V_R(2)$, $V_R(3)$, $V_L(1)$, $V_L(2)$ and $V_L(3)$, each of which is represented by a separate codeword. The subscripts R and L indicate that a_1 is to the right or left respectively of b_1 and the number in brackets indicates the value of the distance a_1b_1. After vertical mode coding has occurred, the position of a_o is set on a_1.

iii. Horizontal mode

When this mode is identified, both the runlengths a_1a_0 and a_1a_2 are coded using the codewords H + $M(a_oa_1)$ + $M(a_1a_2)$. H is the flag codeword '011' taken from the 2-dimensional code table. $M(a_oa_1)$ and $M(a_1a_2)$ are codewords which represent the length and "colour" of the runs a_oa_1 and a_1a_2 respectively and are taken from the appropriate white or black modified Huffman code tables. After a horizontal mode coding, the position of a_o is set on a_2.

c. Coding Procedure

The coding procedure identifies the coding mode that is to be used to code each changing element along the coding line. An adaptive procedure may be used in some cases to determine which coding mode will provide the most efficient coding. When one of the three coding modes has been identified, an appropriate codeword is selected from the code table given in Table 1. The coding procedure is formally defined by the flow diagram given in Figure 1.

Step 1

i. If a pass mode is identified this is coded using the codeword '0001' (Table 1). Return to the start of the coding procedure.

ii. If a pass mode is not detected then proceed to Step 2.

Step 2

Determine the absolute value of the relative distance a_1b_1

i. If $|a_1b_1| > 3$ then transmit the distances a_0a_1 and a_1a_2 by horizontal mode coding (Paragraph 4.2.2b). Return to the start of the coding procedure.

ii. If $|a_1b_1| \leq 3$ then transmit the relative distance a_1b_1 by vertical mode coding (Paragraph 4.2.2b). Return to the start of the coding procedure.

Step 3

This is an adaptive coding procedure which ensures that the most efficient coding mode is used to code the position of a_1. This optional step replaces Step 2 ii).

If $|a_1b_1| \leq 3$ then determine the value of $[a_1b_1]$, ie the number of bits required to code the relative distance a_1b_1 by vertical mode coding. Also, determine $[a_0a_1]$, the number of bits required to code the distance a_0a_1 by horizontal mode coding. This is equal to $H + M(a_0a_1)$, where H is the flag codeword '011' and $M(a_0a_1)$ is the codeword taken from the appropriate modified Huffman code table and represents the "colour" and run-length value of a_0a_1.

Case 1: $[a_0a_1] > [a_1b_1]$

Code a_1b_1 by vertical mode coding.

Case 2: $[a_0a_1] < [a_1b_1]$

Code both the distances a_0a_1 and a_1a_2 by horizontal mode coding.

The use of this optional step does not affect interworking between Group 3 facsimile machines.

d. Coding the first and last picture elements on a line

i. The first run length on a line a_0a_1 is replaced by a_0a_1-1. Therefore, if the first run is black and is deemed to be coded by horizontal mode coding, then the first codeword $M(a_0a_1)$ corresponds to a white run of zero length.

ii. The coding of the coding line continues until the position of the imaginary changing element situated just after the last actual element has been coded. This may be coded as a_1 or a_2. Also, if b_1 and/or b_2 are not detected at any time during the coding of the line, they positioned on the imaginary changing element situated just after the last actual picture element on the reference line.

e. Line synchronization codeword

To the end of every coded line is added the end-of-line (EOL) codeword '000000000001'. The EOL codeword is followed by a single flag bit which indicates whether one- or two-dimensional coding is used for the next line.

The flag bit is:-

1 : one-dimensional coding of next line
0 : two-dimensional coding of next line

f. Fill bits

Fill bits, consisting of variable length strings of '0's may be inserted before the EOL codeword as specified in Paragraph 4.1c.

g. Return to control

The format used is the same as specified in Paragraph 4.1d.

TABLE 1

The R2 code table

MODE	ELEMENTS TO BE CODED		NOTATION	CODEWORD
PASS	b_1, b_2		P	0001
HORIZONTAL	a_0a_1, a_1a_2		H	$011 + M(a_0a_1) + M(a_1a_2)$
VERTICAL	a_1 JUST UNDER b_1	$a_1b_1 = 0$	V(0)	1
	a_1 on the right of b_1	$a_1b_1 = 1$	$V_R(a_1b_1)$	001
		$a_1b_1 = 2$		000011
		$a_1b_1 = 3$		000001
	a_1 on the left of b_1	$a_1b_1 = 1$	$V_L(a_1b_1)$	010
		$a_1b_1 = 2$		000010
		$a_1b_1 = 3$		0000001
END-OF-LINE CODEWORD			EOL	000000000001

\- A '1' or a '0' flag bit is added to the EOL codeword to indicate that the following scan line is coded by one-dimensional coding or two-dimensional coding respectively.

(2681)

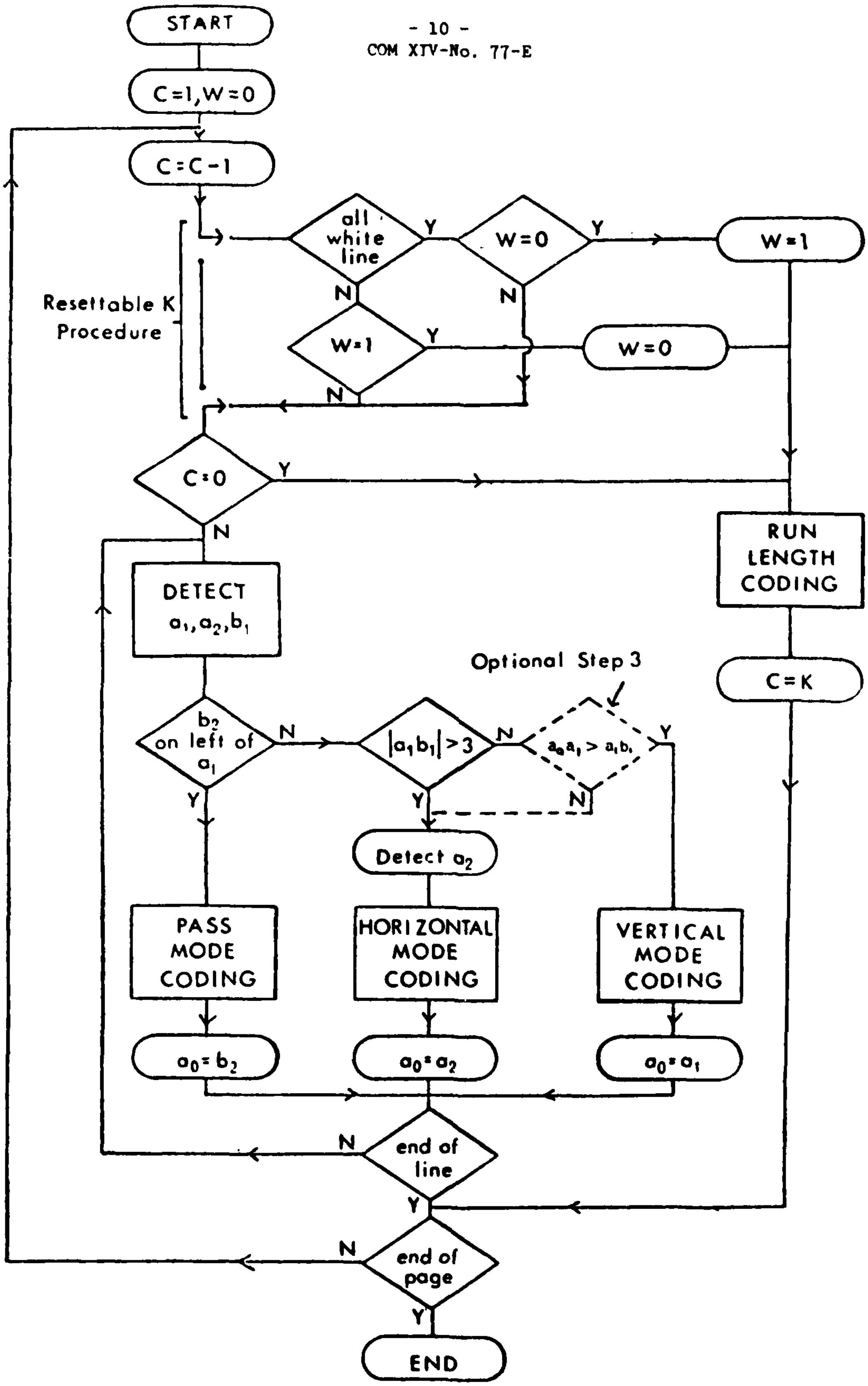


Figure 1 - Flow diagram for the two-dimentional coding scheme

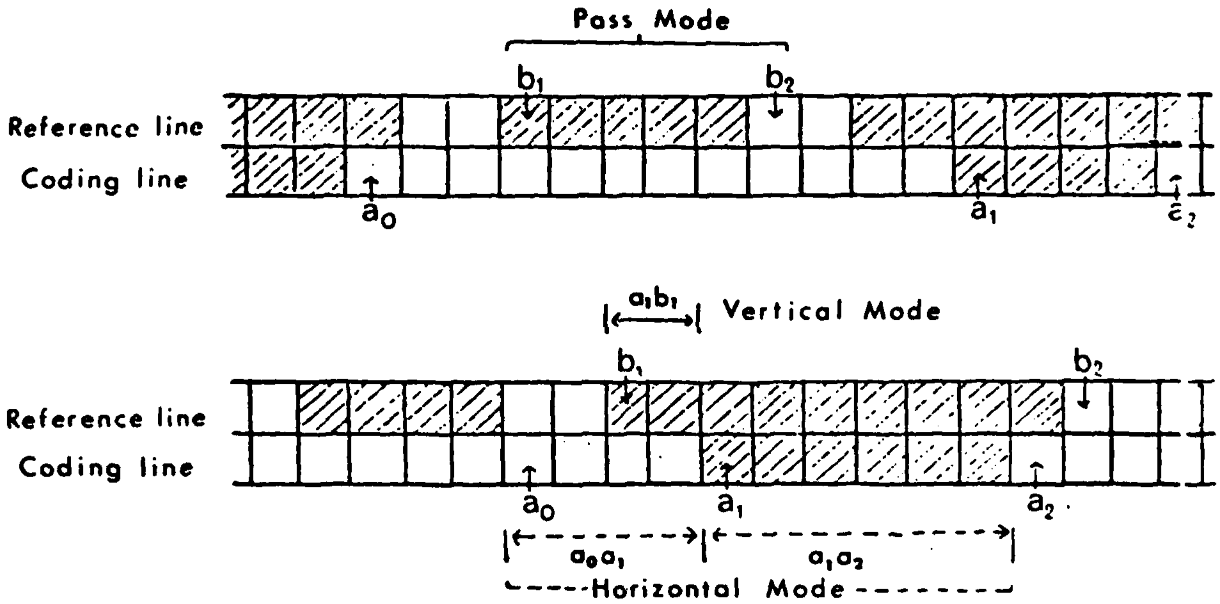


Figure 2 - Examples of coding modes

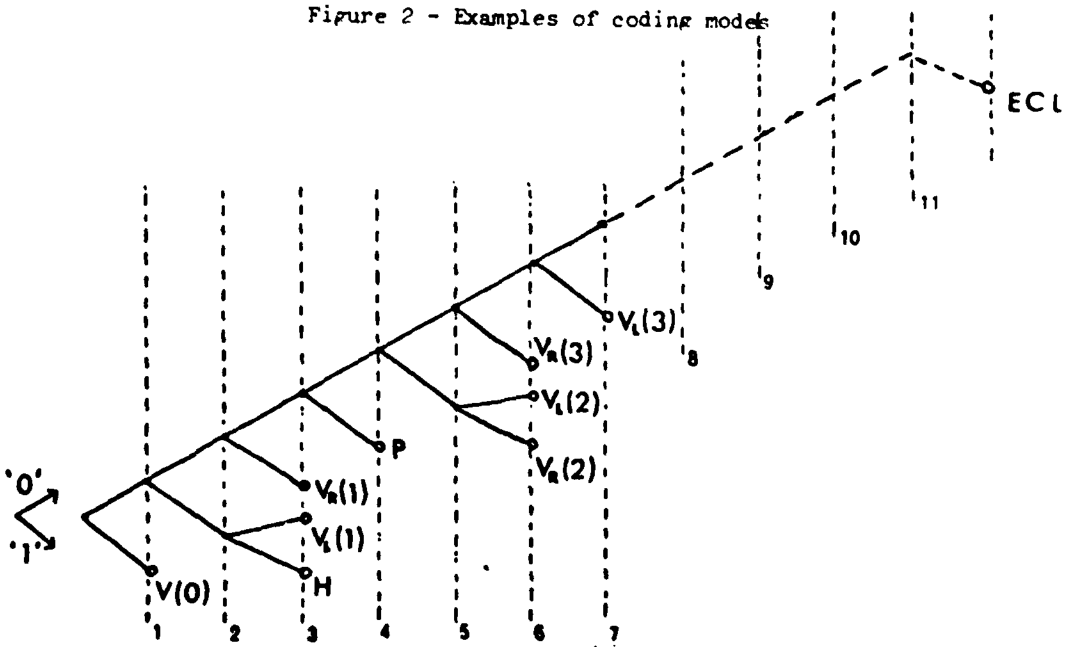


Figure 3 - The R2 code tree

NOTE: The results given in Tables 2 to 8 were obtained at a resolution of 3.85 lines per mm (ie 3190 lines per document) and a resettable K parameter equal to 2.

TABLE 2

Number of coded bits - the read code without stuffing bits

DOCUMENT	0 msecs WITHOUT LSS1/2	0 msecs WITH LSS1/2	5 msecs	10 msecs	20 msec
1	111454	120974	120992	138656	180089
4	369379	378899	380661	390444	41444[illegible]
5	190184	199704	200212	208225	2316[illegible]
7	36183[illegible]	371352	371355	377361	391445

TABLE 3

Number of coded bits - the read code with stuffing bits

DOCUMENT	0 msecs WITHOUT LSS1/2	0 msecs WITH LSS1/2	5 msecs	10 msecs	20 msec
1	114240	123760	123778	141425	182763
4	380296	389816	391578	401349	425284
5	194043	203563	204071	212068	235344
7	372251	381771	381774	387776	401837

TABLE 4

Number of coded bits - the R2 code (step 3 included)

DOCUMENT	0 msecs WITHOUT EOL	0 msecs WITH EOL	5 msecs	10 msecs	20 msecs
1	109952	125422	125430	139085	180042
4	361359	376829	377898	386194	409698
5	186752	202222	202384	208923	231495
7	354784	370254	370254	374906	388787

TABLE 5

Number of coded bits - the R2 code (step 3 omitted)

DOCUMENT	0 msecs WITHOUT EOL	0 msecs WITH EOL	5 msecs	10 msecs	20 msecs
1	110173	125643	125651	139364	180364
4	362438	377908	378977	387292	410868
5	186309	201779	201941	208514	231150
7	353358	368828	368828	373484	387368

TABLE 6

Frequencies of coding elements - the read code

DOCUMENT	P	H	V(0)	$V_R(1)$	$V_L(1)$	$V_R(\geq 2)$	$V_L(\geq 2)$
1	810	1814	4048	1315	1262	723	468
4	3640	7568	15217	5508	4577	2953	1911
5	1603	2912	10240	2536	2457	972	734
7	4035	7470	13295	2801	5311	982	2335

TABLE 7

Frequencies of the coding elements $V_R(n)$, $V_L(n)$, where $n \geq 2$, for the read code

DOCUMENT	$V_R(2)$	$V_L(2)$	$V_R(3)$	$V_L(3)$	$V_R(\geq 4)$	$V_L(\geq 4)$
1	444	304	141	104	138	60
4	2016	1378	781	418	156	115
5	629	520	247	141	96	73
7	604	1518	236	421	143	396

TABLE 8

Frequencies of coding elements - the R2 code (including step 3)

DOCUMENT	P	H	V(0)	$V_R(1)$	$V_L(1)$	$V_R(2)$	$V_L(2)$	$V_R(3)$	$V_L(3)$
1	792	2130	3958	1279	1218	323	227	132	47
4	3565	8508	15008	5395	4265	1552	1042	696	128
5	1422	3266	10081	2591	2429	474	393	241	56
7	3858	8363	13141	2746	5134	481	946	228	263

APPENDIX B

CCITT STUDY GROUP XIV

Contribution No. 82

Source: Federal Republic of Germany

International Telegraph and Telephone
Consultative Committee
(CCITT)

Period 1977-1980

COM XIV-No. 82-E

Original : English

Questions : 2/XIV - Point A.4

Date : March 1979

STUDY GROUP XIV - CONTRIBUTION No. 82

SOURCE : FEDERAL REPUBLIC OF GERMANY

TITLE : TWO-DIMENSIONAL CODING SCHEME
(Reply to Collective-letter No. 60)

Introduction:

A two-dimensional coding scheme for Group 3 facsimile apparatus is described as announced at the last meeting of Study Group XIV in Geneva, December 11. - 15., 1978. Differing from other proposals (IBM and Japan) this code makes use of a prediction method.
Annex 1 gives a detailed description of the code. Results of this code in comparison to other two-dimensional codes will be presented in a later contribution.

This code gives a very good performance, i. e. the compression factor is very high. Error susceptibility is comparable to other two-dimensional codes. This code uses a set of code word tables, which require a 256-word memory. One should take into account that implementation costs of memories are dropping constantly.

Besides high efficiency there is a distinct advantage of a clear patent situation. The owner of the respective patent will grant a duty-free licence to everybody.
An obligatory declaration is given in Annex 2.

Annexes : 2

(2600)

A N N E X 1

TWO-DIMENSIONAL CODING SCHEME

The two-dimensional coding scheme is a line-by-line coding method. It is an extension of the one-dimensional coding standard.

1. Parameter k

For reasons of restricting error propagation, one-dimensional coding with Modified-Huffman-Code is used for the first of every k lines. The parameter k can be chosen to k=2 for normal vertical resolution and k=4 for higher vertical resolution. Parameter k is set k=∞ , if transmission on data links with error control is used.

2. One-dimensional coding

One-dimensional coding of a scan line conforms with coding the run lengths by Modified-Huffman-Code.

3. Two-dimensional coding

The first step in the encoding process is to make a prediction of the present picture element X_o from the neighbouring picture elements X_1 to X_4 (Fig. 1). Table 1 shows the predicted value X_o depending on the four preceeding surrounding picture elements X_1 to X_4. Each black-and-white pattern of these four picture elements defines a different source state S_j. For each state there are individual conditional probabilities $P(X_o/S_j)$ that the present picture element X_o will be white or black. Now the predicted value is the more probable one in the given state S_j. Then the predicted value is compared to the real value of X_o. Each time the prediction is right, a white pel is inserted for X_o. When the prediction is wrong, X_o is replaced by a black pel. The resulting picture of prediction errors is a one-to-one transformation of the original picture, which means that all the information of the picture can be transmitted by coding the positions of the prediction errors.

The second step in the encoding algorithm explained here is to encode the run lengths between prediction errors as it is shown in Fig. 2. This is not done by coding the run lengths between every prediction error. The runs are encoded here separately for each source state S_j. For example the source is five times in state S_o until the first prediction error occurs in state S_o. So the run length to be transmitted is 5. For state S_1 there is a prediction error the third time the state S_1 occurs, so the run length is 3, etc.

For each state S_j an optimal run length code stored in an memory is used to transmit the run lengths between the prediction errors of state S_j (table 2). The storage of the run lengths codes requires a memory of 256 code words.

The run lengths codes used here are Truncated-Huffman-Codes, earlier described in /1/.

The encoder has to arrange the coded run lengths to be transmitted in a sequence corresponding to that of the states (Fig. 3). For example, first the encoder transmit the run length 5 of state S_o since the current line starts with S_o, then the run length 2 of state S_2 follows, afterwards the run lengths of state S_6 and state S_{15} etc.

Each scan line is terminated by a hypothetical prediction error at the end of the current line.

Each hypothetical picture element outside the page, which is requested for prediction, is assumed to be white. For example, for prediction of the first pel X_o in an arbitrary line X_1 and X_4 are assumed to be white.

4. Line synchronisation

The line synchronisation signal used here conforms with the EOL-Code used by the Modified-Huffman-Code. A string of eleven "O" followed by a "1" is used. Additional one bit following the EOL-Code indicates one-dimensional or two-dimensional coding of the succeeding line. A "O" indicates one-dimensional coding, a "1" indicates two-dimensional coding. To make the line synchronisation signal unique, a "1" is inserted in the data stream after occurence of ten continuous "O"s.

5. Fill bits

Fill bits are used to obtain the minimum transmission time per line requested by the system. A variable string of "O"s is inserted in the EOL-Code.

6. Return to control

End of document transmission is indicated by six consecutive EOL-Codes.

7. Compression factor

Fig. 4 shows the compression factor CF_4 achieved by the two-dimensional coding with parameter $k=\infty$, no overhead for the eight testdocuments with 1728 pels/ line and 2128 lines/page.

/1/ COM XIV, Doc. G3, No. 38
Dr.-Ing. Rudolf Hell GmbH
CCITT, Genf
October 1975

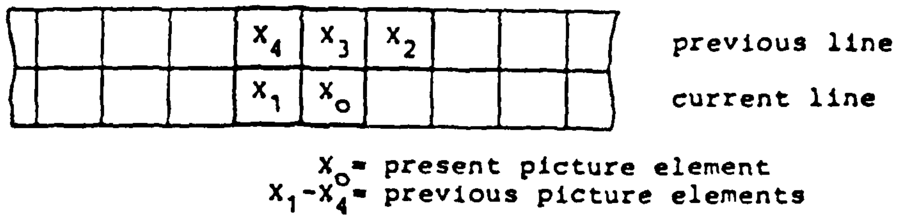


Figure 1 - Prediction pattern

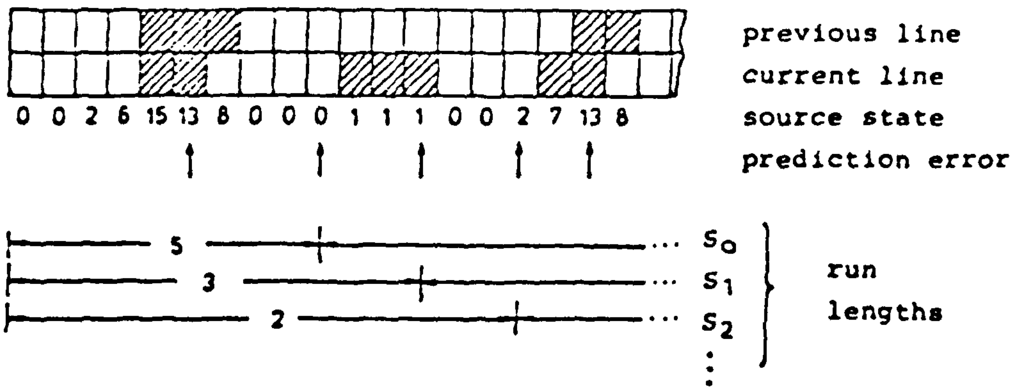


Figure 2 - Coding principle

(2600)

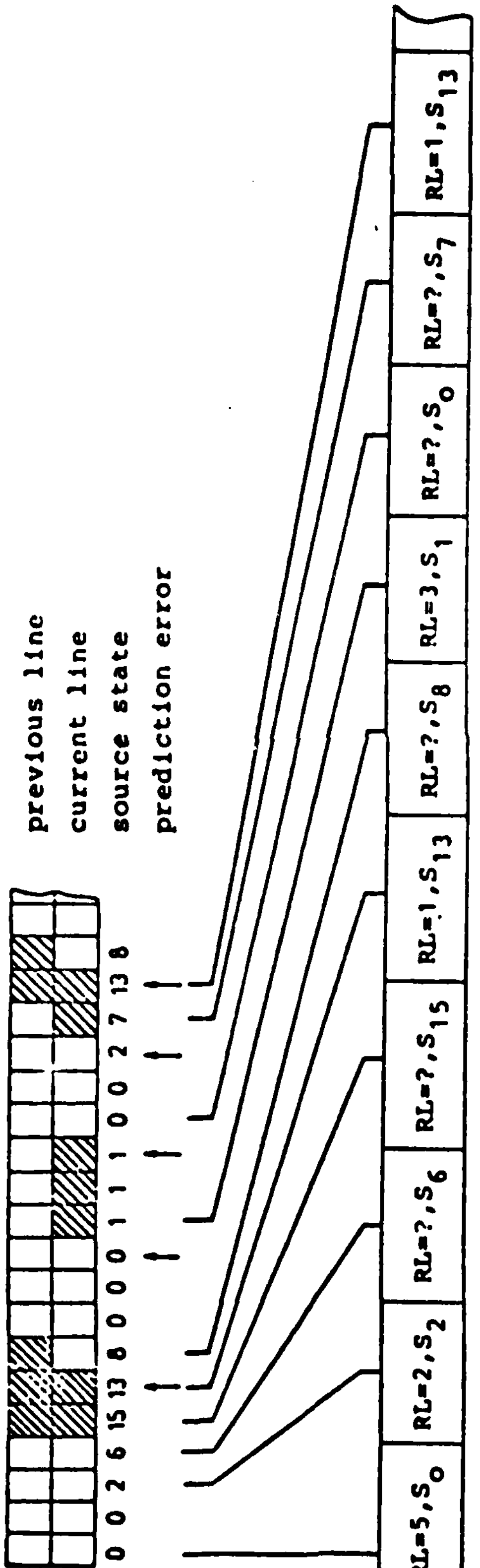


Figure 3 - Transmission sequence of run lengths (RL) between prediction errors

(2600)

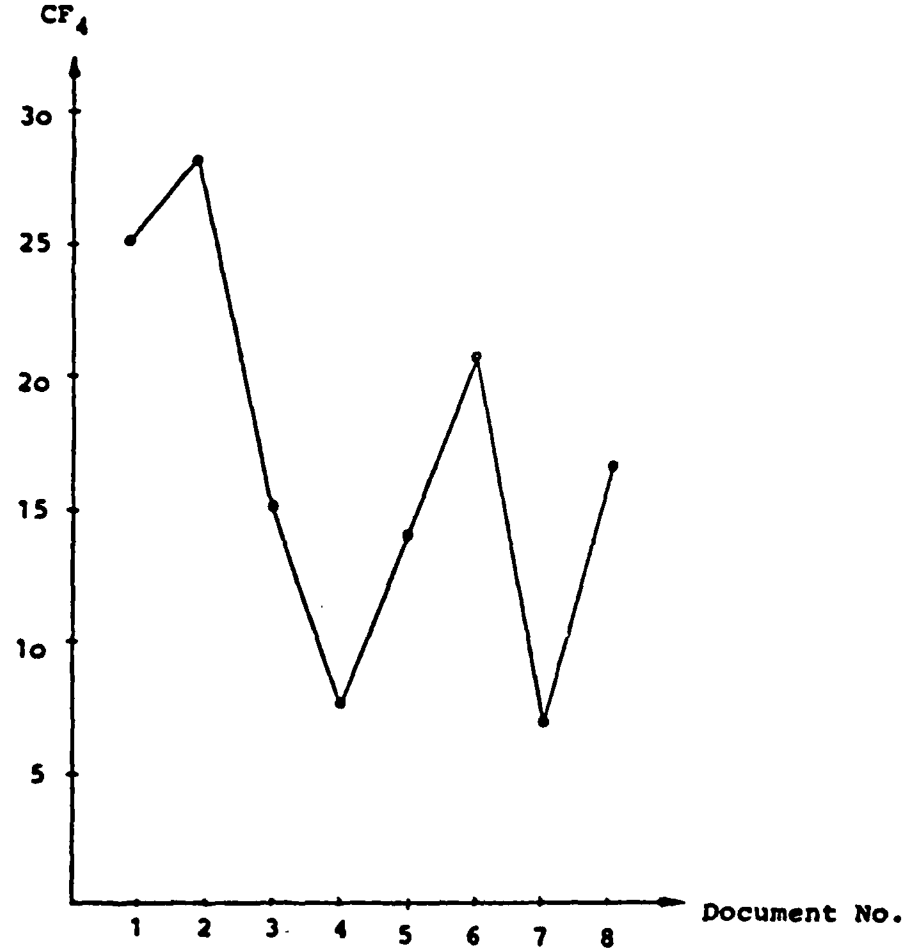
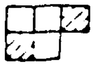


Figure 4 - Compression factor CF_4
1728 pels/line, 2128 lines/page
k = ∞ , exclusive overhead

(2600)

TABLE 1

Prediction Table

state S_j		predicted value X_o
S_0		white
S_1		black
S_2		white
S_3		black
S_4		white
S_5		black
S_6		black
S_7		black
S_8		white
S_9		white
S_{10}		white
S_{11}		black
S_{12}		white
S_{13}		black
S_{14}		white
S_{15}		black

(2600)

TABLE 2.1

Code Words

source state S_0 run length	source state S_0 code word	source state S_{15} run length	source state S_{15} code word
1	OLLOLL	1	LOLL
2	OOLOO	2	LOO
3	OOOOL	3	OLL
4	OLOLL	4	LLO
5	LLLLL	5	OOOL
6	LLLOO	6	OOLL
7	LLLOL	7	LLLL
8	OLOLO	8	OOLOO
9	OLLLL	9	OLOLO
10	OOOLL	10	OLOLL
11	OOLLL	11	LLLOL
12	OLLOOO	12	OOOOLO
13	OOOLOL	13	OLOOOO
14	OOOOOL	14	OLOOOL
15	OOLOLL	15	LLLOOO
16	LLLLOOO	16	OOOOOOL
17	LLLLOOL	17	OOLOLOO
18	OLOOLLO	18	OOOOOLO
19	OLLOOLL	19	OOOOLLL
20	OLOOLOL	20	LOLOLOL
21	OLOOLLL	21	LLLOOLO
22	OLLOLOL	22	LOLOLLO
23	OLOOOLO	23	OLOOLOL
24	OLOOLOO	24	OOOOOLLL
25	OLLOOLO	25	OOOOLLOO
26	OLOOOLL	26	OOLOLLOL
27	OLLLOLO	27	OOLOLOLL
28	OLLLOOL	28	OOLOLLLL
29	OOOLOOL	29	OOLOLLOO
30	OOLOLOO	30	OOOOOOOL
31	OOLLOLO	31	OLOOLLLO
32	OOLOLOL	32	OOOOLLOL
33	OOLLOOL	33	OLOOLOOL
34	OLOOOOOO	34	OLOOLLOO
35	OLOOOOOL	35	OLOOLLLL
36	LLLLOLLO	36	LOLOLLLO
37	LLLLOLLL	37	LLLOOLLO
38	LLLLOLOL	38	OLOOLOOOL
39	OLLOLOOL	39	OOLOLLLOL
40	OOOLOOOL	40	OOOOOLLOO
41	OLLLOLLL	41	LOLOLOOOO
42	OLLLOOOL	42	LLLOOLLLL
43	OOLLOLLO	43	LOLOLLLLL
44	OOOLOOOO	44	LLLOOLLLO
45	OOOOOOLL	45	LOLOLOOLO
46	OOOOOOOL	46	OLOOLLOLL
47	OOLLOOOL	47	OOOOOLLOL
48	OOLLOOOO	48	LOLOLLLLO
49	OOLLOLLL	49	OOLOLOLOL
50	OOOOOOLO	50	LOLOLOOOL
51	OLOOOOLOL	51	OOOOOOOOL
52	OLLOLOOOO	52	OOLOLOLOOO
53	LLLLOLOOL	53	OOLOLLLOOO
54	OLLLOLLOL	54	OOLOLLLOOL
55	LLLLOLOOO	55	OLOOLLOLOO
56	OLLOLOOOL	56	OOOOOOOOOL
57	OLOOOOLLO	57	OOLOLOLOOL
58	OLLLOLLOO	58	LOLOLOOLLO
59	OLOOOOLLL	59	LOLOLOOLLL
60	OLOOOOLOO	60	OLOOLOOOOL
61	OLLLOOOOL	61	OLOOLLOLOL
62	OLLLOOOOO	62	OLOOLOOOOO
63	OOOOOOOOL	63	OOOOOOOOOOL
64	OOOOOOOOOL	64	OOOOOOOOOOOL
1728	LLO	65-1728	LOLOO (prefix)*
65-1727	LO (prefix)*		

*The prefix is followed by the run length coded in 11-bit binary notation, most significant bit first.

TABLE 2.2

Code Words

source state S_1 and S_{14}

run length	code word
1	L
2	OLO
3	OOL
4	OLLO
5	OOOL
6	OLLLL
7	OLLLOO
8	OOOOOL
9	OOOOLL
10	OOOOLOL
11	OLLLOLOL
12	OOOOOOLL
13	OOOOLOOO
14	OLLLOLOOO
15	OOOOOOOLO
16	OOOOOOOOL
17	OOOOLOOLL
18	OLLLOLOOLL
19	OOOOOOLOLO
20	OOOOOOLOLL
21	OOOOOOOLLO
22	OOOOOOOLLL
23	OOOOLOOLOO
24	OOOOOOLOOL
25	OOOOOOOOOL
26	OOOOLOOLOL
27	OLLLOLOOLOL
28	OOOOOOOOOOL
29	OLLLOLOOLOO
30	OOOOOOLOOOO
31	OOOOOOLOOOL
32	OOOOOOOOOOOL
33-1728	OLLLOLL (prefix)*

source state S_2 and S_{13}

run length	code word
1	OLO
2	LO
3	OOL
4	LLL
5	OLLL
6	LLOO
7	OLLOL
8	OOOOL
9	LLOLO
10	OLLOOL
11	OOOOOL
12	OOOLLO
13	OOOLLL
14	OLLOOOO
15	OOOOOOL
16	OOOLOOO
17	LLOLLLO
18	OOOLOLL
19	OLLOOOLO
20	OLLOOOLL
21	OOOLOOLO
22	OOOLOLOO
23	OOOOOOOL
24	LLOLLLLO
25	LLOLLLLL
26	OOOLOOLLL
27	OOOLOOLLO
28	OOOLOLOLO
29	OOOOOOOOL
30	OOOLOLOLL
31	OOOOOOOOOL
32	OOOOOOOOOOL
33-1728	LLOLLO (prefix)*

source state S_6 and S_9

run length	code word
1	L
2	OL
3	OOOL
4	OOLL
5	OOLOO
6	OOOOOL
7	OOOOLO
8	OOLOLL
9	OOOOOOL
10	OOOOLLL
11	OOLOLOL
12	OOOOOOOL
13	OOOOLLOL
14	OOLOLOOL
15	OOOOLLOOL
16	OOLOLOOOL
17	OOOOOOOOOL
18	OOOOLLOOOO
19	OOOOLLOOOL
20	OOLOLOOOOL
21	OOLOLOOOOO
22	OOOOOOOOLLL
23	OOOOOOOOLLO
24	OOOOOOOOOOLL
25	OOOOOOOOOOOL
26	OOOOOOOOOOLO
27	OOOOOOOOLOOO
28	OOOOOOOOOOOOL
29	OOOOOOOOLOOLL
30	OOOOOOOOLOOLO
31	OOOOOOOOOOOOOOL
32	OOOOOOOOOOOOOOOL
33-1728	OOOOOOOOLOL (prefix)*

source state $S_3, S_4, S_5, S_7, S_8, S_{10}, S_{11}$ and S_{12}

run length	code word
1	OLO
2	L
3	OOL
4	OOOL
5	OLLLL
6	OLLOOL
7	OOOOOL
8	OOOOLO
9	OLLOOOL
10	OLLOLOO
11	OLLOOOO
12	OLLLOOL
13	OOOOOOL
14	OLLLOOO
15	OLLOLLL
16	OLLLOLO
17	OOOOLLO
18	OOOOOOOL
19	OOOOLLLL
20	OLLLOLLO
21	OOOOLLLO
22	OLLOLLOOL
23	OOOOOOOOL
24	OLLOLLOLO
25	OLLLOLLLO
26	OLLLOLLLL
27	OLLOLLOLLL
28	OLLOLLOLLO
29	OOOOOOOOOL
30	OLLOLLOOOO
31	OLLOLLOOOL
32	OOOOOOOOOOL
33-1728	OLLOLOL (prefix)*

* The prefix is followed by the run length coded in 11-bit binary notation, most significant bit first.

ANNEX 2

DR.-ING. RUDOLF HELL

GESELLSCHAFT MIT BESCHRÄNKTER HAFTUNG

DR. ING. RUDOLF HELL GMBH POSTFACH 6229 2300 KIEL 14

Fernmeldetechnisches Zentralamt
der Deutschen Bundespost
Referat A 26
Am Kavalleriesand 3

6100 Darmstadt

INFORMATIONSTECHNIK
ELEKTRONIK FÜR SATZ
UND REPRODUKTION

IHR ZEICHEN UND IHRE NACHRICHT VOM	UNSER ZEICHEN	DURCHWAHL	GRENZSTR. 1-5 2300 KIEL 14
	Lf/Hbs.	309	19th March, 1979

The following is a translation of a declaration of our company which was directed to the FTZ on November 10, 1978.

A copy of the original German text is attached.

Declaration

In the case that the method of a two-dimensional coding according to our German patent no. 25 56 803 "Process for Data Compression of Binary Coded Picture Signals" should be part of a CCITT recommendation, we irrevocably commit ourselves to grant by request a duty free licence to the above mentioned patent to everybody. This declaration will also be valid for our legal successors.

DR.-ING. RUDOLF HELL GMBH
signed by Taudt Marhencke

WERK DIETRICHSDORF TEL (04 31) 2001 1 TELEX 0292858 FAX (04 31) 2001 447
WERK SUCHSDORF TEL (04 31) 3013 1 FAX (04 31) 3013 214 TELEGRAMME HELLGERAETE KIEL ABC CODE 6 EDITION
VORSITZENDER DES AUFSICHTSRATES DR ING DR ING E H RUDOLF HELL GESCHÄFTSFÜHRER DR RER NAT ROLAND FUCHS
DIPL KFM ERNST ERICH MARHENCKE DIPL ING HEINZ TAUDT SITZ DER GESELLSCHAFT KIEL HR AMTSGERICHT KIEL ABT B NR

(2600)

APPENDIX C

SUBROUTINES WHICH ARE COMMON TO ALL ALGORITHMS

APPENDIX C

SUBROUTINES WHICH ARE COMMON TO ALL ALGORITHMS

```
START CF DCEC UPRINT PROGRAM          DSNAME=D0031.REDTAP.FORT
C      PROGRAM REDTAP32
C
       IMPLICIT INTEGER(A-Z)
       INTEGER PELBUF(1500),OTBUF(60)
       DATA PELMAX,PELFIL,OTFIL,TERM/1728,1,2,5/
C
C**************************** BEGIN PROGRAM ******************************
C
       INLNCT=0
  150  CONTINUE
       DO 100 I=1,60
  100  OTBUF(I)=0
       ID=1
       IF=250
       READ(PELFIL,300,END=500) IC,J
  300  FORMAT(250I4)
       J1=J
  316  IF(J.GT.250) GO TO 315
       JID1=J+ID-1
       READ(PELFIL,300) (PELBUF(K),K=ID,JID1)
       GO TO 400
  315  CONTINUE
       READ(PELFIL,300) (PELBUF(K),K=ID,IF)
       ID=IF+1
       IF=IF+250
       J=J-250
       IF(J.EQ.0) GO TO 400
       GO TO 316
  400  CONTINUE
       IF(INLNCT.GT.200) GO TO 450
C      WRITE(TERM,410) IC,J1
  410  FORMAT(5X,I4,5X,I6)
C      WRITE(TERM,420) (PELBUF(K),K=1,J1)
  420  FORMAT(2X,20(I4,2X))
  450  CONTINUE
       OTELP=1
       DO 460 I=1,J1
       RUN=PELBUF(I)
       IF(RUN.EQ.0) GC TO 700
       DO 470 K=1,RUN
       CALL MI2B(IC,OTBUF,OTELP,1)
       OTELP=OTELP+1
       IF(OTELP.GT.PELMAX) GO TO 480
  470  CONTINUE
       IC=MOD(IC+1,2)
  460  CONTINUE
  480  CONTINUE
       INLNCT=INLNCT+1
       WRITE(OTFIL) INLNCT,PELMAX,OTBUF
       GO TO 150
  500  CONTINUE
       WRITE(TERM,510) INLNCT,INLNCT
  510  FORMAT('0LINES WRITTEN =',I6,'; LAST LINE NUMBER =',I6)
       STOP
  600  CONTINUE
       STOP 600
  700  CONTINUE
       STOP 700
       E N D
0          END OF DCEC UPRINT PROGRAM            LINES PRINTED=     59
```

```
START OF DCEC UPRINT PROGRAM         DSNAME=D0031.CODELN.FORT
      SUBROUTINE CODELN(LENGTH,POLAR,CDELCT,CDDATA)
C
      IMPLICIT INTEGER(A-Z)
      COMMON/BUFF/PELBUF(60,2),CDBUF(240),OTBUF(50,2),
     *            STFBUF(240), STAT(3000)
      COMMON/HUFF/CODE(3,92,2),CODERO(3,9)
      COMMON/ERAY/ERRORS(2500)
C
C******************************* BEGIN PROGRAM **********************
C
C    INITIALIZE MAKE UP CODE, MAKE UP CODE LENGTH
C
      MCODE=0
      MLENG=0
C
C    CHECK INPUTS
C
      IF(POLAR.LT.1.OR.POLAR.GT.2) CALL EXIT
      IF(LENGTH.LT.0.OR.LENGTH.GT.1728) CALL EXIT
C
      IF(LENGTH.LE.63) GO TO 10
C
C    CALCULATE MAKE UP CODE INDEX, CODE, LENGTH
C     AND WRITE TO CODE LINE
C
      INDEX=LENGTH/64+64
      MCODE=CODE(3,INDEX,POLAR)
      MLENG=CODE(1,INDEX,POLAR)
      CALL MI2B(MCODE,CDBUF,CDELCT+1,MLENG)
      CDELCT=CDELCT+MLENG
      CDDATA=CDDATA+MLENG
C
C    CALCULATE TERMINATING CODE INDEX, CODE, LENGTH
C     AND ADD TO CODE LINE
C
   10 CONTINUE
      INDEX=MOD(LENGTH,64)+1
      TCODE=CODE(3,INDEX,POLAR)
      TLENG=CODE(1,INDEX,POLAR)
      CALL MI2B(TCODE,CDBUF,CDELCT+1,TLENG)
      CDELCT=CDELCT+TLENG
      CDDATA=CDDATA+TLENG
C
      RETURN
      END
```

```
      SUBROUTINE STATS(LENGTH,INLNCT,DIAG)
      IMPLICIT INTEGER(A-Z)
C
      INTEGER MTT(5),ITT(2,5),LENGTH(INLNCT)
      REAL STT(2,5),SUM,SUMSQ
      LOGICAL DIAG
C*********************** FILE DEFINITIONS ********************
C
      COMMON/FILES/TERM,LPFIL,PELFIL,OTFIL,ERFFIL
C
      DATA MTT/0,24,48,96,192/
C
C************************BEGIN PROGRAM*********************************
C
      DO 300 I=1,5
      ITT(1,I)=10000
      ITT(2,I)=0
      SUM=0.
      SUMSQ=0.
      DO 100 J=1,INLNCT
C
C    FIND FILLED LINE LENGTH
C
      LEN=MAX0(LENGTH(J),MTT(I))
      IF(DIAG) WRITE(TERM,50) LEN
   50 FORMAT(I8)
C
C    FIND MINIMUM LINE LENGTH
C
```

UNCLASSIFIED

UNCLASSIFIED

```
      ITT(1,I)=MIN0(LEN,ITT(1,I))
C
C    FIND MAXIMUM LINE LENGTH
C
      ITT(2,I)=MAX0(LEN,ITT(2,I))
C
C    FIND SUM OF LENGTHS
C
      SUM=SUM+FLOAT(LEN)
      SUMSQ=SUMSQ+(FLOAT(LEN))**2
  100 CONTINUE
C
C    FIND SAMPLE MEAN AND STANDARD DEVIATION
C
      STT(1,I)=SUM/FLOAT(INLNCT)
      STT(2,I)=SQRT((SUMSQ-(SUM**2)/FLOAT(INLNCT))/FLOAT(INLNCT-1))
  300 CONTINUE
C
      WRITE(LPFIL,400)(ITT(1,I),I=1,5)
  400 FORMAT(
     *'0                              MINIMUM TRANSMISSION TIME (4800 BPS)'/
     *'   CODED LINE'/
     *'     LENGTH                    0 MS     5 MS     10 MS    20 MS    40 MS'/
     *'   STATISTICS:'//
     *'     MINIMUM',10X,5(I8)//)
      WRITE(LPFIL,410)(ITT(2,I),I=1,5)
  410 FORMAT(
     *'     MAXIMUM',10X,5(I8)//)
      WRITE(LPFIL,420)(STT(1,I),I=1,5)
  420 FORMAT(
     *'     SAMPLE MEAN',9X,5(F8.2)//)
      WRITE(LPFIL,430)(STT(2,I),I=1,5)
  430 FORMAT(
     *'     STANDARD DEVIATION',2X,5(F8.2))
C
      RETURN
      END
0          END OF DCEC UPPRINT PROGRAM          LINES PRINTED=    200
```

```
      BLOCK DATA
C
      IMPLICIT INTEGER (A-Z)
C
C
      COMMON /G32BIT/KIBIT(32),KZBIT(32),LIBIT(32),LZBIT(32)
C
C
      DATA KIBIT /
     &              Z80000000,Z40000000,Z20000000,Z10000000,
     &              Z08000000,Z04000000,Z02000000,Z01000000,
     &              Z00800000,Z00400000,Z00200000,Z00100000,
     &              Z00080000,Z00040000,Z00020000,Z00010000,
     &              Z00008000,Z00004000,Z00002000,Z00001000,
     &              Z00000800,Z00000400,Z00000200,Z00000100,
     &              Z00000080,Z00000040,Z00000020,Z00000010,
     &              Z00000008,Z00000004,Z00000002,Z00000001/
C
      DATA KZBIT /
     &              Z7FFFFFFF,ZBFFFFFFF,ZDFFFFFFF,ZEFFFFFFF,
     &              ZF7FFFFFF,ZFBFFFFFF,ZFDFFFFFF,ZFEFFFFFF,
     &              ZFF7FFFFF,ZFFBFFFFF,ZFFDFFFFF,ZFFEFFFFF,
     &              ZFFF7FFFF,ZFFFBFFFF,ZFFFDFFFF,ZFFFEFFFF,
     &              ZFFFF7FFF,ZFFFFBFFF,ZFFFFDFFF,ZFFFFEFFF,
     &              ZFFFFF7FF,ZFFFFFBFF,ZFFFFFDFF,ZFFFFFEFF,
     &              ZFFFFFF7F,ZFFFFFFBF,ZFFFFFFDF,ZFFFFFFEF,
     &              ZFFFFFFF7,ZFFFFFFFB,ZFFFFFFFD,ZFFFFFFFE/
C
      DATA LIBIT /
     &              Z80000000,ZC0000000,ZE0000000,ZF0000000,
     &              ZF8000000,ZFC000000,ZFE000000,ZFF000000,
     &              ZFF800000,ZFFC00000,ZFFE00000,ZFFF00000,
     &              ZFFF80000,ZFFFC0000,ZFFFE0000,ZFFFF0000,
     &              ZFFFF8000,ZFFFFC000,ZFFFFE000,ZFFFFF000,
     &              ZFFFFF800,ZFFFFFC00,ZFFFFFE00,ZFFFFFF00,
     &              ZFFFFFF80,ZFFFFFFC0,ZFFFFFFE0,ZFFFFFFF0,
     &              ZFFFFFFF8,ZFFFFFFFC,ZFFFFFFFE,ZFFFFFFFF/
C
      DATA LZBIT /
     &              Z7FFFFFFF,Z3FFFFFFF,Z1FFFFFFF,Z0FFFFFFF,
     &              Z07FFFFFF,Z03FFFFFF,Z01FFFFFF,Z00FFFFFF,
     &              Z007FFFFF,Z003FFFFF,Z001FFFFF,Z000FFFFF,
     &              Z0007FFFF,Z0003FFFF,Z0001FFFF,Z0000FFFF,
     &              Z00007FFF,Z00003FFF,Z00001FFF,Z00000FFF,
     &              Z000007FF,Z000003FF,Z000001FF,Z000000FF,
     &              Z0000007F,Z0000003F,Z0000001F,Z0000000F,
     &              Z00000007,Z00000003,Z00000001,Z00000000/
C
C
C
C
      END
```

```
START OF OCSC UPRINT PROGRAM          DSNAME=D0031.MI28.FORT
CMI28
C
      SUBROUTINE MI28(IVAL,IBA,JB,NB)
      IMPLICIT INTEGER(A-Z)
      DIMENSION IBA(2)
C
C******* MI28 MOVES THE BIT STRING RIGHT-JUSTIFIED IN IVAL
C         TO THE JB-TH THRU THE (JB+NB-1)-TH BIT OF IBA.
C
C******* LABELED COMMON /G32BIT/ *******
C
      COMMON /G32BIT/MASK(32),COMASK(32),LIBIT(32),LZBIT(32)
      INTEGER MASK,COMASK,LIBIT,LZBIT
C
C
C******* MI28 EXECUTE *******
      JRHB=JB+NB-2
      NBT=NB
      JRE=JRHB/32+1
      JRB=MOD(JRHB,32)+1
      NBR=MIN0(NBT,JRB)
      LVAL=IVAL
      JIM=32-NBR
C
      J=LAND(LVAL,LZBIT(JIM))
      K=32-JRB
      LRE=LOR(LAND(IBA(JRE),LZBIT(JRB)),SHFTL(J,K))
      K=32-JIM
      LVAL=SHFTR(LVAL,K)
      NBT=NBT-JRB
C
  199 IF(NBT) 300,390,200
C
  200 IBA(JRE)=LRE
      JRE=JRE-1
      LRE=LVAL
      LVAL=0
      NBT=NBT-32
      GO TO 199
C
  300 JIM=-NBT
      LRE=LOR(LRE,LAND(IBA(JRE),LIBIT(JIM)))
  390 IBA(JRE)=LRE
      RETURN
C
      END
```

START OF UCEC UPRINT PROGRAM DSNAME=D0031.I4B.FCRT

```
CI4B
      INTEGER FUNCTION I4B(IBA,JB,NB)
      IMPLICIT INTEGER (A-Z)
      DIMENSION IBA(2)
C
C******* I4B RETURNS AN INTEGER VALUE FOR THE BIT STRING
C             STARTING AT THE JB-TH BIT OF IBA
C                AND CONSISTING OF NB BITS.
C
C******* LABELED COMMON /G32BIT/ *******
C
      COMMON /G32BIT/MASK(32),COMASK(32),LIBIT(32),LZBIT(32)
      INTEGER MASK,COMASK,LIBIT,LZBIT
C
C
C******* I4B EXECUTE ********
C
      IF(NB-1) 10,30,20
   10 STOP 10
   20 CONTINUE
      JRHB=JB+NB-2
      NBT=MINO(NB,32)
      JRE=JRHB/32+1
      JRB=MOD(JRHB,32)+1
      NBR=MINO(NBT,JRB)
      JIM=32-NBR
C
C   SHIFT RIGHT 32-JRB BITS AND PUT IN ZEROS ON LEFT
C
      J=IBA(JRE)
      K=32-JRB
      I4B=LAND(LZBIT(JIM),SHFTR(J,K))
C
C   CALCULATE NUMBER OF BITS REMAINING IN LEFT PORTION IF ANY
      NBR=NBT-NBR
      IF(NBR.LE.0) RETURN
C
C   IF LEFT PORTION EXISTS, SHIFT LEFT TO LINE UP WITH RIGHT
C   PORTION AND 'OR' WITH RIGHT PORTION
C
      J=LAND(IBA(JRE-1),LZBIT(32-NBR))
      K=32-JIM
      I4B=LOR(I4B,SHFTL(J,K))
      RETURN
C   BIT STRING HAS ONLY ONE BIT
   30 CONTINUE
      I4B=0
      JBIND=(JB-1)/32+1
      MSKIND=JB-(JBIND-1)*32
      IF(LAND(MASK(MSKIND),IBA(JBIND)).EQ.MASK(MSKIND)) I4B=1
      RETURN
C     END
```

```
      SUBROUTINE ERRMES(PELBUF,OTBUF,PELMAX,VRES,ERRCNT)
C
      IMPLICIT INTEGER(A-Z)
      REAL ESF
C******* LABELED COMMON /G32BIT/ *******
C
      COMMON /G32BIT/MASK(32),COMASK(32),LIBIT(32),LZBIT(32)
      INTEGER MASK,COMASK,LIBIT,LZBIT
C
C*********************** FILE DEFINITIONS ********************
C
      COMMON/FILES/TERM,LPFIL,PELFIL,OTFIL,ERFIL
C
      DIMENSION PELBUF(60), OTBUF(60)
      COMMON/LOGIC/SEARCH,DIAG
      LOGICAL SEARCH,DIAG
C
C*************************** BEGIN PROGRAM ****************************
C
      REWIND PELFIL
      REWIND OTFIL
      ERROR=0
      OTELW=(PELMAX+32-1)/32
      OTLNCT=0
C
C    READ AN ERROR FREE LINE
C
  100 CONTINUE
      READ(PELFIL,END=600,ERR=800) INLNNO,INELCT,PELBUF
      IF(MOD(INLNNO-1,VRES).NE.0) GO TO 100
C
C    READ AN ERROR-CORRUPTED LINE
C
  200 CONTINUE
      READ(OTFIL,END=500,ERR=800) OTLNNO,OTELCT,OTBUF
      OTLNCT=OTLNCT+1
```

UNCLASSIFIED

```
  300 CONTINUE
C
C   COUNT DIFFERENCES BETWEEN TRANSMITTED AND RECEIVED LINES
C
      DO 450 I=1,OTELW
      IF(OTBUF(I).EQ.PELBUF(I)) GO TO 450
      IF(.NOT.DIAG) GO TO 420
      WRITE(TERM,410) INLNNO,OTLNNO,I,PELBUF(I),OTBUF(I)
  410 FORMAT(3I8,2Z12)
  420 CONTINUE
      DO 440 J=1,32
      IF(I4B(OTBUF(I),J,1).NE.I4B(PELBUF(I),J,1)) ERROR=ERROR+1
  440 CONTINUE
  450 CONTINUE
      IF(OTLNNO-INLNNO) 200,100,580
C
C   ERROR LINE NUMBER GREATER THAN GOOD LINE NUMBER;
C   COUNT DIFFERENCES BETWEEN GOOD AND ALL WHITE LINE
C
  500 CONTINUE
      DO 550 I=1,OTELW
      IF(PELBUF(I).EQ.0) GO TO 550
      IF(.NOT.DIAG) GO TO 520
      WRITE(TERM,410) INLNNO,OTLNNO,I,PELBUF(I),OTBUF(I)
  520 CONTINUE
      DO 540 J=1,32
      IF(I4B(PELBUF(I),J,1).NE.0) ERROR=ERROR+1
  540 CONTINUE
  550 CONTINUE
C
  580 READ(PELFIL,END=600,ERR=800) INLNNO,INELCT,PELBUF
      IF(MOD(INLNNO-1,VRES).NE.0) GO TO 580
C
      GO TO 300
C
C   CALCULATE ERROR SENSITIVITY FACTOR
C
  600 CONTINUE
      ESF=0.
      IF(ERRCNT.LE.0) GO TO 650
      ESF=FLOAT(ERROR)/FLOAT(ERRCNT)
  650 CONTINUE
C
      WRITE(LPFIL,700) ERROR,ERRCNT,ESF,OTLNCT
  700 FORMAT('0NUMBER OF INCORRECT PELS =',I10/
     *       '0NUMBER OF BITS IN ERROR TRANSMITTED =',I10/
     *       '0ERROR SENSITIVITY FACTOR =',F12.4/
     *       '0TOTAL NUMBER OF OUTPUT LINES PROCESSED = ',I8)
C
      RETURN
  800 CONTINUE
      STOP 800
      END
```

```
C      PROGRAM WRITAD32
C
       IMPLICIT INTEGER (A-Z)
       INTEGER PELBUF(60),OTBUF(1500)
       DATA PELMAX,PELFIL,OTFIL,TERM/1728,1,2,6/
C
C****************************** BEGIN PROGRAM ******************************
C
       INLNCT=0
  150  CONTINUE
       DO 100 I=1,60
  100  PELBUF(I)=0
       READ(PELFIL,END=500,ERR=600) INLNNO,INELCT,PELBUF
       INLNCT=INLNCT+1
       IC=I4B(PELBUF,1,1)
       POLAR=IC
       J=1
       RUN=0
       DO 200 I=1,PELMAX
       PEL=I4B(PELBUF,I,1)
       IF(PEL.EQ.POLAR) GO TO 190
       OTBUF(J)=RUN
       J=J+1
       RUN=1
       POLAR=MOD(POLAR+1,2)
       GO TO 200
  190  CONTINUE
       RUN=RUN+1
  200  CONTINUE
       OTBUF(J)=RUN
       ID=1
       IF=250
       WRITE(OTFIL,300) IC,J
  300  FORMAT(250I4)
       J1=J
  316  IF(J.GT.250) GO TO 315
       JID1=J+ID-1
       WRITE(OTFIL,300) (OTBUF(K),K=ID,JID1)
       GO TO 400
  315  CONTINUE
       WRITE(OTFIL,300) (OTBUF(K),K=ID,IF)
       ID=IF+1
       IF=IF+250
       J=J-250
       IF(J.EQ.0) GO TO 400
       GO TO 316
  400  CONTINUE
C      WRITE(TERM,410) IC,J1
  410  FORMAT(5X,I4,5X,I5)
C      WRITE(TERM,420) (OTBUF(K),K=1,J1)
  420  FORMAT(2X,20(I4,2X))
       GO TO 150
  500  CONTINUE
       WRITE(TERM,510) INLNCT,INLNNO
  510  FORMAT(' LINES WRITTEN =',I6,' LAST LINE NUMBER =',I6)
       STOP
  600  CONTINUE
       STOP 600
       END
```

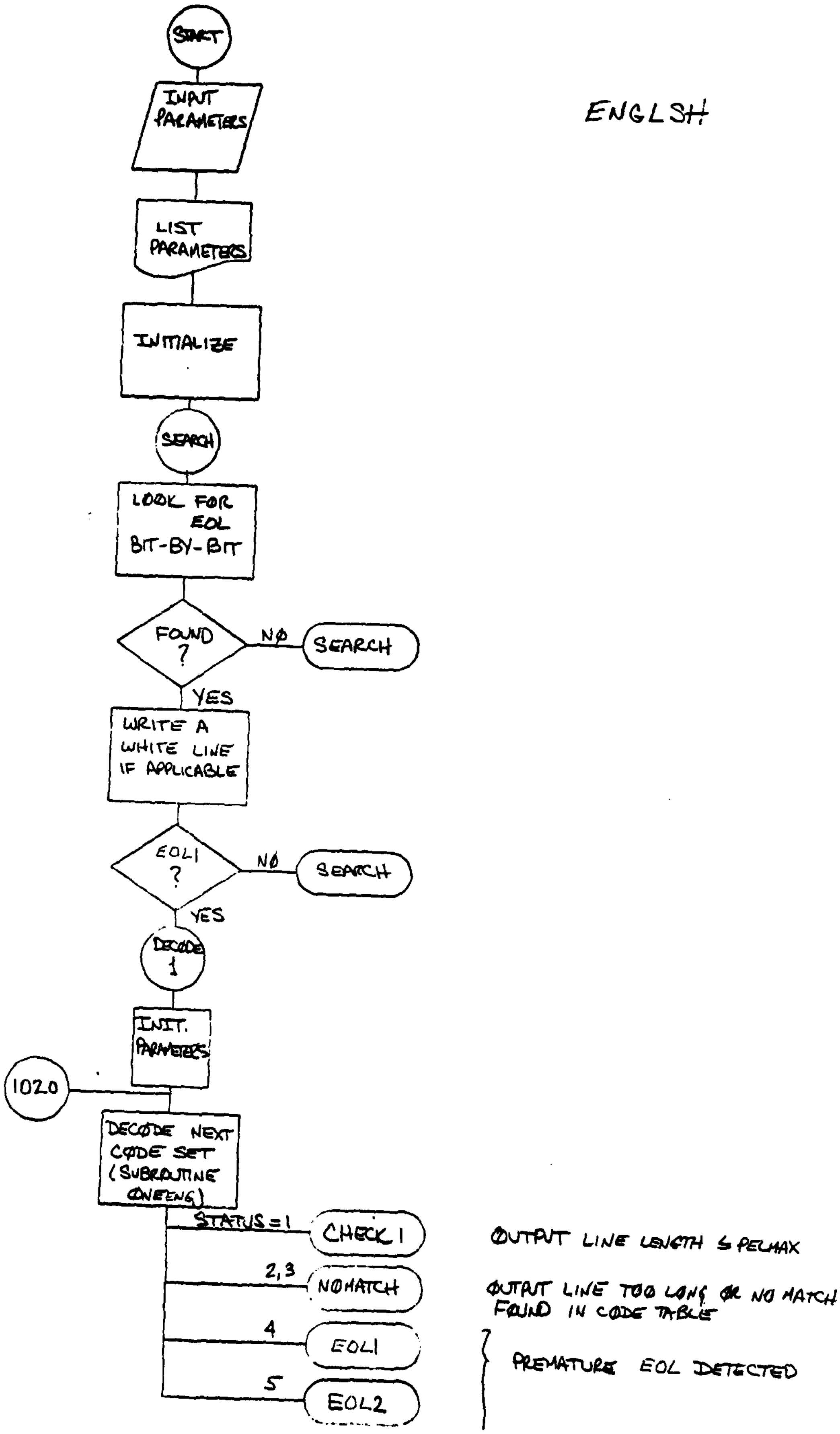
```
START OF DCEC UPRINT PROGRAM                    DSNAME=D0031.CONVERT.FORT
C        PROGRAM CONVERT
C
C     THIS PROGRAM CONVERTS BINARY FORMAT USED BY COMPRESSION
C     ALGORITHMS TO THE FOLLOWING BINARY FORMAT:
C
C           1728 BITS (216 BYTES) PER RECORD:
C
C     EACH LINE OF 1728 PELS BECOMES ONE RECORD
C
         IMPLICIT INTEGER(A-Z)
         INTEGER PELBUF(60),OTBUF(54)
         EQUIVALENCE (PELBUF(1),OTBUF(1))
         INLNCT=0
  100    READ(1,END=500,ERR=600) INLNNO,INELCT,PELBUF
         INLNCT=INLNCT+1
         WRITE(2,ERR=700) OTBUF
         GO TO 100
C
  500    CONTINUE
         WRITE(5,510) INLNCT,INLNNO
  510    FORMAT(' LINES WRITTEN =',I6,'; LAST LINE NUMBER =',I6)
         STOP
  600    CONTINUE
         STOP 600
  700    STOP 700
         E N D
0             END OF DCEC UPRINT PROGRAM              LINES PRINTED=     26
```

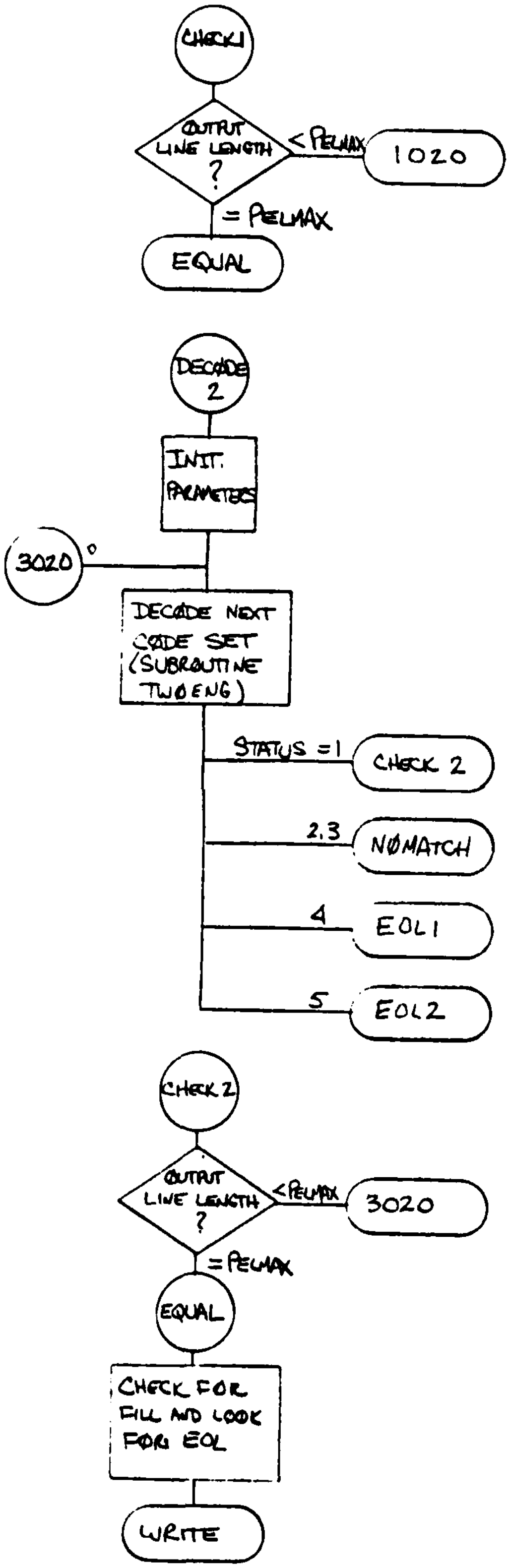
APPENDIX D

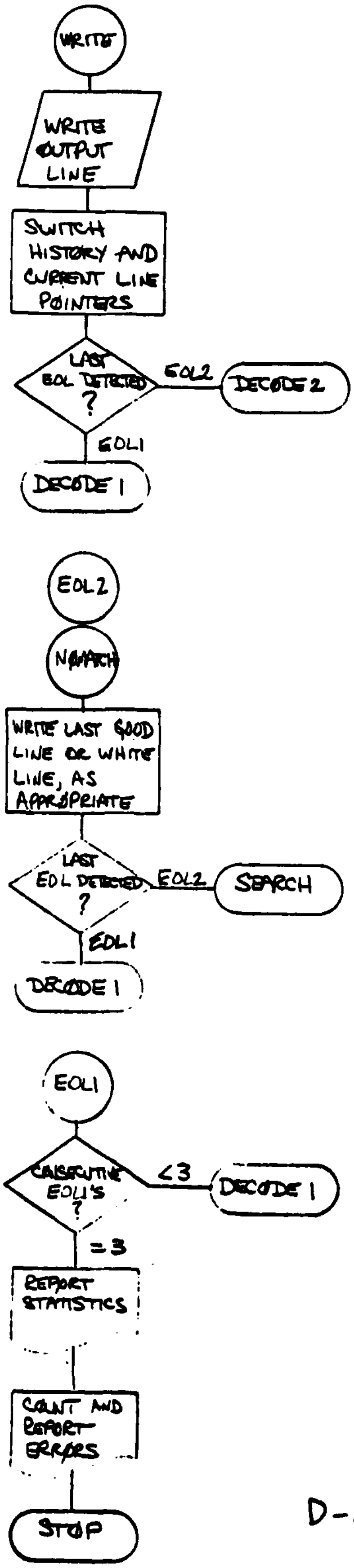
FLOW CHART

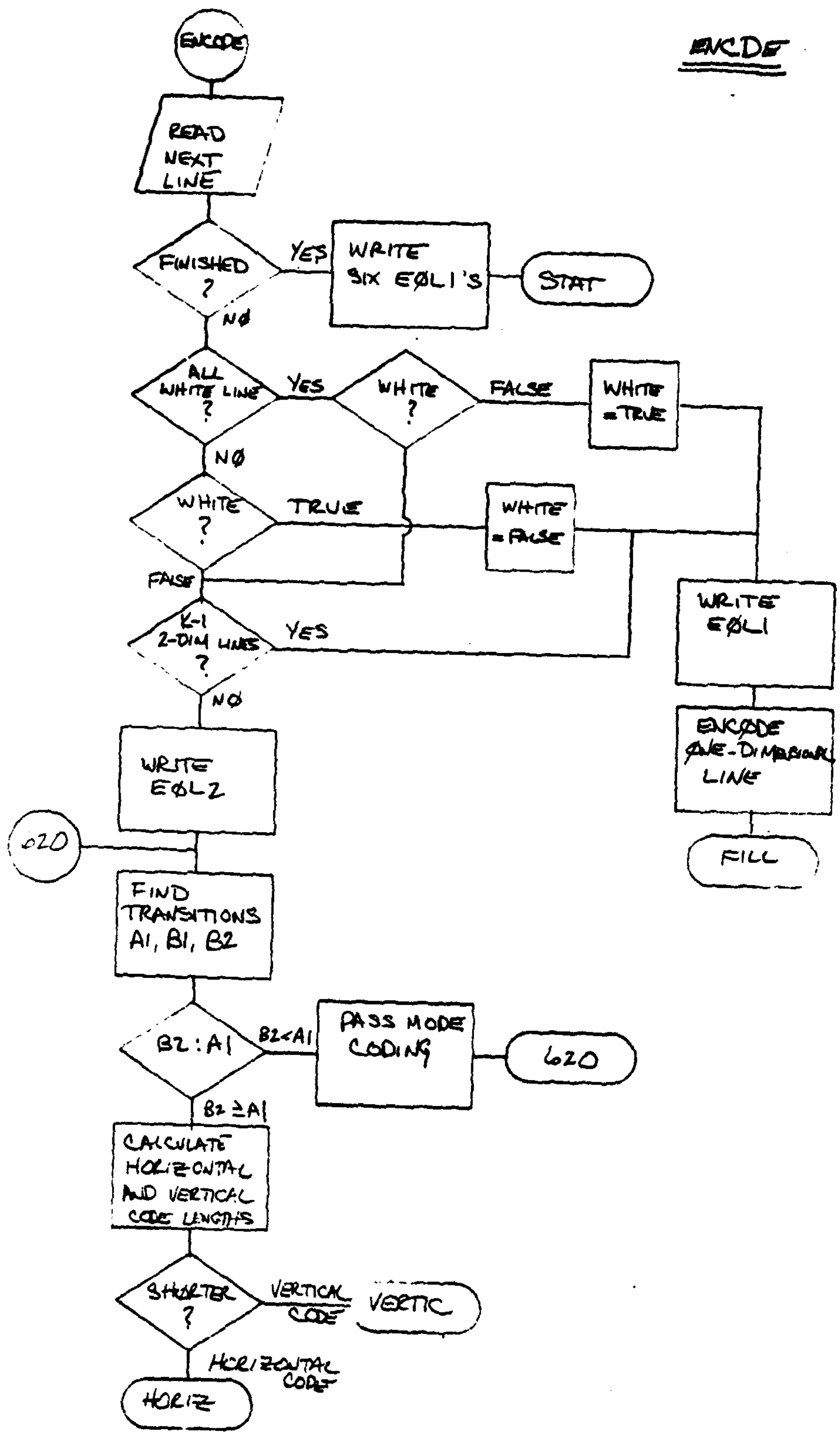
BRITISH POST OFFICE

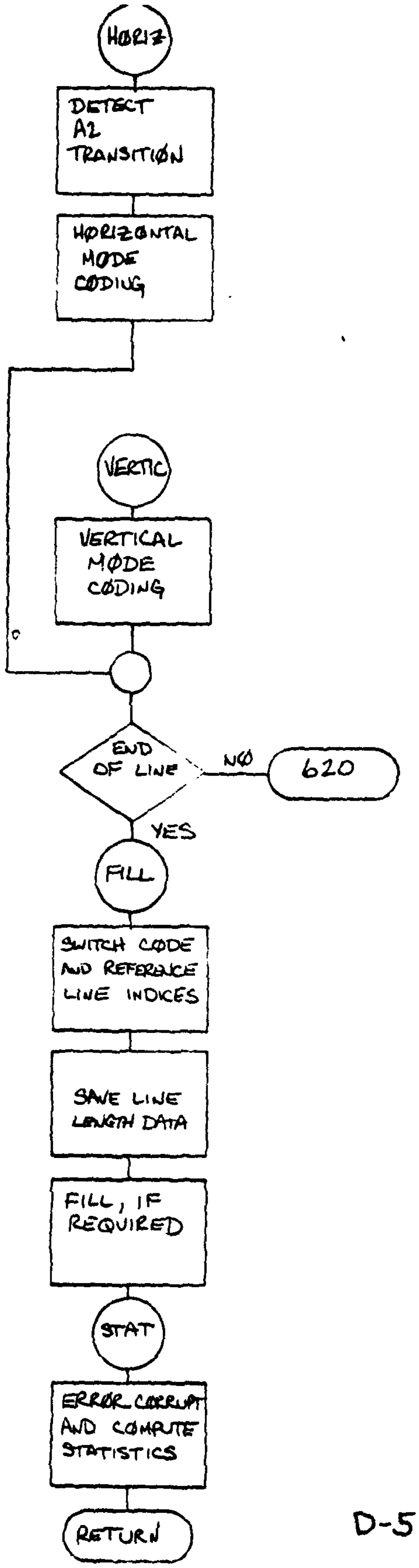
ENGLSH


START
INPUT PARAMETERS
LIST PARAMETERS
INITIALIZE
SEARCH
LOOK FOR EOL BIT-BY-BIT
FOUND ?
NO
SEARCH
YES
WRITE A WHITE LINE IF APPLICABLE
EOL1 ?
NO
SEARCH
YES
DECODE 1
INIT. PARAMETERS
1020
DECODE NEXT CODE SET (SUBROUTINE ONEENG)
STATUS=1
CHECK1
2,3
NOMATCH
4
EOL1
5
EOL2
OUTPUT LINE LENGTH ≤ PELMAX
OUTPUT LINE TOO LONG OR NO MATCH FOUND IN CODE TABLE
PREMATURE EOL DETECTED

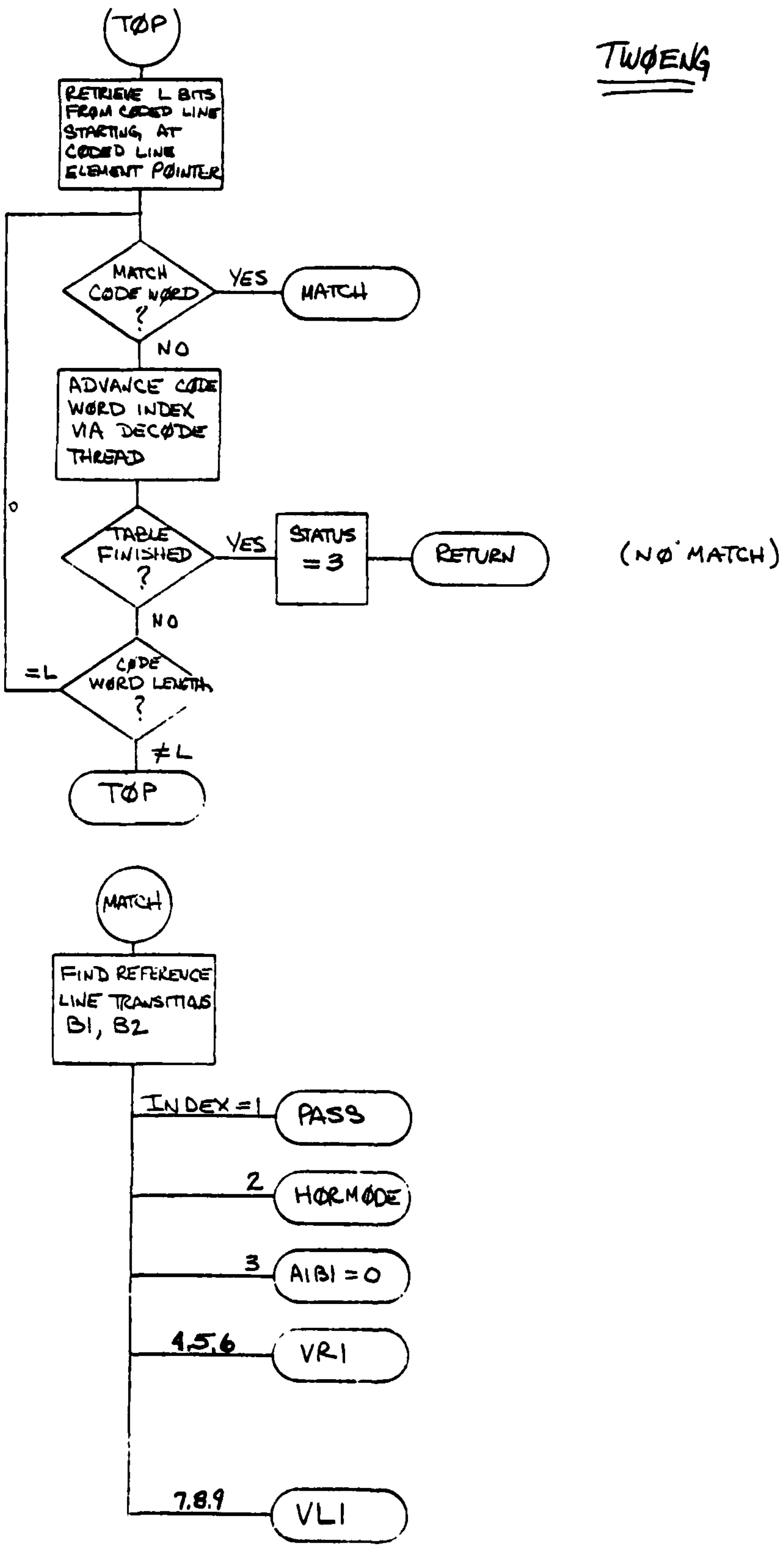

CHECK1
OUTPUT LINE LENGTH ?
< PELMAX
1020
= PELMAX
EQUAL
DECODE 2
INIT. PARAMETERS
3020
DECODE NEXT CODE SET (SUBROUTINE TWOENG)
STATUS = 1
CHECK 2
2,3
NOMATCH
4
EOL1
5
EOL2
CHECK 2
OUTPUT LINE LENGTH ?
< PELMAX
3020
= PELMAX
EQUAL
CHECK FOR FILL AND LOOK FOR EOL
WRITE

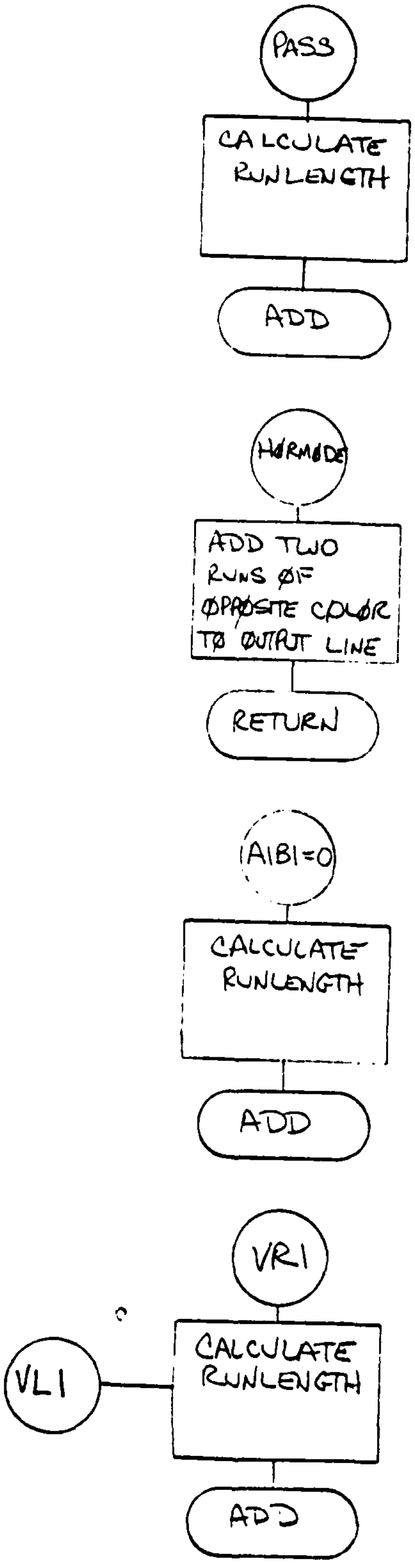

WRITE
WRITE OUTPUT LINE
SWITCH HISTORY AND CURRENT LINE POINTERS
LAST EOL DETECTED ?
EOL2
DECODE 2
EOL1
DECODE 1
EOL2
NMATCH
WRITE LAST GOOD LINE OR WHITE LINE, AS APPROPRIATE
LAST EOL DETECTED ?
EOL2
SEARCH
EOL1
DECODE 1
EOL1
CONSECUTIVE EOL1'S ?
<3
DECODE 1
=3
REPORT STATISTICS
COUNT AND REPORT ERRORS
STOP


ENCDE
ENCODE
READ NEXT LINE
FINISHED ?
YES
WRITE SIX EØLI'S
STAT
NØ
ALL WHITE LINE ?
YES
WHITE ?
FALSE
WHITE = TRUE
NØ
WHITE ?
TRUE
WHITE = FALSE
FALSE
K-1 2-DIM LINES ?
YES
NØ
WRITE EØLI
ENCØDE ØNE-DIMENSIONAL LINE
FILL
WRITE EØL2
620
FIND TRANSITIONS A1, B1, B2
B2 : A1
B2<A1
PASS MODE CODING
620
B2 ≥ A1
CALCULATE HORIZONTAL AND VERTICAL CODE LENGTHS
SHORTER ?
VERTICAL CODE
VERTIC
HORIZONTAL CODE
HORIZ


HORIZ
DETECT A2 TRANSITION
HORIZONTAL MODE CODING
VERTIC
VERTICAL MODE CODING
END OF LINE
NO
620
YES
FILL
SWITCH CODE AND REFERENCE LINE INDICES
SAVE LINE LENGTH DATA
FILL, IF REQUIRED
STAT
ERROR CORRUPT AND COMPUTE STATISTICS
RETURN

TWØENG


TØP
RETRIEVE L BITS FROM CODED LINE STARTING AT CODED LINE ELEMENT POINTER
MATCH CODE WORD ?
YES
MATCH
NO
ADVANCE CODE WORD INDEX VIA DECODE THREAD
TABLE FINISHED ?
YES
STATUS = 3
RETURN
(NØ MATCH)
NO
CODE WORD LENGTH ?
=L
≠L
TØP
MATCH
FIND REFERENCE LINE TRANSITIONS B1, B2
INDEX = 1
PASS
2
HØRMØDE
3
A1B1 = 0
4,5,6
VR1
7,8,9
VL1


PASS
CALCULATE RUNLENGTH
ADD
HØRMØDE
ADD TWO RUNS ØF ØPPØSITE CØLØR TØ ØUTPUT LINE
RETURN
|A1B1=0
CALCULATE RUNLENGTH
ADD
VR1
VL1
CALCULATE RUNLENGTH
ADD

APPENDIX E

CODE LISTING

BRITISH POST OFFICE

```
START OF CCEC JPRINT PROGRAM          DSNAME=N0026.ENGLSH.FORT
C       PROGRAM ENGLSH
        IMPLICIT INTEGER(A-Z)
        REAL CF3,CF4,ERRATE
C******* LABELED COMMON /G32BIT/ *******
C
        COMMON /G32BIT/MASK(32),COMASK(32),LIBIT(32),LZBIT(32)
        INTEGER MASK,COMASK,LIBIT,LZBIT
C
        COMMON/BUFF/PELBUF(60,2),CDBUF(240),OTBUF(60,2),
     *            STBUF(240), STAT(3000)
        COMMON/HUFF/CODE(3,92,2),CCDERD(3,11)
        COMMON/ERAY/ERRORS(2500)
C*********************** FILE DEFINITIONS *******************
C
        COMMON/FILES/TERM,LPFIL,PELFIL,OTFIL,ERFIL
C
C********************** LABELLED COMMON VARIABLES **********************
C
        COMMON/IVAR/PELMAX,VRES,EPHASE,CMPMAX,ERRMOD,LINMAX,K
        COMMON/PVAR/IN_NNO,OTLNNO,OTELW,INELP,CDELP,OTELP,CDELW,
     *            CDELCT,INELCT,TCDATA,TCDEL,ERRPNT,ERRUFF,ERRLIM,
     *            ERRCNT,INLNCT,CONSEC,ONECNT,LNNOBF,KCNT,
     *            INCOD,INREF,CTCOD,OTREF,STFBIT
        COMMON/ICHAR/DD,II,MM,TT,NN,YY
        COMMON/LOGIC/SEARCH,DIAG,SYNC,WRITE,ZERO,LEFT,CHCOL,ONE,WHITE
        LOGICAL SEARCH,DIAG,SYNC,WRITE,LEFT,CHCOL,ONE,WHITE
C
C   READ INPUT PARAMETERS
   90   WRITE(6,100)
  100   FORMAT('$PARAMETERS: INPUT(=I), OR DEFAULT(=D)?')
        READ(5,110,ERR=90) INSW
  110   FORMAT(A1)
        IF (INSW.EQ.DD) GO TO 315
        IF (INSW.NE.II) GO TO 90
C
C   READ DIAGNOSTIC SWITCH
C
  114   WRITE(6,115)
  115   FORMAT('$DIAGNOSTIC PRINTOUT? (Y OR N): ')
        READ(5,110) INSW
        IF(INSW.EQ.YY) GO TO 116
        IF(INSW.EQ.NN) GO TO 120
        GO TO 114
  116   CONTINUE
        DIAG=.TRUE.
C
C   READ MAXIMUM NUMBER OF PELS PER LINE
C
  120   CONTINUE
        WRITE(6,130)
  130   FORMAT('$ENTER MAXIMUM NUMBER OF PELS PER LINE: ')
        READ(5,140,ERR=120) PELMAX
  140   FORMAT(I4)
        IF(PELMAX.GE.1.AND.PELMAX.LE.1728) GO TO 160
        WRITE(6,150) PELMAX
  150   FORMAT('0NUMBER OUT OF RANGE (=',I6,')')
        GO TO 120
C
C   READ VERTICAL SAMPLING
C
  160   CONTINUE
        WRITE(6,170)
  170   FORMAT('$ENTER VERTICAL SAMPLING: ')
        READ(5,180,ERR=160) VRES
  180   FORMAT(I2)
        IF(VRES.GE.1.AND.VRES.LE.10) GO TO 190
        WRITE(6,150) VRES
        GO TO 160
C
C   READ PARAMETER K
C
  190   CONTINUE
        WRITE(6,192)
  192   FORMAT('$ENTER PARAMETER K: ')
        READ(5,140,ERR=190) K
        IF(K.GE.1.AND.K.LE.3000) GO TO 200
        WRITE(6,150) K
        GO TO 190
C
C   READ ERROR PATTERN PHASE
C
```

```
  200 CONTINUE
      WRITE(6,210)
  210 FORMAT('$ENTER ERROR PATTERN PHASE: ')
      READ(5,220,ERR=200) EPHASE
  220 FORMAT(I1)
      IF(EPHASE.GE.0.AND.EPHASE.LE.3) GO TO 240
      WRITE(6,150) EPHASE
      GO TO 200
C
C   READ MINIMUM COMPRESSED LINE LENGTH
C
  240 CONTINUE
      WRITE(6,250)
  250 FORMAT('$ENTER MINIMUM COMPRESSED LINE LENGTH: ')
      READ(5,140,ERR=240) CMPMAX
      IF(CMPMAX.GE.0.AND.CMPMAX.LE.1728) GO TO 320
      WRITE(6,150) CMPMAX
      GO TO 240
C
C   READ NUMBER OF SCAN LINES TO BE PROCESSED
  320 CONTINUE
      WRITE(6,330)
  330 FORMAT('$NUMBER OF SCAN LINES TO BE PROCESSED=? ')
      READ(5,140,ERR=320) LINMAX
      IF(LINMAX.GE.1.AND.LINMAX.LE.3000) GO TO 280
      WRITE(6,150) LINMAX
      GO TO 320
C
C   READ ERROR MODE
C
  280 CONTINUE
      WRITE(6,290)
  290 FORMAT('$ERROR MODE=? (M=MANUAL,T=TAPE,N=NO ERRORS)')
      READ(5,110,ERR=280) ERRMOD
      IF(ERRMOD.EQ.MM) GO TO 300
      IF(ERRMOD.EQ.TT) GO TO 315
      IF(ERRMOD.NE.NN) GO TO 280
      GO TO 350
C
C   READ ERROR LOCATIONS
C
  300 CONTINUE
      ERRLIM=1
  305 READ(5,140) ERRORS(ERRLIM)
      IF(ERRORS(ERRLIM).EQ.9999) GO TO 310
      ERRLIM=ERRLIM+1
      GO TO 305
  310 CONTINUE
      ERRLIM=ERRLIM-1
      GO TO 350
C
C   READ ERROR TAPE FILE AND OPEN
C
  315 CONTINUE
C
      ERRLIM=1
      READ(3,318,END=317) ERRORS(ERRLIM)
      ERRLIM=ERRLIM+1
  316 READ(3,318,END=317) ERRORS(ERRLIM)
  318 FORMAT(I16)
      ERRORS(ERRLIM)=ERRORS(ERRLIM)+ERRORS(ERRLIM-1)
      ERRLIM=ERRLIM+1
      GO TO 316
  317 ERRLIM=ERRLIM-1
C
  350 CONTINUE
C
  360 CONTINUE
C   WRITE INPUT PARAMETERS
C
      WRITE(6,400) PELMAX,VRES,K,EPHASE,CMPMAX,LINMAX
  400 FORMAT('1INPUT PARAMETERS:'/
     *        '0MAXIMUM NUMBER OF PELS PER LINE=',I5/
     *        '0VERTICAL SAMPLING: N=',I4/
     *        '0PARAMETER K =',I4/
     *        '0ERROR PATTERN PHASE =',I4/
     *        '0MINIMUM COMPRESSED LINE LENGTH =',I4,' BITS'/
     *        '0NUMBER OF SCAN LINES TO BE PROCESSED =',I6)
      IF(ERRMOD.EQ.NN) WRITE(6,410)
  410 FORMAT ('0NO ERRORS INSERTED')
      IF(ERRMOD.EQ.MM) WRITE(6,140) (ERRORS(I),I=1,ERRLIM)
      IF(ERRMOD.EQ.TT) WRITE(6,420) ERRLIM
```

```
  420 FORMAT(I12,' ERRORS OBTAINED FROM ERROR TAPE')                      00001
C******************** BEGIN PROGRAM ***********************************   00001
C                                                                          00001
C     INITIALIZE                                                           00001
C                                                                          00001
      TCDEL=0                                                              00001
      TCDATA=0                                                             00001
      ERRPNT=1                                                             00001
      ERRCNT=0                                                             00001
      INLNCT=0                                                             00001
      ERRCFF=EPHASE*1024                                                   00001
      CDELCT=32                                                            00001
      CTELP=1                                                              00001
      CDELP=32+1                                                           00001
      CONSEC=1                                                             00001
      INREF=1                                                              00001
      INCOD=2                                                              00001
      OTREF=1                                                              00001
      OTCOD=2                                                              00001
      WHITE=.FALSE.                                                        00001
      KCNT=1                                                               00001
C                                                                          00001
      DO 800 I=1,240                                                       00001
      STFBUF(I)=0                                                          00001
      CDBUF(I)=0                                                           00001
  800 CONTINUE                                                             00001
      DO 850 I=1,60                                                        00001
      OTBUF(I,OTREF)=0                                                     00001
      OTBUF(I,OTCOD)=0                                                     00001
      PELBUF(I,INREF)=0                                                    00001
      PELBUF(I,INCOD)=0                                                    00001
  850 CONTINUE                                                             00001
      SEARCH=.TRUE.                                                        00001
      SYNC=.FALSE.                                                         00001
      WRITE=.FALSE.                                                        00001
C                                                                          00001
C     SEARCH MODE: LOOK FOR EOL1 BIT-BY-BIT                                00002
C                                                                          00002
 900  CONTINUE                                                             00002
      CALL GET_E(13,MODE,LBITS,L)                                          00002
      GO TO (910,930,930,920),MODE                                         00002
      STOP 900                                                             00002
 910  CONTINUE                                                             00002
C                                                                          00002
C     EOL NOT FOUND; ADVANCE POINTER AND TRY AGAIN                         00002
C                                                                          00002
      CDELP=CDELP+1                                                        00002
      GO TO 900                                                            00002
 920  CONTINUE                                                             00002
      STOP 920                                                             00002
 930  CONTINUE                                                             00002
C                                                                          00002
C     EOL FOUND                                                            00002
C                                                                          00002
      SEARCH=.FALSE.                                                       00002
      CDELP=CDELP+L                                                        00002
      IF(WRITE) GO TO 935                                                  00002
      WRITE=.TRUE.                                                         00002
      GO TO 960                                                            00002
 935  CONTINUE                                                             00002
C                                                                          00002
C     SET OUTPUT DECODE LINE TO 0 AND WRITE OUT                            00002
      DO 950 I=1,60                                                        00002
      OTBUF(I,OTCOD)=0                                                     00002
 950  CONTINUE                                                             00002
      WRITE(2) OTLNNO,PELMAX,(OTBUF(I,OTCOD),I=1,60)                       00002
      OTLNNO=LINNOE                                                        00002
 960  CONTINUE                                                             00002
      IF(MODE-2)965,1000,900                                               00002
 965  STOP 965                                                             00002
 1000 CONTINUE                                                             00002
C                                                                          00002
C     PERFORM ONE-DIMENSIONAL DECODE OF A COMPLETE LINE                    00002
C     FIRST,SET OUTPUT BUFFER TO WHITE                                     00002
C     (ONLY BLACK RUNS WILL BE INSERTED)                                   00002
C                                                                          00002
      DO 1010 I=1,60                                                       00002
      OTBUF(I,OTCOD)=0                                                     00002
 1010 CONTINUE                                                             00002
C                                                                          00002
      INDEX=3                                                              00002
      COLOR=1                                                              00002
```

```
      OTELP=1
C
 1020 CONTINUE
      CALL ONEENG(INDEX,COLOR,STATUS,L)
      GO TO (1030,1070,1070,1035,1040),STATUS
C            1    2    3    4    5
      STOP 1000
C
C    RUN ADDED; CHECK LENGTH OF OUTPUT LINE
C
 1030 CONTINUE
      ONE=.TRUE.
      IF(OTELP-1-PELMAX) 1031,1032,1050
 1031 CONTINUE
      IF(CHCOL) COLOR=MOD(COLOR+2,2)+1
      INDEX=3
      GO TO 1020
 3000 CONTINUE
C
C    PERFORM TWO-DIMENSIONAL DECODE
C
C
C    FIRST,SET OUTPUT BUFFER TO WHITE
C    (ONLY BLACK RUNS WILL BE INSERTED)
C
      DO 3010 I=1,60
      OTBUF(I,OTCOD)=0
 3010 CONTINUE
C
      INDEX=3
      COLOR=1
      OTELP=1
C
 3020 CONTINUE
      CALL TWOENG(INDEX,COLOR,STATUS,L)
      GO TO (3030,1070,1070,1035,1040),STATUS
C            1    2    3    4    5
      STOP 3000
C
C    RUN ADDED; LOOK FOR NEXT RUN
C
 3030 CONTINUE
      ONE=.FALSE.
      IF(OTELP-1-PELMAX) 3031,1032,1050
 3031 CONTINUE
      IF(CHCOL) COLOR=MOD(COLOR+2,2)+1
      INDEX=3
      GO TO 3020
C
C
C    LINE LENGTH=PELMAX; CHECK FOR FILL AND LOOK FOR EOL
C
 1032 CONTINUE
      ZERO=-1
 1033 CONTINUE
      ZERO=ZERO+1
      CALL GETLE(1,MODE,LBITS,L)
C
      GO TO (1034,1050,1050,1050),MODE
C
C     CHECK FOR FILL
C
 1034 CONTINUE
C
      CDELP=CDELP+L
      IF(LBITS.EQ.0) GO TO 1033
      IF(ZERO.LE.10) GO TO 1070
C
C     EOL FOUND; CHECK TYPE
C
      CALL GET_E(1,MODE,LBITS,L)
      IF(LBITS.EQ.1) MODE=2
      IF(LBITS.EQ.0) MODE=3
      GO TO (1070,1060,1060,1080),MODE
C
C     PREMATURE EOL DETECTED
C
C
C     EOL1 DETECTED
C
 1035 CONTINUE
      CDELP=CDELP+L
```

```
      STATUS=4
      IF(OTELP.LE.1) CONSEC=CONSEC+1
      IF(CONSEC-2)1080,1000,2000
C
C   EOL2 DETECTED
C
 1040 CONTINUE
      CDELP=CDELP+L
      STATUS=5
C
      GO TO 1080
C
C   PROBLEMS.PROBLEMS
C
 1050 STOP 1050
C
C   LINE LENGTH CORRECT, EOL DETECTED PROPERLY; WRITE OUTPUT LINE
C
 1060 CONTINUE
      CDELP=CDELP+L
      WRITE(2)OTLNNO,PELMAX,(OTBUF(I,OTCOD),I=1,60)
      OTLNNO=LNNOBF
      CONSEC=1
      IF(ONE) SYNC=.TRUE.
      TEMP=OTREF
      OTREF=OTCOD
      OTCOD=TEMP
      IF(MODE.EQ.2) GO TO 1000
      GO TO 3000
C
C   LINE TOO LONG OR NO MATCH
C
 1070 CONTINUE
      WRITE=.FALSE.
C
C   LINE SHORT
C
 1080 CONTINUE
      IF(.NOT.SYNC) GO TO 1090
C
C   WRITE LAST GOOD LINE
C
      WRITE(2) OTLNNO,PELMAX,(OTBUF(I,OTREF),I=1,60)
      SYNC=.FALSE.
      GO TO 1110
 1090 CONTINUE
C
C   WRITE A WHITE LINE
C
      DO 1100 I=1, 60
 1100 OTBUF(I,OTCOD)=0
      WRITE(2) OTLNNO,PELMAX,(OTBUF(I,OTCOD),I=1,60)
 1110 OTLNNO=LNNOBF
      IF(STATUS.EQ.4) GO TO 1000
      SEARCH=.TRUE.
      GO TO 900
C
C   END OF MESSAGE
C
 2000 CONTINUE
      WRITE(6,2010) CONSEC
 2010 FORMAT('0END OF MESSAGE DETECTED (',I2,' EOL''S)')
C
C   REPORT COMPRESSION FACTOR, ERROR SENSITIVITY FACTOR,BIT ERROR RATE
C
      ERRATE=FLOAT(ERRCNT)/FLOAT(TCDEL)
      WRITE(6,2020) TCDEL,TCDATA,STEBIT,INLNCT,ERRATE
 2020 FORMAT('0TOTAL NUMBER OF CODED BITS = ',I8/
     *       '0TOTAL NUMBER OF CODED DATA BITS = ',I8/
     *       '0TOTAL NUMBER OF 2-DIM LINES = ',I8/
     *       '0TOTAL NUMBER OF INPUT LINES PROCESSED = ',I8/
     *       '0BIT ERROR RATE = ',G14.6)
C
      CALL STATS(STAT,INLNCT,DIAG)
      CF3=FLOAT(PELMAX)*FLOAT(INLNCT)/FLOAT(TCDEL)
      CF4=FLOAT(PELMAX)*FLOAT(INLNCT)/FLOAT(TCDATA)
C
      WRITE(6,2030) CF3,CF4
 2030 FORMAT('0COMPRESSION FACTOR FOR G3 MACHINE (CF3) =',F8.4/
     *       '0COMPRESSION FACTOR FOR G4 MACHINE (CF4) =',F8.4)
C
      CALL ERRMES(PELBUF,OTBUF,PELMAX,VRES,ERRCNT)
```

```
C
      STOP
      E N D
      SUBROUTINE GETLE(LBITS,MCDE,WRD,L)
      IMPLICIT  INTEGER(A-Z)
C******* LABELLED COMMON /G32BIT/ *******
C
      COMMON /G32BIT/MASK(32),COMASK(32),LIBIT(32),LZBIT(32)
      INTEGER MASK,COMASK,LIBIT,LZBIT
C
      COMMON/BUFF/PELBUF(60,2),CDBUF(240),OTBUF(60,2),
     *            STFBUF(240), STAT(3000)
      COMMON/HUFF/CODE(3,92,2),CODERD(3,11)
      COMMON/ERAY/ERRORS(2500)
C********************* LABELLED COMMON VARIABLES **********************
C
      COMMON/IVAR/PELMAX,VRES,EPHASE,CMPMAX,ERRMOD,LINMAX,K
      COMMON/PVAR/INLNNO,OTLNNO,OTELW,INELP,CDELP,OTELP,CDELW,
     *            CDELCT,INELCT,TCDATA,TCDEL,ERRPNT,ERROFF,ERRLIM,
     *            ERRCNT,INLNCT,CONSEC,ONECNT,LNNO3F,KCNT,
     *            INCOD,INREF,CTCOD,OTREF,STFBIT
      COMMON/ICHAR/DD,II,MM,TT,NN,YY
      COMMON/LOGIC/SEARCH,DIAG,SYNC,WRITE,ZERO,LEFT,CHCOL,ONE,WHITE
      LOGICAL SEARCH,DIAG,SYNC,WRITE,ZERO,LEFT,CHCOL,ONE,WHITE
C*************************** BEGIN PROGRAM ***************************
C
      MODE=4
C
C   RETRIEVE NEXT BIT FROM CDBUF
C
  100 CONTINUE
C
C   ENCODE A NEW LINE IF NECESSARY
C
      IF(LBITS+CDELP-1.LE.CDELCT) GO TO 200
      IF(CDELCT-CDELP+1) 170,190,180
  170 STOP 170
  180 CONTINUE
      STFBUF(1)=I4B(STFBUF,CDELP,CDELCT-CDELP+1)
  190 CONTINUE
      CDELP=32-(CDELCT-CDELP)
      CALL ENCODE
  200 CONTINUE
      WRD=I4B(STFBUF,CDELP,LBITS)
      L=LBITS
      IF(L.LT.13) GO TO 250
      IF(L.EQ.13.AND.WRD.EQ.CODERD(3,10)) GO TO 300
      IF(L.EQ.13.AND.WRD.EQ.CODERD(3,11)) GO TO 400
  250 CONTINUE
      MODE=1
      RETURN
  300 CONTINUE
      MODE=2
      RETURN
  400 CONTINUE
      MODE=3
      RETURN
      END
      SUBROUTINE ENCODE
C
      IMPLICIT  INTEGER(A-Z)
C
C******* LABELED COMMON /G32BIT/ *******
C
      COMMON /G32BIT/MASK(32),COMASK(32),LIBIT(32),LZBIT(32)
      INTEGER MASK,COMASK,LIBIT,LZBIT
C
      COMMON/BUFF/PELBUF(60,2),CDBUF(240),OTBUF(60,2),
     *            STFBUF(240), STAT(3000)
      COMMON/HUFF/CODE(3,92,2),CODERD(3,11)
      COMMON/ERAY/ERRORS(2500)
C*********************** FILE DEFINITIONS ********************
C
      COMMON/FILES/TERM,LPFIL,PELFIL,OTFIL,ERFIL
C
C********************* LABELLED COMMON VARIABLES **********************
C
      COMMON/IVAR/PELMAX,VRES,EPHASE,CMPMAX,ERRMOD,LINMAX,K
      COMMON/PVAR/INLNNO,OTLNNO,OTELW,INELP,CDELP,OTELP,CDELW,
     *            CDELCT,INELCT,TCDATA,TCDEL,ERRPNT,ERROFF,ERRLIM,
     *            ERRCNT,INLNCT,CONSEC,ONECNT,LNNO3F,KCNT,
     *            INCOD,INREF,CTCOD,CTREF,STFBIT
```

```
      COMMON/ICHAR/DD,II,MM,TT,NN,YY
      COMMON/LOGIC/SEARCH,DIAG,SYNC,WRITE,ZERO,LEFT,CHCOL,ONE,WHITE
      LOGICAL SEARCH,DIAG,SYNC,WRITE,ZERO,LEFT,CHCOL,ONE,WHITE
C
C************************************ BEGIN PROGRAM ********************
C
C     INITIALIZE VARIABLES
C
      KCNT=KCNT-1
      CDELCT=32
      CDDATA=0
      DO 50 I=2,240
      CDBUF(I)=0
      STFBUF(I)=0
   50 CONTINUE
C
C     READ INPUT PICTURE FILE
C
  100 CONTINUE
      READ(1,END=120,ERR=500)
     * INLNNO,INELCT,(PELBUF(I,INCOD),I=1,60)
      IF(MOD(INLNNO-1,VRES).NE.0) GO TO 100
      IF(INELCT.LT.PELMAX) CALL EXIT
      INLNCT=INLNCT+1
C
C     LOAD OUTPUT LINE NUMBER BUFFER
C
      LNNOBF=INLNNO
      IF(SEARCH)OTLNNO=LNNOBF
C
      IF(INLNNO.LE.LINMAX) GO TO 140
C
C     WRITE SIX EOLI'S
C
  120 CONTINUE
      DO 130 I=1,6
      CALL CODING(10,0,0,0,0,CDELCT,CDDATA)
  130 CONTINUE
      DO 135 I=1,6
      STFBUF(I)=CDBUF(I)
  135 CONTINUE
      GO TO 400
C
  140 CONTINUE
C
C     TEST FOR ALL WHITE LINE
C
      INELW=(INELCT+32-1)/32
      DO 145 I=1,INELW
      IF(PELBUF(I,INCOD)) 146,145,146
  145 CONTINUE
C
C     LINE IS ALL WHITE
C
      IF(WHITE) GO TO 147
      WHITE=.TRUE.
      GO TO 149
C
C     LINE IS NOT ALL WHITE
C
  146 CONTINUE
      IF(.NOT.WHITE) GO TO 147
      WHITE=.FALSE.
      GO TO 149
C
C     TEST FOR CONSECUTIVE 2-DIM LINES
C
  147 CONTINUE
      IF(KCNT) 148,149,600
  148 STOP 149
  149 CONTINUE
C
C     ONE-DIMENSIONAL CODING
C     WRITE ONE EOL1
C
      CALL CODING(10,0,0,0,0,CDELCT,CDDATA)
C
      PELNR=1
C
C     TEST COLOR OF FIRST ELEMENT
C
      IF(I4B(PELBUF(1,INCOD),1,1).EQ.0) GO TO 150
```

```
C
C     FIRST ELEMENT BLACK; ENCODE 0-LENGTH WHITE RUN
C
      CALL CODELN(0,1,CDELCT,CDDATA)
      POLAR=2
C
C     CALCULATE RUN LENGTH AND ENCODE
C
  150 CONTINUE
      RUN=0
      DO 200 I=1,PELMAX
      PEL=I4B(PELBUF(1,INCOD),I,1)+1
      IF(PEL.EQ.POLAR) GO TO 180
      CALL CODELN(RUN,POLAR,CDELCT,CDDATA)
      IF(.NOT.DIAG) GO TO 170
      WRITE(6,160) RUN,POLAR,CDELCT,CDDATA
  160 FORMAT(4I3)
  170 CONTINUE
      RUN=1
      POLAR=MOD(POLAR+2,2)+1
      GO TO 200
  180 CONTINUE
      RUN=RUN+1
  200 CONTINUE
      CALL CODELN(RUN,POLAR,CDELCT,CDDATA)
      KCNT=K
      IF(.NOT.DIAG) GO TO 210
      WRITE(6,160) RUN,POLAR,CDELCT,CDDATA
      GO TO 210
C
C     TWO-DIMENSIONAL CODING
C
  600 CONTINUE
      STFBIT=STFBIT+1
C
C     WRITE ONE EOL2
C
      CALL CODING(11,0,0,0,0,CDELCT,CDDATA)
C
C     SET A0 TO LEFT EDGE-1 AND POLARITY=WHITE
C
      A0=0
      POL=0
      LEFT=.TRUE.
C
C     DETECT A1
C
  620 CONTINUE
      I=A0+1
      IF(I.GT.PELMAX) GO TO 640
  630 CONTINUE
      PEL=I4B(PELBUF(1,INCOD),I,1)
      IF(PEL.NE.POL) GO TO 640
      I=I+1
      IF(I.LE.PELMAX) GO TO 630
  640 CONTINUE
      A1=I
C
C     DETECT B1
C
      I=A0+1
      IF(I.GT.PELMAX) GO TO 665
      PELM1=I4B(PELBUF(1,INREF),A0,1)
      IF(LEFT) PELM1=0
  650 CONTINUE
      PEL=I4B(PELBUF(1,INREF),I,1)
      IF(PEL.NE.PELM1) GO TO 670
  660 CONTINUE
      PELM1=PEL
      I=I+1
      IF(I.LE.PELMAX) GO TO 650
  665 CONTINUE
      B1=I
      GO TO 710
  670 CONTINUE
      IF(PEL.NE.POL) GO TO 690
      GO TO 660
  690 CONTINUE
      B1=I
      POL=PEL
C
C     DETECT B2
```

```
C
      I=B1+1
      IF(I.GT.PELMAX) GO TO 710
  700 CONTINUE
      PEL=I4B(PELBUF(1,INREF),I,1)
      IF(PEL.NE.PCL) GO TO 720
      I=I+1
      IF(I.LE.PELMAX) GO TO 700
  710 CONTINUE
      B2=I
      GO TO 730
  720 CONTINUE
      B2=I
      PCL=PEL
  730 CONTINUE
      IF(.NOT.LEFT) POLAR=I4B(PELBUF(1,INCOD),A0,1)+1
      IF(.NOT.LEFT) GO TO 740
      POLAR=1
      A0=1
      LEFT=.FALSE.
  740 CONTINUE
C
C   TEST FOR PASS MODE
C
      IF(B2.GE.A1) GO TO 750
C
C   PASS MODE CODING (CAN'T END A LINE IN PASS MODE;NEW A0 MUST HAVE
C                         SAME POLARITY AS B2)
C
      CALL CODING(1,0,0,0,0,CDELCT,CDDATA)
      A0=B2
      GO TO 620
  750 CONTINUE
C
      MAB=IABS(A1-B1)
      IF(MAB-3) 751,751,799
C
C   CALCULATE LENGTH OF VERTICAL AND HORIZONTAL MODES
C
C
C   DO HORIZONTAL FIRST
C
  751 CONTINUE
      A1MA0=A1-A0
      HORIZ=0
      IF(A1MA0.LE.63) GO TO 755
      HORIZ=CODE(1,A1MA0/64+64,POLAR)
 755  CONTINUE
      TEMP=MOD(A1MA0,64)+1
      HORIZ=HORIZ+CODE(1,TEMP,POLAR)+3
C
C   CALCULATE VERTICAL LENGTH
C
      MAB=IABS(A1-B1)+1
      GO TO (760,770,780,780),MAB
      STOP 760
C
  760 VERTIC=1
      GO TO 790
  770 VERTIC=3
      GO TO 790
  780 VERTIC=6
      IF(B1-A1.EQ.3) VERTIC=VERTIC+1
  790 CONTINUE
      IF(HORIZ.GT.VERTIC) GO TO 835
C
C   CODE BY HORIZONTAL MODE;FIRST DETECT A2
C
  799 CONTINUE
      I=A1+1
      IF(I.GT.PELMAX) GO TO 810
C
C   CALCULATE POLARITY OF A1
C
      POL=I4B(PELBUF(1,INCOD),A1,1)
  800 CONTINUE
      PEL=I4B(PELBUF(1,INCOD),I,1)
      IF(PEL.NE.POL) GO TO 820
      I=I+1
      IF(I.LE.PELMAX) GO TO 800
  810 A2=PELMAX+1
      GO TO 930
```

```
  820 CONTINUE
      A2=I
  830 CONTINUE
      CALL CODENG(2,POLAR,A0,A1,A2,CDELCT,CDDATA)
      A0=A2
      GO TO 960
C
C   CODE BY VERTICAL MODE
C
  835 CONTINUE
      IF(A1-B1) 850,840,840
C
  840 CALL CODENG(A1-B1+3,0,0,0,0,CDELCT,CDDATA)
      GO TO 950
  850 CONTINUE
      CALL CODENG(B1-A1+6,0,0,0,0,CDELCT,CDDATA)
  950 CONTINUE
      A0=A1
C
C   TEST FOR END OF LINE
C
  960 CONTINUE
      IF(A0.GT.PELMAX) GO TO 210
      POL=I4B(PELBUF(1,INCOD),A0,1)
      GO TO 620
  210 CONTINUE
C
C     SWITCH CODE & REFERENCE LINES
C
      TEMP=INREF
      INREF=INCOD
      INCOD=TEMP
C
      CDELW=(CDELCT+32-1)/32
      DO 300 I=2,CDELW
      STFBUF(I)=CDBUF(I)
  300 CONTINUE
C   SAVE LINE LENGTH(DATA BITS + EOL)
C
      STAT(INLNCT)=CDDATA+13
C
C   CHECK CODED LINE LENGTH
C
      FILL=CMPMAX-(CDELCT-32)
      IF(FILL) 400,400,250
C
C   CODE LINE TOO SHORT; FILL IT TO CMPMAX
  250 CONTINUE
      CDELCT=CDELCT+FILL
C
C   ACCUMULATE STATISTICS AND ERROR CORRUPT
C
  400 CONTINUE
      IF(ERRMOD.EQ.NN) GO TO 390
C
C   ERROR CORRUPT
C
  350 CONTINUE
      ERRBIT=ERRORS(ERRPNT)-ERROFF-ICDEL
      IF(ERRBIT.LE.0) GO TO 360
      IF(ERRBIT.GT.CDELCT-32) GO TO 390
C
C   ERROR IN RANGE OF CODED LINE; CHANGE APPROPRIATE BIT
C
      BIT=I4B(STFBUF,ERRBIT+32,1)
      BIT=MOD(BIT+1,2)
      CALL MI2B(BIT,STFBUF,ERRBIT+32,1)
      ERRCNT=ERRCNT+1
C
C   INCREMENT ERROR LIST POINTER
C
  360 CONTINUE
      ERRPNT=ERRPNT+1
      IF(ERRPNT.LE.ERRLIM) GO TO 350
C
C   ERROR LIST EXHAUSTED
C
      ERRPNT=ERRPNT-1
      WRITE(6,370) ERRPNT,ERRORS(ERRPNT)
  370 FORMAT('0ERROR LIST EXHAUSTED AT',I10,'TH ERROR;'/
     *       ' LAST ERROR OCCURRED AT',I10,' BITS')
      ERRMOD=NN
```

```
C
C   COMPUTE STATISTICS
C
  390 CONTINUE
      TCDEL=TCDEL+CDELCT-32
      TCDATA=TCDATA+CDDATA
      IF(DIAG) WRITE(6,160) INLNCT, CDDATA
C
      IF (.NOT.DIAG) GO TO 460
      CDELW=(CDELCT+32-1)/32
      WRITE(6,450) (CDBUF(I),I=1,CDELW)
      WRITE(6,450) (STFBUF(I),I=1,CDELW)
  450 FORMAT(6Z12)
  460 CONTINUE
      RETURN
C
  500 CONTINUE
      CALL EXIT
C
      E N D
      SUBROUTINE CODENG(MODE,POLAR,A,B,C,CDELCT,CDDATA)
      IMPLICIT INTEGER(A-Z)
      COMMON/BUFF/PELBUF(60,2),CDBUF(240),OTBUF(60,2),
     *            STFBUF(240), STAT(3000)
      COMMON/HUFF/CODE(3,92,2),CODERD(3,11)
      COMMON/ERAY/ERRORS(2500)
C
C**************** BEGIN PROGRAM *********************
C
      CALL MIBO(CODERD(3,MODE),CDBUF,CDELCT+1,CODERD(1,MODE))
      CDELCT=CDELCT+CODERD(1,MODE)
      GO TO (100,200,100,100,100,100,100,100,100,800,800) ,MODE
C
C   MODE        1   2   3   4   5   6   7   8   9  10  11
C
      STOP 129
C
C   PASS MODE(1),VERTICAL MODE(A1B1=0(3),A1B1=1(4,7),=2(5,8),=3(6,9)
C
 100  CONTINUE
      CDDATA=CDDATA+CODERD(1,MODE)
      RETURN
C
C   HORIZONTAL MODE(2)
C
  200 CONTINUE
      CDDATA=CDDATA+CODERD(1,MODE)
      CALL CODELN(B-A,POLAR,CDELCT,CDDATA)
      NEWPOL=MOD(POLAR+2,2)+1
      CALL CODELN(C-B,NEWPOL,CDELCT,CDDATA)
      RETURN
C
C   ADD EOL1 OR EOL2 TO LINE (10,11)
C
 800  CONTINUE
      RETURN
      END
      SUBROUTINE CNEENG(INDEX,COLOR,STATUS,L)
      IMPLICIT INTEGER(A-Z)
C******* LABELED COMMON /G32BIT/ *******
C
      COMMON /G32BIT/MASK(32),COMASK(32),LIBIT(32),LZBIT(32)
      INTEGER MASK,COMASK,LIBIT,LZBIT
C
      COMMON/BUFF/PELBUF(60,2),CDBUF(240),OTBUF(60,2),
     *            STFBUF(240), STAT(3000)
      COMMON/HUFF/CODE(3,92,2),CODERD(3,11)
      COMMON/ERAY/ERRORS(2500)
C********************** FILE DEFINITIONS *******************
C
      COMMON/FILES/TERM,LPFIL,PELFIL,CTFIL,EFFIL
C
C********************** LABELLED COMMON VARIABLES ********************
C
      COMMON/IVAR/PELMAX,VRES,EPHASE,CMPMAX,ERRMOD, INMAX,K
      COMMON/PVAR/INLNNO,OTLNNO,CTELW,INELP,CDELP,OTELP,CDELW,
     *            CDELCT,INELCT,TCDATA,TCDEL,ERRPNT,ERROFF,ERRLIM,
     *            ERRCNT,INLNCT,CONSEC,ONECNT,LNNUSF,KCNT,
     *            INCOD,INREF,OTCOD,OTREF,STFBIT
      COMMON/ICHAR/OO,II,MM,TT,NN,YY
      COMMON/LOGIC/SEARCH,DIAG,SYNC,WRITE,ZERO,LEFT,CHCOL,ONE,WHITE
      LOGICAL SEARCH,DIAG,SYNC,WRITE,ZERO,LEFT,CHCOL,ONE,WHITE
```

```
C
C    BEGIN DECODE LOOP: RETRIEVE NEXT CODE WORD LENGTH (L)
C
 1000 CONTINUE
 1002 LENBIT=CODE(1,INDEX,COLOR)
      CALL GETLE(LENBIT,MODE,LBITS,L)
      IF(DIAG) WRITE(6,1003) LENBIT,MODE,LBITS,L
 1003 FORMAT(2I6,Z8,I6)
      GO TO (1040,1200,1205,1190), MODE
      STOP 1040
 1040 CONTINUE
      IF(LBITS.EQ.CODE(3,INDEX,COLOR)) GO TO 1100
C
C    NO MATCH; ADVANCE CODE WORD INDEX VIA DECODE THREAD
C
      INDEX=CODE(2,INDEX,COLOR)
      IF(INDEX.GE.93) GO TO 1190
      IF(CODE(1,INDEX,COLOR).EQ.LENBIT) GO TO 1040
C
C    CODE WORD LONGER; FROM THE TOP
C
      GO TO 1002
C
C    MATCH FOUND
C
 1100 CONTINUE
      CDELP=CDELP+1
C
C    NOT AN EOL
C
C
C    TEST FOR MAKE UP OR TERMINATING CODE
C
      RUNLEN=INDEX-1
      IF(INDEX.GE.65) RUNLEN=(INDEX-64)*64
      IF(RUNLEN.EQ.0) GO TO 1160
      IF(COLOR.EQ.1) GO TO 1155
      IF(RUNLEN.LT.0) STOP 1100
C
C    ADD BLACK RUN TO OUTPUT BUFFER
C
      DO 1150 I=1,RUNLEN
      CALL MI23(COLOR-1,OTBUF(1,OTCOD),CTELP,1)
      CTELP=CTELP+1
      IF(CTELP-1.GT.PELMAX) GO TO 1180
 1150 CONTINUE
      GO TO 1160
C
C    ADD WHITE RUN TO OUTPUT BUFFER (BY DEFAULT)
C
 1155 CONTINUE
      CTELP=CTELP+RUNLEN
      IF(CTELP-1.GT.PELMAX) GO TO 1180
C
C OUTPUT LINE LESS THAN OR EQUAL TO MAX SPECIFIED
C
 1160 CONTINUE
      IF(INDEX.LT.65) GO TO 1170
      INDEX=3
      GO TO 1000
C
C    RUN ADDED TO OUTPUT LINE; LENGTH LESS THAN OR EQUAL TO PELMAX (1)
C
 1170 CONTINUE
      CHCOL=.TRUE.
      STATUS=1
      RETURN
C
C    RUN ADDED UNTIL PELMAX EXCEEDED; LINE TOO LONG (2)
C
 1180 CONTINUE
      IF(DIAG) WRITE(6,1185) (OTBUF(I,OTCOD),I=1,60)
 1185 FORMAT(6Z10)
      STATUS=2
      RETURN
C
C    NO MATCH FOUND IN CODE TABLE (3)
C
 1190 CONTINUE
      STATUS=3
      RETURN
C
```

```
C     EOL1 DETECTED (4)
C
 1200 CONTINUE
      STATUS=4
      RETURN
C
C     EOL2 DETECTED (5)
C
 1205 CONTINUE
      STATUS=5
      RETURN
      E N D
      SUBROUTINE  TWOENG(INDEX,COLOR,STATUS,L)
      IMPLICIT  INTEGER(A-Z)
C******* LABELED COMMON /G32BIT/ *******
C
      COMMON /G32BIT/MASK(32),COMASK(32),LIBIT(32),LZBIT(32)
      INTEGER MASK,COMASK,LIBIT,LZBIT
C
      COMMON/BUFF/PELBUF(60,2),CDBUF(240),CTBUF(60,2),
     *           STFBUF(240),  STAT(3000)
      COMMON/HUFF/CODE(3,92,2),CODERD(3,11)
      COMMON/ERAY/ERRORS(2500)
C********************** FILE DEFINITIONS ********************
C
      COMMON/FILES/TERM,LPFIL,PELFIL,CTFIL,ERFIL
C
C********************** LABELLED COMMON VARIABLES **********************
C
      COMMON/IVAR/PELMAX,VRES,EPHASE,CMPMAX,ERRMOD,LINMAX,K
      COMMON/PVAR/INLNNO,OTLNNO,OTELW,INELP,CDELP,OTELP,CDELW,
     *           CDELCT, INELCT,TCDATA,TCDEL,ERRPNT,ERROFF,ERRLIM,
     *           ERRCNT,INLNCT,CONSEC,CNECNT,LNNGBF,KCNT,
     *           INCOD,INREF,OTCOD,OTREF,STFBIT
      COMMON/ICHAR/OO,II,MM,TT,NN,YY
      COMMON/LOGIC/SEARCH,DIAG,SYNC,WRITE,ZERO,LEFT,CHCOL,ONE,WHITE
      LOGICAL SEARCH,DIAG,SYNC,WRITE,ZERO,LEFT,CHCOL,ONE,WHITE
C
C     BEGIN DECODE LOOP; RETRIEVE NEXT CODE WORD LENGTH (L)
C
 1000 CONTINUE
 1002 LENBIT=CODERD(1,INDEX)
      CALL GETLE(LENBIT,MODE,LBITS,L)
      IF(DIAG) WRITE(6,1003) LENBIT,MODE,LBITS,L
 1003 FORMAT(2I6,2I2,I6)
      GO TO (1040,1200,1205,1190), MODE
      STOP 1040
 1040 CONTINUE
      IF(LBITS.EQ.CODERD(3,INDEX)) GO TO 1100
C
C     NO MATCH; ADVANCE CODE WORD INDEX VIA DECODE THREAD
C
      INDEX=CODERD(2,INDEX)
      IF(INDEX.GE.12) GO TO 1190
      IF(CODERD(1,INDEX).EQ.LENBIT) GO TO 1040
C
C     CODE WORD LONGER; FROM THE TOP
C
      GO TO 1002
C
C     MATCH FOUND
C
 1100 CONTINUE
      CDELP=CDELP+L
C
C     NOT AN EOL
C
C
C     FIND B1 AND B2
C
      A0=OTELP
      IF(OTELP.EQ.1) A0=0
      POL=COLOR-1
C
C     DETECT B1
C
      I=A0+1
      IF(I.GT.PELMAX) GO TO 65
      PELM1=0
      IF(A0.EQ.0) GO TO 50
      PELM1=IAB(OTBUF(1,OTREF),A0,1)
   50 CONTINUE
```

```
      PEL=I4B(OTBUF(1,OTREF),I,1)
      IF(PEL.NE.PELM1) GO TO 70
   60 CONTINUE
      PELM1=PEL
      I=I+1
      IF(I.LE.PELMAX) GO TO 50
   65 CONTINUE
      B1=I
      GO TO 92
   70 CONTINUE
      IF(PEL.NE.POL) GO TO 90
      GO TO 60
   90 CONTINUE
      B1=I
      POL=PEL
C
C   DETECT B2
C
      I=B1+1
      IF(I.GT.PELMAX) GO TO 92
   91 CONTINUE
      PEL=I4B(OTBUF(1,OTREF),I,1)
      IF(PEL.NE.POL) GO TO 92
      I=I+1
      IF(I.LE.PELMAX) GO TO 91
   92 CONTINUE
      B2=I
      GO TO (100,200,300,400,400,400,600,600,600),INDEX
      STOP 100
C
C   PASS MODE
C
  100 CONTINUE
      RUNLEN=32-OTELP
      CHCOL=.FALSE.
      GO TO (1155,1145),COLOR
C
C   HORIZONTAL MODE
C
  200 CONTINUE
      ENTRY=3
      CALL ONEENG(ENTRY,COLOR,STATE,L)
      GO TO (210,1190,1190,1200,1205),STATE
  210 CONTINUE
      COLOR=MOD(COLOR+2,2)+1
      ENTRY=3
      CALL ONEENG(ENTRY,COLOR,STATE,L)
      GO TO (220,1190,1190,1200,1205),STATE
  220 CONTINUE
      CHCOL=.TRUE.
      GO TO 1160
C
C   VERTICAL MODE A1B1=0
C
  300 CONTINUE
      RUNLEN=31-OTELP
      CHCOL=.TRUE.
      GO TO (1155,1145),COLOR
C
C   VERTICAL MODE VR1 A1B1=1,2,3
C
  400 CONTINUE
      RUNLEN=B1-OTELP+INDEX-3
      CHCOL=.TRUE.
      GO TO (1155,1145),COLOR
C
C
C   VERTICAL MODE LEFT VL1 A1B1=1,2,3
C
  600 CONTINUE
      RUNLEN=31-OTELP-(INDEX-6)
      CHCOL=.TRUE.
      GO TO (1155,1145),COLOR
C
C   ADD BLACK RUN TO OUTPUT BUFFER
C
 1145 CONTINUE
      IF(RUNLEN) 1190,1160,1147
 1147 CONTINUE
      DO 1150 I=1,RUNLEN
      CALL MI23(COLOR-1,OTBUF(1,OTCOD),CTELP,1)
      CTELP=CTELP+1
```

```
      IF(OTELP-1.GT.PELMAX) GO TO 1180                                  00011
 1150 CONTINUE                                                          00011
      GO TO 1160                                                        00011
C                                                                       00011
C   ADD WHITE RUN TO OUTPUT BUFFER (BY DEFAULT)                         00011
C                                                                       00011
 1155 CONTINUE                                                          00011
      IF(RUNLEN.LT.0) GO TO 1190                                        00011
      OTELP=OTELP+RUNLEN                                                00011
      IF(OTELP-1.GT.PELMAX) GO TO 1180                                  00011
C                                                                       00011
C   RUN ADDED TO OUTPUT LINE: LENGTH LESS THAN OR EQUAL TO PELMAX (1)   00011
C                                                                       00011
 1160 CONTINUE                                                          00011
      STATUS=1                                                          00011
      RETURN                                                            00011
C                                                                       00011
C   RUN ADDED UNTIL PELMAX EXCEEDED; LINE TOO LONG (2)                  00011
C                                                                       00011
 1180 CONTINUE                                                          00011
      IF(DIAG) WRITE(6,1185) (OTBUF(I,OTCOD),I=1,60)                    00011
 1185 FORMAT(6Z10)                                                      00011
      STATUS=2                                                          00011
      RETURN                                                            00011
C                                                                       00011
C   NO MATCH FOUND IN CODE TABLE (3)                                    00011
C                                                                       00011
 1190 CONTINUE                                                          00011
      STATUS=3                                                          00011
      RETURN                                                            00011
C                                                                       00011
C   EOL1 DETECTED (4)                                                   00011
C                                                                       00011
 1200 CONTINUE                                                          00011
      STATUS=4                                                          00011
      RETURN                                                            00011
C                                                                       00011
C   EOL2 DETECTED (5)                                                   00011
C                                                                       00011
 1205 CONTINUE                                                          00011
      STATUS=5                                                          00011
      RETURN                                                            00011
      END                                                               00011
      BLOCK DATA                                                        00011
C                                                                       00011
      IMPLICIT INTEGER(A-Z)                                             00011
C*********************** FILE DEFINITIONS *********************         00011
C                                                                       00011
      COMMON/FILES/TERM,LPFIL,PELFIL,OTFIL,ERFIL                        00011
C                                                                       00011
      COMMON/BUFF/PELBUF(60,2),CDBUF(240),OTBUF(60,2),                  00011
     *            STFBUF(240), STAT(3000)                               00012
      COMMON/HUFF/CODE(3,92,2),CODERD(3,11)                             00012
      COMMON/ERAY/ERRORS(2500)                                          00012
C********************** LABELLED COMMON VARIABLES ********************** 00012
C                                                                       00012
      COMMON/IVAR/PELMAX,VRES,EPHASE,CMPMAX,ERRMOD,LINMAX,K             00012
      COMMON/PVAR/INLNNO,OTLNNO,OTELW,INELP,CDELP,OTELP,CDELW,          00012
     *            CDELCT,INELCT,TCDATA,TCDEL,ERRPNT,ERROFF,ERRLIM,      00012
     *            ERRCNT,INLNCT,CONSEC,ONECNT,LNNOBF,KCNT,              00012
     *            INCOD,INREF,OTCOD,OTREF,STFBIT                        00012
      COMMON/ICHAR/DD,II,MM,TT,NN,YY                                    00012
      COMMON/LOGIC/SEARCH,DIAG,SYNC,WRITE,ZERO,LEFT,CHCOL,ONE,WHITE     00012
      LOGICAL SEARCH,DIAG,SYNC,WRITE,ZERO,LEFT,CHCOL,ONE,WHITE          00012
C                                                                       00012
      DATA TERM,LPFIL,PELFIL,OTFIL,ERFIL/5,6,1,2,3/                     00012
      DATA DD,II,MM,TT,NN,YY/'D','I','M','T','N','Y'/                   00012
      DATA PELMAX,VRES,EPHASE,CMPMAX,ERRMOD,LINMAX/1728,2,0,96,'T',3000/00012
      DATA K/2/                                                         00012
      DATA DIAG/.FALSE./                                                00012
C                                                                       00012
       DATA CODE(1,  1,1),CODE(2,  1,1),CODE(3,  1,1)/ 8, 70,Z0035/     00013
       DATA CODE(1,  2,1),CODE(2,  2,1),CODE(3,  2,1)/ 6, 90,Z0007/     00013
       DATA CODE(1,  3,1),CODE(2,  3,1),CODE(3,  3,1)/ 4,  4,Z0007/     00013
       DATA CODE(1,  4,1),CODE(2,  4,1),CODE(3,  4,1)/ 4,  5,Z0008/     00013
       DATA CODE(1,  5,1),CODE(2,  5,1),CODE(3,  5,1)/ 4,  6,Z000B/     00013
       DATA CODE(1,  6,1),CODE(2,  6,1),CODE(3,  6,1)/ 4,  7,Z000C/     00013
       DATA CODE(1,  7,1),CODE(2,  7,1),CODE(3,  7,1)/ 4,  8,Z000E/     00013
       DATA CODE(1,  8,1),CODE(2,  8,1),CODE(3,  8,1)/ 4,  9,Z000F/     00013
       DATA CODE(1,  9,1),CODE(2,  9,1),CODE(3,  9,1)/ 5, 10,Z0013/     00013
       DATA CODE(1, 10,1),CODE(2, 10,1),CODE(3, 10,1)/ 5, 11,Z0014/     00013
       DATA CODE(1, 11,1),CODE(2, 11,1),CODE(3, 11,1)/ 5, 12,Z0007/     00013
```

```
DATA CODE(1, 12,1),CODE(2, 12,1),CODE(3, 12,1)/ 5, 65,Z0008/          00013
DATA CODE(1, 13,1),CODE(2, 13,1),CODE(3, 13,1)/ 6, 14,Z0008/          00013
DATA CODE(1, 14,1),CODE(2, 14,1),CODE(3, 14,1)/ 6, 15,Z0003/          00013
DATA CODE(1, 15,1),CODE(2, 15,1),CODE(3, 15,1)/ 6, 16,Z0034/          00013
DATA CODE(1, 16,1),CODE(2, 16,1),CODE(3, 16,1)/ 6, 17,Z0035/          00013
DATA CODE(1, 17,1),CODE(2, 17,1),CODE(3, 17,1)/ 6, 18,Z002A/          00013
DATA CODE(1, 18,1),CODE(2, 18,1),CODE(3, 18,1)/ 6, 19,Z002B/          00013
DATA CODE(1, 19,1),CODE(2, 19,1),CODE(3, 19,1)/ 7, 20,Z0027/          00013
DATA CODE(1, 20,1),CODE(2, 20,1),CODE(3, 20,1)/ 7, 21,Z000C/          00014
DATA CODE(1, 21,1),CODE(2, 21,1),CODE(3, 21,1)/ 7, 22,Z0008/          00014
DATA CODE(1, 22,1),CODE(2, 22,1),CODE(3, 22,1)/ 7, 23,Z0017/          00014
DATA CODE(1, 23,1),CODE(2, 23,1),CODE(3, 23,1)/ 7, 24,Z0003/          00014
DATA CODE(1, 24,1),CODE(2, 24,1),CODE(3, 24,1)/ 7, 25,Z0004/          00014
DATA CODE(1, 25,1),CODE(2, 25,1),CODE(3, 25,1)/ 7, 26,Z0020/          00014
DATA CODE(1, 26,1),CODE(2, 26,1),CODE(3, 26,1)/ 7, 27,Z002B/          00014
DATA CODE(1, 27,1),CODE(2, 27,1),CODE(3, 27,1)/ 7, 28,Z0013/          00014
DATA CODE(1, 28,1),CODE(2, 28,1),CODE(3, 28,1)/ 7, 29,Z0024/          00014
DATA CODE(1, 29,1),CODE(2, 29,1),CODE(3, 29,1)/ 7, 68,Z0018/          00014
DATA CODE(1, 30,1),CODE(2, 30,1),CODE(3, 30,1)/ 8, 31,Z0002/          00014
DATA CODE(1, 31,1),CODE(2, 31,1),CODE(3, 31,1)/ 8, 32,Z0003/          00014
DATA CODE(1, 32,1),CODE(2, 32,1),CODE(3, 32,1)/ 8, 33,Z001A/          00014
DATA CODE(1, 33,1),CODE(2, 33,1),CODE(3, 33,1)/ 8, 34,Z001B/          00014
DATA CODE(1, 34,1),CODE(2, 34,1),CODE(3, 34,1)/ 8, 35,Z0012/          00014
DATA CODE(1, 35,1),CODE(2, 35,1),CODE(3, 35,1)/ 8, 36,Z0013/          00014
DATA CODE(1, 36,1),CODE(2, 36,1),CODE(3, 36,1)/ 8, 37,Z0014/          00014
DATA CODE(1, 37,1),CODE(2, 37,1),CODE(3, 37,1)/ 8, 38,Z0015/          00014
DATA CODE(1, 38,1),CODE(2, 38,1),CODE(3, 38,1)/ 8, 39,Z0016/          00014
DATA CODE(1, 39,1),CODE(2, 39,1),CODE(3, 39,1)/ 8, 40,Z0017/          00014
DATA CODE(1, 40,1),CODE(2, 40,1),CODE(3, 40,1)/ 8, 41,Z0028/          00014
DATA CODE(1, 41,1),CODE(2, 41,1),CODE(3, 41,1)/ 8, 42,Z0029/          00014
DATA CODE(1, 42,1),CODE(2, 42,1),CODE(3, 42,1)/ 8, 43,Z002A/          00014
DATA CODE(1, 43,1),CODE(2, 43,1),CODE(3, 43,1)/ 8, 44,Z002B/          00014
DATA CODE(1, 44,1),CODE(2, 44,1),CODE(3, 44,1)/ 8, 45,Z002C/          00014
DATA CODE(1, 45,1),CODE(2, 45,1),CODE(3, 45,1)/ 8, 46,Z002D/          00014
DATA CODE(1, 46,1),CODE(2, 46,1),CODE(3, 46,1)/ 8, 47,Z0004/          00014
DATA CODE(1, 47,1),CODE(2, 47,1),CODE(3, 47,1)/ 8, 48,Z0005/          00014
DATA CODE(1, 48,1),CODE(2, 48,1),CODE(3, 48,1)/ 8, 49,Z000A/          00014
DATA CODE(1, 49,1),CODE(2, 49,1),CODE(3, 49,1)/ 8, 50,Z000B/          00014
DATA CODE(1, 50,1),CODE(2, 50,1),CODE(3, 50,1)/ 8, 51,Z0052/          00014
DATA CODE(1, 51,1),CODE(2, 51,1),CODE(3, 51,1)/ 8, 52,Z0053/          00014
DATA CODE(1, 52,1),CODE(2, 52,1),CODE(3, 52,1)/ 8, 53,Z0054/          00014
DATA CODE(1, 53,1),CODE(2, 53,1),CODE(3, 53,1)/ 8, 54,Z0055/          00014
DATA CODE(1, 54,1),CODE(2, 54,1),CODE(3, 54,1)/ 8, 55,Z0024/          00014
DATA CODE(1, 55,1),CODE(2, 55,1),CODE(3, 55,1)/ 8, 56,Z0025/          00014
DATA CODE(1, 56,1),CODE(2, 56,1),CODE(3, 56,1)/ 8, 57,Z0058/          00014
DATA CODE(1, 57,1),CODE(2, 57,1),CODE(3, 57,1)/ 8, 58,Z0059/          00014
DATA CODE(1, 58,1),CODE(2, 58,1),CODE(3, 58,1)/ 8, 59,Z005A/          00014
DATA CODE(1, 59,1),CODE(2, 59,1),CODE(3, 59,1)/ 8, 60,Z005B/          00014
DATA CODE(1, 60,1),CODE(2, 60,1),CODE(3, 60,1)/ 8, 61,Z004A/          00014
DATA CODE(1, 61,1),CODE(2, 61,1),CODE(3, 61,1)/ 8, 62,Z004B/          00014
DATA CODE(1, 62,1),CODE(2, 62,1),CODE(3, 62,1)/ 8, 63,Z0032/          00014
DATA CODE(1, 63,1),CODE(2, 63,1),CODE(3, 63,1)/ 8, 64,Z0033/          00014
DATA CODE(1, 64,1),CODE(2, 64,1),CODE(3, 64,1)/ 8, 69,Z0034/          00014
DATA CODE(1, 65,1),CODE(2, 65,1),CODE(3, 65,1)/ 5, 66,Z001B/          00014
DATA CODE(1, 66,1),CODE(2, 66,1),CODE(3, 66,1)/ 5, 67,Z0012/          00014
DATA CODE(1, 67,1),CODE(2, 67,1),CODE(3, 67,1)/ 6,  2,Z0017/          00014
DATA CODE(1, 68,1),CODE(2, 68,1),CODE(3, 68,1)/ 7, 30,Z0037/          00014
DATA CODE(1, 69,1),CODE(2, 69,1),CODE(3, 69,1)/ 8,  1,Z0036/          00014
DATA CODE(1, 70,1),CODE(2, 70,1),CODE(3, 70,1)/ 8, 71,Z0037/          00014
DATA CODE(1, 71,1),CODE(2, 71,1),CODE(3, 71,1)/ 8, 72,Z0064/          00014
DATA CODE(1, 72,1),CODE(2, 72,1),CODE(3, 72,1)/ 8, 73,Z0065/          00014
DATA CODE(1, 73,1),CODE(2, 73,1),CODE(3, 73,1)/ 8, 74,Z0066/          00014
DATA CODE(1, 74,1),CODE(2, 74,1),CODE(3, 74,1)/ 8, 75,Z0067/          00014
DATA CODE(1, 75,1),CODE(2, 75,1),CODE(3, 75,1)/ 9, 76,Z00CC/          00014
DATA CODE(1, 76,1),CODE(2, 76,1),CODE(3, 76,1)/ 9, 77,Z00CD/          00014
DATA CODE(1, 77,1),CODE(2, 77,1),CODE(3, 77,1)/ 9, 78,Z00D2/          00014
DATA CODE(1, 78,1),CODE(2, 78,1),CODE(3, 78,1)/ 9, 79,Z00D3/          00014
DATA CODE(1, 79,1),CODE(2, 79,1),CODE(3, 79,1)/ 9, 80,Z00D4/          00014
DATA CODE(1, 80,1),CODE(2, 80,1),CODE(3, 80,1)/ 9, 81,Z00D5/          00014
DATA CODE(1, 81,1),CODE(2, 81,1),CODE(3, 81,1)/ 9, 82,Z00D6/          00014
DATA CODE(1, 82,1),CODE(2, 82,1),CODE(3, 82,1)/ 9, 83,Z00D7/          00014
DATA CODE(1, 83,1),CODE(2, 83,1),CODE(3, 83,1)/ 9, 84,Z00D8/          00014
DATA CODE(1, 84,1),CODE(2, 84,1),CODE(3, 84,1)/ 9, 85,Z00D9/          00014
DATA CODE(1, 85,1),CODE(2, 85,1),CODE(3, 85,1)/ 9, 86,Z00DA/          00014
DATA CODE(1, 86,1),CODE(2, 86,1),CODE(3, 86,1)/ 9, 87,Z00DB/          00014
DATA CODE(1, 87,1),CODE(2, 87,1),CODE(3, 87,1)/ 9, 88,Z0098/          00014
DATA CODE(1, 88,1),CODE(2, 88,1),CODE(3, 88,1)/ 9, 89,Z0099/          00014
DATA CODE(1, 89,1),CODE(2, 89,1),CODE(3, 89,1)/ 9, 91,Z009A/          00014
DATA CODE(1, 90,1),CODE(2, 90,1),CODE(3, 90,1)/ 6, 13,Z0018/          00014
DATA CODE(1, 91,1),CODE(2, 91,1),CODE(3, 91,1)/ 9, 92,Z009B/          00014
DATA CODE(1, 92,1),CODE(2, 92,1),CODE(3, 92,1)/13, 93,Z0003/          00014
DATA CODE(1,  1,2),CODE(2,  1,2),CODE(3,  1,2)/10, 65,Z0037/          00014
```

```
DATA CODE(1,  2,2),CODE(2,  2,2),CODE(3,  2,2)/ 3,  6,Z0002/
DATA CODE(1,  3,2),CODE(2,  3,2),CODE(3,  3,2)/ 2,  4,Z0003/
DATA CODE(1,  4,2),CODE(2,  4,2),CODE(3,  4,2)/ 2,  5,Z0002/
DATA CODE(1,  5,2),CODE(2,  5,2),CODE(3,  5,2)/ 3,  2,Z0003/
DATA CODE(1,  6,2),CODE(2,  6,2),CODE(3,  6,2)/ 4,  7,Z0003/
DATA CODE(1,  7,2),CODE(2,  7,2),CODE(3,  7,2)/ 4,  8,Z0002/
DATA CODE(1,  8,2),CODE(2,  8,2),CODE(3,  8,2)/ 5,  9,Z0003/
DATA CODE(1,  9,2),CODE(2,  9,2),CODE(3,  9,2)/ 6, 10,Z0005/
DATA CODE(1, 10,2),CODE(2, 10,2),CODE(3, 10,2)/ 6, 11,Z0004/
DATA CODE(1, 11,2),CODE(2, 11,2),CODE(3, 11,2)/ 7, 12,Z0004/
DATA CODE(1, 12,2),CODE(2, 12,2),CODE(3, 12,2)/ 7, 13,Z0005/
DATA CODE(1, 13,2),CODE(2, 13,2),CODE(3, 13,2)/ 7, 14,Z0007/
DATA CODE(1, 14,2),CODE(2, 14,2),CODE(3, 14,2)/ 8, 15,Z0004/
DATA CODE(1, 15,2),CODE(2, 15,2),CODE(3, 15,2)/ 6, 16,Z0007/
DATA CODE(1, 16,2),CODE(2, 16,2),CODE(3, 16,2)/ 9, 17,Z0018/
DATA CODE(1, 17,2),CODE(2, 17,2),CODE(3, 17,2)/10, 18,Z0017/
DATA CODE(1, 18,2),CODE(2, 18,2),CODE(3, 18,2)/10, 19,Z0018/
DATA CODE(1, 19,2),CODE(2, 19,2),CODE(3, 19,2)/10,  1,Z0008/
DATA CODE(1, 20,2),CODE(2, 20,2),CODE(3, 20,2)/11, 21,Z0067/
DATA CODE(1, 21,2),CODE(2, 21,2),CODE(3, 21,2)/11, 22,Z0068/
DATA CODE(1, 22,2),CODE(2, 22,2),CODE(3, 22,2)/11, 23,Z006C/
DATA CODE(1, 23,2),CODE(2, 23,2),CODE(3, 23,2)/11, 24,Z0037/
DATA CODE(1, 24,2),CODE(2, 24,2),CODE(3, 24,2)/11, 25,Z0028/
DATA CODE(1, 25,2),CODE(2, 25,2),CODE(3, 25,2)/11, 26,Z0017/
DATA CODE(1, 26,2),CODE(2, 26,2),CODE(3, 25,2)/11, 27,Z0018/
DATA CODE(1, 27,2),CODE(2, 27,2),CODE(3, 27,2)/12, 28,Z00CA/
DATA CODE(1, 28,2),CODE(2, 28,2),CODE(3, 28,2)/12, 29,Z00CB/
DATA CODE(1, 29,2),CODE(2, 29,2),CODE(3, 29,2)/12, 30,Z00CC/
DATA CODE(1, 30,2),CODE(2, 30,2),CODE(3, 30,2)/12, 31,Z00CD/
DATA CODE(1, 31,2),CODE(2, 31,2),CODE(3, 31,2)/12, 32,Z0068/
DATA CODE(1, 32,2),CODE(2, 32,2),CODE(3, 32,2)/12, 33,Z0069/
DATA CODE(1, 33,2),CODE(2, 33,2),CODE(3, 33,2)/12, 34,Z006A/
DATA CODE(1, 34,2),CODE(2, 34,2),CODE(3, 34,2)/12, 35,Z006B/
DATA CODE(1, 35,2),CODE(2, 35,2),CODE(3, 35,2)/12, 36,Z00D2/
DATA CODE(1, 36,2),CODE(2, 36,2),CODE(3, 36,2)/12, 37,Z00D3/
DATA CODE(1, 37,2),CODE(2, 37,2),CODE(3, 37,2)/12, 38,Z00D4/
DATA CODE(1, 38,2),CODE(2, 38,2),CODE(3, 38,2)/12, 39,Z00D5/
DATA CODE(1, 39,2),CODE(2, 39,2),CODE(3, 39,2)/12, 40,Z00D6/
DATA CODE(1, 40,2),CODE(2, 40,2),CODE(3, 40,2)/12, 41,Z00D7/
DATA CODE(1, 41,2),CODE(2, 41,2),CODE(3, 41,2)/12, 42,Z006C/
DATA CODE(1, 42,2),CODE(2, 42,2),CODE(3, 42,2)/12, 43,Z006D/
DATA CODE(1, 43,2),CODE(2, 43,2),CODE(3, 43,2)/12, 44,Z00DA/
DATA CODE(1, 44,2),CODE(2, 44,2),CODE(3, 44,2)/12, 45,Z00DB/
DATA CODE(1, 45,2),CODE(2, 45,2),CODE(3, 45,2)/12, 46,Z0054/
DATA CODE(1, 46,2),CODE(2, 46,2),CODE(3, 46,2)/12, 47,Z0055/
DATA CODE(1, 47,2),CODE(2, 47,2),CODE(3, 47,2)/12, 48,Z0056/
DATA CODE(1, 48,2),CODE(2, 48,2),CODE(3, 48,2)/12, 49,Z0057/
DATA CODE(1, 49,2),CODE(2, 49,2),CODE(3, 49,2)/12, 50,Z0064/
DATA CODE(1, 50,2),CODE(2, 50,2),CODE(3, 50,2)/12, 51,Z0065/
DATA CODE(1, 51,2),CODE(2, 51,2),CODE(3, 51,2)/12, 52,Z0052/
DATA CODE(1, 52,2),CODE(2, 52,2),CODE(3, 52,2)/12, 53,Z0053/
DATA CODE(1, 53,2),CODE(2, 53,2),CODE(3, 53,2)/12, 54,Z0024/
DATA CODE(1, 54,2),CODE(2, 54,2),CODE(3, 54,2)/12, 55,Z0037/
DATA CODE(1, 55,2),CODE(2, 55,2),CODE(3, 55,2)/12, 56,Z0038/
DATA CODE(1, 56,2),CODE(2, 56,2),CODE(3, 56,2)/12, 57,Z0027/
DATA CODE(1, 57,2),CODE(2, 57,2),CODE(3, 57,2)/12, 58,Z0028/
DATA CODE(1, 58,2),CODE(2, 58,2),CODE(3, 58,2)/12, 59,Z0058/
DATA CODE(1, 59,2),CODE(2, 59,2),CODE(3, 59,2)/12, 60,Z0059/
DATA CODE(1, 60,2),CODE(2, 60,2),CODE(3, 60,2)/12, 61,Z002B/
DATA CODE(1, 61,2),CODE(2, 61,2),CODE(3, 61,2)/12, 62,Z002C/
DATA CODE(1, 62,2),CODE(2, 62,2),CODE(3, 62,2)/12, 63,Z005A/
DATA CODE(1, 63,2),CODE(2, 63,2),CODE(3, 63,2)/12, 64,Z0066/
DATA CODE(1, 64,2),CODE(2, 64,2),CODE(3, 64,2)/12, 65,Z0067/
DATA CODE(1, 65,2),CODE(2, 65,2),CODE(3, 65,2)/10, 20,Z000F/
DATA CODE(1, 66,2),CODE(2, 66,2),CODE(3, 66,2)/12, 67,Z00C8/
DATA CODE(1, 67,2),CODE(2, 67,2),CODE(3, 67,2)/12, 68,Z00C9/
DATA CODE(1, 68,2),CODE(2, 68,2),CODE(3, 68,2)/12, 69,Z005B/
DATA CODE(1, 69,2),CODE(2, 69,2),CODE(3, 69,2)/12, 70,Z0033/
DATA CODE(1, 70,2),CODE(2, 70,2),CODE(3, 70,2)/12, 71,Z0034/
DATA CODE(1, 71,2),CODE(2, 71,2),CODE(3, 71,2)/12, 72,Z0035/
DATA CODE(1, 72,2),CODE(2, 72,2),CODE(3, 72,2)/13, 73,Z006C/
DATA CODE(1, 73,2),CODE(2, 73,2),CODE(3, 73,2)/13, 74,Z006D/
DATA CODE(1, 74,2),CODE(2, 74,2),CODE(3, 74,2)/13, 75,Z004A/
DATA CODE(1, 75,2),CODE(2, 75,2),CODE(3, 75,2)/13, 76,Z004B/
DATA CODE(1, 76,2),CODE(2, 76,2),CODE(3, 75,2)/13, 77,Z004C/
DATA CODE(1, 77,2),CODE(2, 77,2),CODE(3, 77,2)/13, 78,Z004D/
DATA CODE(1, 78,2),CODE(2, 78,2),CODE(3, 78,2)/13, 79,Z0072/
DATA CODE(1, 79,2),CODE(2, 79,2),CODE(3, 79,2)/13, 80,Z0073/
DATA CODE(1, 80,2),CODE(2, 80,2),CODE(3, 80,2)/13, 81,Z0074/
DATA CODE(1, 81,2),CODE(2, 81,2),CODE(3, 81,2)/13, 82,Z0075/
DATA CODE(1, 82,2),CODE(2, 82,2),CODE(3, 82,2)/13, 83,Z0076/
DATA CODE(1, 83,2),CODE(2, 83,2),CODE(3, 83,2)/13, 84,Z0077/
```

```
      DATA CODE(1, 84,2),CODE(2, 84,2),CODE(3, 84,2)/13, 85,Z0052/
      DATA CODE(1, 85,2),CODE(2, 85,2),CODE(3, 85,2)/13, 86,Z0053/
      DATA CODE(1, 86,2),CODE(2, 86,2),CODE(3, 86,2)/13, 87,Z0054/
      DATA CODE(1, 87,2),CODE(2, 87,2),CODE(3, 87,2)/13, 88,Z0055/
      DATA CODE(1, 88,2),CODE(2, 88,2),CODE(3, 88,2)/13, 89,Z005A/
      DATA CODE(1, 89,2),CODE(2, 89,2),CODE(3, 89,2)/13, 90,Z005B/
      DATA CODE(1, 90,2),CODE(2, 90,2),CODE(3, 90,2)/13, 91,Z0064/
      DATA CODE(1, 91,2),CODE(2, 91,2),CODE(3, 91,2)/13, 93,Z0065/
      DATA CODE(1, 92,2),CODE(2, 92,2),CODE(3, 92,2)/13, 93,Z0003/
      DATA CODERD(1,1),CODERD(2,1),CODERD(3,1)/  4,5,Z1/
      DATA CODERD(1,2),CODERD(2,2),CODERD(3,2)/  1,4,Z3/
      DATA CODERD(1,3),CODERD(2,3),CODERD(3,3)/  1,2,Z1/
      DATA CODERD(1,4),CODERD(2,4),CODERD(3,4)/  3,7,Z1/
      DATA CODERD(1,5),CODERD(2,5),CODERD(3,5)/  6,6,Z3/
      DATA CODERD(1,6),CODERD(2,6),CODERD(3,6)/  6,8,Z1/
      DATA CODERD(1,7),CODERD(2,7),CODERD(3,7)/  3,1,Z2/
      DATA CODERD(1,8),CODERD(2,8),CODERD(3,8)/  6,9,Z2/
      DATA CODERD(1,9),CODERD(2,9),CODERD(3,9)/  7,10,Z1/
      DATA CODERD(1,10),CODERD(2,10),CODERD(3,10)/  13,11,Z3/
      DATA CODERD(1,11),CODERD(2,11),CODERD(3,11)/  13,12,Z2/
C
      E N D
      SUBROUTINE CODELN(LENGTH,POLAR,CDELCT,CDDATA)
C
      IMPLICIT INTEGER(A-Z)
      COMMON/BUFF/PELBUF(60,2),CDBUF(240),OTBUF(60,2),
     *        STFBUF(240), STAT(3000)
      COMMON/HUFF/CODE(3,92,2),CODERD(3,11)
      COMMON/ERAY/ERRORS(2500)
C
C******************************** BEGIN PROGRAM **********************
C
C  INITIALIZE MAKE UP CODE, MAKE UP CODE LENGTH
C
      MCCDE=0
      MLENG=0
C
C  CHECK INPUTS
C
      IF(POLAR.LT.1.OR.POLAR.GT.2) CALL EXIT
      IF(LENGTH.LT.0.OR.LENGTH.GT.1728) CALL EXIT
C
      IF(LENGTH.LE.63) GO TO 10
C
C  CALCULATE MAKE UP CODE INDEX, CODE, LENGTH
C   AND WRITE TO CODE LINE
C
      INDEX=LENGTH/64+64
      MCCDE=CODE(3,INDEX,POLAR)
      MLENG=CODE(1,INDEX,POLAR)
      CALL MI2B(MCODE,CDBUF,CDELCT+1,MLENG)
      CDELCT=CDELCT+MLENG
      CDDATA=CDDATA+MLENG
C
C  CALCULATE TERMINATING CODE INDEX, CODE, LENGTH
C   AND ADD TO CODE LINE
C
 10   CONTINUE
      INDEX=MOD(LENGTH,64)+1
      TCCDE=CODE(3,INDEX,POLAR)
      TLENG=CODE(1,INDEX,POLAR)
      CALL MI2B(TCODE,CDBUF,CDELCT+1,TLENG)
      CDELCT=CDELCT+TLENG
      CDDATA=CDDATA+TLENG
C
      RETURN
      E N D
      SUBROUTINE ERRMES(PELBUF,OTBUF,PELMAX,VRES,ERRCNT)
C
      IMPLICIT INTEGER(A-Z)
      REAL ESF
C******* LABELED COMMON /G32BIT/ *******
C
      COMMON /G32BIT/MASK(32),COMASK(32),L1BIT(32),LZBIT(32)
      INTEGER MASK,COMASK,L1BIT,LZBIT
C
C*********************** FILE DEFINITIONS ********************
C
      COMMON/FILES/TERM,LPFIL,PELFIL,CTFIL,ERFIL
C
      DIMENSION PELBUF(60), OTBUF(60)
      COMMON/LOGIC/SEARCH,DIAG
```

```
      LOGICAL SEARCH,DIAG
C
C**************************** BEGIN PROGRAM ********************************
C
      REWIND PELFIL
      REWIND OTFIL
      ERROR=0
      OTELW=(PELMAX+32-1)/32
      OTLNCT=0
C
C   READ AN ERROR FREE LINE
C
  100 CONTINUE
      READ(1,END=600,ERR=800) INLNNO,INELCT,PELBUF
      IF(MOD(INLNNO-1,VRES).NE.0) GO TO 100
C
C   READ AN ERROR-CORRUPTED LINE
C
  200 CONTINUE
      READ(2,END=500,ERR=800) OTLNNO,OTELCT,OTBUF
      OTLNCT=OTLNCT+1
  300 CONTINUE
C
C   COUNT DIFFERENCES BETWEEN TRANSMITTED AND RECEIVED LINES
C
      DO 450 I=1,OTELW
      IF(OTBUF(I).EQ.PELBUF(I)) GO TO 450
      IF(.NOT.DIAG) GO TO 420
      WRITE(6,410) INLNNO,OTLNNO,I,PELBUF(I),OTBUF(I)
  410 FORMAT(3I10,2Z12)
  420 CONTINUE
      DO 440 J=1,32
      IF(I4B(OTBUF(I),J,1).NE.I4B(PELBUF(I),J,1)) ERROR=ERROR+1
  440 CONTINUE
  450 CONTINUE
      IF(OTLNNO-INLNNO) 200,100,580
C
C   ERROR LINE NUMBER GREATER THAN GOOD LINE NUMBER;
C   COUNT DIFFERENCES BETWEEN GOOD AND ALL WHITE LINE
C
  500 CONTINUE
      DO 550 I=1,OTELW
      IF(PELBUF(I).EQ.0) GO TO 550
      IF(.NOT.DIAG) GO TO 520
      WRITE(6,410) INLNNO,OTLNNO,I,PELBUF(I),OTBUF(I)
  520 CONTINUE
      DO 540 J=1,32
      IF(I4B(PELBUF(I),J,1).NE.0) ERROR=ERROR+1
  540 CONTINUE
  550 CONTINUE
C
  580 READ(1,END=600,ERR=800) INLNNO,INELCT,PELBUF
      IF(MOD(INLNNO-1,VRES).NE.0) GO TO 580
C
      GO TO 300
C
C   CALCULATE ERROR SENSITIVITY FACTOR
C
  600 CONTINUE
      ESF=0.
      IF(ERRCNT.LE.0) GO TO 650
      ESF=FLOAT(ERROR)/FLOAT(ERRCNT)
  650 CONTINUE
C
      WRITE(6,700) ERROR,ERRCNT,ESF,OTLNCT
  700 FORMAT('0NUMBER OF INCORRECT PELS =',I10/
     *       '0NUMBER OF BITS IN ERROR TRANSMITTED =',I10/
     *       '0ERROR SENSITIVITY FACTOR =',F12.4/
     *       '0TOTAL NUMBER OF OUTPUT LINES PROCESSED = ',I8)
C
      RETURN
  800 CONTINUE
      STOP 800
      E N D
      SUBROUTINE STATS(LENGTH,INLNCT,DIAG)
      IMPLICIT INTEGER(A-Z)
C
      INTEGER MTT(5),ITT(2,5),LENGTH(INLNCT)
      REAL STT(2,5),SUM,SUMSQ
      LOGICAL DIAG
C************************ FILE DEFINITIONS ********************
C
```

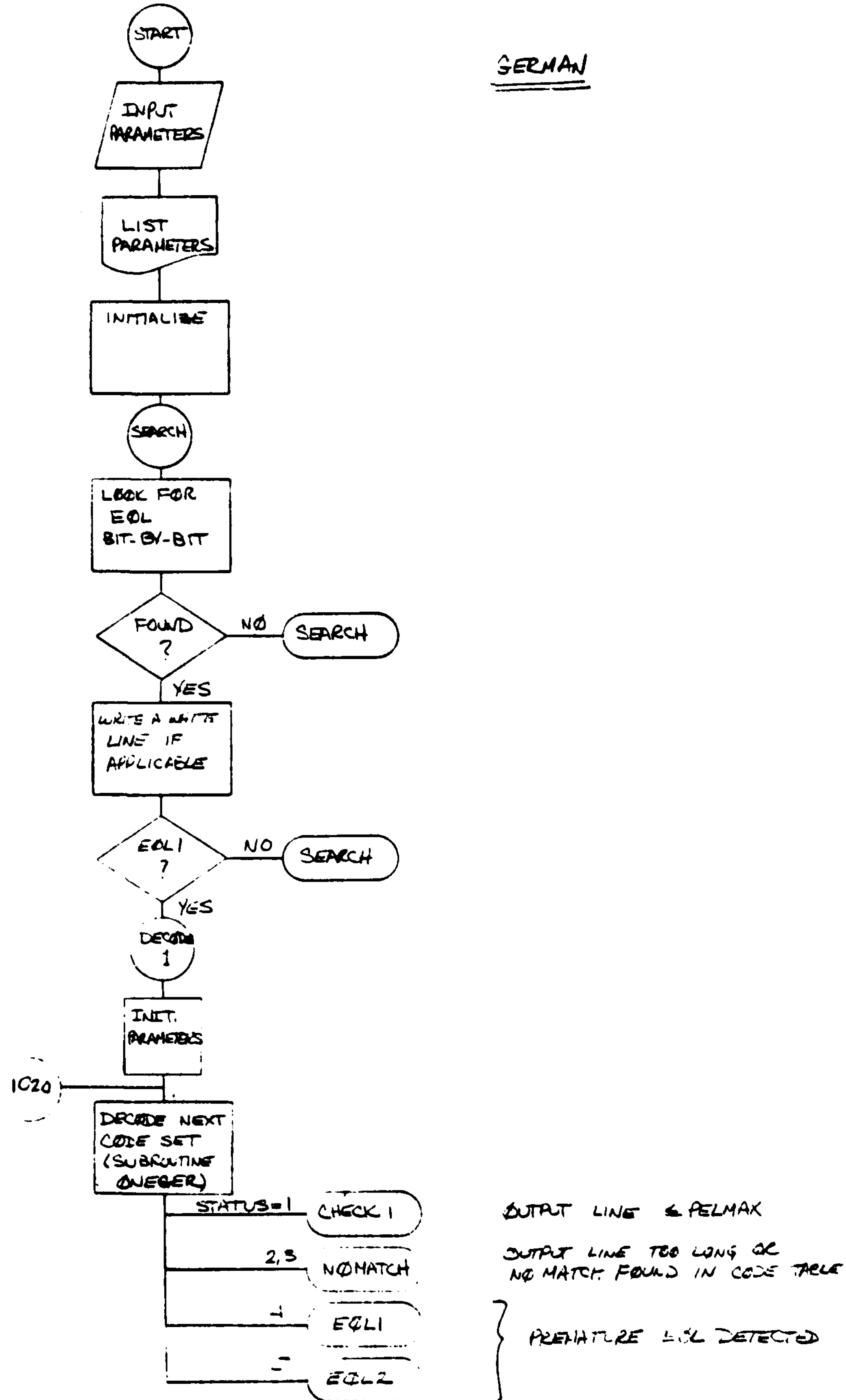
```
      COMMON/FILES/TERM,LPFIL,PELFIL,OTFIL,ERFIL                         00017
C                                                                        00017
      DATA MTT/0,24,48,96,192/                                           00017
C                                                                        00017
C*************************BEGIN PROGRAM**********************************00017
C                                                                        00017
      DO 300 I=1,5                                                       00017
      ITT(1,I)=10000                                                     00017
      ITT(2,I)=0                                                         00017
      SUM=0.                                                             00017
      SUMSQ=0.                                                           00017
      DO 100 J=1,INLNCT                                                  00017
C                                                                        00017
C    FIND FILLED LINE LENGTH                                             00017
C                                                                        00017
      LEN=MAX0(LENGTH(J),MTT(I))                                         00017
      IF(DIAG) WRITE(6,50) LEN                                           00017
   50 FORMAT(I8)                                                         00017
C                                                                        00017
C    FIND MINIMUM LINE LENGTH                                            00017
C                                                                        00017
      ITT(1,I)=MIN0(LEN,ITT(1,I))                                        00017
C                                                                        00017
C    FIND MAXIMUM LINE LENGTH                                            00017
C                                                                        00017
      ITT(2,I)=MAX0(LEN,ITT(2,I))                                        00017
C                                                                        00017
C    FIND SUM OF LENGTHS                                                 00017
C                                                                        00017
      SUM=SUM+FLOAT(LEN)                                                 00017
      SUMSQ=SUMSQ+(FLOAT(LEN))**2                                        00017
  100 CONTINUE                                                           00017
C                                                                        00017
C    FIND SAMPLE MEAN AND STANDARD DEVIATION                             00017
C                                                                        00017
      STT(1,I)=SUM/FLOAT(INLNCT)                                         00017
      STT(2,I)=SQRT((SUMSQ-(SUM**2)/FLOAT(INLNCT))/FLOAT(INLNCT-1))      00017
  300 CONTINUE                                                           00017
C                                                                        00017
      WRITE(6,400)(ITT(1,I),I=1,5)                                       00017
  400 FORMAT(                                                            00017
     *'0                            MINIMUM TRANSMISSION TIME (4800 BPS)'//00017
     *'   CODED LINE'/                                                   00017
     *'     LENGTH                  0 MS     5 MS    10 MS    20 MS    40 MS'/ 00017
     *'   STATISTICS:'//                                                 00017
     *'     MINIMUM',10X,5(I8)//)                                        00017
      WRITE(6,410)(ITT(2,I),I=1,5)                                       00017
  410 FORMAT(                                                            00017
     *'     MAXIMUM',10X,5(I8)//)                                        00017
      WRITE(6,420)(STT(1,I),I=1,5)                                       00017
  420 FORMAT(                                                            00017
     *'     SAMPLE MEAN',9X,5(F8.2)//)                                   00017
      WRITE(6,430)(STT(2,I),I=1,5)                                       00017
  430 FORMAT(                                                            00017
     *'     STANDARD DEVIATION',2X,5(F8.2))                              00017
C                                                                        00017
      RETURN                                                             00017
      END                                                                00017
0          END OF OCEC UPRINT PROGRAM               LINES PRINTED=  1615
```

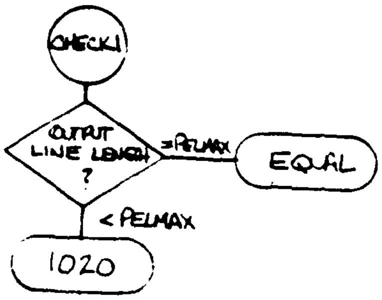
APPENDIX F

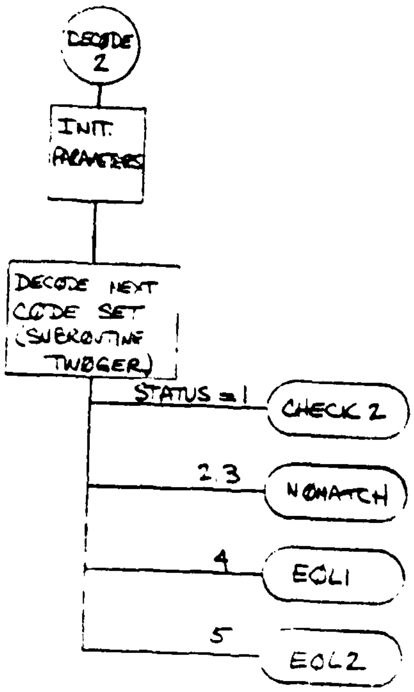
FLOW CHART

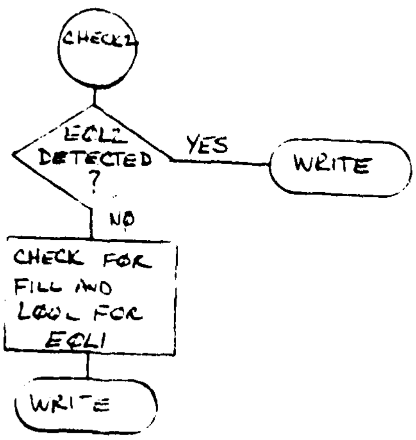
FEDERAL REPUBLIC OF GERMANY

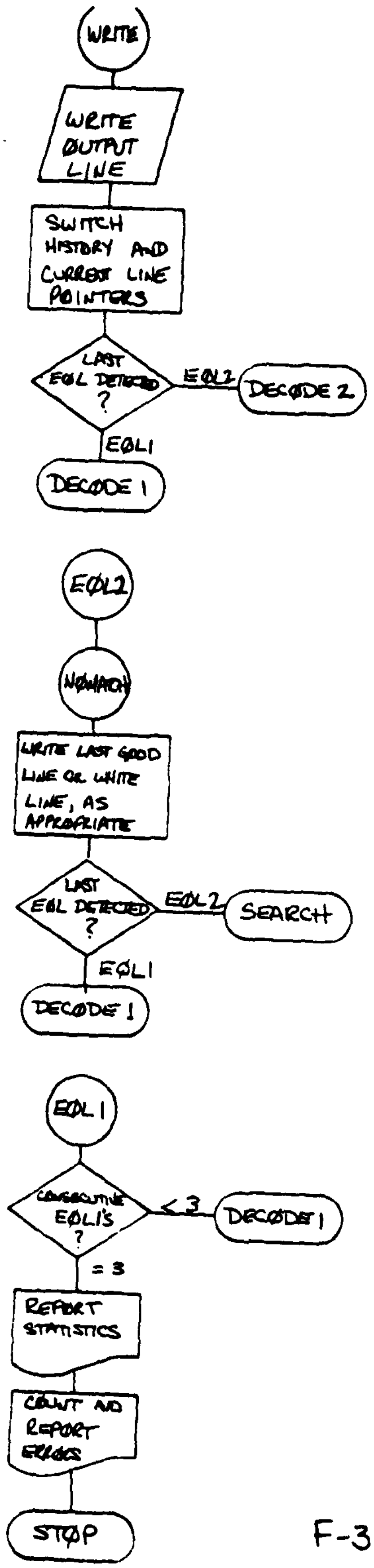
GERMAN


START
INPUT PARAMETERS
LIST PARAMETERS
INITIALIZE
SEARCH
LOOK FOR EOL BIT-BY-BIT
FOUND ?
NO
SEARCH
YES
WRITE A WHITE LINE IF APPLICABLE
EOL1 ?
NO
SEARCH
YES
DECODE 1
INIT. PARAMETERS
IC20
DECODE NEXT CODE SET (SUBROUTINE ONEGER)
STATUS=1
CHECK 1
2,3
NOMATCH
4
EOL1
5
EOL2
OUTPUT LINE ≤ PELMAX
OUTPUT LINE TOO LONG OR NO MATCH FOUND IN CODE TABLE
PREMATURE EOL DETECTED

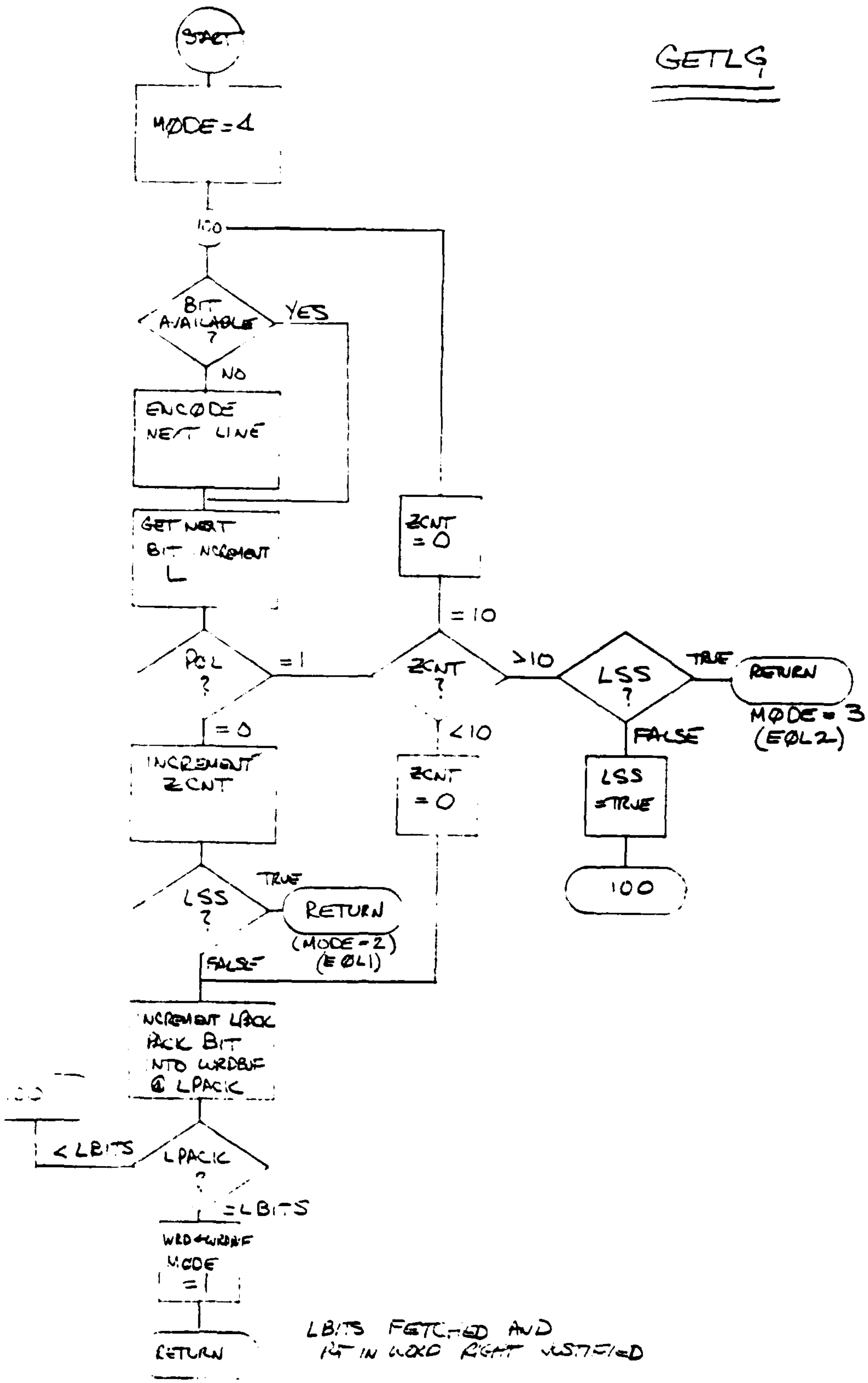

CHECK1
OUTPUT LINE LENGTH ?
=PELMAX
EQUAL
< PELMAX
1020


DECODE 2
INIT. PARAMETERS
DECODE NEXT CODE SET (SUBROUTINE TWOGER)
STATUS = 1
CHECK 2
2.3
NOMATCH
4
EOL1
5
EOL2

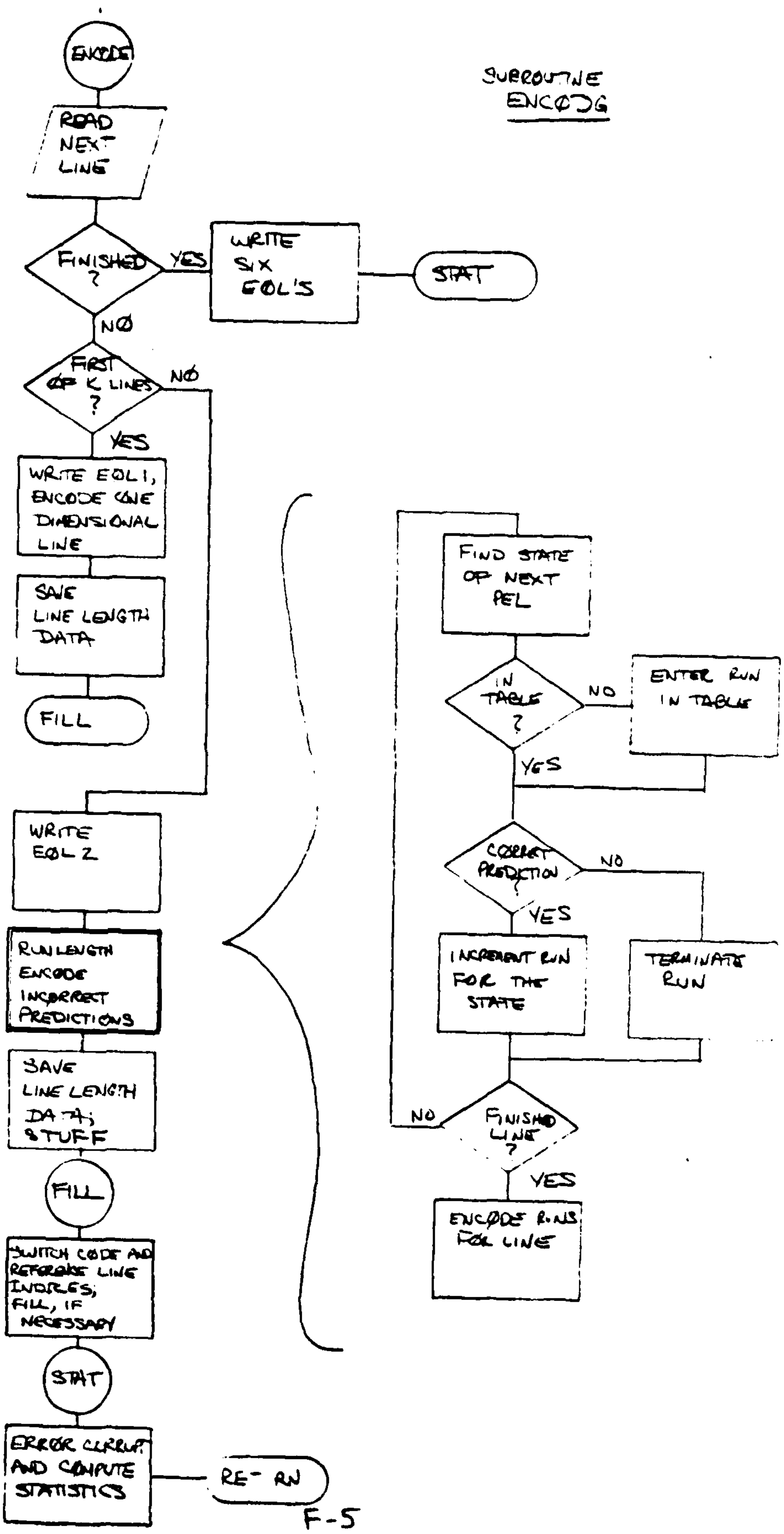

CHECK2
EOL2 DETECTED ?
YES
WRITE
NO
CHECK FOR FILL AND LOOK FOR EOL1
WRITE

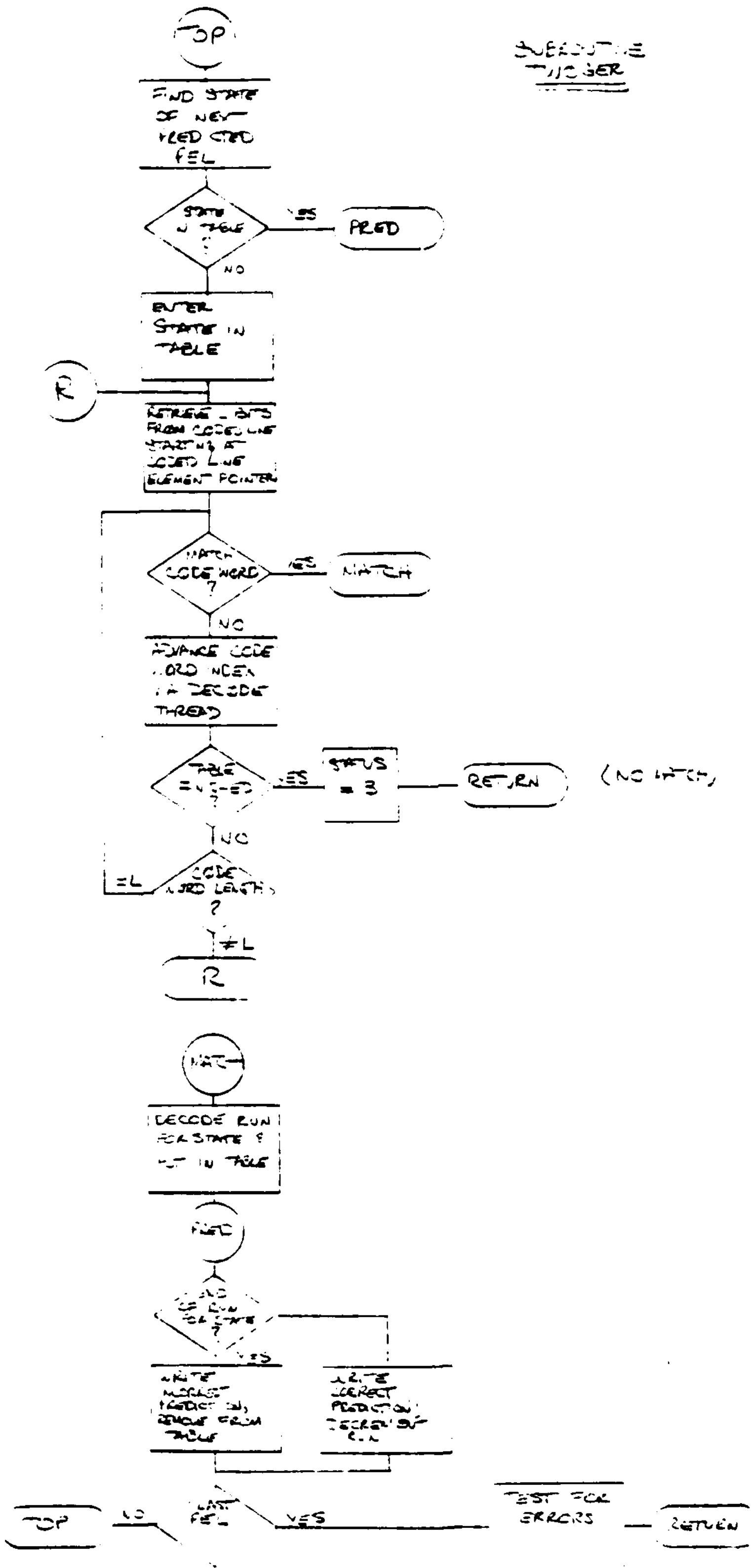

WRITE
WRITE OUTPUT LINE
SWITCH HISTORY AND CURRENT LINE POINTERS
LAST EØL DETECTED ?
EØL2
DECØDE 2
EØL1
DECØDE 1
EØL2
NØMATCH
WRITE LAST GOOD LINE OR WHITE LINE, AS APPROPRIATE
LAST EØL DETECTED ?
EØL2
SEARCH
EØL1
DECØDE 1
EØL1
CONSECUTIVE EØL1'S ?
< 3
DECØDE 1
= 3
REPORT STATISTICS
COUNT AND REPORT ERRORS
STØP

GETLG


START
MØDE = 1
100
BIT AVAILABLE ?
YES
NO
ENCØDE NEXT LINE
GET NEXT BIT INCREMENT L
POL ?
= 1
= 0
ZCNT = 0
= 10
ZCNT ?
> 10
< 10
LSS ?
TRUE
RETURN
MØDE = 3 (EØL2)
FALSE
LSS = TRUE
100
INCREMENT ZCNT
ZCNT = 0
LSS ?
TRUE
RETURN
(MODE = 2) (EØL1)
FALSE
INCREMENT LPACK PACK BIT INTO WRDBUF @ LPACK
LPACK ?
< LBITS
= LBITS
WRD ← WRDBUF MODE = 1
RETURN
LBITS FETCHED AND PUT IN WORD RIGHT JUSTIFIED

SUBROUTINE ENCODG




SUBROUTINE
TWIDGER
TOP
FIND STATE OF NEW PREDICTED PEL
STATE IN TABLE ?
YES
PRED
NO
ENTER STATE IN TABLE
R
RETRIEVE L BITS FROM CODED LINE STARTING AT CODED LINE ELEMENT POINTER
MATCH CODE WORD ?
YES
MATCH
NO
ADVANCE CODE WORD INDEX VIA DECODE THREAD
TABLE FINISHED ?
YES
STATUS = 3
RETURN
(NO MATCH)
NO
CODE WORD LENGTH ?
=L
≠L
R
MATCH
DECODE RUN FOR STATE & PUT IN TABLE
PRED
END OF RUN FOR STATE ?
YES
WRITE INCORRECT PREDICTION, REMOVE FROM TABLE
WRITE CORRECT PREDICTION, DECREMENT RUN
TOP
NO
LAST PEL
YES
TEST FOR ERRORS
RETURN

APPENDIX G

CODE LISTING

FEDERAL REPUBLIC OF GERMANY

```
START OF DCEC JPRINT PROGRAM          DSNAME=N0026.GERMAN.FORT
C     PROGRAM GERMAN
      IMPLICIT INTEGER(A-Z)
      REAL CF3,CF4,ERRATE
C******* LABELED COMMON /G32BIT/ *******
C
      COMMON /G32BIT/MASK(32),COMASK(32),LIBIT(32),LZBIT(32)
      INTEGER MASK,COMASK,LIBIT,LZBIT
C
      COMMON/BUFF/PELBUF(60,2),CDBUF(240),
     *            DTBUF(60,2),STBUF(240), STAT(3000)
      COMMON/HUFF/CODE(3,92,2),CCDS(3,68,6),PREDCT(16),NPRED(16),
     *            CTABLE(16),CSTART(16),STBUF(1728),STRUN(1728)
      COMMON/ERAY/ERRORS(2500)
C*********************** FILE DEFINITIONS *******************
C
      COMMON/FILES/TERM,LPFIL,PELFIL,CTFIL,EFFIL
C
C********************** LABELLED COMMON VARIABLES *********************
C
      COMMON/IVAR/PELMAX,VRES,EPHASE,CMPMAX,ERRMOD,LINMAX,K
      COMMON/PVAR/INLNNO,OTLNNO,OTELW,INELP,CDELP,OFELP,CDE.W,
     *            CDELCT,INELCT,TCDATA,TCDEL,ERRPNT,ERRCFF,ERRLIM,
     *            ERRCNT,INLNCT,CONSEC,LNNOBF,ZCNT,WRDBUF,LPACK,
     *            INCOD,INREF,CTCOD,CTREF,TSTFBT
      COMMON/ICHAR/DD,II,NM,TT,NN,YY
      COMMON/LOGIC/SEARCH,DIAG,SYNC,LSS,WRITE,CHCOL,ONE
      LOGICAL SEARCH,DIAG,SYNC,LSS,WRITE,CHCOL,ONE
C
C  READ INPUT PARAMETERS
   90 WRITE(6,100)
  100 FORMAT('$PARAMETERS: INPUT(=I), OR DEFAULT(=D)?')
      READ(5,110,ERR=90) INSW
  110 FORMAT(A1)
      IF (INSW.EQ.DD) GO TO 315
      IF (INSW.NE.II) GO TO 90
C
C  READ DIAGNOSTIC SWITCH
C
  114 WRITE(6,115)
  115 FORMAT('$DIAGNOSTIC PRINTOUT? (Y OR N): ')
      READ(5,110) INSW
      IF(INSW.EQ.YY) GO TO 116
      IF(INSW.EQ.NN) GO TO 120
      GO TO 114
  116 CONTINUE
      DIAG=.TRUE.
C
C  READ MAXIMUM NUMBER OF PELS PER LINE
C
  120 CONTINUE
      WRITE(6,130)
  130 FORMAT('$ENTER MAXIMUM NUMBER OF PELS PER LINE: ')
      READ(5,140,ERR=120) PELMAX
  140 FORMAT(I4)
      IF(PELMAX.GE.1.AND.PELMAX.LE.1728) GO TO 160
      WRITE(6,150) PELMAX
  150 FORMAT('0NUMBER OUT OF RANGE (=',I6,')')
      GO TO 120
C
C  READ VERTICAL SAMPLING
C
  160 CONTINUE
      WRITE(6,170)
  170 FORMAT('$ENTER VERTICAL SAMPLING: ')
      READ(5,180,ERR=160) VRES
  180 FORMAT(I2)
      IF(VRES.GE.1.AND.VRES.LE.10) GO TO 190
      WRITE(6,150) VRES
      GO TO 160
C
C  READ PARAMETER K
C
  190 CONTINUE
      WRITE(6,192)
  192 FORMAT('$ENTER PARAMETER K: ')
      READ(5,140,ERR=190) K
      IF(K.GE.1.AND.K.LE.3000) GO TO 200
      WRITE(6,150) K
      GO TO 190
C
C  READ ERROR PATTERN PHASE
```

```
C
  200 CONTINUE
      WRITE(6,210)
  210 FORMAT('$ENTER ERROR PATTERN PHASE: ')
      READ(5,220,ERR=200) EPHASE
  220 FORMAT(I1)
      IF(EPHASE.GE.0.AND.EPHASE.LE.3) GO TO 240
      WRITE(6,150) EPHASE
      GO TO 200
C
C   READ MINIMUM COMPRESSED LINE LENGTH
C
  240 CONTINUE
      WRITE(6,250)
  250 FORMAT('$ENTER MINIMUM COMPRESSED LINE LENGTH: ')
      READ(5,140,ERR=240) CMPMAX
      IF(CMPMAX.GE.0.AND.CMPMAX.LE.1728) GO TO 320
      WRITE(6,150) CMPMAX
      GO TO 240
C
C   READ NUMBER OF SCAN LINES TO BE PROCESSED
  320 CONTINUE
      WRITE(6,330)
  330 FORMAT('$NUMBER OF SCAN LINES TO BE PROCESSED=? ')
      READ(5,140,ERR=320) LINMAX
      IF(LINMAX.GE.1.AND.LINMAX.LE.3000) GO TO 280
      WRITE(6,150) LINMAX
      GO TO 320
C
C   READ ERROR MODE
C
  280 CONTINUE
      WRITE(6,290)
  290 FORMAT('$ERROR MODE=? (M=MANUAL,T=TAPE,N=NO ERRORS)')
      READ(5,110,ERR=280) ERRMOD
      IF(ERRMOD.EQ.MM) GO TO 300
      IF(ERRMOD.EQ.TT) GO TO 315
      IF(ERRMOD.NE.NN) GO TO 280
      GO TO 350
C
C   READ ERROR LOCATIONS
C
  300 CONTINUE
      ERRLIM=1
  305 READ(5,140) ERRORS(ERRLIM)
      IF(ERRORS(ERRLIM).EQ.9999) GO TO 310
      ERRLIM=ERRLIM+1
      GO TO 305
  310 CONTINUE
      ERRLIM=ERRLIM-1
      GO TO 350
C
C   READ ERROR TAPE FILE AND OPEN
C
  315 CONTINUE
C
      ERRLIM=1
      READ(3,318,END=317) ERRORS(ERRLIM)
      ERRLIM=ERRLIM+1
  316 READ(3,318,END=317) ERRORS(ERRLIM)
  318 FORMAT(I16)
      ERRORS(ERRLIM)=ERRORS(ERRLIM)+ERRORS(ERRLIM-1)
      ERRLIM=ERRLIM+1
      GO TO 316
  317 ERRLIM=ERRLIM-1
C
  350 CONTINUE
C
  360 CONTINUE
C   WRITE INPUT PARAMETERS
C
      WRITE(6,400) PELMAX,VRES,K,EPHASE,CMPMAX,LINMAX
  400 FORMAT('1INPUT PARAMETERS:'/
     *       '0MAXIMUM NUMBER OF PELS PER LINE=',I6/
     *       '0VERTICAL SAMPLING: N=',I4/
     *       '0PARAMETER K =',I4/
     *       '0ERROR PATTERN PHASE =',I4/
     *       '0MINIMUM COMPRESSED LINE LENGTH =',I6,' BITS'/
     *       '0NUMBER OF SCAN LINES TO BE PROCESSED =',I6)
      IF(ERRMOD.EQ.NN) WRITE(6,410)
  410 FORMAT ('0NO ERRORS INSERTED')
      IF(ERRMOD.EQ.MM) WRITE(6,140) (ERRORS(I),I=1,ERRLIM)
```

```
      IF(ERRMOD.EQ.TT) WRITE(6,420) ERRLIM
  420 FORMAT(I12,' ERRORS OBTAINED FROM ERROR TAPE')
C******************** BEGIN PROGRAM ************************************
C
C    INITIALIZE
C
      TCDEL=0
      TCDATA=0
      ERRPNT=1
      ERRCNT=0
      INLNCT=0
      ERROFF=EPHASE*1024
      CDELCT=32
      OTELP=1
      CDELP=32+1
      CONSEC=1
      INREF=1
      INCCD=2
      OTREF=1
      CTCOD=2
C
      DO 800 I=1,240
      STFBUF(I)=0
      CDBUF(I)=0
  800 CONTINUE
      DO 850 I=1,60
      OTBUF(I,OTREF)=0
      OTBUF(I,OTCOD)=0
      PELBUF(I,INREF)=0
      PELBUF(I,INCOD)=0
  850 CONTINUE
      SEARCH=.TRUE.
      SYNC=.FALSE.
      WRITE=.FALSE.
C
C    SEARCH MODE: LOOK FOR EOL1 BIT-BY-BIT
C
  900 CONTINUE
      L=0
      LSS=.FALSE.
      ZCNT=0
      WRDBUF=0
      LPACK=0
      CALL GETLG(13,MODE,LBITS,L)
      GO TO (910,930,930,920),MODE
      STOP 900
  910 CONTINUE
C
C    EOL NOT FOUND; ADVANCE POINTER AND TRY AGAIN
C
      CDELP=CDELP+1
      GO TO 900
  920 CONTINUE
      STOP 920
  930 CONTINUE
C
C    EOL1 FOUND
C
      SEARCH=.FALSE.
      CDELP=CDELP+L
      IF(WRITE) GO TO 935
      WRITE=.TRUE.
      GO TO 960
  935 CONTINUE
C
C    SET OUTPUT DECODE LINE TO 0 AND WRITE OUT
      DO 950 I=1,60
      OTBUF(I,OTCOD)=0
  950 CONTINUE
      WRITE(2) OTLNNO,PELMAX,(OTBUF(I,OTCOD),I=1,60)
      OTLNNO=LNNOBF
  960 CONTINUE
      IF(MODE-2)965,1000,900
  965 STOP 965
 1000 CONTINUE
C
C    PERFORM ONE-DIMENSIONAL DECODE OF A COMPLETE LINE
C    FIRST, SET OUTPUT BUFFER TO WHITE
C    (ONLY BLACK RUNS WILL BE INSERTED)
C
      DO 1010 I=1,60
      OTBUF(I,OTCOD)=0
```

```
1010 CONTINUE
C
      INDEX=3
      COLOR=1
      CTELP=1
C
      LSS=.FALSE.
      ZCNT=0
1020 CONTINUE
      CALL ONEGER(INDEX,COLOR,STATUS,L)
      GO TO (1030,1070,1070,1035,1040),STATUS
C           1    2    3    4    5
      STOP 1000
C
C   ALL RUNS ADDED
C
1030 CONTINUE
      ONE=.TRUE.
      IF(CTELP-1-PELMAX) 1031,1032,1050
1031 CONTINUE
      IF(CHCOL) COLOR=MOD(COLOR+2,2)+1
      INDEX=2
      GO TO 1020
3000 CONTINUE
C
C   PERFORM TWO-DIMENSIONAL DECODE
C
C
C   FIRST,SET OUTPUT BUFFER TO WHITE
C   (ONLY BLACK RUNS WILL BE INSERTED)
C
      DO 3010 I=1,60
      CTBUF(I,OTCOD)=0
3010 CONTINUE
C
      CTELP=1
C
      LSS=.FALSE.
      ZCNT=0
      CALL TWOGER(INDEX,COLOR,STATUS,L)
      GO TO (3030,1070,1070,1035,1040),STATUS
C           1    2    3    4    5
      STOP 3000
C
C   RUN ADDED; LOOK FOR NEXT RUN
C
3030 CONTINUE
      ONE=.FALSE.
C
C   LINE LENGTH=PELMAX; CHECK FOR FILL AND LOOK FOR EOL1
C
1032 CONTINUE
      ZERO=-1
      LSS=.FALSE.
      ZCNT=0
1033 CONTINUE
      ZERO=ZERO+1
      WRCBUF=0
      LPACK=0
      L=0
      CALL GET_G(1,MODE,LBITS,L)
C
      GO TO (1034,1060,1060,1050),MODE
C
C   CHECK FOR FILL
C
1034 CONTINUE
C
      CDELP=CDELP+L
      IF(LBITS.EQ.0) GO TO 1033
      IF(ZERO.LE.10) GO TO 1070
      STOP 1034
C
C   PREMATURE EOL DETECTED
C
C
C   EOL1 DETECTED
C
1035 CONTINUE
      CDELP=CDELP+L
      STATUS=4
      IF(DTELP.LE.1) CONSEC=CONSEC+1
```

```
      IF(CCNSEC-2)1080,1000,2000
C
C  EOL2 DETECTED
C
 1040 CONTINUE
      CDELP=CDELP+L
      STATUS=5
C
      GO TO 1080
C
C  PROBLEMS,PROBLEMS
C
 1050 STOP 1050
C
C   LINE LENGTH CORRECT, EOL DETECTED PROPERLY; WRITE OUTPUT LINE
C
 1060 CONTINUE
      CDELP=CDELP+L
      WRITE(2)OTLNNO,PELMAX,(OTBUF(I,OTCOD),I=1,60)
      OTLNNO=LNNOBF
      CONSEC=1
      IF(ONE) SYNC=.TRUE.
      TEMP=OTREF
      OTREF=OTCOD
      OTCOD=TEMP
      IF(MODE.EQ.2) GO TO 1000
      GO TO 3010
C
C   LINE TOO LONG OR NO MATCH
C
 1070 CONTINUE
      WRITE=.FALSE.
C
C   LINE SHORT
C
 1080 CONTINUE
      IF(.NOT.SYNC) GO TO 1090
C
C   WRITE LAST GOOD LINE
C
      WRITE(2) OTLNNO,PELMAX,(OTBUF(I,OTREF),I=1,60)
      SYNC=.FALSE.
      GO TO 1110
 1090 CONTINUE
C
C   WRITE A WHITE LINE
C
      DO 1100 I=1, 60
 1100 OTBUF(I,OTCOD)=0
      WRITE(2) OTLNNO,PELMAX,(OTBUF(I,OTCOD),I=1,60)
 1110 OTLNNO=LNNOBF
      IF(STATUS.EQ.4) GO TO 1000
      SEARCH=.TRUE.
      GO TO 900
C
C   END OF MESSAGE
C
 2000 CONTINUE
      WRITE(6,2010) CONSEC
 2010 FORMAT('0END OF MESSAGE DETECTED (',I2,' EOL''S)')
C
C   REPORT COMPRESSION FACTOR, ERROR SENSITIVITY FACTOR,BIT ERROR RATE
C
      ERRATE=FLOAT(ERRCNT)/FLOAT(TCDEL)
      WRITE(6,2020) TCDEL,TCDATA,TSTFBT,INLNCT,ERRATE
 2020 FORMAT('0TOTAL NUMBER OF CODED BITS = ',I8/
     *        '0TOTAL NUMBER OF CODED DATA BITS = ',I8/
     *        '0TOTAL NUMBER OF STUFFING BITS =',I8/
     *        '0TOTAL NUMBER OF INPUT LINES PROCESSED = ',I3/
     *        '0BIT ERROR RATE = ',G14.6)
C
      CALL STATS(STAT,INLNCT,DIAG)
      CF3=FLOAT(PELMAX)*FLOAT(INLNCT)/FLOAT(TCDEL)
      CF4=FLOAT(PELMAX)*FLOAT(INLNCT)/FLOAT(TCDATA)
C
      WRITE(6,2030) CF3,CF4
 2030 FORMAT('0COMPRESSION FACTOR FOR G3 MACHINE (CF3) =',F8.4/
     *        '0COMPRESSION FACTOR FOR G4 MACHINE (CF4) =',F8.4)
C
      CALL ERRMES(PELBUF,OTBUF,PELMAX,VRES,ERRCNT)
C
      STOP
```

```
      END
      SUBROUTINE GETLG(LBITS,MODE,WRD,L)
      IMPLICIT INTEGER(A-Z)
C******* LABELED COMMON /G32BIT/ *******
C
      COMMON /G32BIT/MASK(32),COMASK(32),LIBIT(32),LZBIT(32)
      INTEGER MASK,COMASK,LIBIT,LZBIT
C
      COMMON/BUFF/PELBUF(60,2),CDBUF(240),
     *            OTBUF(60,2),STFBUF(240), STAT(3000)
      COMMON/HUFF/CODE(3,92,2),CODS(3,68,6),PREDCT(16),NPRED(16),
     *            CTABLE(16),CSTART(16),STBUF(1728),STRUN(1728)
      COMMON/ERAY/ERRORS(2500)
C******************** LABELLED COMMON VARIABLES *********************
C
      COMMON/IVAR/PELMAX,VRES,EPHASE,CMPMAX,ERRMOD,LINMAX,K
      COMMON/PVAR/INLNNO,OTLNNO,OTELW,INELP,CDELP,OTELP,CDELW,
     *            CDELCT,INELCT,TCDATA,TCDEL,ERRPNT,ERROFF,ERRLIM,
     *            ERRCNT,INLNCT,CONSEC,LNNOBF,ZCNT,WRDBUF,LPACK,
     *            INCOD,INREF,CTCOD,CTREF,TSTFBT
      COMMON/ICHAR/DD,II,MM,TT,NN,YY
      COMMON/LOGIC/SEARCH,DIAG,SYNC,LSS,WRITE,CHCOL,ONE
      LOGICAL SEARCH,DIAG,SYNC,LSS,WRITE,CHCOL,ONE
C**************************** BEGIN PROGRAM ****************************
C
      MODE=4
C
C     RETRIEVE NEXT BIT FROM CDBUF
C
  100 CONTINUE
C
C     ENCODE A NEW LINE IF NECESSARY
C
      IF(L+CDELP.LE.CDELCT) GO TO 200
      IF(CDELCT-CDELP+1) 170,190,180
  170 STOP 170
  180 CONTINUE
      STFBUF(1)=I4B(STFBUF,CDELP,CDELCT-CDELP+1)
  190 CONTINUE
      CDELP=32-(CDELCT-CDELP)
      CALL ENCODG
  200 CONTINUE
      POL=IAB(STFBUF,CDELP+L,1)
      L=L+1
      IF(POL)220,300,240
  220 STOP 220
  240 CONTINUE
      IF(ZCNT-10)310,260,340
  260 ZCNT=0
      GO TO 100
  300 ZCNT=ZCNT+1
      IF(LSS) GO TO 380
      GO TO 320
  310 CONTINUE
      ZCNT=0
  320 CONTINUE
      LPACK=LPACK+1
      IF(POL) 324,330,325
  324 STOP 324
  325 CONTINUE
      CALL MI2B(POL,WRDBUF,LPACK,1)
  330 CONTINUE
      IF(LPACK.LT.LBITS) GO TO 100
      WRD=I4B(WRDBUF,1,LPACK)
      MODE=1
      RETURN
  340 CONTINUE
      IF(LSS) GO TO 360
      LSS=.TRUE.
      GO TO 100
  360 MODE=3
      RETURN
  380 CONTINUE
      MODE=2
      RETURN
      END
      SUBROUTINE ENCODG
C
      IMPLICIT INTEGER(A-Z)
C
C******* LABELED COMMON /G32BIT/ *******
C
```

```
      COMMON /232BIT/MASK(32),COMASK(32),LIBIT(32),LZBIT(32)
      INTEGER MASK,COMASK,LIBIT,LZBIT
C
      COMMCN/BUFF/PELBUF(60,2),CDBUF(240),
     *          OTBUF(60,2),STFBUF(240), STAT(3000)
      COMMON/HUFF/CODE(3,92,2),CODS(3,60,6),PREDCT(16),NPRED(16),
     *          CTABLE(16),CSTART(16),STBUF(1728),STRUN(1728)
      COMMCN/ERAY/ERRORS(2500)
C*************************** FILE DEFINITIONS ***********************
C
      CCMMCN/FILES/TERM,LPFIL,PELFIL,OTFIL,ERFIL
C
C********************** LABELLED COMMON VARIABLES **********************
C
      COMMON/IVAR/PELMAX,VRES,EPHASE,CMPMAX,ERRMOD,LINMAX,K
      COMMCN/PVAR/INLNNO,OTLNNO,OTELW,INELP,CDELP,OTELP,CDELW,
     *          CDELCT,INELCT,TCDATA,TCDEL,ERRPNT,ERRUFF,ERRLIN,
     *          ERRCNT,INLNCT,CONSEC,LNNOBF,ZCNT,WRDBUF,LPACK,
     *          INCOD,INREF,OTCOD,OTREF,TSTFRT
      COMMON/ICHAR/DD,II,MM,TT,NN,YY
      COMMCN/LOGIC/SEARCH,DIAG,SYNC,LSS,WRITE,CHCOL,ONE
      LOGICAL SEARCH,DIAG,SYNC,LSS,WRITE,CHCOL,ONE
      INTEGER INDEX(16)
C
C*********************************** BEGIN PROGRAM ********************
C
C   INITIALIZE VARIABLES
C
      CDELCT=32
      CDDATA=0
      DC 50 I=2,240
      CDBUF(I)=0
      STFBUF(I)=0
   50 CONTINUE
C
C   READ INPUT PICTURE FILE
C
  100 CONTINUE
      READ(1,END=120,ERR=500)
     * INLNNO,INELCT,(PELBUF(I,INCOD),I=1,60)
      IF(MOD(INLNNO,100).EQ.0) WRITE(6,110) INLNNO
  110 FORMAT(' INPUT LINE NO. =',I6)
      IF(MOD(INLNNO-1,VRES).NE.0) GO TC 100
      IF(INELCT.LT.PELMAX) CALL EXIT
      INLNCT=INLNCT+1
C
C   LOAD OUTPUT LINE NUMBER BUFFER
C
      LNNCBF=INLNNO
      IF(SEARCH)OTLNNO=LNNOBF
C
      IF(INLNNO.LE.LINMAX) GO TO 140
C
C   WRITE SIX EOL'S
C
  120 CONTINUE
C     IF(INLNCT.GT.0) STOP
      DO 130 I=1,6
      T=0
      CALL CODEG(67,T,CDELCT,CDDATA)
  130 CONTINUE
      DO 135 I=1,6
      STFBUF(I)=CDBUF(I)
  135 CCNTINUE
      GO TO 400
C
C   FIRST CF K LINES?
C
  140 CONTINUE
      IF(MOD(INLNCT-1,K).NE.0) GO TO 600
C
C   ONE-DIMENSIONAL CODING
C   WRITE ONE EOL
C
      T=C
      CALL CODEG(67,T,CDELCT,CDDATA)
C
      POLAR=1
C
C   TEST CCLCR OF FIRST ELEMENT
C
      IF(I48(PELBUF(1,INCOD),1,1).EQ.0) GO TO 150
```

```
C
C     FIRST ELEMENT BLACK; ENCODE 0-LENGTH WHITE RUN
C
      CALL GCODLR(0,1,CDELCT,CDDATA)
      POLAR=2
C
C     CALCULATE RUN LENGTH AND ENCODE
C
  150 CONTINUE
      RUN=0
      DO 200 I=1,PELMAX
      PEL=I4B(PELBUF(1,INCOD),I,1)+1
      IF(PEL.EQ.POLAR) GO TO 180
      CALL GCODLR(RUN,POLAR,CDELCT,CDDATA)
      IF(.NOT.DIAG) GO TO 170
      WRITE(6,160) RUN,POLAR,CDELCT,CDDATA
  160 FORMAT(4I8)
  170 CONTINUE
      RUN=1
      POLAR=MOD(POLAR+2,2)+1
      GO TO 200
  180 CONTINUE
      RUN=RUN+1
  200 CONTINUE
      CALL GCODLR(RUN,POLAR,CDELCT,CDDATA)
      IF(.NOT.DIAG) GO TO 210
      WRITE(6,160) RUN,POLAR,CDELCT,CDDATA
      GO TO 210
C
C     TWO-DIMENSIONAL CODING
C
  600 CONTINUE
C
C     WRITE ONE EOL2
C
      T=0
      CALL CODEG(68,T,CDELCT,CDDATA)
C
C     INITIALIZE ARRAY POINTERS
C
      J=1
      DO 610 I=1,16
      INDEX(I)=0
  610 CONTINUE
      DO 700 I=1,PELMAX
C
C     PREDICT NEXT ELEMENT
C
      IF(I-1) 611,612,613
  611 STOP 611
  612 CONTINUE
      PEL1=I4B(PELBUF(1,INREF),I,2)
      PEL2=0
      CALL MI23(PEL1,PEL2,32-2,2)
      GO TO 615
  613 CONTINUE
      PEL2=I4B(PELBUF(1,INCOD),I-1,1)
      PEL1=I4B(PELBUF(1,INREF),I-1,3)
      CALL MI23(PEL1,PEL2,32-3,3)
  615 CONTINUE
      SP1=PEL2+1
      JS=INDEX(SP1)
      IF(JS-1) 620,630,630
C
C     ADD A STATE ENTRY TO TABLE
C
  620 CONTINUE
      STBUF(J)=SP1
      STRUN(J)=1
      INDEX(SP1)=J
      JS=J
      J=J+1
  630 CONTINUE
      IF(PREDCT(SP1).EQ.I4B(PELBUF(1,INCOD),I,1))GO TO 650
      INDEX(SP1)=0
      GO TO 700
  650 CONTINUE
      STRUN(JS)=STRUN(JS)+1
  700 CONTINUE
C
C     CONSTRUCT CODE LINE
C
```

```
      JMAX=J-1
      DO 800 J=1,JMAX
      SP1=STBUF(J)
      CALL CODEG(STRUN(J),CTABLE(SP1),CDELCT,CDDATA)
      IF(.NOT.DIAG) GO TO 800
      S=SP1-1
      WRITE(6,160) STRUN(J),S,CDELCT,CDDATA
  800 CONTINUE
  210 CONTINUE
C
C     SWITCH CODE & REFERENCE LINES
C
      TEMP=INREF
      INREF=INCOD
      INCOD=TEMP
C
C   BIT STUFFING (INSERT ONES)
C
      CALL STUFF1(CDBUF,STFBUF,STFBIT,CDELCT)
C
C   SAVE LINE LENGTH (DATA + EOL)
C
      STAT(INLNCT)=CDDATA+CODS(1,68,1)
C
C   CHECK CODED LINE LENGTH
C
      FILL=CMPMAX-(CDELCT-32)
      IF(FILL) 400,400,250
C
C   CODE LINE TOO SHORT; FILL IT TO CMPMAX
  250 CONTINUE
      CDELCT=CDELCT+FILL
C
C   ACCUMULATE STATISTICS AND ERROR CORRUPT
C
  400 CONTINUE
      IF(ERRMOD.EQ.NN) GO TO 390
C
C   ERROR CORRUPT
C
  350 CONTINUE
      ERRBIT=ERRORS(ERRPNT)-ERROFF-TCDEL
      IF(ERRBIT.LE.0) GO TO 360
      IF(ERRBIT.GT.CDELCT-32) GO TO 390
C
C   ERROR IN RANGE OF CODED LINE; CHANGE APPROPRIATE BIT
C
      BIT=I4B(STFBUF,ERRBIT+32,1)
      BIT=MOD(BIT+1,2)
      CALL MI2B(BIT,STFBUF,ERRBIT+32,1)
      ERRCNT=ERRCNT+1
C
C    INCREMENT ERROR LIST POINTER
C
  360 CONTINUE
      ERRPNT=ERRPNT+1
      IF(ERRPNT.LE.ERRLIM) GO TO 350
C
C    ERROR LIST EXHAUSTED
C
      ERRPNT=ERRPNT-1
      WRITE(6,370) ERRPNT,ERRORS(ERRPNT)
  370 FORMAT('0ERROR LIST EXHAUSTED AT',I10,'TH ERROR;'/
     *       ' LAST ERROR OCCURRED AT',I10,' BITS')
      ERRMOD=NN
C
C    COMPLETE STATISTICS
C
  390 CONTINUE
      TCDEL=TCDEL+CDELCT-32
      TCDATA=TCDATA+CDDATA
      TSTFBT=TSTFBT+STFBIT
      IF(DIAG) WRITE(6,160) INLNCT, CDDATA
C
      IF (.NOT.DIAG) GO TO 460
      CDELW=(CDELCT+32-1)/32
      WRITE(6,450) (CDBUF(I),I=1,CDELW)
      WRITE(6,450) (STFBUF(I),I=1,CDELW)
  450 FORMAT(6Z12)
      WRITE(6,445) STFBIT
  445 FORMAT(I3,' ONES INSERTED')
  460 CONTINUE
```

```
      RETURN
C
  500 CONTINUE
      CALL EXIT
C
      E N D
      SUBROUTINE CODEG(LEN,TABLE,CDELCT,CDDATA)
C
      IMPLICIT INTEGER(A-Z)
      COMMON/BUFF/PELBUF(60,2),CDBUF(240),
     *            OTBUF(60,2),STFBUF(240), STAT(3330)
      COMMON/HUFF/CODE(3,92,2),CODS(3,68,6),PREDCT(16),NPRED(16),
     *            CTABLE(16),CSTART(16),STBUF(1728),STRUN(1728)
      COMMON/ERAY/ERRORS(2500)
      LOGICAL PREFIX
C
C******************************** BEGIN PROGRAM ***********************
C
      L=LEN
      T=TABLE
      PREFIX=.FALSE.
      IF(T) 1,600,5
    1 STOP 1
    5 CONTINUE
      IF(L.GE.65.AND.T.LE.2) GO TO 100
      IF(L.GE.33.AND.T.GE.3) GO TO 500
   10 CONTINUE
      CALL MI2B(CODS(3,L,T),CDBUF,CDELCT+1,CODS(1,L,T))
      CDELCT=CDELCT+CODS(1,L,T)
      IF(L.GT.66) RETURN
      CDDATA=CDDATA+CODS(1,L,T)
      IF(.NOT.PREFIX) RETURN
      CALL MI2B(LENGTH,CDBUF,CDELCT+1,11)
      CDELCT=CDELCT+11
      CDDATA=CDDATA+11
      RETURN
  100 CONTINUE
      IF(L.EQ.1729.AND.T.EQ.1) GO TO 120
      IF(T.EQ.2) GO TO 110
      LENGTH=L
      L=66
      PREFIX=.TRUE.
      GO TO 10
  110 CONTINUE
      LENGTH=L
      L=66
      PREFIX=.TRUE.
      GO TO 10
  120 CONTINUE
      L=65
      GO TO 10
  500 CONTINUE
      LENGTH=L
      L=33
      PREFIX=.TRUE.
      GO TO 10
C
C   WRITE EOL
C
  600 CONTINUE
      T=1
      GO TO 10
      E N D
      SUBROUTINE CNEGER(INDEX,COLOR,STATUS,L)
      IMPLICIT INTEGER(A-Z)
C******* LABELED COMMON /G32BIT/ *******
C
      COMMON /G32BIT/MASK(32),COMASK(32),LIBIT(32),LZBIT(32)
      INTEGER MASK,COMASK,LIBIT,LZBIT
C
      COMMON/BUFF/PELBUF(60,2),CDBUF(240),
     *            OTBUF(60,2),STFBUF(240), STAT(3330)
      COMMON/HUFF/CODE(3,92,2),CODS(3,68,6),PREDCT(16),NPRED(16),
     *            CTABLE(16),CSTART(16),STBUF(1728),STRUN(1728)
      COMMON/ERAY/ERRORS(2500)
C*********************** FILE DEFINITIONS *********************
C
      COMMON/FILES/TERM,LPFIL,PELFIL,OTFIL,ERFIL
C
C********************** LABELLED COMMON VARIABLES **********************
C
      COMMON/IVAR/PELMAX,VRES,EPHASE,CMPMAX,ERRMOD,LINMAX,K
```

```
      COMMON/PVAR/INLNNO,OTLNNO,OTELW,INELP,CDELP,OTELP,CDELW,
     *                COELCT,INELCT,TCDATA,TCDEL,ERRPNT,ERRUFF,ERRLIM,
     *                ERRCNT,INLNCT,CONSEC,LNNCPF,ZCNT,WRDBUF,LPACK,
     *                INCOD,INREF,OTCOD,OTREF,TSTFBT
      COMMCN/ICHAR/OO,II,MM,TT,NN,YY
      EQUIVALENCE/LOGIC/SEARCH,DIAG,SYNC,LSS,WRITE,CHCOL,ONE
      LOGICAL SEARCH,DIAG,SYNC,LSS,WRITE,CHCOL,ONE
C
C**************************** BEGIN PROGRAM ****************************
C
C
C    BEGIN DECODE LOOP; RETRIEVE NEXT CODE WORD LENGTH (L)
C
 1000 CONTINUE
      L=0
      WRDBUF=0
      LPACK=0
 1002 LENBIT=CODE(1,INDEX,COLOR)
      CALL GETLG(LENBIT,MODE,LBITS,L)
      IF(DIAG) WRITE(6,1003) LENBIT,MODE,LBITS,L
 1003 FORMAT(2I6,Z8,I6)
      GO TO (1040,1200,1205,1190), MODE
      STOP 1040
 1040 CONTINUE
      IF(LBITS.EQ.CODE(3,INDEX,COLOR)) GO TO 1100
C
C    NO MATCH; ADVANCE CODE WORD INDEX VIA DECODE THREAD
C
       INDEX=CODE(2,INDEX,COLOR)
       IF(INDEX.GE.93) GO TO 1190
      IF(CODE(1,INDEX,COLOR).EQ.LENBIT) GO TO 1040
C
C    CODE WORD LONGER; FROM THE TOP
C
      GO TO 1002
C
C    MATCH FOUND
C
 1100 CONTINUE
      CDELP=CDELP+L
C
C    NOT AN EOL
C
C
C    TEST FOR MAKE UP OR TERMINATING CODE
C
      RUNLEN=INDEX-1
      IF(INDEX.GE.65) RUNLEN=(INDEX-64)*64
      IF(RUNLEN.EQ.0) GO TO 1160
      IF(COLOR.EQ.1) GO TO 1155
      IF(RUNLEN.LT.0) STOP 1100
C
C    ADD BLACK RUN TO OUTPUT BUFFER
C
      DO 1150 I=1,RUNLEN
      CALL MI23(COLOR-1,OTBUF(1,OTCOD),OTELP,1)
      OTELP=OTELP+1
      IF(OTELP-1.GT.PELMAX) GO TO 1180
 1150 CONTINUE
      GO TO 1160
C
C    ADD WHITE RUN TO OUTPUT BUFFER (BY DEFAULT)
C
 1155 CONTINUE
      OTELP=OTELP+RUNLEN
      IF(OTELP-1.GT.PELMAX) GO TO 1180
C
C OUTPUT LINE LESS THAN OR EQUAL TO MAX SPECIFIED
C
 1160 CONTINUE
      IF(INDEX.LT.65) GO TO 1170
      INDEX=3
      GO TO 1000
C
C    RUN ADDED TO OUTPUT LINE; LENGTH LESS THAN OR EQUAL TO PELMAX (1)
C
 1170 CONTINUE
      CHCOL=.TRUE.
      STATUS=1
      RETURN
C
C    RUN ADDED UNTIL PELMAX EXCEEDED; LINE TOO LONG (2)
```

```
C
 1180 CONTINUE
      IF(DIAG) WRITE(6,1185) (CTBUF(I,CTCOD),I=1,60)
 1185 FORMAT(6Z10)
      STATUS=2
      RETURN
C
C   NO MATCH FOUND IN CODE TABLE (3)
C
 1190 CONTINUE
      STATUS=3
      RETURN
C
C   EOL1 DETECTED (4)
C
 1200 CONTINUE
      STATUS=4
      RETURN
C
C   EOL2 DETECTED (5)
C
 1205 CONTINUE
      STATUS=5
      RETURN
      E N D
      SUBROUTINE TWOGER(INDEX,CCLCR,STATUS,L)
      IMPLICIT INTEGER(A-Z)
C******* LABELED COMMON /G32BIT/ *******
C
      COMMON /G32BIT/MASK(32),COMASK(32),LIBIT(32),LZBIT(32)
      INTEGER MASK,COMASK,LIBIT,LZBIT
C
      COMMON/BUFF/PELBUF(60,2),CDBUF(240),
     *            OTBUF(60,2),STFBUF(240), STAT(3000)
      COMMON/HUFF/CODE(3,92,2),CODS(3,68,6),PREDCT(16),NPRED(16),
     *            CTABLE(16),CSTART(16),STBUF(1728),STRUN(1728)
      COMMON/ERAY/ERRORS(2500)
C******************** FILE DEFINITIONS ********************
C
      COMMON/FILES/TERM,LPFIL,PELFIL,OTFIL,ERFIL
C
C********************** LABELLED COMMON VARIABLES **********************
C
      COMMON/IVAR/PELMAX,VRES,EPHASE,CMPMAX,ERRMOD,LINMAX,K
      COMMON/PVAR/IN_NNO,OTLNNO,OTELW,INELP,CDELP,OTELP,CDE_W,
     *            CDELCT,INELCT,TCDATA,TCDEL,ERRPNT,ERRUFF,ERRLIM,
     *            ERRCNT,INLNCT,CONSEC,LNNOBF,ZCNT,WRDBUF,LPACK,
     *            INCOD,INREF,CTCOD,OTREF,TSTFBT
      COMMON/ICHAR/DD,II,MM,TT,NN,YY
      COMMON/LOGIC/SEARCH,DIAG,SYNC,LSS,WRITE,CHCOL,ONE
      LOGICAL SEARCH,DIAG,SYNC,LSS,WRITE,CHCOL,ONE
      INTEGER STCNT(16)
C
C************************ BEGIN PROGRAM ********************************
C
C   INITIALIZE
C
      DO 100 I=1,16
      STCNT(I)=0
  100 CONTINUE
C
      DO 7000 P=1,PELMAX
C
C   FIND STATE OF NEXT PREDICTED PEL
C
      IF(P-1) 611,612,613
  611 STOP 611
  612 CONTINUE
      PEL1=IAB(OTBUF(1,OTREF),P,2)
      PEL2=0
      CALL MI23(PEL1,PEL2,32-2,2)
      GO TO 615
  613 CONTINUE
      PEL2=IAB(OTBUF(1,OTCOD),P-1,1)
      PEL1=IAB(OTBUF(1,OTREF),P-1,3)
      CALL MI23(PEL1,PEL2,32-3,3)
  615 CONTINUE
      SP1=PEL2+1
C
      IF(STCNT(SP1).NE.0) GO TO 1155
      T=CTABLE(SP1)
      INDEX=CSTART(SP1)
```

```
C
C     BEGIN DECODE LOOP: RETRIEVE NEXT CODE WORD LENGTH (L)
C
 1000 CONTINUE
      L=0
      WRBBUF=0
      LPACK=0
 1002 LENBIT=CODS(1,INDEX,T)
      CALL GETLG(LENBIT,MODE,LBITS,L)
      IF(DIAG) WRITE(6,1003) LENBIT,MODE,LBITS,L
 1003 FORMAT(2I6,Z12,I6)
      GO TO (1040,1200,1205,1190), MODE
      STOP 1040
 1040 CONTINUE
      IF(LBITS.EQ.CODS(3,INDEX,T)) GO TO 1100
C
C     NO MATCH; ADVANCE CODE WORD INDEX VIA DECODE THREAD
C
      INDEX=CODS(2,INDEX,T)
      IF(INDEX.GE.67.AND.T.EQ.1) GO TO 1190
      IF(INDEX.GE.66.AND.T.EQ.2) GO TO 1190
      IF(INDEX.GE.34.AND.T.GE.3) GO TO 1190
      IF(CODS(1,INDEX,T).EQ.LENBIT) GO TO 1040
C
C     CODE WORD LONGER; FROM THE TOP
C
      GO TO 1002
C
C     MATCH FOUND
C
 1100 CONTINUE
      CDELP=CDELP+L
C
C     NOT AN EOL
C
      RUN=INDEX
      GO TO (1110,1120,1130,1130,1130,1130),T
C
C                1     2     3     4     5     6
C
      STOP 1100
 1110 CONTINUE
      IF(INDEX.EQ.66) GO TO 1140
      IF(INDEX.EQ.65) RUN=PELMAX+1
      GO TO 1150
 1120 CONTINUE
      IF(INDEX.EQ.65) GO TO 1140
      GO TO 1150
 1130 CONTINUE
      IF(INDEX.EQ.33) GO TO 1140
      GO TO 1150
 1140 CONTINUE
      L=0
      WRBBUF=0
      LPACK=0
      LENBIT=11
      CALL GETLG(LENBIT,MODE,LBITS,L)
      IF(DIAG) WRITE(6,1003) LENBIT,MODE,LBITS,L
      GO TO (1145,1200,1205,1190),MODE
      STOP 1145
 1145 CONTINUE
      CDELP=CDELP+L
      RUN=LBITS
 1150 CONTINUE
      STCNT(SP1)=RUN
 1155 CONTINUE
      IF(STCNT(SP1)-1) 1190,1160,1165
C
C     INCORRECT PREDICTION
C
 1160 CONTINUE
      IF(NPRED(SP1)) 1161,1163,1162
 1161 STOP 1161
 1162 CALL MI28(1,OTBUF(1,OTCOD),OTELP,1)
 1163 STCNT(SP1)=0
      GO TO 1170
C
C     CORRECT PREDICTION
C
 1165 CONTINUE
      IF(PREDCT(SP1)) 1161,1168,1167
 1167 CALL MI28(1,OTBUF(1,OTCOD),OTELP,1)
```

```
 1168 STCNT(SPI)=STCNT(SPI)-1
 1170 OTELP=OTELP+1
 7000 CONTINUE
      IF(DIAG) WRITE(6,1185) (CTBUF(I,CTCOD),I=1,60)
 1185 FORMAT(6Z10)
C
C    CHECK FOR ERRORS
C
      DO 1175 I=1,16
      IF(STCNT(I).NE.0.AND.STCNT(I).NE.1) GO TO 1190
 1175 CONTINUE
      STATUS=1
      RETURN
C
C    NO MATCH FOUND IN CODE TABLE (3)
C
 1190 CONTINUE
      STATUS=3
      RETURN
C
C    EOL1 DETECTED (4)
C
 1200 CONTINUE
      STATUS=4
      RETURN
C
C    EOL2 DETECTED (5)
C
 1205 CONTINUE
      STATUS=5
      RETURN
      E N D
      SUBROUTINE GCODLR(LENGTH,POLAR,CDELCT,CDDATA)
C
      IMPLICIT INTEGER(A-Z)
      COMMON/BUFF/PELBUF(60,2),CDBUF(240),
     *            OTBUF(60,2),STFBUF(240), STAT(3000)
      COMMON/HUFF/CODE(3,92,2),CODS(3,60,6),PREDCT(16),NPRED(16),
     *            CTABLE(16),CSTART(16),STBUF(1728),STRUN(1728)
      COMMON/ERAY/ERRORS(2500)
C
C********************************* BEGIN PROGRAM ***********************
C
C    INITIALIZE MAKE UP CODE, MAKE UP CODE LENGTH
C
      MCODE=0
      MLENG=0
C
C    CHECK INPUTS
C
      IF(POLAR.LT.1.OR.POLAR.GT.2) CALL EXIT
      IF(LENGTH.LT.0.OR.LENGTH.GT.1728) CALL EXIT
C
      IF(LENGTH.LE.63) GO TO 10
C
C    CALCULATE MAKE UP CODE INDEX, CODE, LENGTH
C     AND WRITE TO CODE LINE
C
      INDEX=LENGTH/64+64
      MCODE=CODE(3,INDEX,POLAR)
      MLENG=CODE(1,INDEX,POLAR)
      CALL MI23(MCODE,CDBUF,CDELCT+1,MLENG)
      CDELCT=CDELCT+MLENG
      CDDATA=CDDATA+MLENG
C
C    CALCULATE TERMINATING CODE INDEX, CODE, LENGTH
C     AND ADD TO CODE LINE
C
   10 CONTINUE
      INDEX=MOD(LENGTH,64)+1
      TCODE=CODE(3,INDEX,POLAR)
      TLENG=CODE(1,INDEX,POLAR)
      CALL MI23(TCODE,CDBUF,CDELCT+1,TLENG)
      CDELCT=CDELCT+TLENG
      CDDATA=CDDATA+TLENG
C
      RETURN
      E N D
      SUBROUTINE STUFF1(CDBUF,STFBUF,STFBIT,CDELCT)
      IMPLICIT INTEGER(A-Z)
      DIMENSION CDBUF(240),STFBUF(240)
C******* LABELED COMMON /G32BIT/ *******
```

```
C
      COMMON /S32BIT/MASK(32),COMASK(32),LIBIT(32),LZBIT(32)
      INTEGER MASK,COMASK,LIBIT,LZBIT
C
C
C     INITIALIZE STFBUF TO 0
C
      DO 50 I=2,240
      STFBUF(I)=0
   50 CONTINUE
      LICNT=0
      I=32+1+13
      J=I
      STFBUF(1)=CDBUF(1)
C
C   PICK UP EOL
C
      LSS=I48(CDBUF(2),1,13)
      CALL MI24(LSS,STFBUF(2),1,13)
  100 CONTINUE
      POL=I48(CDBUF,I,1)
      IF(POL.EQ.1) GO TO 110
      LICNT=LICNT+1
      GO TO 150
  110 CONTINUE
      LICNT=0
      CALL MI24(POL,STFBUF,J,1)
  150 CONTINUE
      I=I+1
      J=J+1
      IF(LICNT.LE.9) GO TO 200
      CALL MI28(1,STFBUF,J,1)
      LICNT=0
      J=J+1
C
C   TEST IF FINISHED
C
  200 CONTINUE
      IF(I.LE.CDELCT) GO TO 100
      STFBIT=J-1-CDELCT
      CDELCT=J-1
      RETURN
      END
      BLOCK DATA
C
      IMPLICIT INTEGER(A-Z)
C************************ FILE DEFINITIONS *******************
C
      COMMON/FILES/TERM,LPFIL,PELFIL,OTFIL,ERFIL
C
      COMMON/BUFF/PELBUF(60,2),CDBUF(240),
     *               OTBUF(60,2),STFBUF(240), STAT(3000)
      COMMON/HUFF/CODE(3,92,2),CCDS(3,68,6),PREDCT(16),NPRED(16),
     *               CTABLE(16),CSTART(16),STBUF(1728),STRUN(1729)
      COMMON/ERAY/ERRORS(2500)
C******************** LABELLED COMMON VARIABLES **********************
C
      COMMON/IVAR/PELMAX,VRES,EPHASE,CMPMAX,ERRMOD,LINMAX,K
      COMMON/PVAR/INLNNO,OTLNNO,OTELW,INELP,CDELP,OTELP,CDELW,
     *               CDELCT,INELCT,TCDATA,TCDEL,ERRPNT,ERRUFF,ERRLIM,
     *               ERRCNT,INLNCT,CONSEC,LNNCBF,ZCNT,#ROBUF,LPACK,
     *               INCOD,INREF,OTCOD,OTREF,TSTFBT
      COMMON/ICHAR/DD,II,MM,TT,NN,YY
      COMMON/LOGIC/SEARCH,DIAG,SYNC,LSS,WRITE,CHCOL,ONE
      LOGICAL SEARCH,DIAG,SYNC,LSS,WRITE,CHCOL,ONE
C
      DATA PREDCT/0,1,0,1,0,1,1,1,0,0,0,1,0,1,0,1/
      DATA NPRED /1,0,1,0,1,0,0,0,1,1,1,0,1,0,1,0/
      DATA CTABLE/1,3,4,6,6,6,5,6,6,5,6,6,6,4,3,2/
      DATA CSTART/66,1,2,2,2,2,1,2,2,1,2,2,2,2,1,2/
C
      DATA TERM,LPFIL,PELFIL,OTFIL,ERFIL/5,6,1,2,3/
      DATA DD,II,MM,TT,NN,YY/'D','I','M','T','N','Y'/
      DATA PELMAX,VRES,EPHASE,CMPMAX,ERRMOD,LINMAX/1728,2,0,96,'T',3000/
      DATA K/2/
      DATA DIAG/.FALSE./
C
       DATA CODE(1,  1,1),CODE(2,  1,1),CODE(3,  1,1)/ 8, 70,Z0035/
       DATA CODE(1,  2,1),CODE(2,  2,1),CODE(3,  2,1)/ 6, 90,Z0007/
       DATA CODE(1,  3,1),CODE(2,  3,1),CODE(3,  3,1)/ 4,  4,Z0007/
       DATA CODE(1,  4,1),CODE(2,  4,1),CODE(3,  4,1)/ 4,  5,Z0008/
       DATA CODE(1,  5,1),CODE(2,  5,1),CODE(3,  5,1)/ 4,  6,Z0008/
```

```
DATA CODE(1,  6,1),CODE(2,  6,1),CODE(3,  6,1)/ 4,  7,Z000C/      00012
DATA CODE(1,  7,1),CODE(2,  7,1),CODE(3,  7,1)/ 4,  8,Z000E/      00012
DATA CODE(1,  8,1),CODE(2,  8,1),CODE(3,  8,1)/ 4,  9,Z000F/      00012
DATA CODE(1,  9,1),CODE(2,  9,1),CODE(3,  9,1)/ 5, 10,Z0013/      00012
DATA CODE(1, 10,1),CODE(2, 10,1),CODE(3, 10,1)/ 5, 11,Z0014/      00012
DATA CODE(1, 11,1),CODE(2, 11,1),CODE(3, 11,1)/ 5, 12,Z0007/      00012
DATA CODE(1, 12,1),CODE(2, 12,1),CODE(3, 12,1)/ 5, 65,Z0008/      00012
DATA CODE(1, 13,1),CODE(2, 13,1),CODE(3, 13,1)/ 6, 14,Z0008/      00012
DATA CODE(1, 14,1),CODE(2, 14,1),CODE(3, 14,1)/ 6, 15,Z0003/      00012
DATA CODE(1, 15,1),CODE(2, 15,1),CODE(3, 15,1)/ 6, 16,Z0034/      00012
DATA CODE(1, 16,1),CODE(2, 16,1),CODE(3, 16,1)/ 6, 17,Z0035/      00012
DATA CODE(1, 17,1),CODE(2, 17,1),CODE(3, 17,1)/ 6, 18,Z002A/      00012
DATA CODE(1, 18,1),CODE(2, 18,1),CODE(3, 18,1)/ 6, 19,Z0028/      00012
DATA CODE(1, 19,1),CODE(2, 19,1),CODE(3, 19,1)/ 7, 20,Z0027/      00012
DATA CODE(1, 20,1),CODE(2, 20,1),CODE(3, 20,1)/ 7, 21,Z000C/      00012
DATA CODE(1, 21,1),CODE(2, 21,1),CODE(3, 21,1)/ 7, 22,Z0008/      00012
DATA CODE(1, 22,1),CODE(2, 22,1),CODE(3, 22,1)/ 7, 23,Z0017/      00012
DATA CODE(1, 23,1),CODE(2, 23,1),CODE(3, 23,1)/ 7, 24,Z0003/      00012
DATA CODE(1, 24,1),CODE(2, 24,1),CODE(3, 24,1)/ 7, 25,Z0004/      00012
DATA CODE(1, 25,1),CODE(2, 25,1),CODE(3, 25,1)/ 7, 26,Z0028/      00012
DATA CODE(1, 26,1),CODE(2, 26,1),CODE(3, 26,1)/ 7, 27,Z002B/      00012
DATA CODE(1, 27,1),CODE(2, 27,1),CODE(3, 27,1)/ 7, 28,Z0013/      00012
DATA CODE(1, 28,1),CODE(2, 28,1),CODE(3, 28,1)/ 7, 29,Z0024/      00012
DATA CODE(1, 29,1),CODE(2, 29,1),CODE(3, 29,1)/ 7, 66,Z0018/      00012
DATA CODE(1, 30,1),CODE(2, 30,1),CODE(3, 30,1)/ 8, 31,Z0002/      00012
DATA CODE(1, 31,1),CODE(2, 31,1),CODE(3, 31,1)/ 8, 32,Z0003/      00012
DATA CODE(1, 32,1),CODE(2, 32,1),CODE(3, 32,1)/ 8, 33,Z001A/      00012
DATA CODE(1, 33,1),CODE(2, 33,1),CODE(3, 33,1)/ 8, 34,Z001B/      00012
DATA CODE(1, 34,1),CODE(2, 34,1),CODE(3, 34,1)/ 8, 35,Z0012/      00012
DATA CODE(1, 35,1),CODE(2, 35,1),CODE(3, 35,1)/ 8, 36,Z0013/      00012
DATA CODE(1, 36,1),CODE(2, 36,1),CODE(3, 36,1)/ 8, 37,Z0014/      00012
DATA CODE(1, 37,1),CODE(2, 37,1),CODE(3, 37,1)/ 8, 38,Z0015/      00012
DATA CODE(1, 38,1),CODE(2, 38,1),CODE(3, 38,1)/ 8, 39,Z0016/      00012
DATA CODE(1, 39,1),CODE(2, 39,1),CODE(3, 39,1)/ 8, 40,Z0017/      00012
DATA CODE(1, 40,1),CODE(2, 40,1),CODE(3, 40,1)/ 8, 41,Z0028/      00012
DATA CODE(1, 41,1),CODE(2, 41,1),CODE(3, 41,1)/ 8, 42,Z0029/      00012
DATA CODE(1, 42,1),CODE(2, 42,1),CODE(3, 42,1)/ 8, 43,Z002A/      00012
DATA CODE(1, 43,1),CODE(2, 43,1),CODE(3, 43,1)/ 8, 44,Z002B/      00012
DATA CODE(1, 44,1),CODE(2, 44,1),CODE(3, 44,1)/ 8, 45,Z002C/      00012
DATA CODE(1, 45,1),CODE(2, 45,1),CODE(3, 45,1)/ 8, 46,Z002D/      00012
DATA CODE(1, 46,1),CODE(2, 46,1),CODE(3, 46,1)/ 8, 47,Z0004/      00012
DATA CODE(1, 47,1),CODE(2, 47,1),CODE(3, 47,1)/ 8, 48,Z0005/      00012
DATA CODE(1, 48,1),CODE(2, 48,1),CODE(3, 48,1)/ 8, 49,Z000A/      00012
DATA CODE(1, 49,1),CODE(2, 49,1),CODE(3, 49,1)/ 8, 50,Z000B/      00012
DATA CODE(1, 50,1),CODE(2, 50,1),CODE(3, 50,1)/ 8, 51,Z0052/      00012
DATA CODE(1, 51,1),CODE(2, 51,1),CODE(3, 51,1)/ 8, 52,Z0053/      00012
DATA CODE(1, 52,1),CODE(2, 52,1),CODE(3, 52,1)/ 8, 53,Z0054/      00012
DATA CODE(1, 53,1),CODE(2, 53,1),CODE(3, 53,1)/ 8, 54,Z0055/      00012
DATA CODE(1, 54,1),CODE(2, 54,1),CODE(3, 54,1)/ 8, 55,Z0024/      00012
DATA CODE(1, 55,1),CODE(2, 55,1),CODE(3, 55,1)/ 8, 56,Z0025/      00012
DATA CODE(1, 56,1),CODE(2, 56,1),CODE(3, 56,1)/ 8, 57,Z0058/      00012
DATA CODE(1, 57,1),CODE(2, 57,1),CODE(3, 57,1)/ 8, 58,Z0059/      00012
DATA CODE(1, 58,1),CODE(2, 58,1),CODE(3, 58,1)/ 8, 59,Z005A/      00012
DATA CODE(1, 59,1),CODE(2, 59,1),CODE(3, 59,1)/ 8, 60,Z005B/      00012
DATA CODE(1, 60,1),CODE(2, 60,1),CODE(3, 60,1)/ 8, 61,Z004A/      00012
DATA CODE(1, 61,1),CODE(2, 61,1),CODE(3, 61,1)/ 8, 62,Z004B/      00012
DATA CODE(1, 62,1),CODE(2, 62,1),CODE(3, 62,1)/ 8, 63,Z0032/      00012
DATA CODE(1, 63,1),CODE(2, 63,1),CODE(3, 63,1)/ 8, 64,Z0033/      00012
DATA CODE(1, 64,1),CODE(2, 64,1),CODE(3, 64,1)/ 8, 6[illegible],Z0034/      00012
DATA CODE(1, 65,1),CODE(2, 65,1),CODE(3, 65,1)/ 5, 66,Z001B/      00012
DATA CODE(1, 66,1),CODE(2, 66,1),CODE(3, 66,1)/ 5, 67,Z0012/      00012
DATA CODE(1, 67,1),CODE(2, 67,1),CODE(3, 67,1)/ 6,  2,Z0017/      00012
DATA CODE(1, 68,1),CODE(2, 68,1),CODE(3, 68,1)/ 7, 30,Z0037/      00012
DATA CODE(1, 69,1),CODE(2, 69,1),CODE(3, 69,1)/ 8,  1,Z0036/      00012
DATA CODE(1, 70,1),CODE(2, 70,1),CODE(3, 70,1)/ 8, 71,Z0037/      00012
DATA CODE(1, 71,1),CODE(2, 71,1),CODE(3, 71,1)/ 8, 72,Z0064/      00012
DATA CODE(1, 72,1),CODE(2, 72,1),CODE(3, 72,1)/ 8, 73,Z0065/      00012
DATA CODE(1, 73,1),CODE(2, 73,1),CODE(3, 73,1)/ 8, 74,Z0068/      00012
DATA CODE(1, 74,1),CODE(2, 74,1),CODE(3, 74,1)/ 8, 75,Z0067/      00012
DATA CODE(1, 75,1),CODE(2, 75,1),CODE(3, 75,1)/ 9, 76,Z00CC/      00012
DATA CODE(1, 76,1),CODE(2, 76,1),CODE(3, 76,1)/ 9, 77,Z00CD/      00012
DATA CODE(1, 77,1),CODE(2, 77,1),CODE(3, 77,1)/ 9, 78,Z00D2/      00012
DATA CODE(1, 78,1),CODE(2, 78,1),CODE(3, 78,1)/ 9, 79,Z00D3/      00013
DATA CODE(1, 79,1),CODE(2, 79,1),CODE(3, 79,1)/ 9, 80,Z00D4/      00013
DATA CODE(1, 80,1),CODE(2, 80,1),CODE(3, 80,1)/ 9, 81,Z00D5/      00013
DATA CODE(1, 81,1),CODE(2, 81,1),CODE(3, 81,1)/ 9, 82,Z00D6/      00013
DATA CODE(1, 82,1),CODE(2, 82,1),CODE(3, 82,1)/ 9, 83,Z00D7/      00013
DATA CODE(1, 83,1),CODE(2, 83,1),CODE(3, 83,1)/ 9, 84,Z00D8/      00013
DATA CODE(1, 84,1),CODE(2, 84,1),CODE(3, 84,1)/ 9, 85,Z00D9/      00013
DATA CODE(1, 85,1),CODE(2, 85,1),CODE(3, 85,1)/ 9, 86,Z00DA/      00013
DATA CODE(1, 86,1),CODE(2, 86,1),CODE(3, 86,1)/ 9, 87,Z00DB/      00013
DATA CODE(1, 97,1),CODE(2, 87,1),CODE(3, 87,1)/ 9, 88,Z0098/      00013
```

```
DATA CODE(1, 88,1),CODE(2, 88,1),CODE(3, 88,1)/ 9, 89,Z0099/      00013
DATA CODE(1, 89,1),CODE(2, 89,1),CODE(3, 89,1)/ 9, 91,Z009A/      00013
DATA CODE(1, 90,1),CODE(2, 90,1),CODE(3, 90,1)/ 6, 13,Z0018/      00013
DATA CODE(1, 91,1),CODE(2, 91,1),CODE(3, 91,1)/ 9, 92,Z009B/      00013
DATA CODE(1, 92,1),CODE(2, 92,1),CODE(3, 92,1)/13, 93,Z0002/      00013
DATA CODE(1,  1,2),CODE(2,  1,2),CODE(3,  1,2)/10, 65,Z0037/      00013
DATA CODE(1,  2,2),CODE(2,  2,2),CODE(3,  2,2)/ 3,  6,Z0002/      00013
DATA CODE(1,  3,2),CODE(2,  3,2),CODE(3,  3,2)/ 2,  4,Z0003/      00013
DATA CODE(1,  4,2),CODE(2,  4,2),CODE(3,  4,2)/ 2,  5,Z0002/      00013
DATA CODE(1,  5,2),CODE(2,  5,2),CODE(3,  5,2)/ 3,  2,Z0003/      00013
DATA CODE(1,  6,2),CODE(2,  6,2),CODE(3,  6,2)/ 4,  7,Z0003/      00013
DATA CODE(1,  7,2),CODE(2,  7,2),CODE(3,  7,2)/ 4,  8,Z0002/      00013
DATA CODE(1,  8,2),CODE(2,  8,2),CODE(3,  8,2)/ 5,  9,Z0003/      00013
DATA CODE(1,  9,2),CODE(2,  9,2),CODE(3,  9,2)/ 6, 10,Z0005/      00013
DATA CODE(1, 10,2),CODE(2, 10,2),CODE(3, 10,2)/ 6, 11,Z0004/      00013
DATA CODE(1, 11,2),CODE(2, 11,2),CODE(3, 11,2)/ 7, 12,Z0004/      00013
DATA CODE(1, 12,2),CODE(2, 12,2),CODE(3, 12,2)/ 7, 13,Z0005/      00013
DATA CODE(1, 13,2),CODE(2, 13,2),CODE(3, 13,2)/ 7, 14,Z0007/      00013
DATA CODE(1, 14,2),CODE(2, 14,2),CODE(3, 14,2)/ 8, 15,Z0004/      00013
DATA CODE(1, 15,2),CODE(2, 15,2),CODE(3, 15,2)/ 8, 16,Z0007/      00013
DATA CODE(1, 16,2),CODE(2, 16,2),CODE(3, 16,2)/ 9, 17,Z0018/      00013
DATA CODE(1, 17,2),CODE(2, 17,2),CODE(3, 17,2)/10, 18,Z0017/      00013
DATA CODE(1, 18,2),CODE(2, 18,2),CODE(3, 18,2)/10, 19,Z0018/      00013
DATA CODE(1, 19,2),CODE(2, 19,2),CODE(3, 19,2)/10,  1,Z0008/      00013
DATA CODE(1, 20,2),CODE(2, 20,2),CODE(3, 20,2)/11, 21,Z0067/      00013
DATA CODE(1, 21,2),CODE(2, 21,2),CODE(3, 21,2)/11, 22,Z0068/      00013
DATA CODE(1, 22,2),CODE(2, 22,2),CODE(3, 22,2)/11, 23,Z006C/      00013
DATA CODE(1, 23,2),CODE(2, 23,2),CODE(3, 23,2)/11, 24,Z0037/      00013
DATA CODE(1, 24,2),CODE(2, 24,2),CODE(3, 24,2)/11, 25,Z0028/      00013
DATA CODE(1, 25,2),CODE(2, 25,2),CODE(3, 25,2)/11, 26,Z0017/      00013
DATA CODE(1, 26,2),CODE(2, 26,2),CODE(3, 26,2)/11, 27,Z0018/      00013
DATA CODE(1, 27,2),CODE(2, 27,2),CODE(3, 27,2)/12, 28,Z00CA/      00013
DATA CODE(1, 28,2),CODE(2, 28,2),CODE(3, 28,2)/12, 29,Z00CB/      00013
DATA CODE(1, 29,2),CODE(2, 29,2),CODE(3, 29,2)/12, 30,Z00CC/      00013
DATA CODE(1, 30,2),CODE(2, 30,2),CODE(3, 30,2)/12, 31,Z00CD/      00013
DATA CODE(1, 31,2),CODE(2, 31,2),CODE(3, 31,2)/12, 32,Z0068/      00013
DATA CODE(1, 32,2),CODE(2, 32,2),CODE(3, 32,2)/12, 33,Z0069/      00013
DATA CODE(1, 33,2),CODE(2, 33,2),CODE(3, 33,2)/12, 34,Z006A/      00013
DATA CODE(1, 34,2),CODE(2, 34,2),CODE(3, 34,2)/12, 35,Z006B/      00013
DATA CODE(1, 35,2),CODE(2, 35,2),CODE(3, 35,2)/12, 35,Z00D2/      00013
DATA CODE(1, 36,2),CODE(2, 36,2),CODE(3, 36,2)/12, 37,Z00D3/      00013
DATA CODE(1, 37,2),CODE(2, 37,2),CODE(3, 37,2)/12, 33,Z00D4/      00013
DATA CODE(1, 38,2),CODE(2, 38,2),CODE(3, 38,2)/12, 39,Z00D5/      00013
DATA CODE(1, 39,2),CODE(2, 39,2),CODE(3, 39,2)/12, 40,Z00D6/      00013
DATA CODE(1, 40,2),CODE(2, 40,2),CODE(3, 40,2)/12, 41,Z00D7/      00013
DATA CODE(1, 41,2),CODE(2, 41,2),CODE(3, 41,2)/12, 42,Z006C/      00013
DATA CODE(1, 42,2),CODE(2, 42,2),CODE(3, 42,2)/12, 43,Z006D/      00013
DATA CODE(1, 43,2),CODE(2, 43,2),CODE(3, 43,2)/12, 44,Z00DA/      00013
DATA CODE(1, 44,2),CODE(2, 44,2),CODE(3, 44,2)/12, 45,Z00DB/      00013
DATA CODE(1, 45,2),CODE(2, 45,2),CODE(3, 45,2)/12, 45,Z0054/      00013
DATA CODE(1, 46,2),CODE(2, 46,2),CODE(3, 46,2)/12, 47,Z0055/      00013
DATA CODE(1, 47,2),CODE(2, 47,2),CODE(3, 47,2)/12, 46,Z0056/      00013
DATA CODE(1, 48,2),CODE(2, 48,2),CODE(3, 48,2)/12, 49,Z0057/      00013
DATA CODE(1, 49,2),CODE(2, 49,2),CODE(3, 49,2)/12, 50,Z0064/      00013
DATA CODE(1, 50,2),CODE(2, 50,2),CODE(3, 50,2)/12, 51,Z0065/      00013
DATA CODE(1, 51,2),CODE(2, 51,2),CODE(3, 51,2)/12, 52,Z0052/      00013
DATA CODE(1, 52,2),CODE(2, 52,2),CODE(3, 52,2)/12, 53,Z0053/      00013
DATA CODE(1, 53,2),CODE(2, 53,2),CODE(3, 63,2)/12, 54,Z0024/      00013
DATA CODE(1, 54,2),CODE(2, 54,2),CODE(3, 54,2)/12, 55,Z0037/      00013
DATA CODE(1, 55,2),CODE(2, 55,2),CODE(3, 55,2)/12, 56,Z0038/      00013
DATA CODE(1, 56,2),CODE(2, 56,2),CODE(3, 56,2)/12, 57,Z0027/      00013
DATA CODE(1, 57,2),CODE(2, 57,2),CODE(3, 57,2)/12, 58,Z0028/      00013
DATA CODE(1, 58,2),CODE(2, 58,2),CODE(3, 58,2)/12, 59,Z0058/      00013
DATA CODE(1, 59,2),CODE(2, 59,2),CODE(3, 59,2)/12, 60,Z0059/      00013
DATA CODE(1, 60,2),CODE(2, 60,2),CODE(3, 60,2)/12, 61,Z002B/      00013
DATA CODE(1, 61,2),CODE(2, 61,2),CODE(3, 51,2)/12, 52,Z002C/      00013
DATA CODE(1, 62,2),CODE(2, 62,2),CODE(3, 62,2)/12, 63,Z005A/      00013
DATA CODE(1, 63,2),CODE(2, 63,2),CODE(3, 63,2)/12, 64,Z0066/      00013
DATA CODE(1, 64,2),CODE(2, 64,2),CODE(3, 64,2)/12, 66,Z0067/      00013
DATA CODE(1, 65,2),CODE(2, 65,2),CODE(3, 65,2)/10, 20,Z000F/      00013
DATA CODE(1, 56,2),CODE(2, 66,2),CODE(3, 66,2)/12, 57,Z00C8/      00013
DATA CODE(1, 67,2),CODE(2, 67,2),CODE(3, 67,2)/12, 68,Z00C9/      00013
DATA CODE(1, 63,2),CODE(2, 68,2),CODE(3, 68,2)/12, 59,Z005B/      00013
DATA CODE(1, 59,2),CODE(2, 69,2),CODE(3, 69,2)/12, 70,Z0033/      00013
DATA CODE(1, 70,2),CODE(2, 70,2),CODE(3, 70,2)/12, 71,Z0034/      00013
DATA CODE(1, 71,2),CODE(2, 71,2),CODE(3, 71,2)/12, 72,Z0035/      00013
DATA CODE(1, 72,2),CODE(2, 72,2),CODE(3, 72,2)/13, 73,Z006C/      00013
DATA CODE(1, 73,2),CODE(2, 73,2),CODE(3, 73,2)/13, 74,Z006D/      00013
DATA CODE(1, 74,2),CODE(2, 74,2),CODE(3, 74,2)/13, 75,Z004A/      00013
DATA CODE(1, 75,2),CODE(2, 75,2),CODE(3, 75,2)/13, 76,Z004B/      00013
DATA CODE(1, 76,2),CODE(2, 76,2),CODE(3, 76,2)/13, 77,Z004C/      00013
DATA CODE(1, 77,2),CODE(2, 77,2),CODE(3, 77,2)/13, 78,Z004D/      00013
```

```
DATA CODE(1, 78,2),CODE(2, 78,2),CODE(3, 78,2)/13, 79,Z0072/         00013
DATA CODE(1, 79,2),CODE(2, 79,2),CODE(3, 79,2)/13, 80,Z0073/         00013
DATA CODE(1, 80,2),CODE(2, 80,2),CODE(3, 80,2)/13, 81,Z0074/         00013
DATA CODE(1, 81,2),CODE(2, 81,2),CODE(3, 81,2)/13, 82,Z0075/         00013
DATA CODE(1, 82,2),CODE(2, 82,2),CODE(3, 82,2)/13, 83,Z0076/         00013
DATA CODE(1, 83,2),CODE(2, 83,2),CODE(3, 83,2)/13, 84,Z0077/         00013
DATA CODE(1, 84,2),CODE(2, 84,2),CODE(3, 84,2)/13, 85,Z0052/         00013
DATA CODE(1, 85,2),CODE(2, 85,2),CODE(3, 85,2)/13, 86,Z0053/         00013
DATA CODE(1, 86,2),CODE(2, 86,2),CODE(3, 86,2)/13, 87,Z0054/         00014
DATA CODE(1, 87,2),CODE(2, 87,2),CODE(3, 87,2)/13, 88,Z0055/         00014
DATA CODE(1, 88,2),CODE(2, 88,2),CODE(3, 88,2)/13, 89,Z005A/         00014
DATA CODE(1, 89,2),CODE(2, 89,2),CODE(3, 89,2)/13, 90,Z005B/         00014
DATA CODE(1, 90,2),CODE(2, 90,2),CODE(3, 90,2)/13, 91,Z0064/         00014
DATA CODE(1, 91,2),CODE(2, 91,2),CODE(3, 91,2)/13, 92,Z0065/         00014
DATA CODE(1, 92,2),CODE(2, 92,2),CODE(3, 92,2)/13, 93,Z0002/         00014
DATA CODS(1,  1,1),CODS(2,  1,1),CODS(3,  1,1)/ 6, 12,Z001B/         00014
DATA CODS(1,  2,1),CODS(2,  2,1),CODS(3,  2,1)/ 5,  3,Z0004/         00014
DATA CODS(1,  3,1),CODS(2,  3,1),CODS(3,  3,1)/ 5,  4,Z0001/         00014
DATA CODS(1,  4,1),CODS(2,  4,1),CODS(3,  4,1)/ 5,  5,Z000B/         00014
DATA CODS(1,  5,1),CODS(2,  5,1),CODS(3,  5,1)/ 5,  6,Z001F/         00014
DATA CODS(1,  6,1),CODS(2,  6,1),CODS(3,  6,1)/ 5,  7,Z001C/         00014
DATA CODS(1,  7,1),CODS(2,  7,1),CODS(3,  7,1)/ 5,  8,Z001D/         00014
DATA CODS(1,  8,1),CODS(2,  8,1),CODS(3,  8,1)/ 5,  9,Z000A/         00014
DATA CODS(1,  9,1),CODS(2,  9,1),CODS(3,  9,1)/ 5, 10,Z000F/         00014
DATA CODS(1, 10,1),CODS(2, 10,1),CODS(3, 10,1)/ 5, 11,Z0003/         00014
DATA CODS(1, 11,1),CODS(2, 11,1),CODS(3, 11,1)/ 5,  1,Z0007/         00014
DATA CODS(1, 12,1),CODS(2, 12,1),CODS(3, 12,1)/ 6, 13,Z0018/         00014
DATA CODS(1, 13,1),CODS(2, 13,1),CODS(3, 13,1)/ 6, 14,Z0005/         00014
DATA CODS(1, 14,1),CODS(2, 14,1),CODS(3, 14,1)/ 6, 15,Z0001/         00014
DATA CODS(1, 15,1),CODS(2, 15,1),CODS(3, 15,1)/ 6, 16,Z000B/         00014
DATA CODS(1, 16,1),CODS(2, 16,1),CODS(3, 16,1)/ 7, 17,Z0078/         00014
DATA CODS(1, 17,1),CODS(2, 17,1),CODS(3, 17,1)/ 7, 18,Z0079/         00014
DATA CODS(1, 18,1),CODS(2, 18,1),CODS(3, 18,1)/ 7, 19,Z0026/         00014
DATA CODS(1, 19,1),CODS(2, 19,1),CODS(3, 19,1)/ 7, 20,Z0033/         00014
DATA CODS(1, 20,1),CODS(2, 20,1),CODS(3, 20,1)/ 7, 21,Z0025/         00014
DATA CODS(1, 21,1),CODS(2, 21,1),CODS(3, 21,1)/ 7, 22,Z0027/         00014
DATA CODS(1, 22,1),CODS(2, 22,1),CODS(3, 22,1)/ 7, 23,Z0035/         00014
DATA CODS(1, 23,1),CODS(2, 23,1),CODS(3, 23,1)/ 7, 24,Z0022/         00014
DATA CODS(1, 24,1),CODS(2, 24,1),CODS(3, 24,1)/ 7, 25,Z0024/         00014
DATA CODS(1, 25,1),CODS(2, 25,1),CODS(3, 25,1)/ 7, 26,Z0032/         00014
DATA CODS(1, 26,1),CODS(2, 26,1),CODS(3, 26,1)/ 7, 27,Z0023/         00014
DATA CODS(1, 27,1),CODS(2, 27,1),CODS(3, 27,1)/ 7, 28,Z003A/         00014
DATA CODS(1, 28,1),CODS(2, 28,1),CODS(3, 28,1)/ 7, 29,Z0039/         00014
DATA CODS(1, 29,1),CODS(2, 29,1),CODS(3, 29,1)/ 7, 30,Z0009/         00014
DATA CODS(1, 30,1),CODS(2, 30,1),CODS(3, 30,1)/ 7, 31,Z0014/         00014
DATA CODS(1, 31,1),CODS(2, 31,1),CODS(3, 31,1)/ 7, 32,Z001A/         00014
DATA CODS(1, 32,1),CODS(2, 32,1),CODS(3, 32,1)/ 7, 33,Z0015/         00014
DATA CODS(1, 33,1),CODS(2, 33,1),CODS(3, 33,1)/ 7, 34,Z0019/         00014
DATA CODS(1, 34,1),CODS(2, 34,1),CODS(3, 34,1)/ 8, 35,Z0040/         00014
DATA CODS(1, 35,1),CODS(2, 35,1),CODS(3, 35,1)/ 8, 36,Z0041/         00014
DATA CODS(1, 36,1),CODS(2, 36,1),CODS(3, 36,1)/ 8, 37,Z00F6/         00014
DATA CODS(1, 37,1),CODS(2, 37,1),CODS(3, 37,1)/ 8, 38,Z00F7/         00014
DATA CODS(1, 38,1),CODS(2, 38,1),CODS(3, 38,1)/ 8, 39,Z00F5/         00014
DATA CODS(1, 39,1),CODS(2, 39,1),CODS(3, 39,1)/ 8, 40,Z0069/         00014
DATA CODS(1, 40,1),CODS(2, 40,1),CODS(3, 40,1)/ 8, 41,Z0011/         00014
DATA CODS(1, 41,1),CODS(2, 41,1),CODS(3, 41,1)/ 8, 42,Z0077/         00014
DATA CODS(1, 42,1),CODS(2, 42,1),CODS(3, 42,1)/ 8, 43,Z0071/         00014
DATA CODS(1, 43,1),CODS(2, 43,1),CODS(3, 43,1)/ 8, 44,Z0036/         00014
DATA CODS(1, 44,1),CODS(2, 44,1),CODS(3, 44,1)/ 8, 45,Z0010/         00014
DATA CODS(1, 45,1),CODS(2, 45,1),CODS(3, 45,1)/ 8, 46,Z0003/         00014
DATA CODS(1, 46,1),CODS(2, 46,1),CODS(3, 46,1)/ 8, 47,Z0001/         00014
DATA CODS(1, 47,1),CODS(2, 47,1),CODS(3, 47,1)/ 8, 48,Z0031/         00014
DATA CODS(1, 48,1),CODS(2, 48,1),CODS(3, 48,1)/ 8, 49,Z0030/         00014
DATA CODS(1, 49,1),CODS(2, 49,1),CODS(3, 49,1)/ 8, 50,Z0037/         00014
DATA CODS(1, 50,1),CODS(2, 50,1),CODS(3, 50,1)/ 8, 51,Z0002/         00013
DATA CODS(1, 51,1),CODS(2, 51,1),CODS(3, 51,1)/ 9, 52,Z0085/         0001
DATA CODS(1, 52,1),CODS(2, 52,1),CODS(3, 52,1)/ 9, 53,Z00D0/         00014
DATA CODS(1, 53,1),CODS(2, 53,1),CODS(3, 53,1)/ 9, 54,Z01E9/         00014
DATA CODS(1, 54,1),CODS(2, 54,1),CODS(3, 54,1)/ 9, 55,Z00ED/         00014
DATA CODS(1, 55,1),CODS(2, 55,1),CODS(3, 55,1)/ 9, 56,Z01E0/         00014
DATA CODS(1, 56,1),CODS(2, 56,1),CODS(3, 56,1)/ 9, 57,Z00D1/         00014
DATA CODS(1, 57,1),CODS(2, 57,1),CODS(3, 57,1)/ 9, 58,Z0086/         00014
DATA CODS(1, 58,1),CODS(2, 58,1),CODS(3, 58,1)/ 9, 59,Z00EC/         00014
DATA CODS(1, 59,1),CODS(2, 59,1),CODS(3, 59,1)/ 9, 60,Z0087/         00014
DATA CODS(1, 60,1),CODS(2, 60,1),CODS(3, 60,1)/ 9, 61,Z0084/         0001
DATA CODS(1, 61,1),CODS(2, 61,1),CODS(3, 61,1)/ 9, 62,Z00E1/         00014
DATA CODS(1, 62,1),CODS(2, 62,1),CODS(3, 62,1)/ 9, 63,Z00E0/         00014
DATA CODS(1, 63,1),CODS(2, 63,1),CODS(3, 63,1)/ 9, 64,Z0001/         00014
DATA CODS(1, 64,1),CODS(2, 64,1),CODS(3, 64,1)/10, 67,Z0001/         0001
DATA CODS(1, 65,1),CODS(2, 65,1),CODS(3, 65,1)/ 3,  2,Z0006/         00014
DATA CODS(1, 66,1),CODS(2, 66,1),CODS(3, 66,1)/ 2, 65,Z0002/         00014
DATA CODS(1, 67,1),CODS(2, 67,1),CODS(3, 67,1)/13, 68,Z0002/         0001
```

```
DATA CODS(1, 68,1),CODS(2, 68,1),CODS(3, 68,1)/13, 69,Z0003/          00014
DATA CODS(1,  1,2),CODS(2,  1,2),CODS(3,  1,2)/ 4,  5,Z000B/          00014
DATA CODS(1,  2,2),CODS(2,  2,2),CODS(3,  2,2)/ 3,  4,Z0004/          00014
DATA CODS(1,  3,2),CODS(2,  3,2),CODS(3,  3,2)/ 3,  4,Z0003/          00014
DATA CODS(1,  4,2),CODS(2,  4,2),CODS(3,  4,2)/ 3,  1,Z0006/          00014
DATA CODS(1,  5,2),CODS(2,  5,2),CODS(3,  5,2)/ 4,  6,Z0001/          00014
DATA CODS(1,  6,2),CODS(2,  6,2),CODS(3,  6,2)/ 4,  7,Z0003/          00014
DATA CODS(1,  7,2),CODS(2,  7,2),CODS(3,  7,2)/ 4,  8,Z000F/          00014
DATA CODS(1,  8,2),CODS(2,  8,2),CODS(3,  8,2)/ 5,  9,Z0004/          00014
DATA CODS(1,  9,2),CODS(2,  9,2),CODS(3,  9,2)/ 5, 10,Z000A/          00014
DATA CODS(1, 10,2),CODS(2, 10,2),CODS(3, 10,2)/ 5, 11,Z000B/          00014
DATA CODS(1, 11,2),CODS(2, 11,2),CODS(3, 11,2)/ 5, 65,Z001D/          00014
DATA CODS(1, 12,2),CODS(2, 12,2),CODS(3, 12,2)/ 6, 13,Z0002/          00014
DATA CODS(1, 13,2),CODS(2, 13,2),CODS(3, 13,2)/ 6, 14,Z0010/          00014
DATA CODS(1, 14,2),CODS(2, 14,2),CODS(3, 14,2)/ 6, 15,Z0011/          00014
DATA CODS(1, 15,2),CODS(2, 15,2),CODS(3, 15,2)/ 6, 16,Z0038/          00014
DATA CODS(1, 16,2),CODS(2, 16,2),CODS(3, 16,2)/ 7, 17,Z0001/          00014
DATA CODS(1, 17,2),CODS(2, 17,2),CODS(3, 17,2)/ 7, 18,Z0014/          00014
DATA CODS(1, 18,2),CODS(2, 18,2),CODS(3, 18,2)/ 7, 19,Z0002/          00014
DATA CODS(1, 19,2),CODS(2, 19,2),CODS(3, 19,2)/ 7, 20,Z0007/          00014
DATA CODS(1, 20,2),CODS(2, 20,2),CODS(3, 20,2)/ 7, 21,Z0055/          00014
DATA CODS(1, 21,2),CODS(2, 21,2),CODS(3, 21,2)/ 7, 22,Z0072/          00014
DATA CODS(1, 22,2),CODS(2, 22,2),CODS(3, 22,2)/ 7, 23,Z0056/          00014
DATA CODS(1, 23,2),CODS(2, 23,2),CODS(3, 23,2)/ 7, 24,Z0025/          00014
DATA CODS(1, 24,2),CODS(2, 24,2),CODS(3, 24,2)/ 8, 25,Z0007/          00014
DATA CODS(1, 25,2),CODS(2, 25,2),CODS(3, 25,2)/ 8, 26,Z000C/          00014
DATA CODS(1, 26,2),CODS(2, 26,2),CODS(3, 26,2)/ 8, 27,Z002D/          00015
DATA CODS(1, 27,2),CODS(2, 27,2),CODS(3, 27,2)/ 8, 28,Z002B/          00015
DATA CODS(1, 28,2),CODS(2, 28,2),CODS(3, 28,2)/ 8, 29,Z002F/          00015
DATA CODS(1, 29,2),CODS(2, 29,2),CODS(3, 29,2)/ 8, 30,Z002C/          00015
DATA CODS(1, 30,2),CODS(2, 30,2),CODS(3, 30,2)/ 8, 31,Z0001/          00015
DATA CODS(1, 31,2),CODS(2, 31,2),CODS(3, 31,2)/ 8, 32,Z004E/          00015
DATA CODS(1, 32,2),CODS(2, 32,2),CODS(3, 32,2)/ 8, 33,Z000D/          00015
DATA CODS(1, 33,2),CODS(2, 33,2),CODS(3, 33,2)/ 8, 34,Z0049/          00015
DATA CODS(1, 34,2),CODS(2, 34,2),CODS(3, 34,2)/ 8, 35,Z004C/          00015
DATA CODS(1, 35,2),CODS(2, 35,2),CODS(3, 35,2)/ 8, 36,Z004F/          00015
DATA CODS(1, 36,2),CODS(2, 36,2),CODS(3, 36,2)/ 8, 37,Z00AE/          00015
DATA CODS(1, 37,2),CODS(2, 37,2),CODS(3, 37,2)/ 8, 38,Z00E6/          00015
DATA CODS(1, 38,2),CODS(2, 38,2),CODS(3, 38,2)/ 9, 39,Z0091/          00015
DATA CODS(1, 39,2),CODS(2, 39,2),CODS(3, 39,2)/ 9, 40,Z005D/          00015
DATA CODS(1, 40,2),CODS(2, 40,2),CODS(3, 40,2)/ 9, 41,Z000C/          00015
DATA CODS(1, 41,2),CODS(2, 41,2),CODS(3, 41,2)/ 9, 42,Z0150/          00015
DATA CODS(1, 42,2),CODS(2, 42,2),CODS(3, 42,2)/ 9, 43,Z01CF/          00015
DATA CODS(1, 43,2),CODS(2, 43,2),CODS(3, 43,2)/ 9, 44,Z015F/          00015
DATA CODS(1, 44,2),CODS(2, 44,2),CODS(3, 44,2)/ 9, 45,Z01CE/          00015
DATA CODS(1, 45,2),CODS(2, 45,2),CODS(3, 45,2)/ 9, 46,Z0152/          00015
DATA CODS(1, 46,2),CODS(2, 46,2),CODS(3, 46,2)/ 9, 47,Z009B/          00015
DATA CODS(1, 47,2),CODS(2, 47,2),CODS(3, 47,2)/ 9, 48,Z000D/          00015
DATA CODS(1, 48,2),CODS(2, 48,2),CODS(3, 48,2)/ 9, 49,Z015E/          00015
DATA CODS(1, 49,2),CODS(2, 49,2),CODS(3, 49,2)/ 9, 50,Z0055/          00015
DATA CODS(1, 50,2),CODS(2, 50,2),CODS(3, 50,2)/ 9, 51,Z0151/          00015
DATA CODS(1, 51,2),CODS(2, 51,2),CODS(3, 51,2)/ 9, 52,Z0001/          00015
DATA CODS(1, 52,2),CODS(2, 52,2),CODS(3, 52,2)/10, 53,Z00A8/          00015
DATA CODS(1, 53,2),CODS(2, 53,2),CODS(3, 53,2)/10, 54,Z00B0/          0001
DATA CODS(1, 54,2),CODS(2, 54,2),CODS(3, 54,2)/10, 55,Z00B9/          0001
DATA CODS(1, 55,2),CODS(2, 55,2),CODS(3, 55,2)/10, 56,Z0134/          0001
DATA CODS(1, 56,2),CODS(2, 56,2),CODS(3, 56,2)/10, 57,Z0001/          0001
DATA CODS(1, 57,2),CODS(2, 57,2),CODS(3, 57,2)/10, 58,Z00A9/          0001
DATA CODS(1, 58,2),CODS(2, 58,2),CODS(3, 58,2)/10, 59,Z02A6/          0001
DATA CODS(1, 59,2),CODS(2, 59,2),CODS(3, 59,2)/10, 60,Z02A7/          0001
DATA CODS(1, 60,2),CODS(2, 60,2),CODS(3, 60,2)/10, 61,Z0121/          0001
DATA CODS(1, 61,2),CODS(2, 61,2),CODS(3, 61,2)/10, 62,Z0135/          0001
DATA CODS(1, 62,2),CODS(2, 62,2),CODS(3, 62,2)/10, 63,Z0120/          0001
DATA CODS(1, 63,2),CODS(2, 63,2),CODS(3, 63,2)/11, 64,Z0001/          0001
DATA CODS(1, 64,2),CODS(2, 64,2),CODS(3, 64,2)/12, 66,Z0001/          0001
DATA CODS(1, 65,2),CODS(2, 65,2),CODS(3, 65,2)/ 5, 12,Z0014/          0001
DATA CODS(1,  1,3),CODS(2,  1,3),CODS(3,  1,3)/ 1,  2,Z0001/          0001
DATA CODS(1,  2,3),CODS(2,  2,3),CODS(3,  2,3)/ 3,  3,Z0002/          0001
DATA CODS(1,  3,3),CODS(2,  3,3),CODS(3,  3,3)/ 3,  4,Z0001/          0001
DATA CODS(1,  4,3),CODS(2,  4,3),CODS(3,  4,3)/ 4,  5,Z0006/          0001
DATA CODS(1,  5,3),CODS(2,  5,3),CODS(3,  5,3)/ 4,  6,Z0001/          0001
DATA CODS(1,  6,3),CODS(2,  6,3),CODS(3,  6,3)/ 5,  7,Z000F/          0001
DATA CODS(1,  7,3),CODS(2,  7,3),CODS(3,  7,3)/ 6,  8,Z001C/          0001
DATA CODS(1,  8,3),CODS(2,  8,3),CODS(3,  8,3)/ 6,  9,Z0001/          0001
DATA CODS(1,  9,3),CODS(2,  9,3),CODS(3,  9,3)/ 6, 10,Z0003/          0001
DATA CODS(1, 10,3),CODS(2, 10,3),CODS(3, 10,3)/ 7, 33,Z0005/          0001
DATA CODS(1, 11,3),CODS(2, 11,3),CODS(3, 11,3)/ 8, 12,Z0075/          0001
DATA CODS(1, 12,3),CODS(2, 12,3),CODS(3, 12,3)/ 8, 13,Z0003/          0001
DATA CODS(1, 13,3),CODS(2, 13,3),CODS(3, 13,3)/ 8, 14,Z0008/          0001
DATA CODS(1, 14,3),CODS(2, 14,3),CODS(3, 14,3)/ 9, 15,Z00E8/          0001
DATA CODS(1, 15,3),CODS(2, 15,3),CODS(3, 15,3)/ 9, 16,Z0002/          0001
DATA CODS(1, 16,3),CODS(2, 16,3),CODS(3, 16,3)/ 9, 17,Z0001/          0001
```

```
DATA CODS(1, 17,3),CODS(2, 17,3),CODS(3, 17,3)/ 9, 13,Z0013/          00015
DATA CODS(1, 18,3),CODS(2, 18,3),CODS(3, 18,3)/10, 19,Z01D3/          00015
DATA CODS(1, 19,3),CODS(2, 19,3),CODS(3, 19,3)/10, 20,Z000A/          00015
DATA CODS(1, 20,3),CODS(2, 20,3),CODS(3, 20,3)/10, 21,Z000B/          00015
DATA CODS(1, 21,3),CODS(2, 21,3),CODS(3, 21,3)/10, 22,Z0006/          00015
DATA CODS(1, 22,3),CODS(2, 22,3),CODS(3, 22,3)/10, 23,Z0007/          00015
DATA CODS(1, 23,3),CODS(2, 23,3),CODS(3, 23,3)/10, 24,Z0024/          00015
DATA CODS(1, 24,3),CODS(2, 24,3),CODS(3, 24,3)/10, 25,Z0009/          00015
DATA CODS(1, 25,3),CODS(2, 25,3),CODS(3, 25,3)/10, 26,Z0001/          00015
DATA CODS(1, 26,3),CODS(2, 26,3),CODS(3, 26,3)/10, 27,Z0025/          00015
DATA CODS(1, 27,3),CODS(2, 27,3),CODS(3, 27,3)/11, 28,Z03A5/          00015
DATA CODS(1, 28,3),CODS(2, 28,3),CODS(3, 28,3)/11, 29,Z0001/          00015
DATA CODS(1, 29,3),CODS(2, 29,3),CODS(3, 29,3)/11, 30,Z03A4/          00015
DATA CODS(1, 30,3),CODS(2, 30,3),CODS(3, 30,3)/11, 31,Z0010/          00015
DATA CODS(1, 31,3),CODS(2, 31,3),CODS(3, 31,3)/11, 32,Z0011/          00015
DATA CODS(1, 32,3),CODS(2, 32,3),CODS(3, 32,3)/12, 34,Z0001/          00015
DATA CODS(1, 33,3),CODS(2, 33,3),CODS(3, 33,3)/ 7, 11,Z0038/          00015
DATA CODS(1,  1,4),CODS(2,  1,4),CODS(3,  1,4)/ 3,  3,Z0002/          00015
DATA CODS(1,  2,4),CODS(2,  2,4),CODS(3,  2,4)/ 2,  1,Z0002/          00015
DATA CODS(1,  3,4),CODS(2,  3,4),CODS(3,  3,4)/ 3,  4,Z0001/          00015
DATA CODS(1,  4,4),CODS(2,  4,4),CODS(3,  4,4)/ 3,  5,Z0007/          00015
DATA CODS(1,  5,4),CODS(2,  5,4),CODS(3,  5,4)/ 4,  6,Z0007/          00015
DATA CODS(1,  6,4),CODS(2,  6,4),CODS(3,  6,4)/ 4,  7,Z000C/          00015
DATA CODS(1,  7,4),CODS(2,  7,4),CODS(3,  7,4)/ 5,  8,Z000D/          00015
DATA CODS(1,  8,4),CODS(2,  8,4),CODS(3,  8,4)/ 5,  9,Z0001/          00015
DATA CODS(1,  9,4),CODS(2,  9,4),CODS(3,  9,4)/ 5, 10,Z001A/          00015
DATA CODS(1, 10,4),CODS(2, 10,4),CODS(3, 10,4)/ 6, 11,Z0019/          00015
DATA CODS(1, 11,4),CODS(2, 11,4),CODS(3, 11,4)/ 6, 12,Z0001/          00015
DATA CODS(1, 12,4),CODS(2, 12,4),CODS(3, 12,4)/ 6, 13,Z0006/          00015
DATA CODS(1, 13,4),CODS(2, 13,4),CODS(3, 13,4)/ 6, 33,Z0007/          00015
DATA CODS(1, 14,4),CODS(2, 14,4),CODS(3, 14,4)/ 7, 15,Z0030/          00015
DATA CODS(1, 15,4),CODS(2, 15,4),CODS(3, 15,4)/ 7, 16,Z0001/          00015
DATA CODS(1, 16,4),CODS(2, 16,4),CODS(3, 16,4)/ 7, 17,Z0008/          00015
DATA CODS(1, 17,4),CODS(2, 17,4),CODS(3, 17,4)/ 7, 18,Z006E/          00015
DATA CODS(1, 18,4),CODS(2, 18,4),CODS(3, 18,4)/ 7, 19,Z000B/          00015
DATA CODS(1, 19,4),CODS(2, 19,4),CODS(3, 19,4)/ 8, 20,Z0062/          00015
DATA CODS(1, 20,4),CODS(2, 20,4),CODS(3, 20,4)/ 8, 21,Z0063/          00015
DATA CODS(1, 21,4),CODS(2, 21,4),CODS(3, 21,4)/ 8, 22,Z0012/          00015
DATA CODS(1, 22,4),CODS(2, 22,4),CODS(3, 22,4)/ 8, 23,Z0014/          00015
DATA CODS(1, 23,4),CODS(2, 23,4),CODS(3, 23,4)/ 8, 24,Z0001/          00015
DATA CODS(1, 24,4),CODS(2, 24,4),CODS(3, 24,4)/ 8, 25,Z000E/          00015
DATA CODS(1, 25,4),CODS(2, 25,4),CODS(3, 25,4)/ 8, 26,Z00DF/          00015
DATA CODS(1, 26,4),CODS(2, 26,4),CODS(3, 26,4)/ 9, 27,Z0027/          00015
DATA CODS(1, 27,4),CODS(2, 27,4),CODS(3, 27,4)/ 9, 28,Z0026/          00015
DATA CODS(1, 28,4),CODS(2, 28,4),CODS(3, 28,4)/ 9, 29,Z002A/          00016
DATA CODS(1, 29,4),CODS(2, 29,4),CODS(3, 29,4)/ 9, 30,Z0001/          00016
DATA CODS(1, 30,4),CODS(2, 30,4),CODS(3, 30,4)/ 9, 31,Z002B/          00016
DATA CODS(1, 31,4),CODS(2, 31,4),CODS(3, 31,4)/10, 32,Z0001/          00016
DATA CODS(1, 32,4),CODS(2, 32,4),CODS(3, 32,4)/11, 34,Z0001/          00016
DATA CODS(1, 33,4),CODS(2, 33,4),CODS(3, 33,4)/ 6, 14,Z0036/          00016
DATA CODS(1,  1,5),CODS(2,  1,5),CODS(3,  1,5)/ 1,  2,Z0001/          00016
DATA CODS(1,  2,5),CODS(2,  2,5),CODS(3,  2,5)/ 2,  3,Z0001/          00016
DATA CODS(1,  3,5),CODS(2,  3,5),CODS(3,  3,5)/ 4,  4,Z0001/          00016
DATA CODS(1,  4,5),CODS(2,  4,5),CODS(3,  4,5)/ 4,  5,Z0003/          00016
DATA CODS(1,  5,5),CODS(2,  5,5),CODS(3,  5,5)/ 5,  6,Z0004/          00016
DATA CODS(1,  6,5),CODS(2,  6,5),CODS(3,  6,5)/ 6,  7,Z0001/          00016
DATA CODS(1,  7,5),CODS(2,  7,5),CODS(3,  7,5)/ 6,  8,Z0002/          00016
DATA CODS(1,  8,5),CODS(2,  8,5),CODS(3,  8,5)/ 6,  9,Z000B/          00016
DATA CODS(1,  9,5),CODS(2,  9,5),CODS(3,  9,5)/ 7, 11,Z0001/          00016
DATA CODS(1, 10,5),CODS(2, 10,5),CODS(3, 10,5)/ 7, 11,Z0007/          00016
DATA CODS(1, 11,5),CODS(2, 11,5),CODS(3, 11,5)/ 7, 12,Z0015/          00016
DATA CODS(1, 12,5),CODS(2, 12,5),CODS(3, 12,5)/ 8, 13,Z0001/          00016
DATA CODS(1, 13,5),CODS(2, 13,5),CODS(3, 13,5)/ 8, 14,Z000D/          00016
DATA CODS(1, 14,5),CODS(2, 14,5),CODS(3, 14,5)/ 8, 15,Z0029/          00016
DATA CODS(1, 15,5),CODS(2, 15,5),CODS(3, 15,5)/ 9, 16,Z0019/          00016
DATA CODS(1, 16,5),CODS(2, 16,5),CODS(3, 16,5)/ 9, 17,Z0051/          00016
DATA CODS(1, 17,5),CODS(2, 17,5),CODS(3, 17,5)/10, 18,Z0001/          00016
DATA CODS(1, 18,5),CODS(2, 18,5),CODS(3, 18,5)/10, 19,Z0030/          00016
DATA CODS(1, 19,5),CODS(2, 19,5),CODS(3, 19,5)/10, 20,Z0031/          00016
DATA CODS(1, 20,5),CODS(2, 20,5),CODS(3, 20,5)/10, 21,Z00A1/          00016
DATA CODS(1, 21,5),CODS(2, 21,5),CODS(3, 21,5)/10, 22,Z00A0/          00016
DATA CODS(1, 22,5),CODS(2, 22,5),CODS(3, 22,5)/11, 23,Z0007/          00016
DATA CODS(1, 23,5),CODS(2, 23,5),CODS(3, 23,5)/11, 33,Z0006/          00016
DATA CODS(1, 24,5),CODS(2, 24,5),CODS(3, 24,5)/12, 25,Z0003/          00016
DATA CODS(1, 25,5),CODS(2, 25,5),CODS(3, 25,5)/12, 26,Z0001/          00016
DATA CODS(1, 26,5),CODS(2, 26,5),CODS(3, 26,5)/12, 27,Z0002/          00016
DATA CODS(1, 27,5),CODS(2, 27,5),CODS(3, 27,5)/12, 28,Z0008/          00016
DATA CODS(1, 28,5),CODS(2, 28,5),CODS(3, 28,5)/13, 29,Z0001/          00016
DATA CODS(1, 29,5),CODS(2, 29,5),CODS(3, 29,5)/13, 30,Z0013/          00016
DATA CODS(1, 30,5),CODS(2, 30,5),CODS(3, 30,5)/13, 31,Z0012/          00016
DATA CODS(1, 31,5),CODS(2, 31,5),CODS(3, 31,5)/14, 32,Z0001/          00016
DATA CODS(1, 32,5),CODS(2, 32,5),CODS(3, 32,5)/15, 34,Z0001/          00016
```

```
      DATA CODS(1, 33,5),CODS(2, 33,5),CODS(3, 33,5)/11, 24,Z0005/
      DATA CODS(1,  1,6),CODS(2,  1,6),CODS(3,  1,6)/ 3,  3,Z0002/
      DATA CODS(1,  2,6),CODS(2,  2,6),CODS(3,  2,6)/ 1,  1,Z0001/
      DATA CODS(1,  3,6),CODS(2,  3,6),CODS(3,  3,6)/ 3,  4,Z0001/
      DATA CODS(1,  4,6),CODS(2,  4,6),CODS(3,  4,6)/ 4,  5,Z0001/
      DATA CODS(1,  5,6),CODS(2,  5,6),CODS(3,  5,6)/ 5,  6,Z000F/
      DATA CODS(1,  6,6),CODS(2,  6,6),CODS(3,  6,6)/ 6,  7,Z0019/
      DATA CODS(1,  7,6),CODS(2,  7,6),CODS(3,  7,6)/ 6,  8,Z0001/
      DATA CODS(1,  8,6),CODS(2,  8,6),CODS(3,  8,6)/ 6,  9,Z0002/
      DATA CODS(1,  9,6),CODS(2,  9,6),CODS(3,  9,6)/ 7, 10,Z0031/
      DATA CODS(1, 10,6),CODS(2, 10,6),CODS(3, 10,6)/ 7, 11,Z003A/
      DATA CODS(1, 11,6),CODS(2, 11,6),CODS(3, 11,6)/ 7, 12,Z0030/
      DATA CODS(1, 12,6),CODS(2, 12,6),CODS(3, 12,6)/ 7, 13,Z0039/
      DATA CODS(1, 13,6),CODS(2, 13,6),CODS(3, 13,6)/ 7, 14,Z0001/
      DATA CODS(1, 14,6),CODS(2, 14,6),CODS(3, 14,6)/ 7, 15,Z0038/
      DATA CODS(1, 15,6),CODS(2, 15,6),CODS(3, 15,6)/ 7, 16,Z0037/
      DATA CODS(1, 16,6),CODS(2, 16,6),CODS(3, 16,6)/ 7, 17,Z003A/
      DATA CODS(1, 17,6),CODS(2, 17,6),CODS(3, 17,5)/ 7, 33,Z0006/
      DATA CODS(1, 18,6),CODS(2, 18,6),CODS(3, 18,6)/ 8, 19,Z0001/
      DATA CODS(1, 19,6),CODS(2, 19,6),CODS(3, 19,6)/ 8, 20,Z000F/
      DATA CODS(1, 20,6),CODS(2, 20,6),CODS(3, 20,6)/ 8, 21,Z0076/
      DATA CODS(1, 21,6),CODS(2, 21,6),CODS(3, 21,6)/ 8, 22,Z000E/
      DATA CODS(1, 22,6),CODS(2, 22,6),CODS(3, 22,6)/ 9, 23,Z0009/
      DATA CODS(1, 23,6),CODS(2, 23,6),CODS(3, 23,6)/ 9, 24,Z0001/
      DATA CODS(1, 24,6),CODS(2, 24,6),CODS(3, 24,5)/ 9, 25,Z000A/
      DATA CODS(1, 25,6),CODS(2, 25,6),CODS(3, 25,6)/ 9, 26,Z00EE/
      DATA CODS(1, 26,6),CODS(2, 26,6),CODS(3, 26,6)/ 9, 27,Z00EF/
      DATA CODS(1, 27,6),CODS(2, 27,6),CODS(3, 27,6)/10, 28,Z01B7/
      DATA CODS(1, 28,6),CODS(2, 28,6),CODS(3, 28,6)/10, 29,Z01B6/
      DATA CODS(1, 29,6),CODS(2, 29,6),CODS(3, 29,6)/10, 30,Z0001/
      DATA CODS(1, 30,6),CODS(2, 30,6),CODS(3, 30,6)/10, 31,Z01B0/
      DATA CODS(1, 31,6),CODS(2, 31,6),CODS(3, 31,6)/10, 32,Z01B1/
      DATA CODS(1, 32,6),CODS(2, 32,6),CODS(3, 32,6)/11, 34,Z0001/
      DATA CODS(1, 33,6),CODS(2, 33,6),CODS(3, 33,6)/ 7, 18,Z0035/
     END
      SUBROUTINE ERRMES(PELBUF,OTBUF,PELMAX,VRES,ERRCNT)
C
      IMPLICIT INTEGER(A-Z)
      REAL ESF
C******* LABELED COMMON /G32BIT/ *******
C
      COMMON /G32BIT/MASK(32),COMASK(32),LIBIT(32),LZBIT(32)
      INTEGER MASK,COMASK,LIBIT,LZBIT
C
C*********************** FILE DEFINITIONS *******************
C
      COMMON/FILES/TERM,LPFIL,PELFIL,OTFIL,ERFIL
C
      DIMENSION PELBUF(60), OTBUF(60)
      COMMON/LOGIC/SEARCH,DIAG
      LOGICAL SEARCH,DIAG
C
C**************************** BEGIN PROGRAM ******************************
C
      REWIND PELFIL
      REWIND OTFIL
      ERROR=0
      OTELW=(PELMAX+32-1)/32
      OTLNCT=0
C
C   READ AN ERROR FREE LINE
C
  100 CONTINUE
      READ(1,END=600,ERR=800) INLNNO,INELCT,PELBUF
      IF(MOD(INLNNO-1,VRES).NE.0) GO TO 100
C
C   READ AN ERROR-CORRUPTED LINE
C
  200 CONTINUE
      READ(2,END=500,ERR=800) OTLNNO,OTELCT,OTBUF
      OTLNCT=OTLNCT+1
  300 CONTINUE
C
C   COUNT DIFFERENCES BETWEEN TRANSMITTED AND RECEIVED LINES
C
      DO 450 I=1,OTELW
      IF(OTBUF(I).EQ.PELBUF(I)) GO TO 450
      IF(.NOT.DIAG) GO TO 420
      WRITE(6,410) INLNNO,OTLNNO,I,PELBUF(I),OTBUF(I)
  410 FORMAT(3I8,2Z12)
  420 CONTINUE
      DO 440 J=1,32
```

```
      IF(I4B(OTBUF(I),J,1).NE.I4B(PELBUF(I),J,1)) ERROR=ERROR+1
  440 CONTINUE
  450 CONTINUE
      IF(OTLNNO-INLNNO) 200,100,580
C
C  ERROR LINE NUMBER GREATER THAN GOOD LINE NUMBER
C  COUNT DIFFERENCES BETWEEN GOOD AND ALL WHITE LINE
C
  500 CONTINUE
      DO 550 I=1,OTELW
      IF(PELBUF(I).EQ.0) GO TO 550
      IF(.NOT.DIAG) GO TO 520
      WRITE(6,410) INLNNO,OTLNNO,I,PELBUF(I),OTBUF(I)
  520 CONTINUE
      DO 540 J=1,32
      IF(I4B(PELBUF(I),J,1).NE.0) ERROR=ERROR+1
  540 CONTINUE
  550 CONTINUE
C
  580 READ(1,END=600,ERR=800) INLNNO,INELCT,PELBUF
      IF(MOD(INLNNO-1,VRES).NE.0) GO TO 580
C
      GO TO 300
C
C  CALCULATE ERROR SENSITIVITY FACTOR
C
  600 CONTINUE
      ESF=0.
      IF(ERRCNT.LE.0) GO TO 650
      ESF=FLOAT(ERROR)/FLOAT(ERRCNT)
  650 CONTINUE
C
      WRITE(6,700) ERROR,ERRCNT,ESF,OTLNCT
  700 FORMAT('ONUMBER OF INCORRECT PELS =',I10/
     *       'ONUMBER OF BITS IN ERROR TRANSMITTED =',I10/
     *       'OERROR SENSITIVITY FACTOR =',F12.4/
     *       'OTOTAL NUMBER OF OUTPUT LINES PROCESSED = ',I8)
C
      RETURN
  800 CONTINUE
      STOP 800
      E N D
      SUBROUTINE STATS(LENGTH,INLNCT,DIAG)
      IMPLICIT INTEGER(A-Z)
C
      INTEGER MTT(5),ITT(2,5),LENGTH(INLNCT)
      REAL STT(2,5),SUM,SUMSQ
      LOGICAL DIAG
C************************ FILE DEFINITIONS *******************
C
      COMMON/FILES/TERM,LPFIL,PELFIL,OTFIL,ERFIL
C
      DATA MTT/0,24,48,96,192/
C
C************************BEGIN PROGRAM*************************************
C
      DO 300 I=1,5
      ITT(1,I)=10000
      ITT(2,I)=0
      SUM=0.
      SUMSQ=0.
      DO 100 J=1,INLNCT
C
C   FIND FILLED LINE LENGTH
C
      LEN=MAX0(LENGTH(J),MTT(I))
      IF(DIAG) WRITE(6,50) LEN
   50 FORMAT(I8)
C
C   FIND MINIMUM LINE LENGTH
C
      ITT(1,I)=MIN0(LEN,ITT(1,I))
C
C   FIND MAXIMUM LINE LENGTH
C
      ITT(2,I)=MAX0(LEN,ITT(2,I))
C
C   FIND SUM OF LENGTHS
C
      SUM=SUM+FLOAT(LEN)
      SUMSQ=SUMSQ+(FLOAT(LEN))**2
  100 CONTINUE
```

```
C
C   FIND SAMPLE MEAN AND STANDARD DEVIATION
C
      STT(1,I)=SUM/FLOAT(INLNCT)
      STT(2,I)=SQRT((SUMSQ-(SUM**2)/FLOAT(INLNCT))/FLOAT(INLNCT-1))
  300 CONTINUE
C
      WRITE(6,400)(ITT(1,I),I=1,5)
  400 FORMAT(
     *'0                          MINIMUM TRANSMISSION TIME (4800 BPS)'//
     *'   CODED LINE'/
     *'     LENGTH                0 MS     5 MS    10 MS    20 MS    40 MS'/
     *'   STATISTICS:'//
     *'     MINIMUM',10X,5(I8)//)
      WRITE(6,410)(ITT(2,I),I=1,5)
  410 FORMAT(
     *'     MAXIMUM',10X,5(I8)//)
      WRITE(6,420)(STT(1,I),I=1,5)
  420 FORMAT(
     *'     SAMPLE MEAN',9X,5(F8.2)//)
      WRITE(6,430)(STT(2,I),I=1,5)
  430 FORMAT(
     *'     STANDARD DEVIATION',2X,5(F8.2))
C
      RETURN
      E N D
0        END OF DCEC UPRINT PROGRAM              LINES PRINTED=  1829
```