A Computer-based System for Production of Braille Music

John Humphreys

A thesis submitted
for the degree of Doctor of Philosophy
to the University of Warwick

Department of Engineering

May 1979

Summary

A computer-based system has been developed for production of braille music without the use of a skilled braillist. The system is designed to be used by an operator having no knowledge of braille or computers. Some ability to read printed music is helpful, but not essential, and no expert musicianship is required.

The input to the system is a coded representation of music notation typed on the keyboard of a graphical display unit by an operator reading printed music. Syntax checking is performed by the program during input to help reduce the number of errors made by the operator. The music is displayed graphically on the screen as it is typed, in a form similar to the printed copy, giving the operator immediate feedback and an opportunity to correct any mistakes.

The music, once stored, may be edited until correct using the keyboard of the terminal in conjunction with a joystick or light pen. Any part of the stored music may be rapidly accessed and displayed on the screen or a digital plotter for checking purposes. The stored music may be archived temporarily or permanently on magnetic tape for subsequent retrieval, in both print and braille forms.

The translation algorithm performs automatic translation to a close approximation to the international braille music code. Although the system is intended primarily to be used without the aid of a braillist, it does allow access to, and editing of, the braille before final output.

The output from the system is the braille music notation corresponding to the input, suitable for embossing directly on an braille terminal connected on-line to the computer. The quality of samples of music of various types produced in braille using this system has been evaluated with the assistance of a number of braille music readers.

The system is suitable for implementation at a braille printing house, and incorporates a general-purpose editor for music notation which may also be useful for other musicological applications.

Contents

List of tables and diagrams

Acknowledgments

I should like to thank:

John Busbridge, Director of Music of the Royal National Institute for the Blind, for advice on the braille music code and RNIB practice; Jennie Cole, for typing music notation to test the method of input to the system; Professor John Douce, my supervisor, for interest and advice throughout the project; Alan Hulme, for help in understanding the idiosyncracies of the Sigma 5 computer and operating system; the sighted musicians who helped to test the system by typing music in their own time, and Tom Maley for introducing them; all the readers of braille music who provided feedback on the quality of samples produced using the system; the Science Research Council, whose financial support made this work possible; and Linda Wooldridge, for secretarial asssistar.ce.

Figure A7:1 is reproduced by permission of the Associated Board of the Royal Schools of Music. The extract from Hershey's Repertory of Graphic Symbols (see appendix A1.3) is taken from Wolcott & Hilsenrath (1976).

Chapter 1. Introduction

1.1 Braille music

Braille is a method of representing information using a pattern of raised dots, usually embossed on paper or plastic sheeting, which may be read with the fingertips by blind people. The original braille code was devised in 1824 by Louis Braille for the French language. Since then, braille codes have been invented for most major natural languages (Mackenzie 1954) and for musical, mathematical, scientific and computer notation. Nearly all braille codes use the same basic six-dot cell, two positions wide by three high, each dot of which may be present or absent, but they differ in the meanings assigned to each of the sixty-four possible distinct cells.

The literary braille codes for most natural languages have an uncontracted form and a contracted form. The uncontracted form, usually known as Grade 1, has an almost one to one correspondence between printed characters and braille cells. The contracted form, usually known as Grade 2, extends the Grade 1 code by representing certain words and groups of letters by fewer braille cells. The use of contractions is governed by rules which depend on position within a word, syllabification and occasionally meaning. Much of the literary braille code forms a subset of the braille music code, since nearly all music includes letters and words.

The first elementary braille code for music notation was originated by Louis Braille, himself an organist and cellist, and first published in 1829. Different variations of this code evolved in different countries. International conferences were held in 1888,

1929 and 1954 in an attempt to standardise the braille music code. The code now in use throughout the world for western music notation is that defined by the official manual (Spanner 1956) embodying the decisions of the 1954 conference held in Paris. It is capable of representing almost all the essential musical information shown in standard western musical notation.

The number of readers of braille music is only a small fraction of the total number of braille readers. There are an estimated 500 to 1000 braille music readers in the United Kingdom compared with about 12,000 readers of literary braille. The major uses of braille music are the teaching of music by blind teachers or to blind pupils, and the learning of parts for performance. It is not normally possible for a blind player to follow an instrumental piece in braille during performance because the reading speed is too slow and their hands are otherwise occupied, although it is possible for a proficient braille music reader to follow a vocal part in braille during performance. For this reason, the bulk of music available in braille is vocal or for piano, organ, or solo instruments; there is little demand for orchestral parts. However, the amount of musical material available in braille is only a small fraction of that available in print, and braille music for certain instruments is particularly difficult to obtain.

1.2 Current methods of braille music production

The sole publisher of braille music in the United Kingdom is the Royal National Institute for the Blind (RNIB), although the Scottish Braille Press also formerly included braille music among its publications. Some blind musicians transcribe individual pieces into

braille for their own use, with the help of a sighted assistant, but the amount and quality of braille music produced in this way is difficult to estimate.

The braille music department of the RNIB has two methods of braille production: general publication and private orders. It employs six transcribing staff who work in pairs (one of each pair being sighted and the other blind), two sighted transcribers who work individually, and two machine operators who are also trained as braille music transcribers.

Music for general publication is transcribed by a pair, who work typically at an estimated rate of seven or eight braille pages a day, using a hand frame. The braille is proof-read by a second pair, and proof-read again by a third pair, including the senior proof-reader. The corrected braille is then transferred to zinc plates by a machine operator, and proof-read again by another pair. The finally corrected plates are used to emboss the required number of copies, which is typically between 50 and 100. About half of these cover advance orders, the remainder being kept in stock. The number of zinc plates produced by the RNIB's braille music department is about 700 to 800 a year.

Music for private orders is transcribed on either a hand-frame or a Perkins brailler by one of the two sighted transcribers working individually. It is proof-read only once, by the original transcriber, possibly with the help of a machine operator. Extra copies of the braille are made on plastic sheeting if required, using a vacuum-forming machine. Private orders which are taken on from blind individuals usually take between a week and two months to

complete, depending on length, urgency and pressure of work.

1.3 Advantages of using a computer

The use of a computer-based system for converting printed musical notation to braille has a number of potential advantages over conventional methods:

(1) The requirement for braille music transcribers is considerably reduced. Braille music transcription is a highly-skilled activity, requiring a year or more of training for a person to become proficient. The RNIB is currently working with two less than its full complement of braille music transcribers. The American Printing House for the Blind (APH) decided in 1971 to investigate the use of computers for braille music transcription because the decline in the number of qualified braille music transcribers led to fears that production of braille music might otherwise soon have to cease.

(2) If sufficiently easy to use, a computer-based system may be potentially more cost-effective than manual methods, in terms of the number of person-hours required to produce a given quantity of braille.

(3) Accuracy is essential in transcribing braille music. Minor errors are often are difficult to correct and may make it necessary to re-braille an entire page. A well-designed computer system can allow easy correction of mistakes at any stage.

(4) Digital storage of braille on magnetic tape is considerably more compact than storage of zinc plates or braille books. Single or

multiple braille copies of individual pieces may be produced to order using a computer-driven embosser, avoiding the need to keep large quantities of braille in stock.

(5) Once printed music notation is stored in digital form, a translation program can be adapted to produce particular variations required in the braille. These include:

(i) Variations with country. Although the braille music code is theoretically international, there are minor variations in usage between countries.

(ii) Variation in amount of detail included. Certain aspects of printed music notation, for example, clef signs, are irrelevant to the braille transcription and are accordingly not normally included in the braille. However, for blind teachers of sighted pupils, a "facsimile" braille copy is required, in which the braille signs for these additional aspects are shown. In particular, the Library of Congress requires facsimile transcriptions.

(iii) Selection of parts. For a piece of music for several instruments or voices, either the whole piece or any combination of parts may be selected for brailling without any further input.

(iv) Variations in format. The standard format for braille keyboard music now used by the RNIB is that known as bar-over-bar. A large amount of braille music remains in stock from before 1960 in the earlier bar-by-bar format. In the event of different formats being required by individuals, or future changes in the standard format, only the program would need to be changed.

(6) Once printed musical notation is stored in digital form, it may be used for various purposes, provided that sufficient detail is included. Thus digital databases of musical notation created for

music printing or analysis may be used for braille translation at little extra cost. Conversely, databases created for braille music production can be made available for these other uses. A number of musicologists have accumulated collections of music in digital form since the mid-1960's (Computers and the Humanities 1972).

1.4 Review of previous work

A number of programs have been developed in various countries for translation of text to contracted literary braille. Some of these are now in regular use for braille production. Less attention has been devoted to automating production of the specialised braille codes, including music. This is due partly to the much smaller demand for material in these braille codes, partly to their considerably more complex and less well-defined rules, and partly to the additional problems of presenting the information to the computer.

The first attempts at a computer-assisted system for braille music production were initiated by William Watkins and John Siems at the American Printing House for the Blind in 1971 (APH 1972-5, Watkins & Siems 1976) because of the decline in the number of music braillists. By 1975, when further development was prevented by lack of funds, two systems had emerged from this work: SAMBA (Systems Activated for Music Braille Automation) and RUMBA (Representations Utilising Music Braille Alphamerics).

SAMBA, the more automated of the two, uses a music typewriter for input of the music notation. This produces coded information on punched cards. The system requires five separate passes to convert the input to braille, the output from each stage forming the input to

the next stage. SAMBA is restricted to single-stave monophonic music and needs a significant amount of human intervention in the braille translation process. It has been used to produce two titles in braille, but the authors suggest that the results of their work on this system are inconclusive and further research is called for.

RUMBA is the only computer-assisted system which has so far been used for production of a significant amount of braille music. It involves relatively little automation of the braille translation. Input is on cards punched from coding sheets prepared by an input specialist, who requires four months training and a thorough knowledge of the braille music code. The layout of the mnemonic symbols on the coding sheet is similar to that of the corresponding braille. RUMBA uses two passes through the data, one to determine the "mode" of the input symbols (musical, literary, etc.) and the other to perform the translation to braille.

Both SAMBA and RUMBA are written in assembly code for the IBM 7040 computer, and are therefore not portable. They require a fairly large amount of core storage (total 350K bytes for SAMBA, 200K bytes for RUMBA, including literary braille translation routines).

Another computer-assisted system for braille music production has been developed by Howard Patrick at the American University (Patrick & Friedman 1975, Patrick & Patrick 1976). This uses the Intermediary Music Language - Music Information Retrieval system developed between 1963 and 1972 at Princeton University, which was designed for the coding and analysis of sixteenth century polyphonic music. Input to the system is via the IML input language punched on cards at a rate of one to one and a half hours per page of printed music. The design of

the IML language does not allow the use of chords, thereby excluding input of keyboard music. The braille translation program is large (400K bytes of core) but is written in FORTRAN and thus it is more portable than SAMBA and RUMBA. It is also designed for vocal music and is not easily adaptable to keyboard music or other music using chords. This project was disbanded in 1977.

A feasibility study of the transcription of music into braille using a computer was carried out by Sally Wilkinson at the Hatfield Polytechnic during 1974-5 (Wilkinson 1975, 1976). Her prototype system accepts input via a teletype, in music, braille or literary modes, using a mnemonic alphanumeric code. The input operator requires some knowledge of braille music. Some braille signs, for example, octave signs, which have no inkprint equivalent, must be specified by the user. Automatic syntax checking is performed on the input, but editing is restricted to cancellation and retyping of the measure currently being input. The program is written in extended Algol 60 for a DEC System 10 computer, and has the advantage of being small (15K bytes of core). Several features of the braille music code were not implemented and further work would be necessary to include these. The conclusion drawn from the project was that, although there were some limitations in the prototype system, the feasibility of computer-assisted braille music transcription had been successfully demonstrated.

The Association RITM-Braille, Paris, has been developing a computer-assisted braille music transcription system since 1975 (Meneau 1978). Music notation is encoded on punched cards or teletype using an alphanumeric code. The input code is proof-read, and corrected using an on-line text editor. The encoder requires some

knowledge of braille music. The major problems encountered were translation of text within the music, and the layout of the braille.

Each of the above systems may be regarded as performing computer-assisted, rather than automatic, braille translation. With more or less assistance from people who have a knowledge of braille music, they produce reasonable "uncontracted" braille music, but are unable to take advantage of many of the abbreviation devices used in the braille music code unless these are explicitly specified in the input. There is still a need for a fully automatic translator usable by a non-braillist and implementing as far as possible all the rules of braille music.

1.5 System design

The computer-based system described in this thesis has been designed with the following considerations in mind:

(1) It should be usable by a relatively unskilled operator: the user should require no knowledge of either musical or literary braille, no experience of using a computer, and as far as possible, little knowledge of music or music notation. It should be quick, easy and pleasant to use, and where possible help the user to avoid mistakes in input of music.

(2) The system should ideally produce braille which is identical to that which would be produced by manual transcription. Minimally, it must produce braille which correctly represents all the necessary information.

(3) It should be suitable for implementation by a service organisation for the blind. It should require no purpose-built or non-standard hardware other than a braille terminal. The software should be portable: computer programs should be written in a widely-available high-level programming language and should be machine independent, for example, with respect to machine word length. The system should be well documented for ease of transference and enhancement. Software should be compact for use on a small or medium sized computer system.

(4) It should have extensive capability for editing and correction of errors which occur in the input of music notation.

(5) There should be no arbitrary limitations on the type and complexity of music notation which can be represented, except restriction to the standard form of western music notation.

(6) It should permit editing of the braille before embossing so that useful braille may be produced in parallel with a long-term development and evaluation of the system. There is in any case no definitive braille transcription of a given piece of music. Individual printing houses should have the opportunity to make minor changes in the braille to conform with their existing conventions.

There are three main stages involved in automating braille music production (see figure 1:1). Firstly, the music notation as printed must be entered into the computer and represented in a digital form. Secondly, this representation must be converted into a corresponding braille representation. Finally the digital representation of the

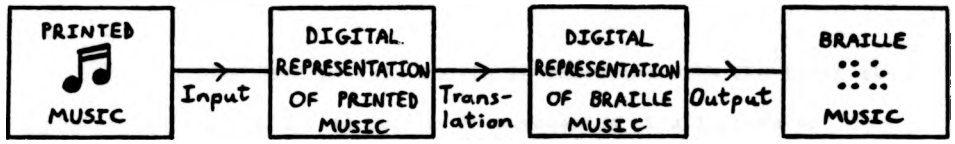


Figure 1:1 Stages in automatic braille music production

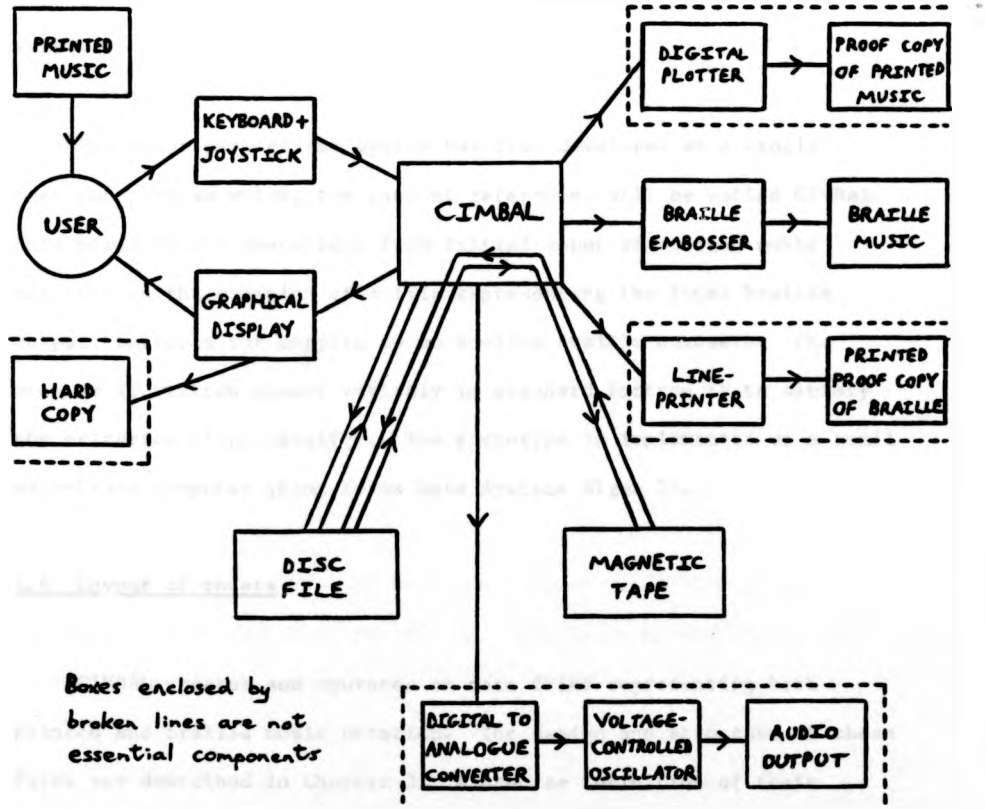


Figure 1:2 System hardware block diagram

braille must be converted into tangible braille.

The work described in this thesis is concerned with the first two stages. The third stage is common to automation of all forms of braille and presents no additional problems specific to braille music. Several models of braille computer terminal are already available for this purpose. In the prototype system, two of these (Triformation LED120 and SAGEM TEM 8BR) have been used to emboss the braille music, but other devices, including stereograph machines for embossing zinc plates, might equally well be used in a production version of the system.

The major part of the system has been developed as a single computer program which, for ease of reference, will be called CIMBAL. This performs all operations from initial input of printed music notation to the creation of a file representing the final braille output, suitable for copying to an on-line braille embosser. The program is written almost entirely in standard Fortran IV to satisfy the criterion of portability. The prototype is implemented on a small main-frame computer (Rank Xerox Data Systems Sigma 5).

1.6 Layout of thesis

CIMBAL creates and operates on data files representing both printed and braille music notation. The design and structure of these files are described in Chapter 2. A precise definition of their format is given in Appendix 5.

CIMBAL can in principle be considered as a number of independent programs which operate successively on the stored music. Many of the

routines are common to several of the different operations, however, and the combination of the separate functions into a single program simplifies development of the system and facilitates compatibility between the successive stages. The operations of the program may be broadly classified into those concerned with creating a digital representation of the printed music notation, and those concerned with automatically creating from this a digital representation of the corresponding braille music notation.

The interactive aspects of the program: initial input, display and editing, leading to a correctly stored digital representation of the printed music notation, are described in Chapter 3. A self-contained Users' Reference Manual, describing in detail the interactive use of the program, is given as Appendix 4. A formal definition of the music input language used is given as Appendix 6.

Problems associated with the translation of the printed music representation into the corresponding braille music representation are described in Chapter 4. This chapter also refers to the interactive aspects of the program relating specifically to the braille.

Chapter 5 describes a pilot evaluation of the method of input to and quality of output from the system. The input is considered in terms of ease of use, accuracy, and cost-effectiveness. The quality of the braille produced is assessed by comparison with the rules and examples given in the braille music manuals, and by direct comparison with braille music produced manually by the RNIB. The help of feedback from braille music readers and transcribers in the development and evaluation of the system is described.

The final chapter contains conclusions and ideas for further development and evaluation of the system

Appendix 1 contains a complete listing of CIMBAL. Appendix 2 gives flow diagrams for algorithms used in the program. Appendix 3 contains program documentation intended in conjunction with Appendices 1 and 2 to enable a programmer to understand, implement and modify the program. Appendix 7 shows examples of printed music with corresponding braille produced by the system.

Chapter 2. Digital storage of music notation

Interest in representing music notation (which for convenience will be referred to simply as "music") in a digital form for processing by a computer developed during the early 1960's. Uses included computer-aided composition, music analysis, automated printing of music, cataloguing of musical themes (by encoding incipits) and, more recently, computer-assisted translation of music to braille. A number of researchers have independently developed methods of input and storage of music convenient for their particular applications. The form of storage is often considered incidental to the application, and not published in any detail.

Because of the variety and comparative rarity of musicological applications of computers, no widely-used standard digital representation for computer processing of music has yet emerged, comparable to the codes used for representation of alphanumeric characters, such as the EBCDIC and ASCII codes. The complexity and diversity of music notation, and the varying requirements of different users make it unlikely that a universal standard will emerge in the near future. The majority of applications use alphanumeric input languages punched on cards to specify the music, and store the data using a standard representation of the characters of the input language. The most comprehensive and widely-used such language (DARMS) is mentioned in chapter 3.

Use of a standard character code for storage makes the data easily transferable between installations. However, while this is adequate for initial encoding of the data and for applications

involving only sequential processing, a more sophisticated structuring of the data is desirable for the purposes of braille translation and on-line editing, two major factors influencing the choice of a digital representation of music for processing by CIMBAL.

The Music Information Retrieval system (Patrick 1976) uses a 40-word magnetic tape record for storage of each note, with fixed numeric fields containing attributes of the note and other parameters currently in effect, such as key and time signatures. This method of storage is far too wasteful of space to be useful for processing large quantities of music on a small computer. Fredlund and Sampson (1973) describe a more structured representation involving a sequential list of pointers to linked lists representing concurrent symbols, but their representation contains some limitations, for example on simultaneous notes, which make it inadequate for a general-purpose braille translator.

In the absence of a published digital representation of music adequately structured and comprehensive for use in a braille translation system, the representation adopted for this system has been designed specifically with the requirements of braille translation and on line-editing in mind. The precise form of this representation is defined in appendix 5. The following sections discuss factors affecting its design.

2.1 Comprehensiveness

At one extreme, some musicological computer applications are concerned mainly or solely with pitch and duration of notes, while at the other, some are concerned with every detail of the printed

notation, including precise layout. A braille translation system requires the representation of all details relating to performance, including all signatures, expression marks, dynamics, words and abbreviations, but not details of the print which do not affect performance and are shown merely for ease of reading. These include, for example, the use of staff lines, beaming together of groups of consecutive short notes, stem direction, and the spacing between notes of different lengths and other symbols. The representation used for CIMBAL can be adapted as it stands for other applications requiring the same or a lesser amount of detail, but would need to be extended for applications requiring more detail.

The representation is designed for the standard form of western music notation based on the five-line stave. It does not cover older forms of notation in general use before the seventeenth century, special notations developed during the twentieth century for contemporary music, or other ethnomusicological notations, since these are not represented in the standard braille music code. It currently includes almost all aspects of music notation which can be shown in braille. Those not yet included because of their infrequent occurrence are listed in chapter 4. Additionally, clef signs are included because they are necessary for input and display of the printed music, and the beaming together of consecutive notes has been included because, while it does not affect the braille in any way, it makes proof-reading of the printed music much easier. The representation contains unused codes which may be assigned meanings later to represent any other desired features.

Given the above restrictions, the representation is designed to avoid arbitrary limitations on size or complexity which would preclude

any piece of music from being fully and correctly stored. This may vary from a nursery rhyme to a symphony without loss of efficiency in storage space.

2.2 Compactness

By comparison with printed literary text, printed music can be a relatively inefficient means of representation in terms of information density, depending on the complexity of notation. Music notation usually contains much regularity and repetition, the difference in pitch between consecutive notes tends to be small, and many of the symbols rarely occur. The braille music code itself is a digital code which, by taking advantage of these aspects of music notation, and by giving each braille cell a number of different meanings distinguished by context, achieves a compactness of representation of simple music several times better relative to print than the contracted literary braille code does for ordinary text. A sheet of printed music may often be transcribed onto a single page of braille, whereas an A4 sized page of 800 words of printed text requires four to five pages of contracted braille.

The braille code itself is not suitable as a basis for digital storage of printed music notation, because neither the conversion from the printed form to braille or braille to printed form is easy to automate. The braille code has, however, influenced to some extent the design of the content and structure of storage used for this project. The information density of this particular representation compares favourably with that of braille, using typically from one to one and a half times the number of bits to represent the printed form of a given piece of music.

A compact digital representation of music is desirable for efficiency of both internal (core) and external random-access (disc) storage. Long-term storage is less critical; with the representation used, over 10,000 pages of fairly dense printed music may be held on a single 2400-foot magnetic tape. Previous programs for manipulation of musical data have tended to be extremely bulky because they use large areas of internal storage containing fixed fields reserved for musical elements which occur infrequently. This may be tolerable where a large computer system is available, but leads to problems on a small system.

Music notation may be more efficiently represented by a structure which reflects more accurately the varying degree of complexity of the notation. This has been achieved by encoding the data in such a way that minimal space is allocated to absent symbols, and data fields are of a suitable minimum size adequate to hold the information without undue complexity of coding. This saving in the required amount of internal and external storage space is partially offset by a slight increase in computation time needed to decode the information. However, the reduced number of data transfers saves input/output time for disc operations, which for braille translation can be a substantial overhead (see chapter 4). The compactness of the representation makes it feasible to use it on a small machine with a limited amount of available core and external random-access storage.

A further reduction in the amount of storage space occupied might theoretically be achieved by taking advantage of repetition in the notation and similarities between consecutive elements. A repeated element might be stored once only with pointers to the full form stored in place of further occurrences, or the pitch of a note might

be stored relative to the pitch of the previous note. Although the resulting saving of space might be quite substantial in the case of lengthy repeated passages, this form of compression has not been adopted for two reasons: uniqueness of representation and ease of modification.

Firstly, the stored form of a given piece of music should be unique. By contrast, the method by which the operator specifies the music should be flexible (see section 3.2). A repeated element may be indicated as such in the input, or specified again, or some combination of these two methods may be used. However the user chooses to specify the music, it should automatically be converted into a standard form for storage. It is easier to expand elements input in abbreviated form into the full representation of their individual components, by copying the original information, than to automatically recognise and compress all possible repeated patterns which the operator may have failed to notice. A process similar to the latter is necessary in translating to braille, but the use of braille repeat signs does not necessarily correspond to exact repetition of the printed notation (see chapter 4).

Secondly, the user should be able to make arbitrary modifications to the stored music to correct errors. They may wish to edit an element of the stored music which was originally specified as a repeat of an earlier element, or which has later been repeated itself. The former case may arise particularly where a passage which is almost, but not exactly, the same as a previous one is initially specified as a repeat to save time, then later edited. The latter case would cause particular problems if the repeated element were stored as such: it would then have to be located and replaced by a fuller representation

to prevent it being unexpectedly modified in the same way as the original. The complexity of structure required for this would outweigh the advantages of any saving in space.

2.3 Hierarchy of storage units

The form of storage has been designed to encode the parameters of music notation directly, rather than through the intermediate form of alphanumeric characters. This has the disadvantage that the information cannot be examined as easily without the aid of software, but allows a more structured and efficient coding. All external storage of musical information is based on the 8-bit byte for portability, this being a unit of information used almost universally on machines ranging from microprocessors to large mainframe computers.

A three-level hierarchy of units of storage addressable by the user has been adopted (see figure 2:1). The three levels, referred to as "files", "measures" and "items", may be regarded as roughly analogous for addressing purposes with the files, lines and characters respectively of a conventional text-editing system. The largest unit, the file, corresponds to a single piece of music, regardless of its length. This is the unit by which information is transferred between random-access disc storage and magnetic tape. A single disc file is used, containing the piece of music currently of interest.

Each file contains a header and three sections of data containing printed music, unformatted braille and formatted braille representations of the music respectively. The braille sections are discussed further in section 2.5 and chapter 4. Each section of the file contains a two-dimensional array of measures. The columns of the

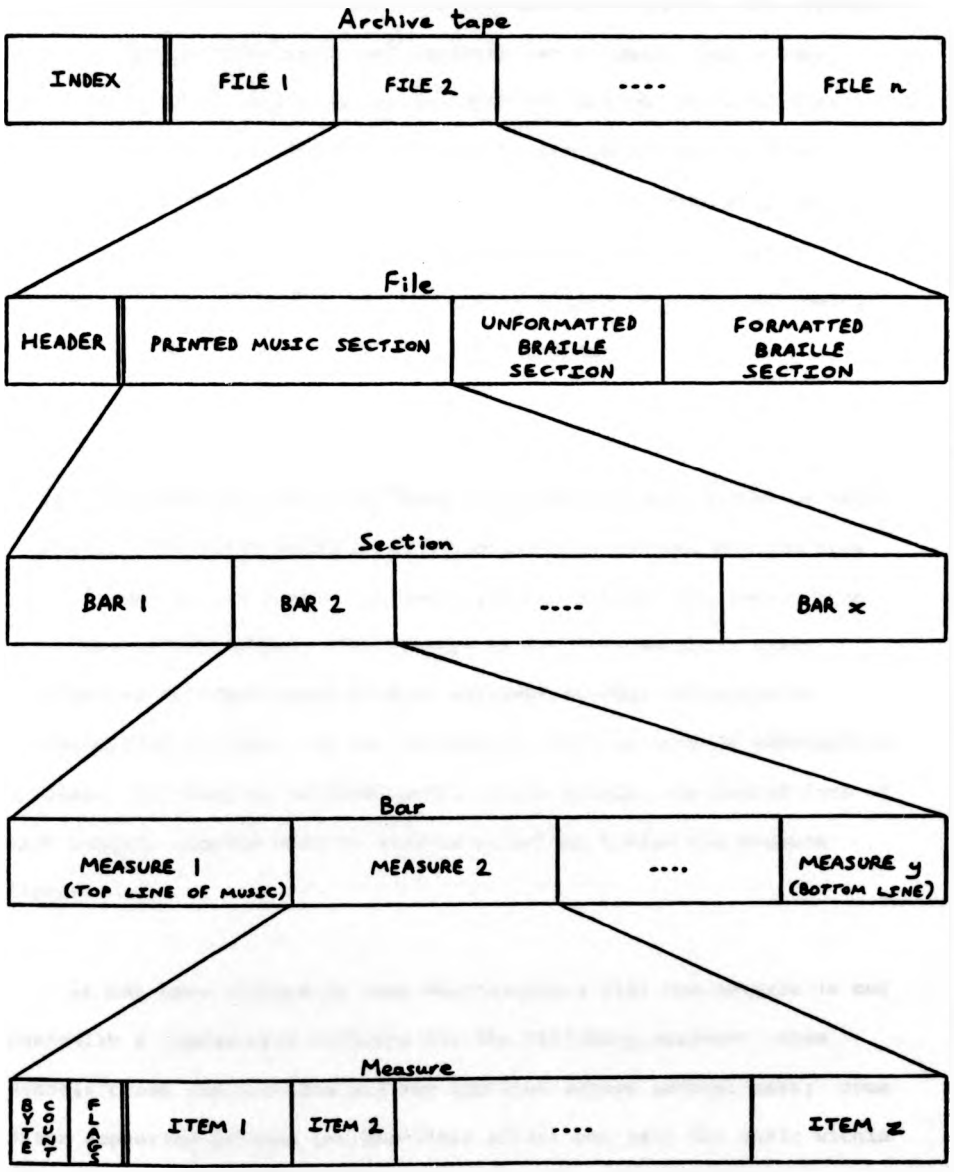


Figure 2:1 Hierarchy of storage units

array are referred to as "bars" and the rows as "lines". For printed music, a bar corresponds to one complete bar of music, and a line corresponds to the music associated with one printed stave of five parallel lines. Thus music for a single instrument has one line, piano music has two, a string quartet four, and an orchestral piece a large number. The file header contains information defining the headings (title, composer, etc.) associated with the piece of music, and the number of bars and lines in each section.

Here, and throughout this thesis, in contrast with ordinary musical terminology, the term "measure" refers to the portion of music written on one stave between two consecutive bar-lines, and the term "bar" refers to the portion of music written between two consecutive bar-lines on all staves. The measure is the unit by which music information is transferred between internal storage and external random-access storage. It is the smallest unit of storage addressable by name. For ease of implementation of the editor, the stored form of each measure depends only on symbols occurring within the measure itself.

It has been claimed by some musicologists that the measure is not musically a fundamental division for the following reasons: some symbols cross the bar-line and may continue across several bars; some signs appearing between two bar-lines affect not only the music within the measure, but also the following music; some music is in any case unmeasured; and the amount of notation within a measure may, and often does, vary within a single piece of music from a single symbol (the whole-measure rest) to a large and complex combination of symbols. David Gomberg (1977) has pointed out that in some music where different instruments have different meters, bar lines may not

even coincide in all parts.

The measure is, nevertheless, a most useful intermediate unit of storage for both on-line editing and braille translation. For editing, the user needs a convenient method of identifying which part of the music is to be changed. As indicated in chapter 3, the use of bar numbers is the most appropriate way of doing this. The measure is the unit by which braille is formatted in the bar-over-bar layout, and is approximately the unit of information which must be compared by the braille translator to determine where measure-repeat signs are appropriate. The question of symbols which cross the bar-line, or whose effect continues across the bar-line, is discussed in section 2.6.

Unmeasured music remains a problem; it can be represented only by an artificial division into measures at suitable points with the inclusion of a special indication that the music is actually unmeasured. Similarly, where bar lines do not coincide in all parts, the "measures" of the file would be artificial divisions not necessarily corresponding to the actual measures of the music. The braille code itself contains no provision for this situation. In music for which there is a demand in braille it is sufficiently rare not to preclude the use of a unit of storage based on the measure.

Each measure comprises a sequence of items, together with an extra "flags" field containing information about changes of state occurring within the measure (see section 2.6). The item is the unit by which information is decoded from its external representation to allow computation on the individual fields contained within the item. In encoded form it occupies one or more bytes of storage. It is the

smallest unit of storage addressable by the user; the method of addressing is described in chapter 3. In general, each item corresponds to a symbol, or group of associated symbols, in the printed music. The assignment of printed symbols to items is described in section 2.7.

A special single-byte code is used to represent a "null" measure to give a compact representation of orchestral music, in which not all the instruments play at any given time, while retaining uniformity of structure. This is interpreted as a whole-measure rest.

2.4 Order of storage

Printed music comprises a set of parallel staves of notation, each of which may, in orchestral music, appear and disappear from time to time. These are largely independent but may occasionally interact, as when keyboard music passes between right and left hand parts. The complexity of structure necessary to represent this interaction properly has been avoided by adopting the braille method of placing a continuous melodic line on one stave with additional signs to indicate change of part, and implied rests inserted in the other stave. Each stave may then be regarded as being independent of other staves. Within each stave there may be more than one melodic line written in parallel ("in-accord") for all or part of a measure.

The need to map the two-dimensional printed music into a linear form for storage means that an ordering must be imposed on the data. Three possible orderings of the information within the file appear to be worth consideration:

(1) A horizontal progression from left to right, with elements occurring at the same horizontal position being stored in order from top to bottom (or bottom to top) as printed;

(2) A vertical progression stave by stave from the uppermost to the lowermost, with the information within each stave being stored in order from left to right; or

(3) A combination of (1) and (2), involving a horizontal progression from left to right of blocks of music, with a vertical progression from top to bottom within each block.

The use of the measure as a fundamental unit of storage excludes the first possibility. The third possibility has been adopted in preference to the second for the order of measures, with each block being equivalent to one bar of music. This minimises the distance between the stored form of measures which are adjacent in print and braille, either horizontally or vertically, improving efficiency of access. A file section may thus be regarded as a two-dimensional array of measures, stored columnwise. Each stored measure is prefaced with a byte containing the length of the encoded measure, which is effectively a pointer to the start of the following measure, used for rapid scanning of the file.

Within a measure the order of items is straightforward unless the measure contains more than one part written in-accord. In this case there are again three possible orderings, corresponding to (1) to (3) above, with "part" replacing "stave". None of these is entirely satisfactory for braille translation. (1) is the most suitable for other purposes since it follows the natural order of the music. In

braille music, however, order (3) is used, the measure being divided into sections if and where the number of parts changes. The order of parts is from bottom upwards in left hand and bass parts. Order (3) has been used for storage of the printed symbols to avoid shuffling them on translation to braille, but it causes problems in identifying exactly which notes marked accidentals apply to.

2.5 Storage of braille

The storage of braille in digital form presents no problems, since braille comprises a linear sequence of 6-bit digital codes. Each braille cell is stored in the low-order 6 bits of a single-byte item. For reasons given in chapter 4, two versions of braille are stored for a given piece of music, the first containing an unformatted representation of the braille and the second a formatted representation derived from the first. The remaining two bits of the byte are used within the unformatted braille section to define extra codes controlling the formatting process.

The braille representation is stored in the same file as the corresponding printed music representation, using the same structure. This allows both forms of the music to be created and accessed using the same routines, and simplifies the operations which apply to both, for example, editing, display, and archiving. Braille translation and various other operations thus effect a mapping of the file into itself. This is made possible by the random-access nature of the file.

Measures correspond to actual measures in the unformatted braille section, or to braille lines in the formatted braille section.

braille music, however, order (3) is used, the measure being divided into sections if and where the number of parts changes. The order of parts is from bottom upwards in left hand and bass parts. Order (3) has been used for storage of the printed symbols to avoid shuffling them on translation to braille, but it causes problems in identifying exactly which notes marked accidentals apply to.

2.5 Storage of braille

The storage of braille in digital form presents no problems, since braille comprises a linear sequence of 6-bit digital codes. Each braille cell is stored in the low-order 6 bits of a single-byte item. For reasons given in chapter 4, two versions of braille are stored for a given piece of music, the first containing an unformatted representation of the braille and the second a formatted representation derived from the first. The remaining two bits of the byte are used within the unformatted braille section to define extra codes controlling the formatting process.

The braille representation is stored in the same file as the corresponding printed music representation, using the same structure. This allows both forms of the music to be created and accessed using the same routines, and simplifies the operations which apply to both, for example, editing, display, and archiving. Braille translation and various other operations thus effect a mapping of the file into itself. This is made possible by the random-access nature of the file.

Measures correspond to actual measures in the unformatted braille section, or to braille lines in the formatted braille section.

Each braille item corresponds to either a braille cell or a format control code. The form of braille storage is independent of any particular embossing device except insofar as the braille translation and formatting are affected by the maximum number of braille cells per line. The number of lines per braille page is not fixed at this stage, braille pagination being controlled by the "flags" field of the formatted braille section.

2.6 State-changing and extended signs

Music notation may be regarded as having a particular "state" at any given point, defined by a set of variables each of which may be altered by a particular music symbol. The meaning of the notation for a given measure is dependent on the clef, key and time signature currently in effect, which are usually defined by symbols occurring outside the measure itself. These symbols are referred to here as "state-changing" symbols.

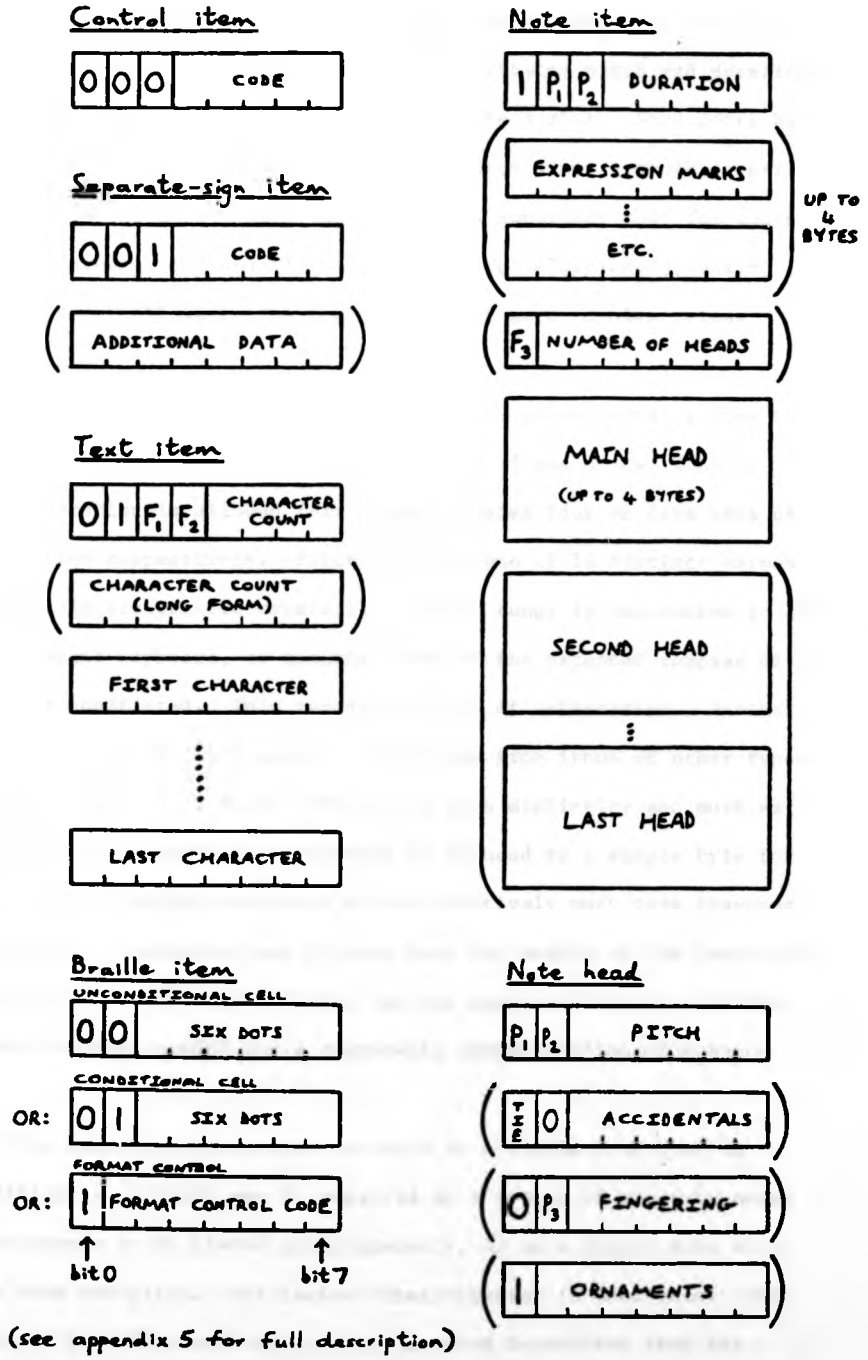
Symbols which may be of arbitrary length - slurs and crescendo and decrescendo signs - will be referred to here as "extended signs". The ottava (8va) is treated as one of these because it directly affects the display of notes within its scope. Extended signs are encoded as two separate items, representing the start and end points of the symbol respectively, and placed before the first note and after the last note covered by the symbol. These items may be regarded as causing a change of state in a binary variable which is set within the scope of the symbol and reset outside it. Because they may occur in different measures, a record of current state must be maintained to ensure that these items are correctly paired and to inform the user when they are not. The state values must be maintained separately for

each line of music since the lines are independent.

The current state must be set correctly to ensure correct interpretation of the music. For random-access to measures, this could be achieved by storing the current state at the start of each measure, but this would conflict with the requirement for independence of measures: changing a key signature at any point, for example, might require the whole file to be re-written from that point on. The state is therefore not stored but evaluated at the time of access. For this reason, random access to the file is restricted to entry points at which the state values are known, either because they correspond to default values or because they have been saved from a previous access at the same point. The file is then scanned sequentially from the entry point to find the required location. A variable stored in the flags field of each measure indicates whether the state at the end of the measure differs from that at the beginning. To prevent unnecessary computation during such a scan, a measure is skipped if it causes no overall change in the state.

2.7 Allocation and coding of items

The allocation of symbols to items has been based on the association of meaning of the printed symbols, rather than their actual position on the printed page. Four categories of item have been defined: one repreaencing notes, rests and their associated symbols ("note item"), one representing text ("text item"), one representing other symbols ("separate-sign item"), and one containing items controlling the display of the music ("control item") (see figure 2:2). The distinction between the last two categories has become blurred during the development of the system.



(see appendix 5 for full description)

Figure 2:2 Coding of items

The most frequently occurring symbol in music notation is the note. Every note has at least the two attributes pitch and duration, implied by the form and position of the note symbol. Some notes have additional attributes, represented by various other symbols appearing adjacent to the note symbol itself. It is important that the coding should represent the majority of simple notes efficiently, while making proper allowance for the minority of more complex notes.

Duration may take one of nine distinct values (ranging from the breve to the 128th note), or 27 if the use of one or two dots to extend duration is allowed for. This occupies four or five bits of information respectively. Pitch may take one of 52 distinct values if accidentals are treated separately and the range is restricted to that of the piano keyboard, or somewhat more if the extended compass of the organ is considered. This requires 6 bits of information. Another bit is required to distinguish a note item from items of other types, giving a total of 12 bits. This could with difficulty and much extra complexity in encoding and decoding be reduced to a single byte for many notes by taking advantage of the relatively much more frequent occurrence of durations and pitches near the middle of the range and by using variable length fields, but two appears to be the minimum number of bytes needed for a reasonable representation of a note.

The decision to represent a chord as a single note item is significant. A chord may be regarded as a number of separate notes which happen to be played simultaneously, or as a single note with more than one pitch. The latter interpretation is used here. The rules of braille music require that a chord be written from the uppermost pitch downwards in a right hand or tenor part, and from the lowermost pitch upwards in a left hand or bass part. To avoid passing

this rule on to the input operator, it is desirable for a chord to be treated as a single entity. The reduction in storage space achieved by representing a chord as a single item is a less important factor.

Given the encoding of a chord as a single note-item, it is convenient to store the attributes which, for a chord, apply to the chord as a whole (note-item indicator bit, and duration) in one byte, and the pitch for each individual head in another. One of the two spare bits in the first byte is used as a presence bit which, when set, indicates the presence of extra symbols associated with the chord as a whole, such as accents and staccato signs, these being encoded in the following byte or bytes. The other spare bit is used to distinguish a single note from a chord, for which the pitch bytes are preceded by a byte containing a count of the number of heads. It is useful to treat a rest as a form of note since it has the same duration attributes. This is encoded in the same form as a chord, but with a value of 1 in the count field and no pitch coding.

The two spare bits for each pitch byte are used as presence bits for symbols which, for a chord, are attached to individual heads: one for the more frequent accidentals and ties, the other for the less frequent fingering and ornaments. These if present are encoded immediately following the pitch byte. Alternative fingering for a given note-head is currently limited to two different values, as in braille.

Text items represent literary text occurring in the printed music, including headings, dynamics, performance directions, and words to be sung (but excluding the stylised "tr" and "Ped" used for trills and pedalling). Sung words and syllables are treated as text items

rather than as part of note items to reflect the separation of words and music in braille, where they are placed on alternate lines, and to allow equally efficient coding of both instrumental and vocal music in the same form. The association of words with notes is defined by the order in which the items are stored, the text item or items preceding the corresponding note.

A short form of text item is used for text strings of 14 characters or less, and a long form is used for longer text strings. Apart from headings, the long form is expected to apply only rarely. Four bits of the initial byte of a text item contain a character count for the short form, or a long form indicator. The character count for the long form is stored in a separate byte. Of the remaining bits of the initial byte, one bit is used to distinguish the item from a note item, one bit to distinguish it from separate-sign and control items, and one bit to distinguish sung text from headings and musical directions. For sung text the remaining bit distinguishes whether it represents part of a word to be continued in a subsequent text item; for musical directions it distinguishes whether they appear above or below the stave. The text itself is stored in the following bytes using the full 89-character set standard EBCDIC code. Some unassigned and control codes are used to represent accented vowels and other special codes affecting the text to braille translation (described in section 4.7).

All stored symbols other than those attached to notes or representing text are represented by control items and separate-sign items. These include the end-points of extended signs. The type of symbol is encoded in a single byte, two bits of which distinguish the item from note and text items. For symbols of a variable nature, such

as clefs, key signatures and time signatures, the additional information is encoded in a second byte. The single bar-line is not encoded because is it implied by the storage structure. Two control item codes are used to define the structure of music written in-accord. The use of some additional items for controlling the display is mentioned in chapter 3.

2.8 Method of access

The disc file contains fixed-size blocks of storage, which are referred to here as "sectors" because in the prototype implementation they correspond to the sectors of the disc used. The first sector is a fixed header sector. The remaining sectors are linked together in up to three separate linked lists which contain the information for the printed music, unformatted braille and formatted braille respectively, and a linked list of the remaining sectors, representing free storage space (see figure 2:3). The use of linked lists is necessary for editing, where the length of a measure may be increased. In this case the sectors containing the edited measure are re-written and the links changed accordingly. For editing of a measure which does not increase its length, the measure is replaced at the original address. Links are maintained by storing the number of the next sector of a list in a two-byte link field at the beginning of each sector. Sectors are transferred between the free storage list and the other lists as required by re-linking.

Access to the file will be referred to as sequential or random with respect to these linked lists and not to the physical file. Thus if sector x is linked to sector y, then the first byte of sector y (excluding the link field) is the next byte in sequence after the last

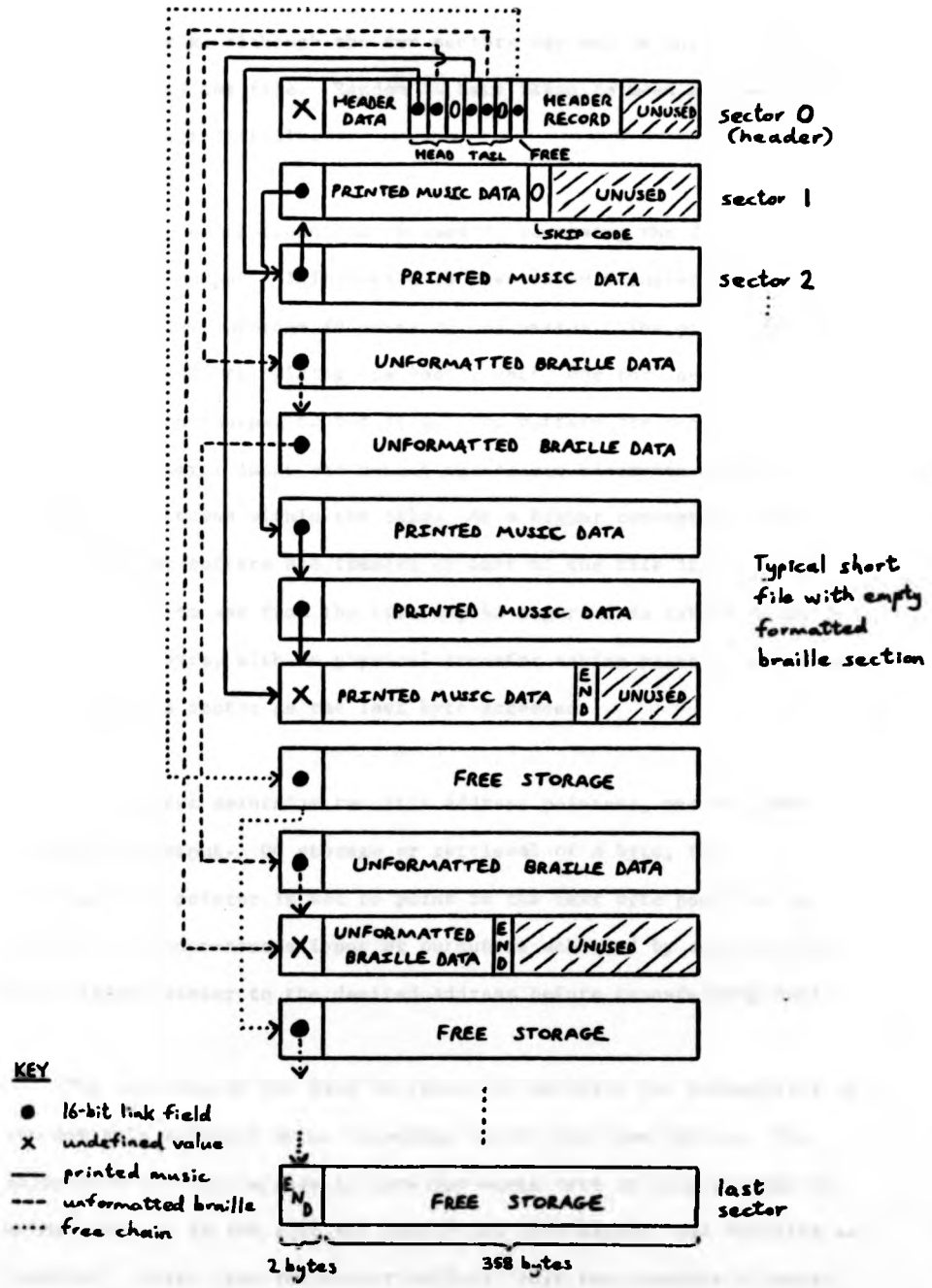


Figure 2:3 Physical layout of disc file

byte of sector x, although the two sectors may not be physically adjacent within the file. Random is here taken to mean non-sequential rather than unpredictable.

A simple paging technique is used in accessing the disc file both for input and output. Information is physically transferred between the file and core storage in units of one sector. The program uses two internal buffers holding one sector each, one for input from the file and one for output to the file. Two buffers are necessary because in general input and output operations alternate at two different positions within the file. At a higher conceptual level, these internal buffers are treated as part of the file itself, and data transfer to and from the file may be regarded as taking place in units of one byte, with no physical transfer taking place if the byte is in the same sector as the last byte accessed.

The program maintains two file address pointers, one for input and one for output. On storage or retrieval of a byte, the corresponding pointer is set to point to the next byte position in sequence. Random-access input or output is achieved by setting the appropriate pointer to the desired address before transferring data.

The ordering of the data is chosen to optimise the probability of successively accessed bytes occurring within the same sector. The allocation of large arrays in core for music data is then avoided by using pointers to the file and retrieving information from the file as required. Apart from the sector buffers, only two measures of music are held in core at any given time. Not more than two items are held in expanded form at any one time. The ability to process music with a large number of staves thus requires insignificant extra core space (2

bytes per stave). Processing time is degraded as the number of staves becomes large, while retaining efficiency in both internal storage space and access time for most music handled, which is likely to have only a few lines.

It was originally anticipated that the braille translator would require random-access output as well as input, but it has proved more satisfactory to place all the random-access requirements on the input to the translator by making forays ahead of the current input position and returning to it, the output remaining sequential. The other operations using random-access output, editing and braille formatting, require only one known address other than the current output position to be saved at any given time for subsequent access.

For input from the file, entry points for random-access are less predictable. A set of pointers to positions within the file is used. These are set up dynamically as the file is scanned. The optimal placing of these pointers is important in reducing random-access time, especially for braille translation, in which a large proportion of time is taken up in searching for potential repeated measures, which are not necessarily close to the current translation position.

2.9 Archiving

A long-term form of storage for both the printed and braille forms of music is an essential part of the overall system. Permanent storage of the printed music is desirable because:

(1) Undetected errors may need subsequent correction;

(2) Improvements and modifications in the braille translator may make

future re-translation necessary;

(3) Similarly, changes to the braille music or literary codes may necessitate re-translation;

(4) A database of music notation could be used for other purposes.

Permanent storage of the braille is desirable so that copies of the braille may be embossed on demand without the need for re-translation.

Open-reel magnetic tape has been used as the medium for long-term storage in the current implementation. Cassette tapes might equally well be used on another machine. The system has been designed on the assumption that available disc storage is restricted. Only one file of music is stored on disc at one time, and the storage space for this is assumed to be available only while the program is actually in use. Magnetic tape is therefore used not only for long-term storage but also for short-term storage between sessions of program use. The system has self-contained facilities for transferring a file from disc to magnetic tape and vice versa, and for making a second copy of a tape. A file of music will not necessarily be input and fully corrected in a single session. Provision has been made for saving a piece of music on magnetic tape at different stages, the successive versions of the same piece being distinguished by a version number. The use of magnetic tape for temporary storage in this way necessitates the deletion of versions of a piece superseded by more complete versions. This requires an ability to copy tapes, which is also needed for integrity of the data.

An index of file names and version numbers is stored at the beginning of each magnetic tape used. Location of the index at the

front of the tape allows instant listing of files on the tape and deletion of unwanted versions of files by marking them as such in the index. It also ensures that the index on the tape remains intact in the event of a computer system failure while a file is being copied to the tape, which would not be the case if the index were written after the data files. It does, however, involve an assumption about the way the index is written to tape when it is updated.

Chapter 3. Input, display and editing

A major consideration in the design of a computer-based system for production of braille music, or any other musicological application, is the problem of initially converting the musical information into a digital form. This contrasts with automation of literary braille production for which input is straightforward using standard keyboard devices such as card punches, teletypes or visual display units, the only significant problem being the specification of the braille layout. The method of input for music is to a certain extent independent of the application for which the music is used once stored. The relative merits of a number of methods which have been used for presenting musical information to a computer are discussed in section 3.1.

Use of digital encoding of music notation by music publishers for automated printing of music on a large scale would be ideal for parallel production of the same music in braille. The information needed for braille music is a subset of that needed for printed music, and could be derived from it without difficulty, provided that the coding reflected the meaning and not merely the layout of symbols. The major cost of separate initial encoding for braille production would be eliminated. Progress has been made in this direction for production of literary braille directly from error-free paper tapes used by some compositors for computer typesetting. Unfortunately, computer-based techniques for printing of music are still at an experimental stage. Music publishers continue to use the traditional methods of manual engraving and lithography.

3.1 Review of input methods

(1) Punched cards

This is the earliest and most widely-used method. The music notation is encoded using an alphanumeric language based on the character set of a standard keypunch. A number of such languages were devised, primarily in the United States, during the 1960's and early 1970's (see Brook 1970a for a brief description, and Styles 1974 for a review, of some of these). Several of these input languages contain restrictions which make them suitable mainly for the specific purposes for which they were originally designed.

One input language, DARMS (Digital Alternate Representation of Music Symbols; Bauer-Mengelberg 1970, Erickson 1975), originally designed for automated printing of music notation, is comprehensive in its ability to represent conventional western music notation. It has been used for encoding music for various purposes at a number of different places. Development of a system based on DARMS is still going on and is expected to require a total of about 25 person-years of work. A complete reference manual for the language has recently been published (Erickson 1977).

The card punch has the advantage of being cheap to use and available at most computer installations. Input languages can be designed to represent the notation in as much detail as is required. Initial input of data is fairly quick relative to other manual methods of encoding.

The major disadvantage of punched card input, mentioned by nearly

all users of this method, is the occurrence of errors in the encoding of the data. The identification and correction of these errors is a tedious and time-consuming process, especially when no graphical proof-reading methods are available and the character string as typed must be laboriously checked against the original print. In particular, errors in the syntax of these input languages have caused programs analysing the data to stop or misinterpret subsequent data. The input code has to be "debugged" in a similar manner to the syntax checking of the source of a computer program, using successive batch runs until the data is syntactically correct.

(2) Teletype or visual display unit

On-line use of an alphanumeric terminal (Wilkinson 1975) has many of the advantages and disadvantages of punched card input. It can alleviate the problem of syntax checking but the question of proof-reading still remains.

(3) Graphical display unit (GDU)

On-line graphical terminals have been used with the aid of software-generated music symbols displayed on the screen (Cantor 1971, Fredlund & Sampson 1973). A symbol is selected from a menu of symbols permanently displayed on the screen by pointing at it with a light pen. The symbol is placed in the desired position on a displayed staff by use of a second light-pen pick and a single keystroke. Use of a refreshing display enables wrongly-placed symbols to be moved to different positions or deleted. Commands typed on the keyboard or selected from the menu indicate which function is required.

Graphical input is a more natural representation of musical notation than alphanumeric coding. Its use is initially easier to learn and it is less susceptible to syntax errors. It permits easy proof-reading and correction of the stored music. It is particularly suitable for computer-assisted music printing because symbols can be moved around to give a desired layout.

Graphics terminals were less widely used at the time many of the punched card codes were originated but are now fairly common, although more expensive than card punches. Use of refreshing graphics is attractive for deletion and re-positioning of symbols but requires extra computing resources to maintain the display and limits the number of symbols which may be displayed at one time. The ease of learning may make speed of input faster than keyboard input for a new user but the extra physical operations required (typically two light pen picks and a keystroke for each symbol instead of one or possibly two keystrokes) are liable to make it slower for an experienced user. The complete set of symbols used in music notation is too large to be displayed on a single menu easily. With suitably elaborate programming this form of input could be designed for operation by a user with no knowledge of music notation by building up an image of the printed page on the screen symbol by symbol, leaving the program to sort the information into order and interpret it.

(4) Music typewriter

The music typewriter has been used as a device which can simultaneously produce hard-copy printed music and a digital encoding (Hiller & Baker 1965, APH 1972-5). It resembles an ordinary typewriter, but has music symbols in place of the usual characters.

It has extra function keys for accurate horizontal and vertical positioning of the paper, and for deleting symbols. The digital coding is output onto punched cards or paper tape.

The music typewriter is relatively easy to use. The typed music may be readily checked against the original and corrected without the use of a computer. It includes much information on layout likely to be useful for automated music printing applications but possibly redundant for other purposes. It is, however, much slower to use than other methods, comparing unfavourably even with hand-copying: for example, a note symbol is built up from its individual components (head, stem and flags) with intermediate keystrokes for positioning the paper; a slur is typed as a series of short lines.

(5) Optical character recognition

An optical character recognition system (DO-RE-MI) for automated input of music notation has been developed by David Prerau (1975). A flying-spot scanner scans the printed page and produces a sequence of values corresponding to the degree of lightness or darkness of the paper at each point. The software involves three stages: firstly, a hardware dependent conversion of the input to an array of bits indicating black or white at each position; secondly, removal of stave lines to isolate the individual symbols; thirdly, identification of these symbols.

This method, if comprehensive and reliable, would be the most satisfactory means of converting music already in print into a digital form, because it is fully automated, giving potentially greater speed and less scope for error than methods using human intervention. The

limitation to music already in print is not likely to be severely restrictive on the application to braille production. DO-RE-MI is restricted in the set of music symbols recognised, and in particular does not recognise letters, which are harder to distinguish because of their closer resemblance to one another in size and shape. It also appears to require excessive computation time.

(6) On-line digitally sampled keyboard

Real-time on-line digital sampling of an organ-type keyboard has been developed for input of music directly from performance (Gillies & Goldsack 1976, Mars 1977, Mars & Cattanach 1977, Tucker & others 1978). A piece of music is played on the instrument in the normal way. The position of each key (depressed or not depressed) is sampled at a typical rate of 20 to 100 times a second, producing a sequence of bits. Note pitches and durations are subsequently computed from this. This form of input is especially suitable for applications where these are the only parameters required. For these it is considerably faster than indirect encoding methods. Text, dynamics, expression marks, and other symbols must be added using another method. Where these constitute a major part of the notation, the effectiveness of on-line sampling is reduced.

3.2 Input language design

The method developed for input of music to CIMBAL is a hybrid of the graphical and alphanumeric coding methods. Of the range of standard computer hardware the GDU is the most natural device for transcribing a graphical notation. Some form of graphical display in the system is necessary in any case for adequate proof-reading of the

stored music. The limitations on speed and comprehensiveness of input using a purely graphical form are avoided by the use of a mainly alphanumeric input language for initial input of music notation. This is similar in some ways to those mentioned above but differs from them in that it is interactively interpreted by software.

Neither the form of storage nor the braille translator is dependent in any way on the original method of input. With suitable software and hardware interfacing, this could be replaced or supplemented by on-line sampled keyboard if this proves to give an overall reduction in input time, or optical character recognition when this is sufficiently developed to read printed music comprehensively and reliably.

The input language is described in detail in appendix 4 (Users' reference manual) and a formal syntax definition is given in appendix 6. The language is based on the character set of a standard GDU (figure 3:1), with optional use of a joystick or its equivalent for pitch specification. This provides a total of 95 available characters, including lower case letters, the space bar, and the carriage-return function, but excluding all other control keys. To avoid confusion the lower case letters are taken to be equivalent to the corresponding upper case letters except where they represent letters actually appearing in the print.

The language is essentially designed for interactive use. It is unsuitable as it stands for off-line data preparation, so no attempt has been made to restrict the character set used to that of a standard keypunch. With some modification the language could be adapted for use with a card punch, but more complex programming would be needed to

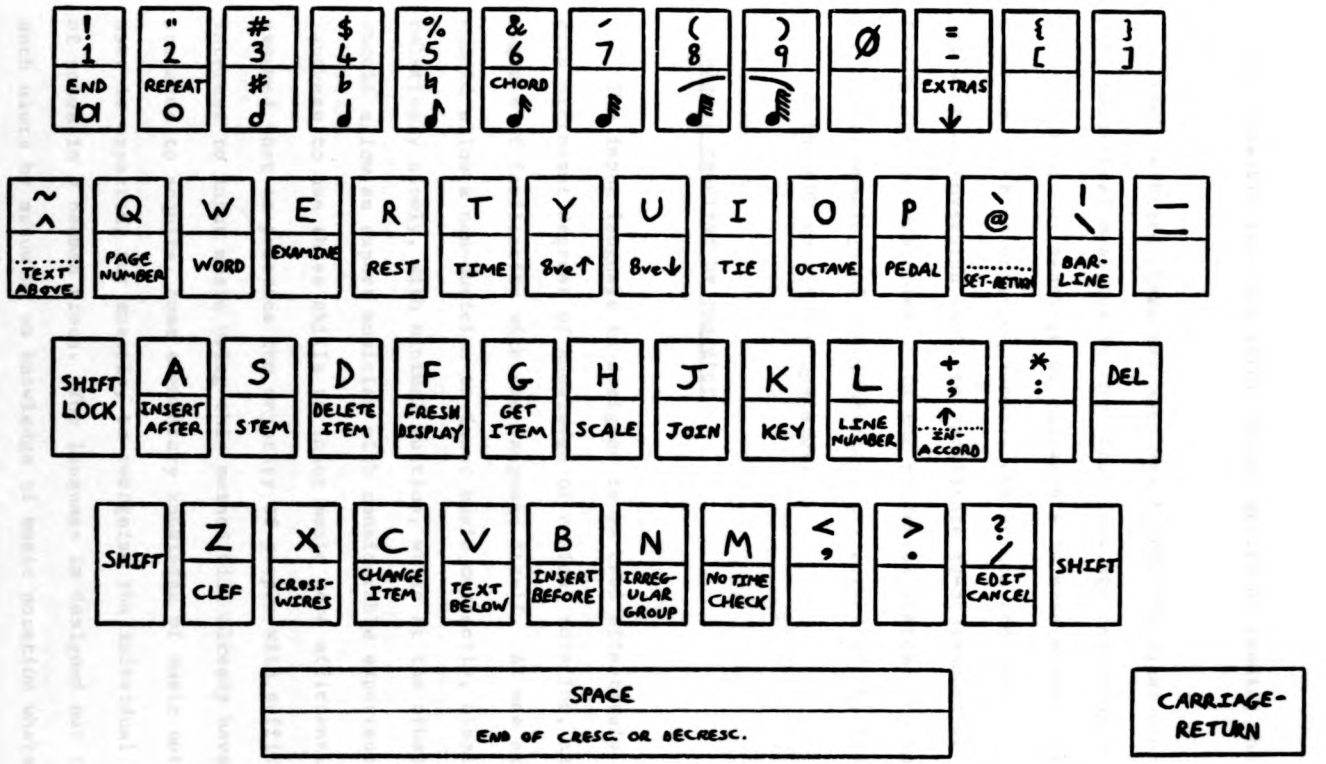


Figure 3:1

Graphical display unit keyboard layout

The upper half of each box shows the standard symbols; the lower half shows additional meanings marked on the front of the keys.

ensure sensible recovery after syntax errors on reading the data.

Each keyboard character has, apart from its usual meaning, one or more additional meanings in the input language depending on context. The assignment of meanings to characters is made mnemonic as far as possible. The association between characters and meanings is made by means of a conversion table to facilitate re-assignment of characters during system development. In the prototype system, the major additional meanings of the characters are marked on the front of the keys of the GDU to aid the operator.

Knowledge required by operator

The input language is designed to be used effectively by users with different degrees of knowledge of musical notation, and different degrees of familiarity with the language itself. At one extreme it should allow a non-musician to input music correctly, albeit relatively slowly, with minimal tuition, while at the other extreme it should allow an expert musician with considerable experience of the language to use these skills to input music more efficiently. It is assumed that in practice the majority of people with sufficient interest to input music using this method will already have, or be prepared to acquire, some elementary knowledge of music notation. The user is expected, for example, to recognise the individual durations of notes in a beamed group. The language is designed not to hamper such users by assuming no knowledge of music notation whatsoever.

The method of specifying note pitch in particular is most effectively used by someone with a knowledge of the concept of octaves, and actual pitch letters for at least music written in the

treble clef. This contrasts with DARMS and some other music input languages which specify pitch by line or space number relative to the printed stave. It is expected to be simpler for users already familiar with the standard nomenclature and fairly readily learnt by others. For completeness, the language includes the slightly slower alternative of specifying pitch either absolutely by position on the displayed stave using crosswires, or relative to the previous note by indicating an increment or decrement.

No knowledge of computers or experience of their use is required. Apart from experience in the use of CIMBAL itself and general familiarity with the layout of a standard keyboard, such knowledge is of no particular advantage.

The requirement that the operator should need no knowledge of braille has been almost entirely met. The only significant exception is that to avoid large-scale re-ordering, the following two braille music rules regarding order are effectively passed on to the operator: in-accord parts are ordered upwards or downwards according to the instrument, hand or voice, and a dynamic marking is associated with the first played note to which it applies, whichever stave this appears on.

Automatic checking of input

The program analyses the input character by character as it is typed. This requires extra computing resources by comparison with analysis of a block of characters at a time, but it is essential in taking advantage of the immediate checking capability of interactive input. The user is notified of any automatically detected error by

the program's refusal to display the typed character on the screen, an audible warning being given instead using the "bell" character.

Automatic checking of the input may be conceptually regarded as being divided into syntactic analysis and semantic analysis. The borderline between these is to some extent arbitrary and the distinction is irrelevant to the user, who is simply made aware that a mistake has occurred. For the purpose of this language a character will be regarded as causing a syntactic error if it cannot be interpreted according to the formal definition of the input language syntax given in appendix 6. Any other error detected will be regarded as a semantic error. With a more elaborate definition of the input language syntax, some semantic errors would become syntactic errors. With the current definition, a syntactic error is effectively an invalid character string for the specification of a single item.

The program detects all syntactic errors in the input: no character typed by the user can cause the program to fail to continue correctly. Syntactic errors are corrected by the operator either by continuing with a correct character or, where no syntactically correct continuation is possible, by cancelling and re-typing the character string used to specify the current item.

The most significant semantic error detected by the program is an invalid total time count for the notes in a measure. This is detected when the expected count is exceeded, or when the user attempts to type a single bar-line if the time count is insufficient. Other semantic errors include the end of an extended sign which has not previously been started, and attempting to increment the pitch of a rest (which is syntactically equivalent to a note). A semantic error is not

generally detected at the time the actual error is made. In this case, the error must be corrected by editing of music already input.

The program does not detect all erroneous input which could be identified automatically. It is quite possible for a perverse operator to input notation which is musically nonsensical (for example, a staccato rest). The extent of semantic checking could be considerably increased to avoid this, but for the present application this does not seem worth the extra complexity involved.

Flexibility

The overall order in which items are normally input is determined by the storage structure. It is possible though less efficient to use a different order using editing facilities (see section 3.4). Within the specification of a note item a greater degree of flexibility in the order of attributes and attached symbols is both possible and desirable. No order is imposed on the input of these although the item is automatically given a unique form on storage. This flexibility simplifies the rules of the input language and reduces the chance of incidental signs being missed by the operator until it is too late.

The interpretive nature of the input processor and the fact that the input characters themselves are not stored precludes the use of a single-character backspace as usually used for text. A form of backspace is implemented but the number of typed characters whose effect is cancelled by it depends on the context, and it cannot be used repeatedly, except within text. To avoid confusion for the operator resulting from this, it is permissible to correct a wrong

specification by typing the correct value without the need to first explicitly cancel the original.

Conciseness

Given this degree of flexibility and a certain amount of redundancy to help reduce the potential for errors, the language is designed to allow the skilled operator to specify the music in a concise form. The operator is given the opportunity, optionally, to take advantage of regularities in the notation by the use of default values and repeat specifications. These are used to reduce the overall number of characters typed and thus save time. The most significant default values are the assumption of the same pitch (excluding accidentals) and duration (excluding dots) as the previous note unless otherwise stated by the user. Repeat specifications may be used to automatically repeat any measure or sequence of measures in the current file, or a note or sequence of notes in the current measure. The former can save considerable effort in music with lengthy repeated passages and may justify preliminary marking of these on the printed copy by a musician for a non-musician operator. Repeat specifications may also be combined with other specifications to simplify the input of groups of notes which have a repeated pattern but are not exact repeats. The use of default values and repeat specifications could be made almost arbitrarily sophisticated. Those used in this input language are considered sufficient to take reasonable advantage of musical patterns without confusing the user with excessive complexity.

For the input of orchestral music or other music in which not all instruments play all the time, any combination of staves may be

selected to be input at any given time. Omitted staves are not displayed and whole-measure rests are automatically inserted into them as the input proceeds.

3.3 Graphical display

Display of the music in a form recognisably similar to the printed page is used for two purposes: confirmation or otherwise at the time of input that the music has been correctly specified, and subsequent proof-reading.

Confirmation of correct input increases the operator's confidence and sustains interest. Immediate detection of errors increases the operator's learning speed and reduces the likelihood of repetition of mistakes due to misunderstanding of the input language. Immediate correction of errors reduces the overall time taken to locate their position.

Proof-reading of the stored music is necessary to identify errors not noticed or corrected at the input stage. The alternative method of achieving this, verification (comparison of the same information coded independently by two different operators or on two different occasions) is less suitable because the encoding operation is much more complex and slower than a direct visual comparison of similar notations. Verification could usefully be applied if a less labour-intensive method of coding, such as on-line sampled keyboard or optical character recognition, were substituted. Other researchers have found that proof-reading of an alphanumeric code is tedious and inadequate for identifying all errors. For maximum speed and accuracy, the form of notation to be proof-read should correspond as

closely as possible to the original print.

The two standard computer peripherals available for producing the graphical output necessary to achieve this are the digital plotter and the cathode ray tube in the form of the graphical display unit. Other potential means of display are computer-controlled photocomposition, computer-controlled music typewriter, and standard lineprinter with a modified character set. None of these is standard computer equipment; photocomposition is attractive but expensive, the music typewriter is fairly slow and has a limited character set, and the modified lineprinter cannot produce a good approximation to printed music.

Use of the digital plotter for proof-reading purposes is mentioned in section 3.5. This device produces a cheap hard-copy but is unsuitable for use during input because of its slow speed. The graphical display terminal is therefore used for this purpose and may also be used for proof-reading. A Tektronix model T4002 terminal with storage tube display is used in the prototype system. From the user's point of view, a refreshing display would be preferable for editing purposes, enabling instant removal of deleted information from the display. This would require extra processing capacity to maintain the display. Experiments with a refreshing display (DEC model GT40) indicate that the refresh cycle time for a screenful of music is so long as to cause irritating flickering of the display. With a storage tube display, the amount of music displayable at one time is limited only by the dimensions of the screen. Physical erasure of individual symbols from the display is not possible; deletion is indicated by drawing a cross over a deleted item. Re-positioning or modification of symbols is indicated as deletion followed by re-drawing at a slightly displaced position.

Symbol generation

The standard display hardware generates only alphanumeric characters and straight lines. Musical symbols and layout are software-generated. The software to achieve this in this system is self-contained within the CIMBAL program. No additional package of graphical routines is required apart from the following basic hardware-dependent functions which should already be available on any computer system supporting graphical terminals:

(1) read a character

(2) write a character string

(3) select a position specified by x and y co-ordinates

(4) draw a straight line to a specified position

(5) erase the screen

(6) interrogate position of joystick or light pen.

Speed of symbol display is a critical factor. For display during input, symbols should be displayed as quickly as possible to cause minimal interruption to the flow of typing. For proof-reading directly from the terminal, music should be displayed at least as fast as the operator can proof-read it.

For the Tektronix terminal used in the prototype system, from two to four bytes of data must be transmitted for each re-positioning of the cursor in graphical mode, depending on proximity of consecutive x and y co-ordinates, with an extra byte if no line is drawn. Thus a sharp sign, for example, requires transmission of about 30 bytes. For an application using mainly small graphical symbols this is a reasonably efficient coding of information and would not be

significantly improved by use of a different terminal. The maximum data transmission rate available on the Sigma 5 computer is 1200 baud (110 bytes per second). While this is not intolerably slow, use of a 2400 baud or higher transmission rate would give a substantial improvement in the speed of display of music.

For speed of display, all note-heads apart from the breve are generated using a letter "O", overtyped with an asterisk in the case of a solid note-head. On the terminal used in the prototype system, these two characters produce a reasonable note-head; note-head generation might require some modification for use with a different type of terminal. All other non-textual music symbols are generated using straight line segments only, and should be directly transferable to other types of display terminals without difficulty provided that adequate resolution is available.

Symbols of fixed shape and size are generated using co-ordinate tables (see appendix 3). These are: breve note, rests, clef signs, accidentals, expression marks, ornaments, bar lines, segno and encircled cross. Stave lines and symbols of variable form or size are program-generated to fit individual circumstances. These include: slurs, crescendos and decrescendos, beaming of notes. For speed of display on the GDU screen, symbols are reduced to the minimum number of lines necessary for clear recognition. For example, a treble clef sign is composed of 7 straight lines (see figure A7:2).

Layout of display

The most complex aspect of the display of music notation is the layout of symbols. Apart from the complex conventions used by music

publishers, a good quality display should be arranged to avoid collisions between different symbols, and right-justified to give a bar line at the right hand side of the page, with simultaneous notes aligned between parallel staves.

It should be emphasised that the purpose of the display in this system is solely to provide effective checking and proof-reading of the music. No attempt has been made to emulate the quality of printed music, an extremely complex task which has already been tackled by a number of researchers (for example, Hiller & Baker 1965, Boker-Heil 1972, Smith 1973, Byrd 1974, Gomberg 1977).

A single-pass display processor is used in CIMBAL. This is inherently unsuitable for production of a print quality display, which requires two or more passes to evaluate potential collisions and perform alignment and justification. It is reasonably adequate for the purpose of proof-reading which it is designed for, with the exception of occasional symbol collisions in which one symbol is partially or completely obliterated by another.

Right-justification of the music has not been attempted. A new parallel is started when the estimated length of the next bar is greater than the remaining space on the current parallel. This occasionally causes a measure to be split between two parallels. Vertical alignment of simultaneous notes is achieved by making horizontal spacing of notes directly proportional to duration for music with two or more staves. This is generally satisfactory but leads to misalignments where one stave has particularly short notes, and wasted space where all staves have long notes. Where this causes any problem the operator may change the scaling factor to suit the

circumstances.

Most potential collisions of symbols are successfully avoided by moving the second of two symbols which would otherwise appear in the same place. This fails where the position of the later symbol is determined by its meaning and the earlier symbol should have been displaced instead, for example, notes written on ledger lines may conflict with text or other symbols previously displayed above or below the stave. The display of beaming of notes may also sometimes partially overwrite symbols already shown. There remains scope for further improvement of these aspects of the display format.

3.4 Editing

It is found in any computer system involving human input that the human element introduces errors and omissions. The significance of this is often underestimated in the basic design of a system, causing inherent defects which may be difficult to remedy later. The incidence of human error in alphanumeric coding of music notation is likely to be particularly great because the operator is asked not simply to copy information, but at the same time to convert it into an artificial form. The ability to identify and correct any operator error at any time and as easily as possible is therefore a fundamental principle in the design of the input side of CIMBAL. The processes of correction, whether deletion of superfluous information, insertion of omitted information, or modification of incorrect information, will be referred to collectively as "editing".

Editing involves two distinct stages: specifying where a change is to be made, and specifying what change is to be made. A computer

text editing system normally identifies a location in two stages. A line is identified by context (the first few characters), name (a number permanently associated with the line), or sequence number relative to a previous position or to the start of the file. Within the line, exact location is identified by context. For a music editor, identification by context is less useful because the repetetive nature of the notation often makes a wider context necessary to identify a position uniquely. The time penalty in displaying a wrongly identified measure is also greater.

The two stage method of location is valid, and has been used, for the music editor, with the measures and items of the music data structure analogous to the lines and characters of text. For a measure containing a large number of items, an intermediate stage would be desirable to avoid display of the whole measure. The beat appears to be the most natural intermediate unit, although use of this would require extra musical knowledge. In practice, the rarity of such long measures makes the extra stage an unnecessary refinement. Addressing of individual items is done directly by selecting a position on the display to avoid re-coding of an item for identification. This reduces the identification of an item to a single operation: the positioning of crosswires using a joystick.

For simplicity and uniformity between different types of music, a measure is identified solely by bar and line number. The position of a measure could be specified relative to rehearsal marks, or print page and parallel numbers, but the former are not always used, the latter are not necessarily stored, and neither is explicitly represented in the structure of the file.

3.5 Alternative forms of output

The forms of output described in this section are not essential to the effective operation of the system, but they form useful alternatives to the graphics and braille terminals for proof-reading of printed and braille music respectively.

Digital plotter

To minimise hardware requirements, the GDU itself may be used for all proof-reading of stored music in its printed form. However, some form of hard-copy display may also be desirable. The GDU in the prototype system has an attached hard-copy unit for reproduction of the display screen on paper (see figure A7:2) but use of this is fairly expensive (about ten pence per sheet at the time of writing).

Output on a digital plotter in a form almost identical to the GDU display has been incorporated with minimal extra complexity by switching from terminal display routines to digital plotting routines at the lowest possible programming level. The time penalty associated with the use of a digital plotter (the example shown in figure A7:4 took 11 minutes to draw) is avoided by pseudo off-line operation of the plotter, with intermediate storage of the plotter output file on magnetic tape.

Some of the music symbols for the digital plotter are generated using co-ordinates taken from Hershey's repertory of co-ordinates for display of over 3000 graphic symbols (Wolcott & Hilsenrath 1976). This improves the quality of display of these symbols at the expense of increased program size and plotting time. Alphanumeric characters

are generated using a Sigma 5 library plotting routine. For machine-independence, this could be replaced by a further subset of the Hershey co-ordinate tables.

Audio output

The prototype system includes an extension enabling the stored music to be played by the computer in an audible form. This should not be regarded as an essential part of the system since non-standard hardware is used, but the sofware used to drive the audio output is included in the program listing (appendix A1.1, segment 6) to demonstrate the ease with which the basic music retrieval system may be extended to applications other than braille translation.

The audio signal is generated by a voltage-controlled oscillator developed by Malcolm Watson (1978) for a project concerning perception of key. The oscillator is driven by two signals supplied by the computer via a digital to analogue converter, the voltages of which control intensity and frequency respectively. The hardware and software currently implemented support only monophonic output. The "proof-listening" applications using this form of output are restricted mainly to identification of wrong pitch and duration of notes by someone familiar with the music, but for monophonic music this can be remarkably effective.

Line printer

The line printer is used as an alternative to the braille terminal for hard-copy output of braille where this is to be proof-read by a sighted person. Braille cells are displayed on three

consecutive print lines using the dot character. The standard line printer character set is not suitable for display of printed music, although some previous attempts have been made to do this in a crude form (see, for example, Patrick & Friedman 1975).

Chapter 4. Braille translation

Braille music may be regarded as a sequence of signs, each comprising one or more six-dot cells, together with a number of rules defining the use and layout of these signs. The tables of signs and rules for their use for Western braille music notation are defined in the Revised International Manual of Braille Music Notation (Spanner 1956) which will be referred to below as "the manual". References to specific rules will cite the number of the corresponding numbered paragraph of the manual. An addendum to the manual (APH 1975) has been published to clarify and, where necessary, correct the manual. Although the braille music code is international, there are minor variations of usage between countries.

The basic philosophy of the braille music code is that it should reflect accurately the performance-related aspects of printed music. It does not attempt to represent the musical performance directly. For this reason, the process of converting printed music to braille music is usually referred to as "transcription". The term "translation" will be used here in preference to describe the automated form of this process, in conformity with computer terminology.

The goal of an automatic translator is to implement the braille music rules to produce a correct transcription insofar as this is well-defined. No attempt has been made here to alter or improve the braille music code currently in use.

The major problems in achieving automatic translation arise from:

(1) Ambiguity. Although the majority of braille music rules may be expressed without difficulty in a computable form, some, particularly those regarding use of braille repeat signs, are expressed in terms which do not lend themselves to a precise definition. These include the use of repeat signs in conjunction with slurs (rule 110), doubling (rule 113; see section 4.4) and expression marks (rule 114). Some rules are expressed only in terms of examples. The diversity of notational combinations which may occur in music gives rise to situations not explicitly covered by the manual, in which the judgment and discretion of the transcriber are called for to determine a suitable transcription.

(2) Complexity. The rules of braille music are both numerous and complex. This is not in itself a theoretical problem. However, it is desirable that implementation of the rules should be as simple and structured as possible to produce a translator which is efficient and of practical value.

4.1 Summary of braille music rules

The following is a brief outline of the basic rules of braille music, included to clarify terms used later. It is not intended to be a comprehensive definition. Rules which cause particular difficulties in automatic translation are described separately in following sections. Most of the rules described in this section are relatively simply automated using tables of braille music signs and straightforward logic to determine the presence or absence, and ordering, of the signs.

A note is represented by a single-cell sign. The upper four dots

indicate the pitch within an octave and the lower two dots represent the duration. A breve is represented as a semibreve followed by a fixed cell (dots 1,3). Of the remaining durations, the longer four are represented by the same combination of dots as the shorter four. Occasional ambiguities arising from this representation are resolved by placing a special value sign before a note. Similarly, four single-cell signs are used for the rests of different durations.

The musical scale is divided into octaves, each ranging from note C to the B above it. The octave of a note is indicated by one of seven octave signs, placed before the note. The octave sign is shown only in the following circumstances:

(1) the interval between the note and the previous note is a sixth or more; or

(2) this interval is a fourth or fifth and the two notes are in different octaves; or

(3) certain other signs intervene between the two notes; or

(4) the note is the first of a braille line.

For a chord, only one note is shown as such. Each of the remaining notes of the chord is shown using one of seven interval signs, indicating its interval from the written note. The written note is normally the uppermost, with intervals reading downwards, for right-hand and treble parts, or those written in the treble clef; the written note is normally the lowermost, with intervals reading upwards, for left-hand and bass parts, and those written in the bass clef. Intervals more than an octave apart are preceded by the appropriate octave sign. Isolated notes with two opposite stems are shown as an ordinary note followed by a special "stem sign".

Other symbols applicable to a note are represented by additional signs preceding or following the note or interval sign in a well-defined order. These include accidentals, dots (representing extended duration), fingering, ornaments, expression marks, and note-fractioning.

Words, letters, and abbreviations of expression are shown preceding the first note to which they apply, either enclosed in braille parentheses or preceded by a word sign. Changes of time and key, and other miscellaneous symbols are shown by corresponding signs.

Irregular grouping of notes (e.g. triplets) is shown by a sign preceding the first note of the group. This indicates the number of notes grouped. The beaming together of printed notes to indicate ordinary rhythmic grouping has no direct braille equivalent.

Symbols which apply to a group of consecutive notes - slurs, crescendos and decrescendos - are represented by an opening sign before the first note and a closing sign following the last note. However, slurs covering four notes or less are represented by individual slur signs attached to each note of the group except the last, and longer slurs may be represented by doubling these signs (see section 4.4).

4.2 Braille layout and format dependency

The rules of braille music fall naturally into two groups: those defining the sequence of braille signs and those defining the layout of the signs on the braille page. Different methods of layout are used for different types of music. The standard format for keyboard

music currently in use in the United Kingdom, called "bar over bar", reflects the layout of the printed music by placing the different staves of a parallel on consecutive braille lines, with the start of corresponding measures aligned within each parallel. The number of the first bar of each parallel is placed in the left hand margin.

A two-pass translator has been developed, reflecting this division of the rules. The first pass, the "translator" proper, operates on the printed music representation, and produces as output an intermediate form of braille referred to as "unformatted braille". The second pass, the "formatter", operates on the unformatted braille and produces as output a representation of the braille arranged in lines as it will appear, apart from pagination, in the final braille output from the system.

The major advantage of this approach is that it allows arbitrary manual editing of the braille. Editing of the formatted braille alone is not necessarily sufficient where the number of cells in a line is to be changed, since this may involve re-formatting of subsequent material. Other advantages of the two-pass approach are:

(1) the formatting module may be replaced to provide alternative layouts with only minor alterations to the main translation algorithm;
(2) the overall program size is reduced, and extra internal buffering of braille is avoided.

The problems associated with the use of two separate passes are:

(1) Format dependency of translation. The actual sequence of braille cells is partially dependent on their layout on the braille page. The

most significant format dependencies arise fom the splitting of a measure between two braille lines: for example, note-grouping must be completed within a single braille line, and part-measure repeats may refer back only to music on the same braille line. However, it is not normal practice to split a measure between two braille lines unless it is in itself too long to fit onto a single braille line. This condition can be detected at translation time. Other format dependencies relate to the position of music within a braille line, and are handled by producing alternative translations at the first pass, with additional information to enable the formatter to choose between them:

(a) The first note of each braille line must have an octave sign.

(b) Doubling (see section 4.4) is re-marked at the start of a new line.

(c) Separate single signs are used in preference to a double sign where, because of (b), the double sign would not save any space.

The first two conditions are satisfied simply by marking the extra signs as "conditional", that is, to be included in the final output if and only if they are associated with the first note of a braille line. The third condition, described in section 4.4, is more complex and has not yet been implemented.

(2) Formatting is partially determined by particular musical signs which after translation to braille become merely patterns of braille cells whose meaning cannot be automatically interpreted except by an inverse translation process. Thus, for example, it is usually desirable to begin a new braille parallel following a double bar line even if this occurs within the middle of a measure. This question is dealt with by the inclusion of format control information interspersed with braille cells in the output from the translator. Format controls

in the unformatted braille may be edited in the same way as braille cells, allowing the user to vary the layout independently of the content if necessary.

(3) The method is less suitable for vocal music than for instrumental music. Formatting of vocal music is dependent on the words rather than the music. The natural point of division between lines for vocal music is the end of a phrase, or failing that, the end of a word. These do not necessarily coincide with the end of a measure, which makes it an invalid assumption that splitting of measures can always be determined during the first pass. Less attention has been devoted to vocal music in view of the fact that some earlier work has concentrated on this (Patrick 1976).

4.3 Repeat signs

The braille music code uses a system of repeat signs to achieve saving of space and as an aid to reading and memorising of the music. These are notational repeats, used where a given passage of music is written out in full in the print more than once. They bear no relation to performance repeats shown in print by forms of double bar line and other signs, which each have an equivalent sign in braille.

The use of braille repeat signs is "one of the main differences of procedure between ink-print and Braille music" (rule 100) and it correspondingly causes some of the major problems in automatic translation. These may be summarised as follows:

(1) The translation of any passage of a piece of music may depend directly on comparison with passages anywhere in the preceding music.

This leads to potential excessive overheads in computation time for the identification of non-local repeats. Random-access input is essential to make this process feasible without such excessive overheads.

(2) The matching process is complex. Passages of music which are notationally identical in print do not necessarily give rise to braille repeat signs. Conversely, passages of music which are not notationally identical in print may be represented in braille using repeat signs.

(3) The two major purposes of braille repeat signs, space saving and ease of reading, may conflict with each other. While space-saving is easily computable by direct comparison of contracted and uncontracted forms, ease of reading depends to some extent on musical judgment which is not always easy to specify formally.

(4) A braille repeat sign may not be used where, taken in context, any ambiguity or confusion in meaning might arise. This is often difficult to specify in a precise form.

Braille repeat signs refer either to whole measures or consecutive groups of measures, or to parts of a measure which, depending on the complexity of the notation, may be as short as a single note. Part-measure repeat signs are used only for repetition of music within a single measure.

Measure repeats

The following types of measure repeats are defined in the braille

music code. Only the first two of these forms have been implemented in the translation program, these being the ones in common use.

(1) Single measure repeat. The single measure repeat sign is used for a measure which is a repetition of the immediately preceding measure. Where a single measure is repeated three or more times in succession, a measure repeat sign is used, followed by a number indicating how many times the measure is repeated.

(2) Measure-number repeat. A measure or sequence of measures which is a repetition of an earlier passage may be represented by a number or pair of numbers representing the original range of measures.

(3) Partial abbreviation. Repetition of a recent sequence of up to eight measures may be shown as a count-back number, giving the measure number relative to the current position, followed by the number of measures to be repeated if this is less than the count-back figure.

(4) Braille segno and da capo. These are used by analogy with the printed signs of the same name.

Avoidance of unnecessary searching of the disc file to identify potential measure repeats is a major factor in reducing overall computation time. This is achieved by computing a "checksum" value for each measure translated. A table of such values for measures already translated is held in internal storage. Measures are accessed from the file for comparison only if their checksums are the same as that of the current measure. The checksum is made dependent only on note attributes which must coincide if a braille repeat sign is to be considered. In the prototype system it is stored in a single byte.

This is adequate for short pieces of music but leads to inefficiency for pieces with a large number of measures, and should be extended where program size is less crucial.

Comparison of potential repeated measures is performed sequentially from the start of the piece, or, for vocal music, in which numeral repeats are not allowed, from the measure preceding the current measure. If a match with the current measure is found, the bars following the current measure are compared with those following the earlier matching measure until a mismatch is found. This procedure, while not necessarily locating the longest repeated passage, avoids referencing measures which have themselves been specified as repeats of earlier measures.

Given that an actual repetition of one or more measures is identified, use of a repeat sign depends on the amount of space saved and ease of reading. The latter has been expressed as a function of the distance between the original measure and its repeat. The use or non-use of repeats in this situation is a matter of musical judgment; the opinion of braille music experts and readers is needed to determine whether this method, and the chosen balance between space-saving and readability are adequate.

A repeat sign may only be considered if the notes repeated are identical with those of the original passage or each differ by the same multiple of the octave from the original. The effect of other symbols on the use of repeat signs is defined in the manual largely in terms of examples, from which general rules must be deduced for the purpose of automatic translation.

Part-measure repeats

Part-measure repeat signs are used for repetition of half measures, simple or compound beats, or natural divisions of a simple beat, provided that the meaning of the notation remains clear and space is saved by the use of the repeat sign.

Although not explicitly stated in the manual, comparison of note pitches depends on actual pitch and not on marked accidentals. Since only marked accidentals are stored, the true pitch of each note is evaluated dynamically during translation to enable this comparison to be made.

Otherwise, similar considerations apply to part-measure repeats as to whole measure repeats.

4.4 Doubling of signs

Where certain signs apply to four or more consecutive notes (or groups of notes in the case of an irregular grouping sign) the braille sign is written twice for the first note and once for the last note and omitted from the intervening notes. This practice, called "doubling", is used to save space and improve readability. The signs to which it is applicable are: irregular note-grouping, intervals, slurs, chord ties, note-fractioning, grace notes, expression marks (staccato, staccatissimo, mezzo-staccato, agogic accent, attack, martellato), arpeggios, variants (large and small notes), bowing, pizzicato, natural harmonics and continuation lines for string numbers. These will be referred to as "doubling signs", regardless of whether in any particular instance they are actually doubled. For

some of these signs, only part of the sign is doubled at the first occurrence. Doubling is not interrupted by rests. The range of doubling of different signs may coincide or overlap, but doubling of all intervals of a sequence of chords is interrupted if any one interval changes.

The slur is treated in braille either as a pair of brackets enclosing the notes within it, or as a sign associated with each of the notes enclosed by it except the last. In the latter case the sign is doubled if five or more notes are slurred together. Doubling of the slur is treated as a special case because the single or double slur sign is placed after a note, whereas the extent of the slur must be evaluated before the note is translated in case the bracket form of slur is to be used.

In literary braille, the principle of doubling is used only for the infrequently occurring italic sign. Literary braille translation algorithms have avoided the question of automatic doubling by requiring doubled signs to be included explicitly in the input. This is unsatisfactory for automatic translation of the braille music code, in which doubling is both more complex and more frequent.

The following possible approaches to the problem of automatic detection of doubling have been considered:

(1) Internal buffering. On encountering a doubling sign during translation, accumulate either printed music input or translated braille in internal store until it can be determined whether the sign is to be doubled; insert a double sign or sequence of single signs as appropriate, and output as much braille as is then uniquely determined

(other doubling signs may have occurred in the meantime); if the sign is doubled, keep the input one note ahead of the output to identify where doubling ends. This requires an indeterminate amount of internal storage space, which may become large for complex multi-stave music, particularly for doubling of irregular note-grouping signs.

(2) Retrospective modification of output. Similar to (1) except: instead of accumulating braille, output it normally assuming no doubling by default. If a doubling sign then occurs four times consecutively, modify the original output and continue as for (1). This eliminates unnecessary testing for doubling but requires the use of non-sequential output, which is not used elsewhere in the translator.

(3) Partial look-ahead. On encountering a doubling sign, look ahead up to four notes (or groups of notes) in the input to determine whether or not to double. Return to the current translation position and output a double sign or single signs as appropriate. If doubling occurs, continue as for (1) after four notes. This is the approach a human transcriber would use. It requires non-sequential input.

(4) Look-ahead. Similar to (3), but look ahead over the complete range of the doubling sign. Store a count of the number of notes to which the sign applies and decrement the count by one for each note translated.

The last method has been found to be the most suitable. Although the current translation context must be saved during the look-ahead, it avoids complexities caused by overlapping ranges of different doubling signs. It avoids any extra internal buffering of either

input or output. By identifying the end of the range of the doubling sign in advance, it allows the output from the translator to remain in step with the input when a sign is doubled. It also simplifies the problem of re-marking of doubled signs at the start of each new braille parallel. Non-sequential input is necessary in any case for detection of repeated measures.

Where doubling of a sign continues on a new braille parallel it is re-marked for the first note of the new line. This is an aspect of current braille music usage which is not defined in the manual. It is intended to enable a reader to interpret the music on any given parallel correctly without reference to the preceding parallels. It creates a format dependency which is handled using conditional braille cells as described in section 4.2.

Where a sign is doubled, but occurs only once or twice on a particular parallel it is current practice to mark one or two single signs for the notes of this parallel, since doubling is marked again for the following parallel and no space would be saved by use of a double sign. Because of the division between translation and formatting, the doubling algorithm as implemented is incapable of identifying this situation and will always produce an initial double sign. This is not ambiguous but it is regarded as untidy by braille music experts. A generalisation of the use of conditional cells (section 4.2) would be necessary to overcome this problem.

The interaction of doubling with braille repeat signs causes difficulties in translation which can be fully resolved only by evaluating repetition in a preliminary pass before the main translation. A repeat sign cannot be used where it would start

within, and continue past the end of, the range of a doubled sign, because the end of doubling would not then be correctly marked. In some cases it would be preferable to cancel the doubling earlier to permit use of the repeat sign, requiring the repeat to be evaluated before the doubling.

4.5 In-accord parts and order of signs

"In-accord parts" are those written in parallel on a single stave. Within each measure these parts are shown sequentially in braille, separated by an in-accord sign.

The major problem arising from the use of in-accord parts is the placement of accidentals. Whereas in print the effect of a marked accidental applies to subsequent notes at the same vertical position in the same measure, in braille the separate parts are treated as independent. Thus, braille notes written in-accord may have accidentals not shown in the print and vice versa. Correct determination of actual note pitches depends on scanning notes in the order in which they are played. This can fairly readily be achieved, since it takes place within a single measure, but is not implemented in the prototype system.

For the purpose of doubling of signs, in-accord parts are treated as independent of each other. In evaluating the extent of a doubling sign, the translator disregards music belonging to other parts on the same stave. It is therefore important that slurs begin and end in the same part, to avoid spurious mismatching of the start and end of slurs.

The ordering of in-accord parts, which is from the top down or bottom up according to the instrument, voice, or clef as for the notes of a chord, has been passed on to the user in the form of rules for input. Within a single item, ordering of signs is determined automatically by the translator. This includes correct ordering of the different notes of a chord as described in section 4.1. Larger scale ordering, including the positioning of text before the corresponding note and ordering of in-accord parts, has been passed on to the user in the form of rules for input. Automatic re-ordering of in-accord parts at the time of translation would be more desirable, enabling the input and storage of these parts to be in a fixed order independent of the rules of braille.

4.6 Added and omitted signs

In cases where braille notation does not accurately represent the printed notation, extra signs not appearing in the print are added to the braille to make the meaning clear. Thus, where dynamics change within the duration of a single note, in braille the note is written in-accord with a sequence of added rests interspersed with the dynamic markings to show the precise start and end of each (rule 199). Irregular note grouping is marked explicitly in braille even though it may be omitted or marked only for the first few occurrences in print. Where rests not shown in the print are implied by the spacing of the notes, these must be included explicitly in the braille. There is no obvious way of doing this automatically without the inclusion in the input of a great deal of otherwise irrelevant information about the layout of the print. These added signs are therefore left to be inserted by the user in the original input.

Certain signs are omitted from braille music in contexts in which they are redundant. The signs representing the end of crescendo and decrescendo lines are omitted if followed by a double bar line or dynamic marking before the next played note. The latter may be in a subsequent measure and may follow one or more intervening rests. This presents a problem of non-sequentiality which is resolved, as for doubling, by looking ahead in the input as far as the next played note on the same stave.

Printed notes are sometimes marked with redundant accidentals, usually confirming cancellation of accidentals marked in the previous measure. It is normal practice except in facsimile transcriptions to omit these from the braille. Also, while, for visual convenience, printed accidentals apply only to notes of a specific pitch, in braille they automatically apply to subsequent notes in the same measure which differ in pitch by a multiple of the octave. Omission of redundant accidentals is achieved by comparing the true pitch of each note, evaluated dynamically from the key signature and marked accidentals in the current measure, with the pitch the note would have if no accidental were shown. Accidentals marked in the print are omitted from the braille if these two values are the same.

4.7 Translation of text

Text appearing in music notation is translated according to the usual rules for literary braille translation, with a few exceptions. For text in the language of the country in which the braille music is published, contracted braille is used; uncontracted braille is used for foreign text. Computer translation of text to contracted literary braille is in itself a complex process for which a number of computer

programs have been developed elsewhere. The braille translation of text appearing in music notation differs from the translation of ordinary literary text in the following ways:

(1) Use of foreign words is much more common;
(2) Accented letters are represented by the correct single braille cell instead of the cell for the unaccented letter, preceded by an accent sign;
(3) Sung words must correspond on a braille line-by-line basis with the matching music. It is undesirable to hyphenate words at the end of a braille line but may occasionally be made necessary by this correspondence. Use of hyphenation complicates the literary braille contraction rules and has not been implemented in most automatic translation programs for contracted literary braille;
(4) Words and phrases are frequently repeated in music. Braille music includes a special sign to represent this.

Use of contracted braille for the words of vocal music is important because:

(1) reading speed should be sufficient to enable the singer to follow the words during performance;
(2) the line of words is generally longer than the corresponding line of music, effectively doubling the space-saving effect of contraction;
(3) the more words that can be fitted onto a single braille line, the more likely it is that repeat signs can be used, giving an even greater space saving.

No new algorithm has been developed for full contraction of literary braille in CIMBAL, since this is a major task and algorithms

already exist which may be adapted for this purpose. Lack of core space has precluded the incorporation of a full grade 2 contraction algorithm in the prototype system, but a subset of the grade 2 rules has been implemented using a simple text translation algorithm. This includes the rules regarding position of a letter group within a word, by considering a context of one character before and after a contractable letter group, but is restricted to contraction of groups of letters sung to a single note. This tends to avoid the use of contractions where they should not occur, but occasionally fails to contract groups of letters which should be contracted, especially multi-syllable and whole-word contractions.

The braille repeat sign for words and phrases causes particular problems, and has not been implemented in the prototype system. The use of a repeat sign for a phrase must reflect the phrasing and not merely literal repetition of words. Thus in the line:
"Sleep, my baby, sleep, O my baby, Sleep, O my baby, arru, arru"
the repeated phrase is "sleep, O my baby", not "my baby, sleep, O".

Hyphens shown in print to connect syllables of a single word sung to different notes are normally omitted from braille. Genuine hyphens and those connecting nonsense syllables, for example "a-rru" in the above example, are included. The difference between the two types of hyphen cannot easily be distinguished automatically. This must be specified by the input operator. Where punctuation differs between repetitions, a repeat sign may still be used provided that the punctuation remains reasonably clear. For example, "Tweet, tweet, tweet!" is represented in braille as "tweet!" with two repetitions.

The repeat sign may be used only for words or phrases which

already exist which may be adapted for this purpose. Lack of core space has precluded the incorporation of a full grade 2 contraction algorithm in the prototype system, but a subset of the grade 2 rules has been implemented using a simple text translation algorithm. This includes the rules regarding position of a letter group within a word, by considering a context of one character before and after a contractable letter group, but is restricted to contraction of groups of letters sung to a single note. This tends to avoid the use of contractions where they should not occur, but occasionally fails to contract groups of letters which should be contracted, especially multi-syllable and whole-word contractions.

The braille repeat sign for words and phrases causes particular problems, and has not been implemented in the prototype system. The use of a repeat sign for a phrase must reflect the phrasing and not merely literal repetition of words. Thus in the line:
"Sleep, my baby, sleep, O my baby, Sleep, O my baby, arru, arru"
the repeated phrase is "sleep, O my baby", not "my baby, sleep, O".

Hyphens shown in print to connect syllables of a single word sung to different notes are normally omitted from braille. Genuine hyphens and those connecting nonsense syllables, for example "a-rru" in the above example, are included. The difference between the two types of hyphen cannot easily be distinguished automatically. This must be specified by the input operator. Where punctuation differs between repetitions, a repeat sign may still be used provided that the punctuation remains reasonably clear. For example, "Tweet, tweet, tweet!" is represented in braille as "tweet!" with two repetitions.

The repeat sign may be used only for words or phrases which

appear on the same braille line. The final decision on whether to use a repeat sign must therefore be left until the formatting stage. It would be necessary to include alternative translations in the unformatted braille with additional information to enable the formatter to choose between them if the two-pass method of translation is to be used.

4.8 Braille note-grouping

Groups of semiquavers or notes of shorter length may be grouped in braille for ease of reading by showing all but the first note of each group as quavers (bottom two dots absent). The rules regarding the use of such grouping (rules 29-35), intended to avoid ambiguities which might otherwise occur, are easy to specify in a computable form, and cause no significant problem except for the format dependency implied by the requirement that such a group of notes must be completed within a single braille line (rule 31).

Since grouping takes place only within a single measure, and does not involve any change in the number of braille cells, it has been implemented by maintaining a table of pointers to cells representing notes of the current braille measure, and retrospectively modifying these cells if grouping of the notes is found to be applicable.

4.9 Editing of braille

The ability to edit the braille music before embossing is an essential component of the system. Ideally it would be redundant but in practice it serves two useful purposes:

(1) It is desirable that the system should be used in a production capacity in parallel with continuing development and improvement, which depends to a certain extent on assessment of its merits and demerits in actual use. Braille editing allows useful braille to be produced during system testing and improvement without unnecessary delay.

(2) It seems probable that there are theoretical limits to the degree of accuracy which may be attained automatically. While the braille produced by the automatic translator should be adequate for many purposes, the ability to adapt to the existing standards and formats of particular publishing houses is likely to be a major factor in their decision to adopt the system.

Where a particular form of transcription is required, the automatic translator can replace the initial transcription stage of the traditional method, with proof-reading and correction of the braille continuing as before, except that the correction process is simplified and need not involve manual re-writing of the braille.

The storage of both the braille and print forms of music in the same file using the same structure allows the same input and editing routines to be used for both print and braille forms with minimal extra complexity.

While this system is no substitute for a purpose-built general braille editor, its capacity for input and editing of braille makes it suitable for direct input and digital storage of specialised braille codes other than music for which automatic translators are not yet available. It has been used experimentally to produce a short piece

of literary text in a suggested modification of the grade 2 Standard English Braille code, by typing and editing the braille cells directly.

4.10 Braille output

The hard-copy braille output from the system depends on the braille hardware used. In the prototype system, two models of braille computer terminal have been used (Triformation model LED120 and SAGEM model TEM 8BR), operating in a pseudo-off-line mode. These might be replaced or supplemented by a zinc-plate embossing machine for larger scale publication.

The stored form of the braille is independent of the output device used, except insofar as the layout depends on the number of braille cells per line. Conversion from this form to the final output form required by the hardware is performed by a small independent program module, the "embosser". Pagination of the braille is delayed to this stage. This is based on the indicator, included with each line of formatted braille, showing the minimum number of lines which must remain on the current braille page.

4.11 Signs not implemented

The braille music signs given in the "tables of signs" in the manual have been implemented with the exception of those listed below, which either occur only rarely, apply only to specific types of instrument or are alternatives to signs which are implemented. The system could be extended to include most of the following without difficulty.

Square brackets above or below stave (general table)

Music and literary prefixes (general table)

Coincidence of notes in both hands (general table)

Value signs (table 1)

Alternative version of breve note (table 1) and rest (table 5)

Clef signs (table 3)

Single-figure time signature (table 6)

Inner triplet (table 7)

Moving note sign (table 7)

Overlapping slurs (table 12)

Part-to-part slurs (table 12)

Half-phrase (table 12)

Grace note slurs (table 12)

Omission of alternative fingerings (table 14)

Unusual ornaments (table 15D)

Braille repeats other than the repeat sign (table 16A)

Half-pedalling (table 18)

Variants (table 19)

Second continuation line (table 20)

Organ pedalling (table 21)

Portamento (table 22)

Two or three vowels on one note (table 22)

Variation of syllables (table 22)

Solo sign in accompaniment (table 22)

Prefix for divided part (table 22)

Special signs for stringed instruments (table 23)

Short form scoring (table 24)

Music for the accordion (table 25)

Abbreviations for orchestral instruments (table 26)

Figured bass (table 27)

Chapter 5. Evaluation and resulting developments

5.1 Evaluation of input method

Effective evaluation of the user interface with the system was necessarily delayed until the later stages of development, when the program was believed to be working reliably. The input of music was tested using:

(1) A typist with no musical knowledge

(2) A group of musicians

(3) The author

To allow comparison of the speed attained by users of the computer-based system with that of manual transcribers, timings for input of music are expressed below in equivalent braille lines per person-hour (LPH). This is more reliable than estimates based on number of measures or amount of printed notation because there is a large variation in the density of the printed notation. There is also some variation in the density of the braille music notation due to the formatting rules. The figures given must therefore be regarded as approximate. For comparison, the Director of Music at the RNIB estimates that a pair of manual transcribers produce seven or eight pages (of 22 lines each) of braille music a day, a rate of about 12 LPH. This does not include the subsequent braille proof-reading by different transcribers. The timings given below include checking and correction of the printed music notation.

Input by non-musician

The typist had no knowledge of music notation and little interest in music of any kind. She therefore provided a severe test for the operator interface. Her previous experience in the use of a computer consisted of simple editing of text for a pilot computer-based short braille document service over a period of several months.

Her overall experience in input of music was between 5 and 10 hours over a period of a few weeks in sessions of 30 to 45 minutes each. By the end of this period her rate of input was about 8 LPH. The stored music was correct after checking and editing by the user herself. She used the slower methods of specifying the pitch of notes (joystick or increments and decrements), since she was unfamiliar with pitch names. She frequently failed to save time by noticing the occurrence of repeated measures in the music.

Comparison of her use of the music program and the ordinary text editing system suggests that her rate of input would have continued to improve significantly with further experience, but she did not find the input of music satisfying and was not put under any pressure to continue.

Some examples of music used in testing the system indicate that certain aspects of printed music are likely to cause particular difficulty for the non-musician. Triplets and other irregular note groupings are sometimes not explicitly marked in print, but must always be shown in braille. In most cases these could be deduced from the time count and the beaming of notes in the printed copy, but a

greater degree of musicianship might be necessary to resolve such questions as the ambiguity between triplets and sextuplets.

The beginning and end of crescendo and decrescendo signs are not always accurately positioned on the printed copy. One piece of music used in testing the system contained a passage which was an exact repeat of an earlier passage except that a crescendo sign extended one note further, precluding the use of a braille repeat sign. Even a non-musician who noticed the repetition could not be expected to tell whether this was a genuine difference or merely carelessness in manuscript or printing. Many crescendo and decrescendo signs start or end immediately above or below a printed note, and the input operator must then decide whether the note is included within the scope of the sign.

Input by musicians

The piano part of the first movement and a large part of the third movement of Edward Elgar's Piano Quintet in A Minor, opus 84, were input by nine musicians, friends of a blind musician who suggested this piece as a comprehensive test of the system's transcription of keyboard music. The work was done voluntarily over a period of three months in irregular evening sessions. Most, but not all, of the volunteers were pianists. Two had professional experience of computers; the rest were unfamiliar with them.

The average length of sessions was two to three hours. The amount of use of the facility by individual members of the group ranged from 2 hours to about 20 hours. Each person was given a few minutes tuition then worked under supervision for the remainder of

their first session. Thereafter they worked independently, with the reference manual (appendix 4) available for consultation in cases of difficulty. Most of them felt confident enough to work unattended after the preliminary session, noting occasional points of confusion to be resolved later.

The rate of input for each user gradually increased from an average of about 10 LPH during the initial session. The most experienced user had reached an average of about 18 LPH after 20 hours of use. A large part of the initial slowness was noted as being due to the need to correct mistakes caused by unfamiliarity with the input code. Two users were also slightly hampered by unfamiliarity with the standard keyboard layout.

The braille transcription of the first movement occupies 27 pages. The total time taken to input this music is difficult to estimate since users did not always enter the appropriate times in the log book, and no log was kept during the first month.

The two users who typed the third movement attempted a different method of input. One, after some experience in using the program, encoded the music onto coding forms, which the other later typed into the computer. No accurate timings were taken for this method of input, but it was clearly less efficient than direct encoding. Although some terminal time was saved on initial input, more was required for error correction, because checking of the displayed music against the print was less frequent.

All users among the group of musicians found the program pleasant and generally easy to use. The following criticisms were made by one

or more users when asked what aspects of the system had caused them difficulty.

One user familiar with computer programs for text editing felt that the editing procedure was somewhat unsatisfactory, though he did not have any immediate suggestions for improvement. Most of the delay in accessing a measure for editing was due to the searching of the file from its start for each measure edited. This was subsequently considerably reduced by use of a more efficient means of file access, using the additional entry points to the file mentioned in section 2.8. A further delay is caused by the use of a 1200-baud communication line as mentioned in section 3.3. This is a limitation of the hardware used in the prototype system.

The correction of a mistake involving the use of the editor during input typically requires the typing of four or five characters plus one positioning of the crosswires. This compares favourably with correction of characters, other than those most recently typed, using a text editor. However, there is scope for further improvement in the editing process. For example, editing during input most frequently involves correction of the most recent displayed item. It would be a substantial improvement to combine four of the operations presently necessary to achieve this (enter edit mode, identify item, select "change-item" edit command, exit edit mode) into the single operation "change last item and continue".

Much of the difficulty encountered by users in editing arose from an unclear understanding of the methods used. One person had trouble in distinguishing the functions of the "insert" and "change" edit commands. This is likely to be cleared up by experience or more

comprehensive training.

There was an occasional delay by CIMBAL in accepting a character typed by the user. This was almost entirely eliminated by improvement in the method of file access. While it existed, this delay was a major cause of errors because users were liable to continue typing unaware that some characters were being ignored while a previous character was being processed.

Some users found the jargon in the users' reference manual (appendix 4) hard to follow, although others, more familiar with technical manuals, found it clear. The manual has been written primarily for reference rather than learning. If the system is more extensively used, a separate training manual for beginners would be desirable. Users were introduced to the use of the input code by example rather than by explanation of concepts such as "item". While this approach was considered most suitable for these fairly casual users, it did lead to some confusion. Thus the number of characters cancelled by pressing the interrupt button, and the relationship between the heads of a chord, were not clear to all users.

Input by system developer

The second movement of the Elgar piano quintet and a number of shorter pieces were input by the author, who is intimately familiar with the program and, although not a musician, had some knowledge of music notation before undertaking this project, a knowledge which has been considerably extended during the course of the project.

The average input rate for these pieces was typically 25 to 30

LPH. This, while perhaps not a fair test in itself, gives an indication of the speed which a regular user of the program might expect to achieve with sufficient experience. The figures quoted are not necessarily an upper bound on the speed attainable since this particular user was somewhat slower than a musician in specifying the pitch of notes. On the other hand the timings are based on continuous use during sessions with an average length of an hour or less; a professional user might reasonably be expected to adopt a more leisurely pace.

Accuracy of input

The accuracy of stored music was high for all users after careful proof-reading and correction of errors. The small amount of music input so far in testing the system does not justify specifying this quantitatively. As far as can be determined, all notes were correctly specified. Of the errors which were not picked up by the user the majority were incorrect ordering of items due to unclear or insufficient guidance to the user.

5.2 Evaluation of braille produced

Two methods were used to evaluate the quality of braille transcription of the system:

Firstly, samples of braille music produced by the system were distributed to readers of braille music, who were asked to comment on and criticise the transcription. This indicates the overall acceptability of the braille to the typical reader and the relative significance of deviations from the rules.

Secondly, test pieces were compared with equivalent braille produced manually by the RNIB or given as examples in the braille music manuals. This identifies the existence of deviations from the rules as officially defined or as currently used by manual transcribers.

Evaluation by readers

During the early development of the system a sample of braille music (Sonata in D by Mozart (K.381), second movement, primo part) produced using the system was distributed with a copy of the original print to about 30 braille music readers. These included transcribers, music teachers, and professional and amateur musicians. They were asked to criticise and comment on the braille. Although the braille contained several known errors due to the incompleteness of the translation algorithm, the sample was distributed at that time to assess the reaction of typical readers and to ensure that the system was developing in the right direction.

Additional errors pointed out by readers arose from:

(1) Unclear guidance from the manual (for example, direction in which chords are read in the left hand);

(2) Differences between the 1956 definition and current practice (for example: clef signs are now omitted; octave signs are not automatically shown at the start of every measure);

(3) Misunderstanding of the rules (for example, missing grace notes);

(4) Certain braille rules not implemented at the time of the test (for example, no measure repeat signs, or doubling of signs);

(5) Incorrect input of the printed music (for example, one wrong note, a missing crescendo sign).

Some readers were far more thorough than others in noting mistakes in the braille. The transcribers were the most scrupulous, but no single reader reported every mistake or omission which was noted by one or more other readers. Some were partially restricted by the lack of a sighted assistant to help check the braille against the original print copy.

The translator was modified and improved in the light of the comments from readers. A revised transcription of the same piece of music was then distributed. Comments on the two versions revealed differences of opinion between readers on some matters:

In this piece of music several passages occur where the left hand plays in parallel with the right hand, an octave below. In the first braille version the music was written out in full. One reader suggested that the rule given in the manual (rule 154) should be followed, each measure of the left hand part being replaced by a single octave interval sign, preceded by the appropriate octave sign. Several said that the two hands should be combined by showing the left hand as a doubled octave interval in the right hand part, saving space and making the music easier to learn. In the revised version this method was used. A reader who had not seen the earlier version then complained that this was not clear and "could disturb the mental process of committing the music to memory".

In the first braille version an octave sign was included for the first note of each measure, according to the manual (rule 156). Several readers said that this was necessary only for the first measure of each braille line unless the first note of the measure required an octave sign anyway. Substantial changes were made to the translator to remove the superfluous format-dependent octave signs. A reader of the revised version stated: "Contrary to what others may have told you, it is an established rule in keyboard music that every bar, in each hand part, in bar-over-bar disposition must have an appropriate octave sign before the first note of that bar."

Some readers felt that the inclusion of inkprint page and line numbers in the braille was an unnecessary distraction; others, especially blind teachers of sighted pupils, felt that it was important to include them. This is a legitimate difference of use which could be allowed for by producing two alternative transcriptions of a piece when necessary.

These examples show that it is impossible to produce what all readers will regard as a definitive braille transcription of a piece of music. There is a certain amount of disagreement between experts on the finer points of the braille music code. There was, however, general agreement on most points.

Significance of errors

Where there was agreement on the braille rules, there was in some cases a variation in the significance attached to particular deviations from the rules by different readers. Errors which alter the meaning of the notation are clearly unacceptable. Thus showing

intervals in the wrong direction by brailling the upper note of a chord instead of the lower is a major fault from the reader's point of view. However, most of the errors in the original version of the Mozart sample affected ease of reading and learning rather than the meaning of the notation. The opinions of readers were useful in assessing the significance and acceptability of each of these variations from the standard.

The following are examples of errors which, being noted by many readers, were treated as unacceptable and subsequently corrected: failure to align music in both hands following words of expression; unnecessary use of a repeat sign to replace a single cell; absence of measure repeats; failure to use a part-measure repeat where accidentals were involved; doubled slurs used instead of bracket slurs for phrasing signs; unnecessary octave signs at the start of some measures.

By contrast only one reader observed that doubling of an interval sign had not been used for a sequence of four consecutive chords which were brailled on the same line. Use of uncontracted braille for headings was considered only a minor defect by nearly all readers. Where two or more literary abbreviations or expressions occur together the rule appears to be that signs of wider application should be transcribed first in braille. The expression "p dolce" occurring in the sample should accordingly appear as "dolce p" in braille, but was shown in the same order as in the print. Only one reader mentioned this. When it was specifically pointed out to other readers, most of them felt that this was a trivial detail. The reversal of such pairs of expressions would be extremely difficult to achieve automatically since it depends on the meaning of the expressions.

A set of 19 short pieces for clarinet were transcribed using the computer system for a student learning to play the instrument. After reading through the braille carefully she reported only one genuine error in the transcription and said that it was "beautifully clear". She had no objection to the extent to which measure repeat signs were used. This is one aspect of the braille code where the transcriber has a considerable amount of discretion in achieving a balance between clarity and conciseness in the braille.

A short piece of vocal music, "Hosanna to the Son of David" (anthem for six voices) by Thomas Weelkes was input and the tenor part was translated to braille. The braille was found satisfactory by a blind musician who read it, apart from formatting, failure to use repeat signs for words, and failure to mark the special indications for words with an ambiguous number of syllables ("heavens" and "blessed"). Repeat signs for words were regarded here as "desirable but not essential" by the RNIB.

The piano part of the Elgar quintet mentioned above is the most recent, the longest and the most complex piece used to test the translator. This piece includes nearly all aspects of braille music notation which occur in keyboard music. The quality of the braille for this piece has not yet been fully evaluated, but the first movement has been found substantially correct by a blind musician who has read through it. There are some mistakes in the transcription which cannot be considered insignificant. Some of these are a result of insufficient instructions to the input operator about the order of certain signs (for example, time signature, key signature and rehearsal numbers). A few are due to remaining faults in the translator which need to be corrected. For example, the translator

currently treats doubling of different intervals as independent of each other. This is incorrect according to the manual (rule 51) and can occasionally cause ambiguity.

Comparison with manual transcription

Pieces from the Associated Board of the Royal Schools of Music 1977 Grade V Pianoforte Examinations were translated and compared with RNIB transcriptions of the same pieces. The system was modified and improved until the computer transcription was close to the manual transcription. A few discrepancies were allowed to remain because their significance did not appear to justify the amount of effort necessary to adapt the translator to avoid them.

One of these pieces (Gigue by Max Reger, opus 44, no. 9) was translated by computer. A second version of this transcription was also produced using the computer system by editing the braille to correspond cell for cell with the RNIB transcription of the same piece. The two versions were submitted, with a list of differences between them, to the Director of Music of the RNIB for comment. While he felt the RNIB version was slightly better where there was any significant difference, there was nothing in the computer-translated version which he found unacceptable.

5.3 Program portability

The bulk of CIMBAL is coded in Rank Xerox Data Systems Extended FORTRAN IV-H to run on a Sigma 5 computer. For portability, the source code is intended to conform as far as possible to USA standard FORTRAN X3.9-1966. However, the inadequacy of standard FORTRAN for

byte handling and interactive use, and local operating system interface, necessitate some deviations from the standard. Details of these are given in appendix 3.

The portability of the software has not yet been thoroughly tested by full implementation at a different installation. A partial test of portability has been made by transferring the program to a Burroughs B6700 computer at the University of Warwick, with the non-standard features modified or replaced as necessary. A few previously unnoticed non-standard FORTRAN statements have been identified as a result of this transference. The more sophisticated operating system of the B6700 effectively precludes the use of single character input. Otherwise the program has been compiled and run successfully on this machine, but not thoroughly tested.

The practical use of this system for braille production depends on its implementation by a braille music publisher. The possibility of implementation at the RNIB, the most likely potential user of the system in the United Kingdom, has been investigated, and there appear to be no major technical problems involved. The RNIB has recently installed a new computer system for braille production, based on two GEC 4070 processors. This installation has all the hardware necessary for operation of the braille music program, including LED120 braille embossing terminals. The two existing graphics terminals are expected to be fully used for editing of literary braille and a separate graphical display unit would probably be required for music input. There is likely to be sufficient spare processing time available to run CIMBAL on the RNIB system. Magnetic tape units are compatible with those of the computer on which the system has been developed, enabling direct transfer via magnetic tape of programs and any music

data already stored.

The software appears to be compatible with the RNIB system. A computer program (DOTSYS III-F) written in FORTRAN for translation of text to contracted literary braille has already been transferred from the Sigma 5 computer at the University of Warwick to the RNIB installation without any major problems. The use of character by character input for the music program is the feature most likely to cause difficulty in transference of the program to other installations. This is unlikely to be a problem at the RNIB because character by character input is already used for on-line verification of text.

It is unlikely that a computer-based system will replace manual transcription before it has been thoroughly tested and shown to attain the same standards of transcription as traditional methods at less cost. It can, however, be used to supplement existing production, with the braille being proof-read by skilled transcribers in the conventional way, and any errors being corrected, using the system's facility for braille editing, before final storage and embossing. Remaining deficiencies in the system giving rise to these errors could then be corrected in parallel with braille production.

Chapter 6. Conclusions and further work

The system has been used reliably over a period of several months. A small scale evaluation has shown that the two major objectives have been largely met: efficient use by an operator unfamiliar with braille or computers, and production of good quality braille music with most of the rules for abbreviation implemented. The design criteria listed in section 1.5 have been achieved with the following reservations:

The need for the user of the system to have some musical knowledge is somewhat greater than was originally anticipated. Although most music notation may be input correctly by a non-musician, there are instances where some musical knowledge is required to interpret the print correctly. These include, for example, symbols merely implied in the print but explicitly included in the braille, and the ordering of symbols where these are displaced in the printed copy for purely typographical reasons. For the operator with little knowledge of music notation, therefore, an initial brief survey of the printed music by a musician to mark difficult points seems desirable.

The preliminary evaluation suggests that, with this assistance, the non-musician with little experience in the use of the system can achieve a transcription rate comparable with that quoted by the RNIB for manual transcription of braille music. However, music is input more efficiently by people with prior knowledge of music notation, or a wish to acquire this knowledge. The rate of input for such people in the early stages of use is up to one and a half times that of manual transcription. The input rate of the author, the only user

with experience of the system equivalent to more than a few working days, suggests that a regular user may achieve an average transcription speed twice that of manual transcription or better.

It has not proved possible to develop a translator which will produce cell for cell an exact copy of the braille music produced by a skilled transcriber. This is partly because a certain amount of musical judgment is sometimes required in determining the best braille transcription of a given passage, and partly because there is a degree of flexibility in the rules of braille music transcription whereby it is unlikely that two different transcribers would produce identical braille for the same piece of music.

However, evaluation of the quality of braille music produced has shown this to be considered satisfactory for instrumental music by the majority of the braille music readers who helped in the evaluation, even when they were requested to be highly critical.

Test pieces of single line music for a solo instrument have been transcribed with no known errors. Simple keyboard music has been produced to a standard judged acceptable by authoritative readers of braille music. There are some minor flaws remaining to be corrected in the transcription of more complex instrumental music notation. The small amount of vocal music so far produced is readable, but requires improvement in its layout, and does not yet include full grade 2 contraction of text or the use of repeat signs for text.

Facilities for braille editing after translation are available for use where perfect results are required. This is regarded as an essential feature of any production system, including some current

computer-based text translation systems. The correction, where necessary, of small errors in the braille in this way makes it feasible to use the system in a production capacity while continuing to make refinements to the translation process to avoid the future repetition of such errors.

It is therefore suggested that the system is now suitable with minimal modification for production use by a braille publisher. It may be used to supplement existing braille music transcription resources, which in the United Kingdom have remained at a constant level while literary braille production has been considerably increased following the introduction of a computer-based production system at the RNIB. A long-term evaluation of the system is recommended in parallel with production, since it is envisaged that further improvements in the system will take place only after a more widely ranging assessment based on feedback from users of braille music.

Further Work

Some further work is necessary to make the translation of vocal music fully acceptable. The formatting is currently more appropriate to instrumental music, and should be adapted to represent the phrasing of the text and divide the music between braille lines at more suitable places. A proper grade 2 text translator should replace the present version, which only contracts to a fairly good approximation to grade 2. The main problem here concerns the interaction of words and music rather than simply the contraction of literary text, for which acceptable algorithms already exist.

Some of the less frequently used braille music signs and rules which have not been implemented (listed in chapter 4) may be added as required. Consideration should be given to the special requirements of music for particular instruments, depending on the demand for such music. Two of the readers consulted, for example, found guitar music extremely difficult to obtain in braille.

The variations in format and amount of detail included, and the variations between the usage of different countries, have not been allowed for in the prototype system. Apart from formatting, most of these require minor additions to the program, which may be implemented as the demand for them arises.

Most of the omissions mentioned above and elsewhere are largely due to the restricted amount of core storage available to the prototype system. Implementation of these features is not expected to cause undue trouble on a machine where program size is a less crucial factor.

The interactive part of the program works satisfactorily for the purpose of braille production, except that some further work is necessary to ensure that different symbols are not displayed at the same position on the screen. This side of the system might be adapted for applications other than braille production. The music editor could be developed if necessary to include the various additional facilities of the more powerful text editing systems. One such feature lacking in the system as it stands which has been found to be desirable is the possibility of merging two separate files, allowing two people to work on a particular piece in parallel.

The diverse nature of music notation, the complexity of the translation process, and the interaction between the different braille music rules make it difficult to formulate an artificial piece of test data which can be used, as is done with literary braille translation algorithms, to check thoroughly for correct translation in nearly all circumstances. The use of the system in a production capacity is expected to reveal notational combinations leading to additional problems presently unforeseen. The overall quality of braille translation will continue to converge on a desired standard as these problems are identified and solved.

References and bibliography

American Printing House for the Blind. Computer-Assisted Translation of Braille Music; Progress Reports: May 10, 1971 - May 9 1972; June 28, 1972 - June 27, 1973; September 1, 1974 - November 30, 1974; December 1, 1974 - February 28, 1975; March 1, 1975 - May 31, 1975; June 1, 1975 - August 31, 1975. Louisville, Kentucky, 1972-5.

American Printing House for the Blind. Revised International Manual of Braille Music Notation 1956, 1975 American Addendum. Louisville, Kentucky 1975.

Bauer-Mengelberg, Stefan. The Ford-Columbia Input Language. Musicology and the Computer (ed. Barry S. Brook), pp. 48-52. New York, 1970.

Boker-Heil, Norbert. Plotting Conventional Music Notation. Journal of Music Theory, vol. 16, pp. 72-101, 1972.

Brook, Barry S. (ed). Musicology and the Computer: Three Symposia. American Musicological Society, Greater New York Chapter 1965-6. City University of New York Press, New York, 1970a.

Brook, Barry S. The Plaine and Easie Code. Musicology and the Computer (ed. Barry S. Brook), pp. 53-56. New York, 1970b.

Byrd, Donald. A System for Music Printing by Computer. Computers and the Humanities, vol. 8, pp. 161-172, 1974.

Cantor, Don. A Computer Program that Accepts Common Musical Notation. Computers and the Humanities vol. 6, pp. 103-109, 1971.

Computers and the Humanities. Music in Machine-Readable Form. Vol. 6, pp. 176-178, 1972.

De Garmo, Mary Turner. Introduction to Braille Music Transcription. Division for the Blind and Physically Handicapped, Library of Congress, Washington 1974.

Erickson, Raymond F. "The DARMS Project": A Status Report. Computers and the Humanities, vol. 9, pp. 291-298, 1975.

Erickson, Raymond F. DARMS: a Reference Manual. Queens College, New York, 1977.

Fredlund, Lorne D. and Jeffrey R. Sampson. An Interactive Graphics System for Computer-Assisted Musical Composition. International Journal of Man-Machine Studies, vol. 5, pp. 585-605, 1973.

Gillies, S.G. and S.J. Goldsack. Computer Coding of Music Scores Using an On-Line Organ Keyboard. Braille Automation Newsletter, no. 2, pp. 62-64, August 1976.

Gomberg, David A. A Computer-Oriented System for Music Printing. Computers and the Humanities, vol. 11, pp. 63-80, 1977.

Gould, Murray J. and George W. Logemann. ALMA: Alphameric Language for Music Analysis. Musicology and the Computer (ed. Barry S. Brook), pp. 57-90. New York, 1970.

Hiller, Lejaren A., Jr. and R.A. Baker. Automated Music Printing. Journal of Music Theory, vol. 9, pp. 128-152, 1965.

Jackson, Roland and Philip Bernzott. A Musical Input Language and a Sample Program for Musical Analysis. Musicology and the Computer (ed. Barry S. Brook), pp. 130-150. New York, 1970.

Humphreys, John. Automatic Translation by Computer of Music Notation to Braille. Braille Research Newsletter, no. 5, pp. 5-12, July 1977.

Kassler, Michael. An Essay Toward Specification of a Music-Reading Machine. Musicology and the Computer (ed. Barry S. Brook) pp. 151-175. New York, 1970.

Lincoln, Harry B. (ed). The Computer and Music. Cornell University Press, 1970.

Lincoln, Harry B. Use of the Computer in Music Research: A Short Report on Accomplishments, Limitations and Future Needs. Computers and the Humanities, vol. 8, pp. 285-289, 1974.

Mackenzie, Clutha. World Braille Usage. UNESCO, Paris, 1954.

Mars, P. and J.M. Cattanach. Automatic Transcription of Keyboard Music. Proceedings of the Institution of Electrical Engineers, vol. 124, pp. 436-440, 1977.

Mars, P. Keyboard Music Transcription by Computer. Electronics and Power, vol. 23, pp. 651-653, 1977.

McLean, B. Translating DARMS into Musical Braille. Braille Automation Newsletter, no. 2, pp. 65-67, August 1976.

Meneau, Christophe. Private communication, 1978.

Morawski, Paul (with P. Howard Patrick). Notes on a Computer Subsystem for Transcription of Music into Braille. Music Department, The American University, January 1977.

Patrick, P. Howard and Patricia Friedman. Computer Printing of Braille Music Using the IML-MIR System. Computers and the Humanities, vol. 9, pp. 115-121, 1975.

Patrick, P. Howard and Rosalind E. Patrick. Computers and Music Braille. Braille Automation Newsletter, no. 2, pp. 52-61, August 1976.

Patrick, P. Howard. An Introduction to the Computer-Printing of Braille Music using the I.M.L.-M.I.R. (Intermediary Music Language - Music Information Retrieval) System. Music Department, The American University, December 1976.

Prerau, David S. Do-Re-Mi: A Program that Recognises Music Notation. Computers and the Humanities, vol. 9, pp. 25-29, 1975.

Robbins, Donald C. EDP Applications to Musical Bibliography: Input Considerations. Special Libraries, vol. 63, part 9, pp. 373-378, 1972.

Royal National Institute for the Blind. A restatement of the Lay-out

Definitions and Rules of the Standard English Braille System. Part I, Basic code, 1969. Part II, Advanced Code, 1971.

Schack, Ann and Joseph Schack. Studies in the Automation of Braille Mathematics and Music, Final Report to the Department of Health, Education and Welfare. American Printing House for the Blind, Louisville, Kentucky, June 1969.

Smith, Leland. Editing and Printing Music by Computer. Journal of Music Theory, vol. 17, pp. 292-309, 1973.

Spanner, H.V. Revised International Manual of Braille Music Notation 1956, Part 1, Western Music. World Council for the Welfare of the Blind, Paris, 1956.

Spanner, H.V. Lessons in Braille Music. American Printing House for the Blind, Louisville, Kentucky, 1961.

Styles, B.C. Describing Music to a Computer. International Journal of Man-Machine Studies, vol. 6, pp. 125-134, January 1974.

Tucker, W.H., R.H.T. Bates, S.D. Frykberg, R.J. Howarth, W.K. Kennedy, M.R. Lamb and R.G. Vaughan. An Interactive Aid for Musicians. International Journal of Man-Machine Studies, vol. 6, pp. 635-651, November 1977.

Watkins, William and John Siems. SAMBA and RUMBA: Systems for Computer Assisted Translation of Braille Music. Braille Automation Newsletter, no. 2, pp. 47-51, August 1976.

Watson, Edward. Elementary Lessons and Exercises in the Revised (1922) Braille Music Notation. National Institute for the Blind, London, 1926.

Watson, Malcolm. The Perception of Tonality in Music. Department of Psychology, University of Warwick, May 1978.

Wenker, Jerome. A Computer Oriented Music Notation including Ethnomusicological Symbols. Musicology and the Computer (ed. Barry S. Brook), pp. 91-129. New York, 1970.

Wilkinson, Sally. Investigation into the Feasibility of Computer-Assisted Transcription of Music into Braille Notation. M.Sc. Project Report, School of Information Sciences, The Hatfield Polytechnic, September 1975.

Wilkinson, Sally. Computer-Assisted Transcription of Braille Music. Braille Automation Newsletter, no. 2, pp. 68-71, August 1976.

Wolcott, Norman M. and Joseph Hilsenrath. A Contribution to Computer Typesetting Techniques: Tables of Co-ordinates for Hershey's Repertory of Occidental Type Fonts and Graphic Symbols. National Bureau of Standards Special Publication 424. U.S. Department of Commerce, Washington, 1976.

Wolff, Anthony B. Problems of Representation in Musical Computing. Computers and the Humanities, vol. 11, pp. 3-12, 1977.

APPENDICES

```
!JOB JH,COMPILE MUSIC PROGRAM ROOT
!EXTRABCD 6800
!ALL (FIL,G0),(RSI,27),(FOR,B),(FSI,1000)
!FORTRAN G0,S,NS
C
C
C     ******************************************
C     *                                        *
C     *                CIMBAL                  *
C     *                                        *
C     *    A PROGRAM TO INPUT, EDIT AND STORE  *
C     *    MUSIC AND TRANSLATE IT TO BRAILLE.  *
C     *                                        *
C     *                JOHN HUMPHREYS 1976-1978*
C     ******************************************
C
C     WRITTEN IN RANK XEROX DATA SYSTEMS EXTENDED
C     FORTRAN IV-H AND MACRO-SYMBOL TO RUN ON A
C     SIGMA 5 COMPUTER.    SEE SEPARATE PROGRAM
C     DOCUMENTATION FOR FULL DETAILS OF OPERATION
C     AND PROGRAMMING, AND DEVIATIONS FROM USA
C     STANDARD FORTRAN X3.9-1966.
C
C
C     INPUT AND OUTPUT UNITS
C     104   GRAPH PLOTTER OUTPUT FILE
C     108   LINEPRINTER OUTPUT FILE
C     119   BRAILLE EMBOSSER OUTPUT FILE
C     120   RANDOM ACCESS DISC FILE
C     121   ARCHIVE TAPE
C     123   NEW ARCHIVE TAPE CREATED BY COPYING
C
C
C  ALL LABELLED COMMON BLOCKS USED IN THE PROGRAM
C  ARE LISTED BELOW.  THESE GIVEN AS COMMENTS ARE
C  NOT USED IN THE ROOT, THE NUMBER IN THE MARGIN
C  BEING THE HIGHEST SEGMENT IN WHICH THEY APPEAR.
C
      COMMON MINBUF(90),MOPBUF(90),NINBUF,NOPBUF,
     * JINSEC,JOPSEC,JINBUF,JOPBUF,JINLK,JOPLK,
     * NILENG,MIBAR(64),NOLENG,MOBAR(64),JIBAR,
     * JOBAR,JBBUF,NFLAGS,MFLAGS(3)
      COMMON /AD/ JIADDR,JOADDR
C 5   COMMON /AR/ KATDEV,KISIZE,KIBASE,KSBASE,
C     * KVBASE,KNBASE,KMTMES(3),MINDEX(540),NFNAME,
C     * MFNAME(8)
C 4   COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /BY/ NBYTES,MBYTES(3),NSECTS,NVERS
      COMMON /B1/ IFIRST,JOPAGE
      COMMON /CB/ MXCOM(4)
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /CT/ JLABEL
C 4   COMMON /DB/ ISDUB,NDBL,ISSIG,NSLURS,JSLUR(2)
C 32  COMMON /DI/ MBEAMX(8),MBMYH(8),MBMYL(8),
C     * MBEAML(4),JBEAM,ISBEAM
C 3   COMMON /DN/ JSDIR,JFDIR,JLEV,JALEV,JPVLEV,
C     * IXSM,IYSM,IYSMAX,IYSMIN,MXTEXT(3)
      COMMON /DP/ IXSHIF,IYSHIF
C 324 COMMON /DX/ JBEAMX,JBEAMY,JPVAX
      COMMON /ED/ NEDBAR,NEDSTV,NELENG,JEDCMD
      COMMON /EE/ JEDIT,JEFREE,JEBACK,ISITEM
C 44  COMMON /EK/ KBRDEV,KMLPP(2)
C 44  COMMON /EM/ JOUT,JOLINE,NBHDR,MBHDR(10)
      COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSGZ,
     * ISREST,ISAHED,ISXPOS,ISEMB
      COMMON /FG/ KLEVEL,IEA
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FI/ ISWRIT,ISSCAN
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *    ISERR,ISNXST,ISINPT
C 43  COMMON /FO/ MEND(4),MSPLIT(3),ISSPLT,ISDOT,
C     * ISHYPH,IPAGE,ILINE,JSTART,NSLENG,MSBAR(10)
      COMMON /FS/ JCSTAT,JNSTAT,MSTAT(3)
      COMMON /GP/ JGROUP,MGROUP(7)
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /HE/ MCLEF(10),MPART(10)
C 421 COMMON /IS/ ICITEM,ICTYPE,MCSAVE(9),
```

```
C     * MCXTRA(4),ISPRT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
C 3   COMMON /IV/ INTYPE,ISBAR
      COMMON /JD/ JSTAFF,JDLINE,JDHIGH,JDLOW
C 3   COMMON /JO/ JX,JY,JDIST
C 42  COMMON /JS/ JSLM,JMATCH,JNLAST,JLASTN
C 4   COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
C 4   COMMON /KC/ KWHYPH,KTOUT,KSPLIT,KBREAK,KRBAR
      COMMON /KE/ KESPAC,KEZERO,KEDOT,KEHYPH,KENAK
      COMMON /KK/ KBAR,KBDOTS,KCLEF,KTIME,KKEY,
     * KTEXUP,KTEXDN,KTEXWD,KFLAT,KNATUR,KSHARP,
     * KTIE,KNDOT,KNREPT,KNSDEL,KSWICH
      COMMON /KM/ KZEROS(2),KMRSIZ,MREST(1)
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /KY/ JKEYNO,JKEYSG,MKPCH(2),MTKPCH(2)
      COMMON /LK/ MBLINK(14),MFLINK(14),
     * MLIST1(14),MLIST2(14),MLIST3(14),MLIST4(14)
      COMMON /LO/ NLOCS,JLOCN
      COMMON /LT/ MLOCN(10),MCTK(5),MFIND(10)
      COMMON /MC/ KBSIZE,KWSIZE,KXMAX,KYMAX,KFSIZE
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /ME/ MEMPTR(10)
      COMMON /MM/ IXST,JMEM
      COMMON /MO/ JIMODE,JOMODE
      COMMON /MR/ NNDOWN
C 43  COMMON /MT/ NTLENG,MTBAR(64)
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /ND/ NXLENG,NSHIFT,JBXL,NEXTX
      COMMON /NG/ JNGRP,MJNOTE(4)
      COMMON /NH/ JNHEDS,KNHMAX
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JCOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /PL/ ISPLOT
      COMMON /PP/ ISPOSE,JTERM,ISDBUG,ISHOD,
     * ISGRD1,ISSAG,ISWICE
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
C 3   COMMON /RP/ ISAUTO,JEDCOD,JESEL,ISDISP
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /SD/ MYAXID(3)
      COMMON /SE/ JCMD,KSEGS(7)
      COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
      COMMON /SU/ MHCLEF(10)
C 4   COMMON /TA/ MPITCH(4),MREPT(4),MRESTS(4),
C     * MCXMEM(3),MPCONT(4),ISPCON,JBNO,JISNO,
C     * JOSNO,ISCANC,ISLINR,JLINR
      COMMON /TD/ IXC,IYC,IXB,IXR
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
C 4   COMMON /TJ/ JCKMAX,MCHECK(100)
C 42  COMMON /TK/ NRLENG,MRBAR(4),MTKEY(4),
C     * ISGR1,ISBEAT,ISPVBT,ISCMPD,ISPAUS,JXPOSE
      COMMON /TL/ NNAME,MNAME(8)
C 4   COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
C     * ISQUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
C     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
      COMMON /TT/ JTTEMP,JXTEMP
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      COMMON /TY/ ISCONT,ISBLOW
      COMMON /WS/ L1,L2,L3,JBYTE
C     PROGRAM-GENERATED TABLES
C 2   COMMON /XA/ MMESIJ(544)
C 31  COMMON /XB/ MCODES(64)
C 32  COMMON /XC/ MCORD1(20)
C 321 COMMON /XD/ MCORD2(56)
C 322 COMMON /XE/ MCORD3(12)
C 324 COMMON /XF/ MCORD4(76)
C 4   COMMON /XG/ MCELLS(112)
C 421 COMMON /XH/ MTABLE(220)
C BGD COMMON /XJ/ MHYTAB(248)
C
C     INITIALISE BACKGROUND OR FOREGROUND PROGRAM
      CALL BGDFGD
      JCMD = -1
```

```
C
C     RETURN POINT FOR ALL TERMINAL INTERRUPTS
C     EXCEPT DURING INPUT OF MUSIC
   19 INIT = 1
    2 CALL BREAK(&19)
C     REQUEST COMMAND
      CALL LSEG(K2)
      CALL CTRL
C     LOAD REQUIRED OVERLAY SEGMENT IF ANY
      CALL LSEG(NBYTE(KSEGS,JCMD))
C     BRANCH INDEXED BY COMMAND NUMBER
C    1  DELETE        2  EDIT          3  INSERT
C    4  BUILD         5  ADD           6  DISPLAY
C    7  BRAILLE       8  FORMAT        9  EMBOSS
C   10  SAVE         11  RESTORE      12  ERASE
C   13  FILES        14  CLEAR        15  MODE
C   16  STOP         17  SUMMARY      18  SCALE
C   19  PLAY         20  CHANGE       21  PRINT
C   22  SQUEEZE      23  COPY         24  FLAGS
C   25  OPTIONS      26  INITIALISATION
      GOTO (2,26,28,22,22,24,4,4,4,48,48,48,48,2,
     *       2,9,6,2,5,2,24,2,48,2,2,48),JCMD
      GOTO 2
C
C     BUILD, ADD
   22 CALL INPUT
      GOTO 2
C
C     DISPLAY, PRINT
   24 CALL MDISP
      GOTO 2
C
C     EDIT MEASURE
   26 CALL EDIT
      GOTO 2
C
C     INSERT BAR
   28 CALL INSERT
      GOTO 2
C
C     BRAILLE, FORMAT, EMBOSS
    4 CALL BRAILE
      GOTO 2
C
C     INITIALISATION, RESTORE, FILES, SAVE, COPY
   48 CALL ARCHIV
      GOTO 2
C
C     PLAY
    5 CALL PLAY
      GOTO 2
C
C     SUMMARY
    6 CALL DHDR
      GOTO 2
C
C     END OF RUN
    9 STOP
      END
C
C
C     ****************************************
C     *                                      *
C     *        MUSIC STORAGE ROUTINES        *
C     *                                      *
C     ****************************************
C
C  THE FOLLOWING MUSIC STORAGE ROUTINES ARE NOT
C  CALLED WITHIN THE ROOT EXCEPT BY OTHER MUSIC
C  STORAGE ROUTINES, AND MAY BE OMITTED FROM ANY
C  PROGRAM USING THE ROOT OF THIS PROGRAM FOR
C  MUSIC RETRIEVAL ONLY.
C
      SUBROUTINE STLINE(N)
C     STORE *MOBAR* IN FILE STARTING AT CURRENT
C     OUTPUT ADDRESS.  *N* IS THE FLAGS FIELD.
C     RETURNS *NBYTES* = NUMBER OF BYTES STORED.
      COMMON MINBUF(90),MOPBUF(90),NINBUF,NOPBUF,
     * JINSEC,JOPSEC,JINBUF,JOPBUF,JINLK,JOPLK,
     * NILENG,MIBAR(64),NOLENG,MOBAR(64),JIBAR,
```

```
     * JOBAR,JBBUF,NFLAGS,MFLAGS(3)
      COMMON /AD/ JIADDR,JOADDR
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /BY/ NBYTES,MBYTES(3),NSECTS,NVERS
      COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSQZ,
     * ISREST,ISAHED,ISXPOS,ISEMB
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KM/ KZEROS(2),KMRSIZ,MREST(1)
      COMMON /MO/ JIMODE,JOMODE
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /SE/ JCMD,KSEGS(7)
      COMMON /WS/ L1,L2,L3,JBYTE
      LOGICAL CBS
C
      MFLAGS(1) = N
      NFLAGS = 1
      IF (JOMODE-2) 14,16,18
C     PRINTED MUSIC
   14 IF (NOLENG+MFLAGS(1) .NE. 0 .AND. .NOT.
     * CBS(MOBAR,KO,MREST,KO,K3)) GOTO 18
C     STORE NULL BAR
      CALL OPBYTE(254)
      NBYTES = 1
      GOTO 3
C     UNFORMATTED BRAILLE
   16 NFLAGS = 3
   18 CALL OPBYTE(NOLENG+NFLAGS)
C     SAVE ADDRESS OF FLAGS FIELD
      JOADDR = KDATA * JOPSEC + JOPBUF
      DO 190 L1=1,NFLAGS
  190 CALL OPBYTE(MFLAGS(L1)+128)
      NBYTES = NOLENG + NFLAGS + 1
      IF (NOLENG .EQ. 0) GOTO 3
C     STORE MUSIC DATA
      DO 20 L1=1,NOLENG
   20 CALL OPBYTE(NBYTE(MOBAR,L1))
      NOLENG = 0
C     EXIT IF SQUEEZING
    3 IF (ISSQZ .NE. 0) GOTO 9
      IF (JCMD .LE. 3) GOTO 6
      MTAIL(JOMODE) = JOPSEC
C     EXIT IF END-OF-SECTION RECORD
      IF (NOLENG .EQ. 0 .AND. N .LT. 0) GOTO 9
C     INCREMENT MEASURE COUNT
      NOS = NOS + 1
      IF (NOS .LE. MLINES(JOMODE)) GOTO 6
      NOS = 1
    4 NBARS = NOB
      MBARS(JOMODE) = NBARS
      NOB = NOB + 1
    6 IF (ISINPT .EQ. 0) GOTO 9
C     FOR SAFETY, SECURE FILE AFTER EACH MEASURE
C     DURING INPUT OF MUSIC
      JIADDR = KDATA * JOPSEC + JOPBUF
      CALL SECURE
C     RESTORE CURRENT OUTPUT POSITION
      CALL OPPOSN(JIADDR)
    9 RETURN
      END
C
C
      SUBROUTINE SECURE
C     UPDATE PHYSICAL DISC FILE.
      CALL HDRSET
      CALL FUPDAT
      RETURN
      END
C
C
      SUBROUTINE HDRSET
C     UPDATE FILE HEADER BLOCK.
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /HE/ MCLEF(10),MPART(10)
      COMMON /MO/ JIMODE,JOMODE
```

```
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION MFHDR(14)
      EQUIVALENCE(MFHDR,KNLIST),(NLNB,MLINES(1))
      CALL OPPOSN(10)
      MBARS(JIMODE) = NBARS
      MLINES(JIMODE) = NLINES
      L2 = 4 * KNLIST + 2
      DO 20 L1=1,L2
   20 CALL OPHW(MFHDR(L1))
      DO 30 L1=1,NLNB
   30 CALL OPBYTE(NBYTE(MPART,L1))
      RETURN
      END
C
C
      SUBROUTINE XBYTE(NNBYTE,NPTR)
C     STORE *NNBYTE* AT FILE ADDRESS *NPTR* AND
C     RESTORE CURRENT FILE OUTPUT POINTER.
      COMMON MC1(183),JOPSEC,JINBUF,JOPBUF
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /WS/ L1,L2,L3,JBYTE
      L1 = KDATA * JOPSEC + JOPBUF
      CALL OPPOSN(NPTR)
      CALL OPBYTE(NNBYTE)
      CALL OPPOSN(L1)
      RETURN
      END
C
C
      SUBROUTINE OPHW(N)
C     STORE NON-NEGATIVE INTEGER *N* AS 16-BIT
C     BINARY VALUE AT CURRENT OUTPUT ADDRESS.
      COMMON /WS/ L1,L2,L3,JBYTE
      L3 = N / 256
      CALL OPBYTE(L3)
      CALL OPBYTE(N-256*L3)
      RETURN
      END
C
C
      SUBROUTINE OPBYTE(NNBYTE)
C     STORE *NNBYTE* IN FILE AT CURRENT OUTPUT
C     ADDRESS AND INCREMENT ADDRESS.  OBTAIN NEW
C     SECTOR FROM HEAD OF FREE CHAIN AS REQUIRED.
      COMMON MINBUF(90),MOPBUF(90),NINBUF,NOPBUF,
     * JINSEC,JOPSEC,JINBUF,JOPBUF,JINLK,JOPLK
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FI/ ISWRIT,ISSCAN
      COMMON /MO/ JIMODE,JOMODE
      JOPBUF = JOPBUF + 1
      IF (JOPBUF .LE. KDATA) GOTO 5
C     OBTAIN NEXT SECTOR FOR OUTPUT
      IF (JOPSEC .EQ. MTAIL(JOMODE))
     *                    CALL FLINK(JOPSEC,JFREE)
      JOPSEC = JOPLK
      JOPBUF = 1
    5 CALL OPSET
      IF (JFREE .EQ. NOPBUF) JFREE = JOPLK
C     STORE BYTE IN FILE AND SET FILE-WRITTEN FLAG
      CALL STBYTE(NNBYTE,MOPBUF,JOPBUF+2)
      ISWRIT = 1
      RETURN
      END
C
C
      SUBROUTINE FLINK(NSECT1,NSECT2)
C     LINK SECTOR *NSECT1* TO SECTOR *NSECT2*.
      COMMON MINBUF(90),MOPBUF(90),NINBUF,NOPBUF,
     * JINSEC,JOPSEC,JINBUF,JOPBUF,JINLK,JOPLK
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /FI/ ISWRIT,ISSCAN
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
C     OBTAIN CORRECT SECTOR IN BUFFER
      CALL OPPOSN(KDATA*NSECT1)
      JOPLK = NSECT2
      IF (NINBUF .EQ. NOPBUF) JINLK = JOPLK
```

```
C     STORE FORWARD LINK AT START OF SECTOR
      CALL STHW(JOPLK,MOPBUF,K1)
      ISWRIT = 1
      RETURN
      END
C
C
      SUBROUTINE OPPOSN(NPTR)
C     SET CURRENT FILE OUTPUT ADDRESS TO *NPTR*.
      COMMON MC1(183),JOPSEC,JINBUF,JOPBUF
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      JOPSEC = NPTR / KDATA
      JOPBUF = NPTR - KDATA * JOPSEC
      CALL OPSET
      RETURN
      END
C
C
      SUBROUTINE OPSET
C     OBTAIN CORRECT OUTPUT SECTOR IN *MOPBUF*,
C     ENSURING *JOPLK* IS CORRECTLY SET.
      COMMON MC1(181),MOPBUF,JINSEC,JOPSEC
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      IF (JOPSEC .EQ. MMPBUF) GOTO 9
      CALL FUPDAT
      CALL DREAD(JOPSEC,K1)
    9 RETURN
      END
C
C
      SUBROUTINE FUPDAT
C     UPDATE PHYSICAL DISC FILE.
      COMMON MC1(180),MINBUF,MOPBUF
      COMMON /FI/ ISWRIT,ISSCAN
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
C     NO ACTION IF NO SECTOR CURRENTLY IN *MOPBUF*
C     OR FILE UNCHANGED SINCE LAST READ
      IF (ISWRIT .EQ. 0 .OR. MOPBUF .LT. 0) GOTO 9
C     WRITE CONTENTS OF *MOPBUF* TO DISC FILE
      CALL DISKIO(K1,MOPBUF,K1)
      ISWRIT = 0
C     CANCEL CONTENTS OF *MINBUF* IF SAME SECTOR
      IF (MINBUF .EQ. MOPBUF) MINBUF = -1
    9 RETURN
      END
C
C
C     ***********************************************
C     *                                             *
C     *          MUSIC RETRIEVAL ROUTINES           *
C     *                                             *
C     ***********************************************
C
C
      SUBROUTINE BFIND(NNBAR,NNLINE)
C     RETURN BAR *NNBAR* STAVE *NNLINE* IN *MIBAR*
C     LEAVING INPUT ADDRESS AT END OF MEASURE.
      COMMON MC1(188),NILENG,MIBAR(64),
     * MC2(65),JIBAR,JOBAR,JBBUF
      COMMON /FI/ ISWRIT,ISSCAN
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /HE/ MCLEF(10),MPART(10)
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /L0/ NLOCS,JLOCN
      COMMON /LT/ MLOCN(10),MCTK(5),MFIND(10)
      COMMON /M0/ JIMODE,JOMODE
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
      COMMON /SU/ MHCLEF(10)
      LREQM = NNBAR * NLINES + NNLINE
      LBEST = NSEEK(LREQM)
      IF (LBEST .EQ. 0) GOTO 4
      NIBS = MFIND(LBEST)
C     SET INPUT ADDRESS TO LOCATION FOUND IN TABLE
      CALL INPOSN(MLOCN(LBEST))
      IF (ISCHEK .EQ. 0 .OR. JIMODE .NE. 1) GOTO 3
```

```
      JLINE = NIBS - NIBS / NLINES * NLINES
      IF (JLINE .EQ. 0) JLINE = NLINES
C     RESET STATE AND CLEFS TO INITIAL VALUES
      CALL XRESET
      CALL MBS(MHCLEF,K1,MCLEF,K1,NLINES)
      JCLEF = NBYTE(MCLEF,JLINE)
C     SET TIME AND KEY FOR START OF THIS MEASURE
      JBBUF = 0
      NILENG = 2
      DO 20 L=1,2
      CALL STBYTE(33+L,MIBAR,K1)
      CALL STBYTE(NBYTE(MCTK,2*LBEST-2+L),
     *                                  MIBAR,K2)
      JIBAR = 0
      CALL IDCODE
   20 CONTINUE
    3 NIBS = NIBS - 1
      GOTO 5
C
C     NO LOCATION FOUND, POSITION FILE TO START
    4 CALL GETHDR
      IF (NNBAR .LE. 0) GOTO 9
C
C     SEARCH THROUGH FILE
    5 IF (JIMODE .EQ. 1 .AND. ISCHEK .NE. 0)
     *                                 ISSCAN = 1
   55 IF (NIBS-LREQM+1) 7,56,88
C     NO SCAN THROUGH REQUESTED BAR
   56 ISSCAN = 0
    7 CALL GETBAR
      GOTO 55
   88 ISSCAN = 0
    9 RETURN
      END
C
C
      FUNCTION NSEEK(NREQM)
C     SEARCH LOCATION TABLE FOR NEAREST MEASURE
C     PRECEDING MEASURE *NREQM*.  RETURN *NSEEK* =
C     TABLE INDEX IF FOUND, OTHERWISE 0.
      COMMON /LO/ NLOCS,JLOCN
      COMMON /LT/ MLOCN(10),MCTK(5),MFIND(10)
      NSEEK = 0
      LNIBS = 0
      IF (NLOCS .LE. 0) GOTO 9
      DO 30 L=1,NLOCS
      LHW = MFIND(L)
C     IGNORE ENTRY IF AFTER REQUIRED MEASURE OR
C     A BETTER ONE HAS ALREADY BEEN FOUND
      IF (LHW.LE.LNIBS .OR. LHW.GT.NREQM) GOTO 3
C     SAVE TABLE POINTER AND MEASURE NUMBER
      NSEEK = L
      LNIBS = LHW
    3 CONTINUE
   30 CONTINUE
    9 RETURN
      END
C
C
      SUBROUTINE GETHDR
C     OBTAIN FILE HEADER RECORD IN *MIBAR* AND SET
C     INPUT ADDRESS TO START OF CURRENT SECTION.
      COMMON MC1(253),NOLENG,MOBAR(64),MC2(3),
     * NFLAGS,MFLAGS(3)
      COMMON /B1/ IFIRST,JOPAGE
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND
     *   ISERR,ISNXST,ISINPT
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /HE/ MCLEF(10),MPART(10)
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,K8IG
      COMMON /MO/ JIMODE,JOMODE
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NCP,NDL
      COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
```

```
      COMMON /SU/ MHCLEF(10)
      COMMON /TL/ NNAME,MNAME(8)
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      ISHDR = 1
      CALL INPOSN(JHDR)
      NIBS = NLINES - 1
      IF (JIMODE .NE. 1) GOTO 4
      CALL GETBAR
      IFIRST = MFLAGS(1)
      IF (ISCHEK .EQ. 0) GOTO 4
      NSCALE = JSCALE
      NHTEXT = NFETCH(KO)
C     GET NAME
      CALL IDCODE
      CALL BMOVE(MTEXT,MNAME)
C     GET OPTIONAL TEXT ITEMS, METRONOME MARK,
C     TIME AND KEY SIGNATURES
      LL = NHTEXT + 2
      DO 20 L=1,LL
   20 CALL IDCODE
C     RESET STATE AND CLEF VALUES
      CALL XRESET
      DO 30 JLINE=1,NLINES
   30 CALL IDCODE
      CALL MBS(MCLEF,K1,MHCLEF,K1,NLINES)
      JLINE = NIS
    4 CALL INPOSN(KDATA+MHEAD(JIMODE))
      INBACK = 0
      IF (JIMODE .NE. 2) GOTO 6
C     UNFORMATTED BRAILLE MODE - SKIP HEADER LINES
C     AND BRAILLE PART CODES
      CALL GETBAR
      LHL = MFLAGS(1) + 1
      DO 50 L=1,LHL
   50 CALL GETBAR
      SET MEASURE NUMBER FOR HEADER RECORD
    6 NIB = 0
      NIS = NLINES
      NIBS = NLINES
      ISHDR = 0
      RETURN
      END
C
C
      SUBROUTINE NXSEL
C     OBTAIN NEXT MEASURE ON A SELECTED STAVE.
      COMMON /FL/ MFL1(4),ISEND,MFL2(3)
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
    3 CALL GETBAR
      IF (ISEND .EQ. 0 .AND. NBYTE(MSSEL,NIS)
     *                       .EQ. 0) GOTO 3
      RETURN
      END
C
C
      SUBROUTINE GETBAR
C     OBTAIN MEASURE IN *MIBAR* FROM FILE AT
C     CURRENT INPUT ADDRESS, WHICH IS LEFT AT THE
C     END OF THE MEASURE UNLESS IT IS AN END-OF-
C     SECTION MARK.
      COMMON MINBUF(90),MOPBUF(90),NINBUF,NOFBUF,
     * JINSEC,JOPSEC,JINBUF,JOPBUF,JINLK,JOPLK,
     * NILENG,MIBAR(64),NOLENG,MOBAR(64),JIBAR,
     * JOBAR,JOBUF,NFLAGS,MFLAGS(3)
      COMMON /AD/ JIADDR,JOADDR
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /EE/ JEDIT,JEFREE,JEBACK,ISITEM
      COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSGZ,
     * ISREST,ISAHED,ISXPOS,ISEMB
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FI/ ISWRIT,ISSCAN
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /HE/ MCLEF(10),MPART(10)
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
```

```
      COMMON /KM/ KZEROS(2),KMRSIZ,MREST(1)
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KY/ JKEYN0,JKEYSG,MKPCH(2),MTKPCH(2)
      COMMON /L0/ NL0CS,JL0CN
      COMMON /LT/ ML0CN(10),MCTK(5),MFIND(10)
      COMMON /MC/ KBSIZE,KWSIZE,KXMAX,KYMAX,KFSIZE
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /ME/ MEMPTR(10)
      COMMON /M0/ JIM0DE,J0M0DE
      COMMON /NB/ JBAR,NIB,N0B,NIS,N0S,NIBS
      COMMON /ND/ NXLENG,NSHIFT,JBXL,NEXTX
      COMMON /NS/ NLINES,JLINE,JPART,JV0IC,NDP,NDL
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /SE/ JCMD,KSEGS(7)
      COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
      COMMON /SU/ MHCLEF(10)
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NN0TES
      COMMON /WS/ L1,L2,L3,JBYTE
      LOGICAL CBS
      ISEND = 0
    2 LBYTE = INBYTE(K0)
      IF (LBYTE .EQ. 255) GOTO 2
      JIADDR = KDATA * JINSEC + JINBUF - 1
      JEBACK = INBACK
      NIB = NIBS / NLINES
      NIBS = NIBS + 1
      IF (ISHDR .NE. 0) GOTO 25
C     SKIP MEMORY TEST IF IN BRAILLE MODE
      IF (JIM0DE .NE. 1) GOTO 23
C     N0 SAVE IF STATE VALUES N0T CORRECTLY SET
      IF (ISCHEK.EQ.C .0R. ISAHED.NE.0) GOTO 25
C     STORE CURRENT STATE VALUES FOR COMPARISON
      IF (.N0T. CBS(MCMEM,K1,KZER0S,K1,KMSIZE))
     *          CALL STBYTE(K1,MEMPTR,NIS)
C     TEST FOR ALL MEMORY ZERO
      DO 210 L1=1,NLINES
      L2 = NBYTE(MEMPTR,L1)
      IF (L2 .NE. 0) GOTO 22
  210 CONTINUE
C     RESTORE TRUE MEMORY STATUS
   22 CALL STBYTE(K0,MEMPTR,NIS)
C     N0 SAVE IF ANY MEMORY N0N-ZERO
      IF (L2 .NE. 0) GOTO 25
C     N0 SAVE IF DISPLAY-TYPE COMMAND AND CLEFS
C     ARE DIFFERENT FROM START OF FILE
      IF (JIM0DE.EQ.1 .AND. JCMD.LE.6 .AND. .N0T.
     *  CBS(MCLEF,K1,MHCLEF,K1,NLINES)) GOTO 25
C     N0 SAVE IF THIS LOCATION ALREADY SAVED
   23 L1 = NSEEK(NIBS)
      IF (L1.GT.0 .AND. MFIND(L1).EQ.NIBS) GOTO 25
C     SAVE LOCATION OF THIS MEASURE FOR SUBSEQUENT
C     RAPID RETRIEVAL
      IF (NL0CS .LT. 10) NL0CS = NL0CS + 1
      JL0CN = JL0CN + 1
      IF (JL0CN .GT. NL0CS) JL0CN = 1
      ML0CN(JL0CN) = JIADDR
      MFIND(JL0CN) = NIBS
      IF (JIM0DE .NE. 1) GOTO 25
C     SAVE TIME AND KEY AS AT START OF MEASURE
      CALL STBYTE(JTYME,MCTK,2*JL0CN-1)
      CALL STBYTE(JKEY,MCTK,2*JL0CN)
   25 NIS = NIBS - NIB * NLINES
      JPART = NBYTE(MPART,NIS)
      IF (NIB .GT. JBE) ISEND = 1
      IF (LBYTE .LT. 254) GOTO 26
C     REPLACE NULL MEASURE BY MEASURE REST
      CALL BMOVE(MREST,MIBAR)
      MFLAGS(1) = 0
      ISREST = 1
      GOTO 32
   26 NFLAGS = 1
      ISREST = 0
      IF (JIM0DE .EQ. 2) NFLAGS = 3
      NILENG = LBYTE - NFLAGS
      DO 280 L1=1,NFLAGS
  280 MFLAGS(L1) = INBYTE(K0) - 128
      IF (NILENG+MFLAGS(1) .LT. 0) GOTO 8
C     COPY DATA ITEMS FROM FILE TO *MIBAR*
      IF (NILENG .EQ. 0) GOTO 32
```

```
      DO 30 L1=1,NILENG
   30 CALL STBYTE(INBYTE(K0),MIBAR,L1)
   32 IF (ISCHEK .EQ. 0) GOTO 9
      JTIME = 0
      IF (ISAHED .NE. 0) GOTO 35
      CALL CHSTAV(NIS)
      JECLEF = JCLEF
C     SAVE MEMORY STATUS AS AT START OF MEASURE
      CALL MMOVE(MCMEM,MSMEM)
C     SET ACCIDENTAL VALUES AS FOR KEY SIGNATURE
      CALL MMOVE(MKPCH,MTKPCH)
   35 ISVOC = NSPLIT(NBYTE(MPART,NIS),K1)
      IF (ISSCAN.EQ.C .OR. MFLAGS(1).EQ.0) GOTO 85
      JIBAR = 0
      JBBUF = 0
C     DECODE ITEMS IN MEASURE TO SET STATE VALUES
    4 CALL IDCODE
      IF (ISEND .EQ. 0) GOTO 4
      GOTO 85
C     BACKSPACE OVER END-OF-SECTION RECORD
    8 CALL INPOSN(JIADDR)
C     SET FLAG TO INDICATE END-OF-SECTION REACHED
      ISEND = 1
   85 NNOTES = 0
      JVOIC = JLINE
    9 JIBAR = 0
      JREPT = 0
      RETURN
      END
C
C
      FUNCTION INBYTE(NZERO)
C     RETURN *INBYTE* = NEXT BYTE OF FILE AT
C     CURRENT INPUT ADDRESS AND INCREMENT ADDRESS.
C     IF *NZERO* = 0, A ZERO BYTE MEANS SKIP TO
C     START OF NEXT SECTOR.
      COMMON MINBUF(90),MOPBUF(90),NINBUF,NOPBUF,
     * JINSEC,JOPSEC,JINBUF,JOPBUF,JINLK,JOPLK
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *    ISERR,ISNXST,ISINPT
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      JINBUF = JINBUF + 1
      IF (JINBUF .LE. KDATA) GOTO 5
    2 JINBUF = 1
      IF (JINSEC .EQ. NOPBUF) GOTO 3
      JINSEC = JINLK
      GOTO 5
    3 JINSEC = JOPLK
C     IF REQUIRED SECTOR IS IN *MOPBUF* THEN
C     USE *MOPBUF* INSTEAD OF *MINBUF*
    5 IF (NOPBUF .NE. JINSEC) GOTO 6
      INBYTE = NBYTE(MOPBUF,JINBUF+2)
      GOTO 8
    6 IF (NINBUF .EQ. JINSEC) GOTO 7
      INBACK = NINBUF
      CALL DREAD(JINSEC,K0)
    7 INBYTE = NBYTE(MINBUF,JINBUF+2)
    8 IF (INBYTE .EQ. 0 .AND. NZERO .EQ. 0) GOTO 2
      RETURN
      END
C
C
      SUBROUTINE INPOSN(NPTR)
C     SET CURRENT INPUT ADDRESS TO *NPTR*.
      COMMON MC1(182),JINSEC,JOPSEC,JINBUF
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      JINSEC = NPTR / KDATA
      JINBUF = NPTR - KDATA * JINSEC
C     ENSURE *JINLK* CORRECTLY SET
      CALL DREAD(JINSEC,K0)
      RETURN
      END
```

```
      SUBROUTINE DREAD(NSECT,NBUFF)
C     READ SECTOR OF DISC FILE INTO BUFFER.
C     *NSECT* = SECTOR NUMBER,
C     *NBUFF* = 0 FOR *MINBUF*, 1 FOR *MOPBUF*.
      COMMON MIOBUF(180),NIOBUF(2),MC1(4),JIOLK(2)
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      CALL DISKIO(K0,NSECT,NBUFF)
      JIOLK(NBUFF+1)=NHW(MIOBUF,KBFSIZ/K2*NBUFF+1)
      NIOBUF(NBUFF+1) = NSECT
      RETURN
      END
C
C
C     ******************************************
C     *                                        *
C     *      DISC FILE INPUT/OUTPUT ROUTINE    *
C     *                                        *
C     ******************************************
C
C
      SUBROUTINE DISKIO(NIO,NSECT,NBUFF)
C     RANDOM-ACCESS DISC FILE READ/WRITE ROUTINE.
C     TRANFERS DATA BETWEEN INTERNAL BUFFER AND
C     EXTERNAL DISC FILE.
C     *NIO* = 0 FOR READ, 1 FOR WRITE,
C     *NSECT* = SECTOR NUMBER WITHIN FILE,
C     *NBUFF* = 0 FOR *MINBUF*, 1 FOR *MOPBUF*.
      COMMON MIOBUF(180)
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /C1/ JLABEL
      COMMON /FG/ KLEVEL,IEA
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      DIMENSION LIO(2)
      DATA LIO(1)/8Z10000000/,LIO(2)/8Z11000000/
      IF (NSECT .LT. 0) GOTO 35
S     LI,1     F:120         UNIT NUMBER
S     LW,2     *NIO          READ/WRITE INDICATOR
S     AW,1     LIO,2         TRANSFER CODE
S     LW,2     =8Z31000000   FLAGS: BUFFER,BYTE,KEY
S     AW,2     IEA           ADD END-ACTION FLAG
S     LW,3     *NBUFF
S     MW,3     KBFWDS
S     AI,3     MIOBUF        BUFFER ADDRESS
S     LW,4     KBFSIZ        BUFFER LENGTH
S     LW,5     *NSECT        KEY NUMBER
S     LW,6     =8Z40000000
S     AW,6     KLEVEL        END-ACTION INTERRUPT
S     CAL1,1   1             READ OR WRITE RECORD
S3    LI,1     F:120         UNIT NUMBER
S     AW,1     =8Z29000000   CHECK-FUNCTION CODE
S     LW,2     =8ZE0000000   ERROR,ABNORMAL,BUSY
S     LI,3     35S           ERROR ADDRESS
S     LI,4     35S           ABNORMAL ADDRESS
S     LI,5     4S            BUSY ADDRESS
S     CAL1,1   1             CHECK IO COMPLETION
      GOTO 9
C     I/O ERROR = SET FLAG AND SIMULATE INTERRUPT
   35 ISERR = 2
C+    FOLLOWING STATEMENT ISNT FORTRAN...
      GOTO JLABEL
C     WAIT FOR I/O COMPLETION
    4 CALL WAIT
      GOTO 3
    9 RETURN
      END
C
C
C     ******************************************
C     *                                        *
C     *          ITEM DECODING ROUTINES        *
C     *                                        *
C     ******************************************
C
C
      SUBROUTINE IDCODE
C     CONVERTS ITEM FROM CODED TO UNCODED FORM.
C     *JBBUF* = 0 TO OBTAIN ITEM FROM *MIBAR*,
```

```
C               1 TO OBTAIN ITEM FROM *MOBAR*,
C     RETURNS *ISEND* = 1 IF NO MORE ITEMS LEFT.
      COMMON MC1(318),JIBAR,JOBAR,JBBUF
      COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSQZ,
     * ISREST,ISAHED,ISXPOS,ISEMB
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /GP/ JGROUP,MGROUP(7)
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /HE/ MCLEF(10),MPART(10)
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KY/ JKEYNO,JKEYSG,MKPCH(2),MTKPCH(2)
      COMMON /MO/ JIMODE,JOMODE
      COMMON /ND/ NXLENG,NSHIFT,JBXL,NEXTX
      COMMON /NH/ JNHEDS,KNHMAX
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /SD/ MYAXID(3)
      COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /TT/ JTTEMP,JXTEMP
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      COMMON /TY/ ISCONT,ISBLOW
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION MITEM2(8),JBBAR(2)
      DIMENSION LMKP(2),LKNATS(2)
      EQUIVALENCE(MITEM2,JCLEF),(JBBAR,JIBAR)
      DATA LMKP(1)/8Z04010502/,LMKP(2)/8Z06030700/
     *,LKNATS(1)/8Z03030303/,LKNATS(2)/8Z03030303/
C
C     RESET END-OF-MEASURE INDICATOR
      ISEND = 0
C     SAVE POINTER TO START OF ITEM WITHIN MEASURE
      ITPTR = JBBAR(JBBUF+1)
C     OBTAIN FIRST BYTE OF CODED ITEM
      L1 = NFETCH(KO)
C     NO FURTHER ACTION IF NO ITEMS REMAINING
      IF (ISEND .NE. 0) GOTO 8
      ITEM = L1
C     IDENTIFY ITEM TYPE
      IF (JIMODE .EQ. 1 .OR. ISHDR .NE. 0) GOTO 2
C     BRAILLE ITEM
      ITTYPE = 5
      GOTO 23
    2 IF (ITEM .GE. 128) GOTO 22
C     CONTROL, SEPARATE SIGN OR TEXT ITEM
      ITTYPE = MINO(ITEM/32+1,K3)
      GOTO 23
C     NOTE ITEM
   22 ITTYPE = 4
C     BRANCH INDEXED BY ITEM TYPE CODE
   23 GOTO (4,5,6,7,8),ITTYPE
      GOTO 8
C
C     DECODE CONTROL ITEM
    4 IF (ISCHEK .EQ. 0) GOTO 8
      IF (ITEM-3) 42,43,8
C     IN-ACCORD SIGN
   42 JVOIC = JVOIC + 64
      JTIME = JTTEMP
      GOTO 8
C     SET-RETURN SIGN
   43 JVOIC = JLINE
      JTTEMP = JTIME
      GOTO 8
C
C     DECODE SEPARATE SIGN ITEM
    5 L2 = ITEM - 32
      IF (L2 .GT. 16) GOTO 52
C     DOUBLE-BYTE ITEM
      L1 = NFETCH(KO)
      IF (L2 .GT. 8) GOTO 8
      IF (ISCHEK .NE. 0 .OR. L2 .LE. 4)
```

```
     *                              MITEM2(L2) = L1
      IF (ISCHEK.EQ.C .AND. ISAHED.EQ.0) GOTO 8
      GOTO (53,54,55,56,58,58,58,547),L2
      GOTO 58
C     SINGLE-BYTE ITEM
   52 GOTO 58
C     CLEF
   53 CALL STBYTE(JCLEF,MCLEF,JLINE)
      GOTO 555
C     TIME SIGNATURE
   54 JTIME2 = JTYME / 32
      JTIME1 = JTYME - 32 * JTIME2 + 1
      IF (JTIME2 .LT. 6) GOTO 543
C     C (4/4), CBAR (2/2), OR NO TIME SIGNATURE
      JTIME2 = JTIME2 - 5
      JTIME1 = 2 * JTIME2
  543 JTIME2 = NPOWER(JTIME2)
      IF (JTYME .NE. 255) GOTO 545
      JTMIN = 0
      JTMAX = KBIG
      GOTO 547
  545 JTMIN = 512 * JTIME1 / JTIME2
      JTMAX = JTMIN
  547 JBXL = JTIME1 * NSCALE * 2
      JBDUR = JTMAX / JTIME1
      GOTO 555
C     KEY SIGNATURE
   55 MYAXID(2) = MYAXID(JKEYSG)
      JKEYNO = JKEY
      JKEYSG = NSPLIT(JKEYNO,K4)
C     RESET TO NATURALS
      CALL MMOVE(LKNATS,MKPCH)
      IF (JKEYNO .LE. 0) GOTO 554
C     INSERT SHARPS OR FLATS AS MARKED IN KEY SIG
      DO 5520 L1=1,JKEYNO
 5520 CALL STBYTE(JKEYSG+1,MKPCH,NBYTE(LMKP,
     *            4*(3-JKEYSG)+(JKEYSG-2)*L1))
C     COPY TRUE KEY SIG TO CURRENT KEY SIGNATURE
  554 CALL MMOVE(MKPCH,MTKPCH)
  555 ISCTK = 1
      GOTO 58
C     IRREGULAR NOTE GROUPING
   56 JGROUP = MGROUP(NGROUP)
   58 IF (ISCHEK .NE. 0) CALL SSCHK(K0)
      GOTO 8
C
C     DECODE TEXT ITEM
    6 JBYTE = ITEM - 64
      ISWORD = NBIT(K2)
      IF (ISWORD .EQ. 0) GOTO 63
      ISCONT = NBIT(K3)
      ISBLOW = 1
      GOTO 64
   63 ISCONT = 0
      ISBLOW = NBIT(K3)
   64 NCHARS = JBYTE
      IF (NCHARS .LE. 0) GOTO 8
      IF (NCHARS .EQ. 15) NCHARS = NFETCH(K0)
      DO 650 L1=1,NCHARS
  650 CALL STBYTE(NFETCH(K0),MTEXT,L1)
      GOTO 8
C
C     DECODE NOTE ITEM
    7 JBYTE = ITEM - 128
      ISSTEM = 0
      ISEXTR = NBIT(K1)
      ISCHOR = NBIT(K2)
      JLENTH = JBYTE / K3
      JDOTS = JBYTE - 3 * JLENTH
      DO 710 L1=1,4
  710 MEXTRA(L1) = 0
      L1 = ISEXTR
   72 IF (L1 .EQ. 0) GOTO 73
      JBYTE = NFETCH(K0)
      L1 = NBIT(K0)
      L2 = NBIT(K2)
      MEXTRA(L2+1) = JBYTE
      GOTO 72
   73 IF (ISCHOR .EQ. 0) GOTO 731
C     CHORD, REST OR STEM SIGN
```

```
      JBYTE = NFETCH(K0)
      ISTIE = NBIT(KC)
      ISSTEM = NBIT(K1)
      NHEDS = MIN0(JBYTE,KNHMAX)
      ISCTIE = 1
      G0T0 732
C     SINGLE-HEAD NOTE ITEM
  731 NHEDS = 1
      ISCTIE = 0
  732 D0 770 L1=1,NHEDS
      LPTR = (L1 - 1) * KHS + 1
      IF (ISCH0R .EQ. NHEDS) G0T0 735
      JBYTE = NFETCH(K0)
      G0T0 74
C     REST
  735 JBYTE = 0
      ISCH0R = 0
   74 IF (ISSTEM.EQ.0) CALL STBYTE(JBYTE,MHD,LPTR)
C     TIE AND ACCIDENTAL SPECIFICATION
      L3 = NBIT(K0)
      IF (L3 .GT. 0) L3 = NFETCH(K0)
      IF (L3 .LT. 128) ISCTIE = 0
      CALL STBYTE(L3,MHD,LPTR+1)
      L3 = NBIT(K1)
      IF (L3 .GT. 0) L3 = NFETCH(K0)
      L2 = L3/64
      IF (L2=1) 746,742,744
  742 L2 = NFETCH(K0)
      G0T0 746
  744 L2 = L3
      L3 = 0
  746 CALL STBYTE(L3,MHD,LPTR+2)
      CALL STBYTE(L2,MHD,LPTR+3)
  770 CONTINUE
      CALL HDC0DE(K1)
      IF (ISCHEK .EQ. 0) G0T0 8
      CALL NLENTH
      NN0TES = NN0TES + 1
      JTIME = JTIME + JNDUR
C
    8 ITLENG = JBBAR(JBBUF+1) = ITPTR
      RETURN
      END
C
C
      SUBR0UTINE HDC0DE(N)
C     SETS N0TE HEAD ATTRIBUTES F0R *N*TH HEAD 0F
C     CURRENT N0TE ITEM.
      C0MM0N /FF/ ISCTK,ICFLAG,ISV0C,ISSGZ,
     * ISREST,ISAHED,ISXP0S,ISEMB
      C0MM0N /IU/ JCLEF,JTYME,JKEY,NGR0UP,NPAGE,
     * NLINE,NBEAM,NSCALE
      C0MM0N /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      C0MM0N /N0/ KHS,J0CTAV,JPITCH,JLENTH,ISEXTR,
     * JD0TS,NHEDS,ISCH0R,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      C0MM0N /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
      C0MM0N /WS/ L1,L2,L3,JBYTE
      EQUIVALENCE(N0RNAM,MHEXT(2)),(NAX1,MHEXT(3))
      EQUIVALENCE(NAX2,MHEXT(4))
      L2 = (N-1) * KHS + 1
      JBYTE = NBYTE(MHD,L2)
C     DISCARD PRESENCE BITS
      L1 = NBIT(K1)
      IF (ISXP0S .EQ. 0 .0R. JBYTE .EQ. 0) G0T0 3
C     TRANSP0SE T0 TREBLE CLEF IF INPUTTING
      L1 = NBYTE(MCMEM,K4)
      IF (L1 .GT. 0) L1 = (L1/128) * 14 - 7
      JBYTE = JBYTE + JCLEF - 122 + L1
    3 J0CTAV = (JBYTE-1) / K7
      JPITCH = JBYTE - 7 * J0CTAV
      JAXIDS = NBYTE(MHD,L2+1)
      IF (JPITCH .NE. 0) ISTIE = NSPLIT(JAXIDS,K0)
C     FINGERING
      MHEXT(1) = NBYTE(MHD,L2+2)
C     0RNAMENTS
      JBYTE = NBYTE(MHD,L2+3)
      IF (JBYTE .NE. 0) JBYTE = JBYTE + 1
      L1 = NBIT(K0)
      NAX2 = NBIT(K1)
```

```
      NAX1 = NBIT(K3)
      IF (NAX2 .NE. 0) NAX2 = NBIT(K5)
      NORNAM = JBYTE
      ISSMAL = 0
C     SET SMALL-NOTE INDICATOR FOR GRACE NOTES
      IF (NORNAM .EQ. 10 .OR. NORNAM .EQ. 11)
     *                          ISSMAL = NORNAM - 9
    9 RETURN
      END
C
C
      FUNCTION NFETCH(N)
C     RETURN *NFETCH*=NEXT BYTE OF CODED MEASURE.
C     *JBBUF* = 0 FOR *MIBAR*, 1 FOR *MOBAR*,
C     DUMMY ARGUMENT *N* IS IGNORED.
      COMMON MC1(188),NILENG,MIBAR(64),NOLENG,
     * MOBAR(64),JIBAR,JOBAR,JBBUF
      COMMON /FL/ MFL1(4),ISEND,MFL2(3)
      IF (JBBUF .NE. 0) GOTO 4
      IF (JIBAR .GE. NILENG) GOTO 7
      JIBAR = JIBAR + 1
      NFETCH = NBYTE(MIBAR,JIBAR)
      GOTO 9
    4 IF (JOBAR .GE. NOLENG) GOTO 7
      JOBAR = JOBAR + 1
      NFETCH = NBYTE(MOBAR,JOBAR)
      GOTO 9
    7 ISEND = 1
      NFETCH = 0
    9 RETURN
      END
C
C
C     ***********************************************
C     *                                             *
C     *          LINKED LIST ROUTINES               *
C     *                                             *
C     ***********************************************
C
C
      SUBROUTINE ECLEAR(N)
C     CLEAR LIST *N* OF EDIT TABLE.
      COMMON /EE/ JEDIT,JEFREE,JEBACK,ISITEM
      LJFREE = NFLINK(N)
C     EXIT IF LIST ALREADY EMPTY
      IF (LJFREE .EQ. N) GOTO 9
C     LINK LAST ENTRY OF LIST TO HEAD OF FREE LIST
      CALL SLINK(NBLINK(N),JEFREE)
C     SET HEAD OF FREE LIST TO FIRST ENTRY OF LIST
      JEFREE = LJFREE
C     RESET LIST *N*
      CALL SLINK(N,N)
    9 RETURN
      END
C
C
      SUBROUTINE EDEL(NPTR)
C     DELETES ENTRY *NPTR* FROM LINKED LIST.
      COMMON /EE/ JEDIT,JEFREE,JEBACK,ISITEM
C     LINK PRECEDING ENTRY TO FOLLOWING ENTRY
      CALL SLINK(NBLINK(NPTR),NFLINK(NPTR))
C     LINK DELETED ENTRY TO HEAD OF FREE LIST
      CALL SLINK(NPTR,JEFREE)
C     SET HEAD OF FREE LIST EQUAL TO DELETED ENTRY
      JEFREE = NPTR
      RETURN
      END
C
C
      SUBROUTINE EINSRT(NPTR,NBYTE1,NBYTE2,NBYTE3,
     * NBYTE4)
C     INSERT NEW ENTRY PRECEDING ENTRY *NPTR* AND
C     INSERT *NBYTE1* TO *NBYTE4* IN INFO FIELDS.
      COMMON /EE/ JEDIT,JEFREE,JEBACK,ISITEM
      COMMON /LK/ MBLINK(14),MFLINK(14),
     * MLIST1(14),MLIST2(14),MLIST3(14),MLIST4(14)
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION LMESS(4)
      DATA LMESS/15HEDIT TABLE FULL/
C     OBTAIN EMPTY CELL FROM FREE CHAIN
```

```
      L3 = NFLINK(JEFREE)
C     ERROR IF NO CELLS LEFT ON FREE CHAIN
      IF (L3 .EQ. 0) GOTO 4
C     ENTER INFORMATION
      CALL STBYTE(NBYTE1,MLIST1,JEFREE)
      CALL STBYTE(NBYTE2,MLIST2,JEFREE)
      CALL STBYTE(NBYTE3,MLIST3,JEFREE)
      CALL STBYTE(NBYTE4,MLIST4,JEFREE)
C     LINK ENTRY BEFORE CHOSEN ENTRY TO NEW ENTRY
      CALL SLINK(NBLINK(NPTR),JEFREE)
C     LINK NEW ENTRY TO SELECTED ENTRY
      CALL SLINK(JEFREE,NPTR)
C     RESET HEAD OF FREE CHAIN
      JEFREE = L3
      GOTO 9
C     DISPLAY EDIT TABLE FULL MESSAGE
    4 CALL TMESS(15,LMESS)
    9 RETURN
      END
C
C
      SUBROUTINE UPDATE
C     ON ENTRY EDIT TABLE CONTAINS POINTERS TO
C     ITEMS IN *MIBAR*.  RETURNS CURRENT MEASURE
C     CORRECTLY ORDERED IN *MIBAR* AND *MOBAR*.
      COMMON MC1(189),MIBAR(64),NOLENG,MOBAR(64)
      COMMON /EE/ JEDIT,JEFREE,JEBACK,ISITEM
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /LK/ MBLINK(14),MFLINK(14),
     * MLIST1(14),MLIST2(14),MLIST3(14),MLIST4(14)
      JEDIT = 1
      NOLENG = 0
      IF (ISHDR .NE. 0) CALL BSTORE(NHTEXT)
    2 JEDIT = NFLINK(JEDIT)
      IF (JEDIT .EQ. 1) GOTO 4
      ITPTR = NBYTE(MLIST1,JEDIT)
      ITLENG = NBYTE(MLIST2,JEDIT)
      CALL MBS(MIBAR,ITPTR+1,MOBAR,NOLENG+1,
     *                                ITLENG)
      NOLENG = NOLENG + ITLENG
      GOTO 2
    4 CALL BMOVE(MOBAR,MIBAR)
      RETURN
      END
C
C     FUNCTION NBLINK(N)
C     IF *N* = 0, RETURNS *NBLINK* = 0, OTHERWISE
C     RETURNS BACKWARD LINK OF TABLE ENTRY *N*.
      COMMON /LK/ MBLINK(14),MFLINK(14),MLK1(56)
      NBLINK = NBYTE(MBLINK,N+1)
      RETURN
      END
C
C     FUNCTION NFLINK(N)
C     IF *N* = 0, RETURNS *NFLINK* = 0, OTHERWISE
C     RETURNS FORWARD LINK FROM TABEL ENTRY *N*.
      COMMON /LK/ MBLINK(14),MFLINK(14),MLK1(56)
      NFLINK = NBYTE(MFLINK,N+1)
      RETURN
      END
C
C     SUBROUTINE SLINK(N1,N2)
C     LINK EDIT TABLE ENTRY *N1* TO ENTRY *N2*.
      COMMON /LK/ MBLINK(14),MFLINK(14),MLK1(56)
      CALL STBYTE(N2,MFLINK,N1+1)
      CALL STBYTE(N1,MBLINK,N2+1)
      RETURN
      END
C
C
```

```
C     ******************************************
C     *                                        *
C     *   TERMINAL INPUT AND OUTPUT ROUTINES   *
C     *                                        *
C     ******************************************
C
C
      SUBROUTINE REJECT
C     SOUND BELL ON TERMINAL.
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      DIMENSION LBELL(1)
      DATA LBELL(1)/8Z07000000/
      CALL TWT(K1,LBELL)
      RETURN
      END
C
C
      SUBROUTINE TNLINE
C     START NEW LINE ON TERMINAL.
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      CALL TMESS(K0,KSPACS)
      RETURN
      END
C
C
      SUBROUTINE TMESS(NLENG,NTEXT)
C     DISPLAY MESSAGE ON NEW LINE ON SCREEN.
C     *NLENG* = MESSAGE LENGTH, *NTEXT* = TEXT.
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /GL/ IXG,IYG,IXGMIN,IGMAX,JXMAX,JYMAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      DIMENSION NTEXT(1)
      IYG = IYG - 3 * IYINC
      IF (IYG .GT. 3 * IYINC) GOTO 5
C     START NEW HALF-PAGE
      IXG = IXG + JXMAX/K2
      IF (IXG .LT. JXMAX) GOTO 4
C     CLEAR SCREEN AND START NEW PAGE
    3 CALL TPAGE(K1)
    4 IYG = IGMAX
C     DISPLAY MESSAGE
    5 CALL TWRITE(IXG,IYG,NLENG,NTEXT)
      IXGMIN = 20 * IXINC
      RETURN
      END
C
C
      SUBROUTINE TWRITE(NX,NY,NLENG,NTEXT)
C     DISPLAY MESSAGE AT POSITION (*NX*,*NY*).
      COMMON /DP/ IXSHIF,IYSHIF
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      DIMENSION NTEXT(1)
      IXT = NX - IXSHIF
      IYT = NY - IYSHIF
      CALL TPOSN
      CALL TWT(NLENG,NTEXT)
      RETURN
      END
C
C
      SUBROUTINE TWT(NLENG,NTEXT)
C     DISPLAY MESSAGE AT CURRENT SCREEN POSITION.
C     *NLENG* = MESSAGE LENGTH, *NTEXT* = TEXT.
      COMMON /PL/ ISPLOT
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      DIMENSION NTEXT(1)
      IF (NLENG .LE. 0) GOTO 9
      IF (ISPLOT .NE. 0) GOTO 4
C     DISPLAY MESSAGE ON TERMINAL
      CALL UWT(NLENG,NTEXT)
      GOTO 6
C     DISPLAY MESSAGE ON DIGITAL PLOTTER
    4 CALL PWT(NLENG,NTEXT)
    6 IXT = IXT + NLENG * IXINC
    9 RETURN
      END
C
C
```

```
      SUBROUTINE IWT1(N)
C     DISPLAY INTEGER *N* AT CURRENT POSITION.
      DIMENSION LTEXT(2)
      CALL ICCONV(N,LLENG,LTEXT)
      CALL IWT(LLENG,LTEXT)
      RETURN
      END
C
C
      SUBROUTINE TPOSN
C     SET SCREEN POSITION TO (*IXT*,*IYT*).
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      CALL TPL(K0)
      RETURN
      END
C
C
      SUBROUTINE TLINE
C     MOVE SCREEN POSITION FROM CURRENT POSITION
C     TO (*IXT*,*IYT*), DRAWING A STRAIGHT LINE.
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      CALL TPL(K1)
      RETURN
      END
C
C
      SUBROUTINE TPL(NTYPE)
C     POSITIONING OR LINE ON TERMINAL OR PLOTTER.
C     *NTYPE* = 0 POSITION CURSOR, 1 DRAW LINE,
C     *ISPLOT* = 0 USE TERMINAL, 1 USE PLOTTER.
      COMMON /PL/ ISPLOT
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      IF (ISPLOT .NE. 0) GOTO 4
      CALL TPLOT(NTYPE,IXT,IYT)
      GOTO 9
    4 CALL PPLOT(NTYPE,IXT,IYT)
    9 RETURN
      END
C
C
      SUBROUTINE TPAGE(NWAIT)
C     START NEW PAGE ON DISPLAY.  *NWAIT* = 1 TO
C     WAIT ON C/R INPUT, 0 OTHERWISE.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /PL/ ISPLOT
      COMMON /PP/ ISPOSE,JTERM,ISDBUG,MPP1(4)
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /TD/ IXC,IYC,IXB,IXR
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      IF (ISDBUG .EQ. 0 .AND. NWAIT .EQ. 0) GOTO 2
      IYQ = 0
      CALL REQCR
    2 CALL ERASE
      IXC = JXMAX
      IYC = 200
      IXT = 0
      IYT = JYMAX - 50
      IXQ = 0
      IYQ = IYT
      IXQMIN = 0
      IQMAX = IYT
      IF (ISPLOT .NE. 0) GOTO 4
      CALL TPOSN
      GOTO 9
    4 CALL PPAGE
    9 RETURN
      END
C
C
      SUBROUTINE REQCR
C     REQUEST USER TO TYPE A CARRIAGE-RETURN.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /SE/ JCMD,KSEGS(7)
      DIMENSION LMESS(6)
      DATA LMESS/21HPRESS CARRIAGE-RETURN/
      DATA LEND/1H!/
```

```
C     DISPLAY MESSAGE UNLESS EXECUTING DISPLAY CMD
      IF (JCMD.NE.6 .OR. INIT.NE.0) CALL TWRITE
     *  (JXMAX/K2,IYG,21,LMESS)
    3 CALL REJECT
      CALL NOECHO
      IF (MCHAR(1) .EQ. C) GOTO 9
      IF (MCHAR(1) .NE. LEND) GOTO 3
      ISEND = 1
    9 RETURN
      END
C
C
      SUBROUTINE TRD
C     READ A SINGLE CHARACTER FROM THE TERMINAL
C     AND ECHO IT ON THE SCREEN.
      CALL NOECHO
      CALL ECHO
      RETURN
      END
C
C
      SUBROUTINE NOECHO
C     READ A SINGLE CHARACTER FROM THE TERMINAL IN
C     NO-ECHO MODE.  IF CARRIAGE-RETURN IS TYPED,
C     RETURNS *MCHAR(1)* = 0, OTHERWISE RETURNS
C     *MCHAR(1)* = CHARACTER LEFT-JUSTIFIED WITH
C     TRAILING BLANKS.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /PL/ ISPLOT
      DATA LCR/8Z0D4C4040/
C     SIMULATE CARRIAGE-RETURN IF GRAPH-PLOTTING
      IF (ISPLOT .NE. 0) GOTO 4
      LLENG = 1
      MCHAR(1) = KSPACS(1)
      CALL URD(LLENG,MCHAR)
      GOTO 6
    4 LLENG = 0
    6 IF (LLENG .LE. 0 .OR. MCHAR(1) .EQ. LCR)
     *                                  MCHAR(1) = 0
      RETURN
      END
C
C
      SUBROUTINE ECHO
C     DISPLAY SINGLE CHARACTER *MCHAR(1)*.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      CALL TWT(K1,MCHAR)
      RETURN
      END
C
C     ********************************************
C     *                                          *
C     *              UTILITIES                   *
C     *                                          *
C     ********************************************
C
      SUBROUTINE BMOVE(NBUFF1,NBUFF2)
C     MOVE BYTE STRING FROM *NBUFF1* TO *NBUFF2*.
C     THE BYTE COUNT IS TAKEN FROM THE LAST BYTE
C     OF THE WORD PRECEDING THE START OF *NBUFF1*
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      DIMENSION NBUFF1(1),NBUFF2(1)
      CALL MBS(NBUFF1,KO,NBUFF2,KO,
     *                    NBYTE(NBUFF1,KO)+1)
      RETURN
      END
C
C
      SUBROUTINE MMOVE(NBUFF1,NBUFF2)
C     COPY STATE VECTOR *NBUFF1* TO *NBUFF2*.
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
      DIMENSION NBUFF1(1),NBUFF2(1)
```

```
      CALL MBS(NBUFF1,K1,NBUFF2,K1,KMSIZE)
      END
C
C
      SUBROUTINE BREAK(N)
C+    CAUSE SUBSEQUENT TERMINAL INTERRUPTS TO
C+    TRANFER CONTROL TO LABEL *JLABEL*.
      COMMON /CT/ JLABEL
      JLABEL = N
      CALL BRKLABEL(N)
      END
C
C
      SUBROUTINE STHW(N,NARRAY,NPTR)
C     STORE 16-BIT VALUE *N* IN *NPTR*TH
C     HALFWORD (DOUBLE-BYTE) OF *NARRAY*.
C     DIMENSION NARRAY(1)
S        LW,7       *NPTR
S        AI,7       -1
S        LW,9       *N
S        STH,9      *NARRAY,7
      RETURN
      END
C
C
      SUBROUTINE STBYTE(NNBYTE,NARRAY,NPTR)
C     STORE *NNBYTE* IN *NPTR*TH BYTE OF *NARRAY*.
C     DIMENSION NARRAY(1)
S        LW,7       *NPTR
S        AI,7       -1
S        LW,9       *NNBYTE
S        STB,9      *NARRAY,7
      RETURN
      END
C
C
      FUNCTION NHW(NARRAY,NPTR)
C     RETURNS *NHW* = *NPTR*TH HALFWORD (DOUBLE-
C     BYTE) OF *NARRAY*.
C     DIMENSION NARRAY(1)
S        LW,7       *NPTR
S        AI,7       -1
S        LH,9       *NARRAY,7
S        AND,9      =X'FFFF'     MASK TO 16 BITS
S        STW,9      NHW
      RETURN
      END
C
C
      FUNCTION NBYTE(NARRAY,NPTR)
C     RETURNS *NBYTE* = *NPTR*TH BYTE OF *NARRAY*
C     DIMENSION NARRAY(1)
S        LW,7       *NPTR
S        AI,7       -1
S        LB,9       *NARRAY,7
S        STW,9      NBYTE
      RETURN
      END
C
C
      SUBROUTINE STBIT(NNBIT,NNBYTE,NPTR)
C     STORE SINGLE BIT *NNBIT* IN *NPTR*TH BIT OF
C     *NNBYTE*.  LAST BIT OF WORD IS BIT 8.
S        LI,3       1
S        LW,2       *NNBIT
S        LCW,1      *NPTR
S        AI,1       264
S        S,2        *1
S        STS,2      *NNBYTE
      RETURN
      END
C
C
      SUBROUTINE ICCONV(N,NLENG,NTEXT)
C     CONVERT INTEGER *N* TO CHARACTER FORM.
C     RETURNS *NTEXT* = CHARACTER STRING,
C     LEFT-JUSTIFIED WITH ONE LEADING BLANK,
C     *NLENG* = NUMBER OF CHARACTERS INCLUDING
C     THE BLANK.  *N* OUTSIDE RANGE 0 TO *KBIG*
C     IS TREATED AS EQUAL TO *KBIG*.
```

```
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
C     DIMENSION NTEXT(2)
S     LI,1      7           OUTPUT POINTER
S     LW,9      *N          NUMBER TO CONVERT
S     BLZ       2S          OUT OF RANGE IF -VE
S     CW,9      KBIG        MAXIMUM ALLOWED VALUE
S     BLE       3S          B IF NUMBER IN RANGE
S2    LW,9      KBIG        REPLACE *N* BY *KBIG*
S3    LI,8      0
S     DW,8      =10         REM TO R8, QUOT TO R9
S     AI,8      X'F0'       CONVERT TO CHARACTER
S     STB,8     *NTEXT,1    STORE DIGIT IN BUFFER
S     CI,9      0
S     BE        6S          B IF FINISHED
S     BDR,1     3S
S6    LI,2      X'40'       BLANK CHARACTER
S     STB,2     *NTEXT      STORE LEADING BLANK
S     LI,2      1           OUTPUT PTR AFTER SHIFT
S8    LB,8      *NTEXT,1    PICK UP NEXT DIGIT
S     STB,8     *NTEXT,2    SHIFT LEFT
S     AI,1      1
S     AI,2      1
S     CI,1      7           LAST DIGIT?
S     BLE       8S          B IF NOT
S     STW,2     *NLENG      NUMBER OF DIGITS
      RETURN
      END
C
C
      SUBROUTINE BSTORE(NNBYTE)
C     ADD *NNBYTE* TO CURRENT OUTPUT BAR.
      COMMON MC1(253),NOLENG,MOBAR(64)
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /MC/ KBSIZE,KWSIZE,KXMAX,KYMAX,KFSIZE
      IF (NNBYTE .EQ. 0) GOTO 9
      IF (NOLENG .LT. KBSIZE) GOTO 2
      ISERR = 1
      GOTO 9
    2 NOLENG = NOLENG + 1
      CALL STBYTE(NNBYTE,MOBAR,NOLENG)
    9 RETURN
      END


      FUNCTION NSPLIT(NNBYTE,NPTR)
      SPLITS *NNBYTE* BETWEEN BITS *NPTR* AND
      *NPTR* + 1.  RETURNS *NSPLIT* = QUOTIENT
      AND *NNBYTE* = REMAINDER OF DIVISION.
      LQUOT = NPOWER(7-NPTR)
      NSPLIT = NNBYTE / LQUOT
      NNBYTE = NNBYTE - NSPLIT * LQUOT
      RETURN
      END


      FUNCTION NBIT(NPTR)
      SPLITS *JBYTE* BETWEEN BITS *NPTR* AND
      *NPTR* + 1.  RETURNS *NBIT* = QUOTIENT
      AND *JBYTE* = REMAINDER OF DIVISION.
      COMMON /WS/ L1,L2,L3,JBYTE
      LQUOT = NPOWER(7-NPTR)
      NBIT = JBYTE / LQUOT
      JBYTE = JBYTE - NBIT * LQUOT
      RETURN
      END


      FUNCTION NPOWER(N)
      RETURNS *NPOWER* = 2 TO THE *N*TH POWER
         LW,1      *N
         LI,2      1
         SLS,2     0,1          SHIFT LEFT *N* BITS
         STW,2     NPOWER
      RETURN
      END
```

```
      FUNCTION ISCBS(NLENG,NTEXT)
C     COMPARES *NTEXT* WITH *MTEXT* IN FIRST
C     *NLENG* CHARACTERS.  RETURNS *ISCBS* = 1
C     IF THE TEXT STRINGS ARE THE SAME, 0 IF
C     DIFFERENT.  UPPER AND LOWER CASE LETTERS
C     ARE TREATED AS THE SAME.
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      DIMENSION NTEXT(1)
      DO 30 L=1,NLENG
   3C IF (ISSAME(NBYTE(MTEXT,L),NBYTE(NTEXT,L))
     *                              .EQ. 0) GOTO 5
      ISCBS = 1
      GOTO 9
    5 ISCBS = 0
    9 RETURN
      END
C
C
      FUNCTION ISSAME(NCHAR1,NCHAR2)
C     COMPARE BYTES *NCHAR1* AND *NCHAR2*.
C     RETURNS *ISSAME* = 1 IF SAME, TREATING
C     UPPER AND LOWER CASE EBCDIC LETTERS AS SAME,
C     0 IF DIFFERENT.
      IF (NCHAR1 .EQ. NCHAR2 .OR. (NCHAR1 .GT. 128
     * .AND. NCHAR2 .GT. 128 .AND. IABS(NCHAR1-
     * NCHAR2) .EQ. 64)) GOTO 3
      ISSAME = 0
      GOTO 9
    3 ISSAME = 1
    9 RETURN
      END
C
C
C     ***************************************** C
C     *                                       * C
C     *    ITEM INPUT AND DECODING ROUTINES   * C
C     *                                       * C
C     ***************************************** C
C
C
      SUBROUTINE CHSTAV(N)
C     SET CORRECT CONTEXT FOR STAVE *N*
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /HE/ MCLEF(10),MPART(10)
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /KM/ KZEROS(2),KMRSIZ,MREST(1)
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /LK/ MBLINK(14),MFLINK(14),
     * MLIST1(14),MLIST2(14),MLIST3(14),MLIST4(14)
      COMMON /ME/ MEMPTR(10)
      COMMON /ND/ NXLENG,NSHIFT,JBXL,NEXTX
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
      COMMON /WS/ L1,L2,L3,JBYTE
      EQUIVALENCE(MLIST,MLIST1)
      DIMENSION LBYTE(4)
      LOGICAL CBS
      IF (ISHDR .NE. 0) GOTO 9
      IF (JLINE .EQ. 0) GOTO 5
      CALL STBYTE(JCLEF,MCLEF,JLINE)
      CALL STBYTE(NSHIFT,MCMEM,K5)
C     SAVE MEMORY VALUES FOR LINE *JLINE*
      IF (CBS(MCMEM,K1,KZEROS,K1,KMSIZE)) GOTO 5
C     SAVE MEMORY TABLE AT END OF LIST 3
      DO 40 L1=1,2
      DO 30 L2=1,4
   30 LBYTE(L2) = NBYTE(MCMEM,4*(L1-1)+L2)
      CALL EINSRT(K3,LBYTE(1),LBYTE(2),
     *                        LBYTE(3),LBYTE(4))
   40 CONTINUE
      CALL STBYTE(NBLINK(NBLINK(K3)),MEMPTR,JLINE)
C     SET CORRECT CONTEXT FOR STAVE *N*
    5 JLINE = N
      JCLEF = NBYTE(MCLEF,JLINE)
C     SET STATE VALUES FOR LINE *JLINE* IN *MCMEM*
      LBYTE(1) = NBYTE(MEMPTR,JLINE)
      IF (LBYTE(1) .EQ. 0) GOTO 8
      LBYTE(2) = NFLINK(LBYTE(1))
```

```
C       EXTRACT MEMORY VALUES FROM EDIT TABLE
        DO 70 L1=1,2
        DO 60 L2=1,4
   60   CALL STBYTE(NBYTE(MLIST,56*(L2-1)+LBYTE(L1))
       * ,MCMEM,4*(L1-1)+L2)
        CALL EDEL(LBYTE(L1))
   70   CONTINUE
        CALL STBYTE(K0,MEMPTR,JLINE)
        GOTO 85
C       SET STATE VALUES TO ZERO IF NO TABLE ENTRY
    8   CALL MMOVE(KZEROS,MCMEM)
   85   NSHIFT = NBYTE(MCMEM,K5)
C       SET NEGATIVE *NSHIFT* CORRECTLY
        IF (NSHIFT .GE. 128) NSHIFT = NSHIFT - 256
    9   RETURN
        END
C
C
        SUBROUTINE XRESET
C       RESET MEMORY VALUES TO ZERO FOR ALL STAVES.
        COMMON /KM/ KZEROS(2),KMRSIZ,MREST(1)
        COMMON /KB/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
        COMMON /ME/ MEMPTR(10)
        COMMON /ND/ NXLENG,NSHIFT,JBXL,NEXTX
        COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
        COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
        COMMON /WS/ L1,L2,L3,JBYTE
        NSHIFT = 0
        CALL MMOVE(KZEROS,MCMEM)
        DO 30 L1=1,NLINES
   30   CALL STBYTE(K0,MEMPTR,L1)
        CALL ECLEAR(K3)
        RETURN
        END
C
C
        SUBROUTINE SSCHK(N)
C       CHECK VALIDITY OF SEPARATE SIGN ITEM.
C       *N* = 0 IF DECODING, *N* = 1 IF INPUTTING.
        COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
        COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
       *    ISERR,ISNXST,ISINPT
        COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
        COMMON /KB/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
        COMMON /MM/ IXST,JMEM
        COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
        COMMON /ND/ NXLENG,NSHIFT,JBXL,NEXTX
        COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
        COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
        COMMON /WS/ L1,L2,L3,JBYTE
        DIMENSION LMESS(2)
        DATA LMESS/6H*ERROR/
        L3 = ITEM - 48
        IF (L3 .LE. 0) GOTO 85
        GOTO (18,2,24,26,24,28,3,32),L3
        GOTO 8
C
C       START-OF-SLUR SIGN
   18   CALL GETXST(K2)
        IF (IXST .EQ. 0) GOTO 6
        CALL GETXST(K1)
        GOTO 5
C       END-OF-SLUR SIGN
    2   DO 2020 L1=1,2
        IXST = NBYTE(MCMEM,L1)
        IF (IXST .NE. 0) GOTO 21
 2020   CONTINUE
        GOTO 7
   21   JMEM = L1
        GOTO 75
C       OCTAVA ABOVE OR BELOW
   24   CALL GETXST(K4)
        IF (IXST .NE. 0) GOTO 7
        GOTO 6
C       END-OF-OCTAVA SIGN
   26   JMEM = 4
        GOTO 4
C       START OF CRESCENDO OR DECRESCENDO SIGN
   28   CONTINUE
    3   CALL GETXST(K3)
```

```
      GOTO 5
C     END-OF-CRESCENDO-OR-DECRESCENDO SIGN
   32 JMEM = 3
    4 IXST = NBYTE(MCMEM,JMEM)
      IF (IXST .EQ. 0) GOTO 7
      GOTO 75
    5 IF (IXST .NE. 0) GOTO 7
    6 L1 = 8
      IF (ISEDIT .NE. 0 .AND. N .NE. 0) L1 = IXP
      IF (L3 .EQ. 3 .OR. L3 .EQ. 7) L1 = L1 + 1024
      CALL STBYTE(L1,K8,MCMEM,JMEM)
      GOTO 8
    7 ISERR = 1
      ITERR = 1
      CALL REJECT
      CALL DBUG(K0,LMESS,ITEM,NIB,NIS,K0)
      GOTO 9
C     CANCEL STATE ENTRY FOR END OF EXTENDED SIGN
   75 CALL STBYTE(K0,MCMEM,JMEM)
C     OCTAVA ADJUSTMENT FOR DISPLAY OF NOTES
    8 IF (L3 .GE. 3 .AND. L3 .LE. 5)
     *                    NSHIFT = 7 * (4-L3)
   85 ISERR = 0
    9 RETURN
      END
C
C
      SUBROUTINE GETXST(NPTR)
C     OBTAIN STARTING CO-ORDINATE FOR EXTENDED
C     SIGN.  *NPTR* IS STATE TABLE INDEX.
      COMMON /MM/ IXST,JMEM
      COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      JMEM = NPTR
      IXST = NBYTE(MCMEM,JMEM)
      RETURN
      END
C
C
      SUBROUTINE NLENTH
C     RETURN *JNDUR* = TIME DURATION OF NOTE ITEM
      COMMON /GP/ JGROUP,MGROUP(7)
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /SE/ JCMD,KSEGS(7)
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      JNDUR = NPOWER(11-JLENTH-JDOTS) *
     *                    (NPOWER(JDOTS+1) - 1)
C     REDUCE SEMIBREVE REST TO MEASURE REST
C     IF MEASURE IS SHORTER THAN SEMIBREVE
      IF (JPITCH .EQ. 0 .AND. JLENTH .EQ. 2 .AND.
     *    JNDUR .GT. JTMAX) JNDUR = JTMAX
C     ADJUST FOR IRREGULAR GROUPING UNLESS PLAY
      IF (JCMD .NE. 19) JNDUR = JNDUR * JGROUP
C     ZERO DURATION FOR GRACE NOTES AND STEM SIGN
      IF (ISSMAL.NE.0 .OR. ISSTEM.NE.0) JNDUR = 0
      JGROUP = 1
      RETURN
      END
C
C
      SUBROUTINE DBUG(NLEVEL,NTEXT,N1,N2,N3,N4)
C     DIPLAYS MESSAGE AND 4 NUMBERS FOR DEBUGGING.
      COMMON /KO/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /PP/ ISPOSE,JTERM,ISDBUG,ISHOD,
     * ISGRD1,ISSAG,ISWIDE
      DIMENSION NTEXT(2)
      IF (NLEVEL .GT. ISDBUG) GOTO 9
      CALL TMESS(K6,NTEXT)
      CALL TWT1(N1)
      CALL TWT1(N2)
      CALL TWT1(N3)
      CALL TWT1(N4)
    9 RETURN
      END
```

```
C
C
      BLOCK DATA
C     CONSTANT VALUES
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /FG/ KLEVEL,IEA
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /GP/ JGROUP,MGROUP(7)
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /KE/ KESPAC,KEZERO,KEDOT,KEHYPH,KENAK
      COMMON /KK/ KBAR,KBDOTS,KCLEF,KTIME,KKEY,
     * KTEXUP,KTEXDN,KTEXWD,KFLAT,KNATUR,KSHARP,
     * KTIE,KNDOT,KNREPT,KNSDEL,KSWICH
      COMMON /KM/ KZEROS(2),KMRSIZ,MREST(1)
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /MC/ KBSIZE,KWSIZE,KXMAX,KYMAX,KFSIZE
      COMMON /NH/ JNHEDS,KNHMAX
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /SD/ MYAXID(3)
      COMMON /SE/ JCMD,KSEGS(7)
      COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
C
C     NUMERICAL CONSTANTS
      DATA KBIG/32767/,KZEROS/0,0/,KO/0/,K1/1/,
     *K2/2/,K3/3/,K4/4/,K5/5/,K6/6/,K7/7/,K8/8/
C
C     LITERAL AND TEXTUAL CONSTANTS
      DATA KTHDR/4HMUZK/,KSPACS/8H        /,
     * KQERY(1)/2H? /,KDOT(1)/4H.   /
C
C     EBCDIC CHARACTER CODES
      DATA KESPAC/64/,KEZERO/240/,KEDOT/75/
      DATA KENAK/10/,KEHYPH/96/
C
C     MAIN SEGMENT ID NUMBER FOR EACH COMMAND
      DATA KSEGS/8Z03030303,8Z03030404,8Z04050505,
     *8Z05000000,8Z03000600,8Z03000500,8Z00050000/
C     INTERNAL CODES FOR MUSIC INPUT CHARACTERS
      DATA KBAR/4/,KBDOTS/5/,KSWICH/7/
      DATA KCLEF/64/,KTIME/65/,KKEY/66/
      DATA KTEXUP/128/,KTEXDN/129/,KTEXWD/130/
      DATA KFLAT/211/,KNATUR/212/,KSHARP/213/
      DATA KTIE/214/,KNDOT/218/
      DATA KNREPT/219/,KNSDEL/221/,KEOD/255/
C
C     CONSTANT MACHINE AND PROGRAM PARAMETERS
      DATA KYMAX/780/,KXMAX/1000/,KNHMAX/8/
      DATA KMSIZE/8/,KNLIST/3/,KBSIZE/252/
      DATA KBFSIZ/360/,KBFWDS/90/,KDATA/358/
      DATA KFSIZE/159/,KWSIZE/4/
C
      DATA MGROUP/1,2,0,0,0,-1,0/,KHS/4/
      DATA MYAXID(1)/0415263/,MYAXID(3)/3625847/
      DATA MYSTV(2)/50/,MYSTV(3)/50/
C
C     CODED FORM OF MEASURE REST
      DATA KMRSIZ/2/,MREST/8ZA6010000/
      END
!MACRSYM SI,GO
*     ********************************************
*     *                                          *
*     *    UTILITY ROUTINES FOR SEGMENT LOADING  *
*     *        AND TERMINAL INPUT/OUTPUT         *
*     *                                          *
*     ********************************************
         DEF       OVMOVE,LSEG,URD,UWT
         REF       7CREAD,7CWRITE,9SETUP2,ALPHA
         REF       VDUDCB,WAIT
*
OVMOVE   EQU       *
*INITIAL ENTRY POINT ON PROGRAM LOADING.  *OVMOVE*
*REPACKS THE OVERLAY SEGMENT TABLE AND MOVES THE
*PROGRAMS TEMP STACK TO USE THE SPACE GAINED.
```

```
        LW,1    X'4E'       PCB ADDRESS
        LW,2    2,1         ØVLØAD TABLE ADDR
        BEZ     TØSTART     B IF NØ TABLE
        STW,2   0,1         NEW TEMP STACK ADDR
        LI,3    250         NEW TEMP STACK SIZE
        SLS,3   16
        STW,3   1,1         TØ WØRD 1 ØF PCB
        LW,3    0,2         NUMBER ØF SEGMENTS
        STH,3   SGID
        LW,4    2,2         FIRST WØRD ØF
        STW,4   SGFPT               SEGLØAD FPT
ØV1     LW,4    1,2         SEGMENT ID NUMBER
        STH,4   SGID,3
        LW,4    6,2         STARTING ADDRESS
        STH,4   SGBASE,3
        LW,4    7,2         # ØF BYTES IN SEGT
        STH,4   SGSIZ,3
        LW,4    8,2         GRANULE NUMBER
        STH,4   SGRAN,3
        AI,2    11          NEXT ØVLØAD ENTRY
        BDR,3   ØV1
TØSTART B       14,1        GØTØ TRUE START
*
LSEG    EQU     $
*CALL LSEG(N); *N*=SEGMENT IDENTIFICATION NUMBER.
*REPLACES SYSTEM SEGMENT LØADER RØUTINE, USING
*ØVLØAD TABLE AS MØDIFIED BY *ØVMØVE*.
        LW,1    *15         ADDR ØF SEGMENT ID
        BGEZ    $+2
        LW,1    0,1         INDIRECT ADDRESSING
        AW,15   14          SKIP ARGUMENTS
        STW,15  RET         SAVE RETURN ADDRESS
        LW,1    0,1         REQUIRED SEGMENT ID
        BEZ     *RET        NØ ACTIØN IF 0 ID
        CW,1    SEG         IS SEGMENT LØADED?
        BE      *RET        RETURN IF YES
        LH,2    SGID        NUMBER ØF SEGMENTS
LS1     LH,3    SGID,2
        CW,1    3
        BE      MATCH
        BDR,2   LS1
ERRØR   DATA    0           NØNEXISTENT INSTR.
MATCH   LD,4    SGFPT       FPT WØRD C & FLAGS
        AW,5    IEA         ADD END-ACTIØN FLAG
        LH,6    SGBASE,2    BUFFER ADDRESS
        LH,7    SGSIZ,2     BYTE CØUNT
        LH,8    SGRAN,2     KEY NUMBER
        LW,9    =X'40000000'
        AW,9    KLEVEL      INTERRUPT LEVEL
        CAL1,1  4           READ SEGMENT NØWAIT
        STW,1   SEG         NEW SEGMENT NUMBER
CHECK   LW,4    SGFPT
        AW,4    =X'10000000' FØR CHECK FUNCTIØN
        LW,5    =X'E0000000' ERR,ABNØRMAL,BUSY
        LI,6    ERRØR       ERRØR ADDRESS
        LI,7    ERRØR       ABNØRMAL ADDRESS
        LI,8    LSWAIT      BUSY ADDRESS
        CAL1,1  4           CHECK IØ CØMPLETIØN
        B       *RET        RETURN
LSWAIT  LI,14   0
        BAL,15  WAIT
        B       CHECK
*
*
UWT     EQU     *           WRITE TØ TERMINAL
        MTB,1   15          SET WRITE FLAG
URD     EQU     *           READ FRØM TERMINAL
        BAL,6   9SETUP2
        DATA    L1          1ST ARGUMENT ADDR
        DATA    L2          2ND ARGUMENT ADDR
        STW,15  RET         SAVE RETURN ADDRESS
        LI,14   0
        BAL,15  ALPHA       SET CHARACTER MØDE
        LW,4    *L1         BYTE CØUNT
        LW,13   L2          BUFFER ADDRESS
        MTB,0   RET         READ ØR WRITE?
        BGZ     WT1         B IF WRITE
        LI,1    X'20000'
        STS,1   VDUDCB      SET NØECHØ FLAG
        LI,5    0           BYTE DISPLACEMENT
```

```
         BAL,12   7CREAD       READ TEXT
         STW,6    *L1          ACTUAL BYTE COUNT
         B        *RET         RETURN
WT1      BAL,12   7CWRITE      WRITE TEXT
         B        *RET         RETURN
         BOUND    8            DOUBLEWORD BOUNDARY
#SEGS    EQU      28           NUMBER OF SEGMENTS
SGFPT    RES      1
         DATA     X'31000000'  FLAGS FOR SEGLOAD
SEG      DATA     0
SGID     RES      #SEGS/2+2
SGBASE   RES      #SEGS/2+2
SGSIZ    RES      #SEGS/2+2
SGRAN    RES      #SEGS/2+2
L1       EQU      OVMOVE
L2       EQU      OVMOVE+1
RET      EQU      OVMOVE+2     SAVE RETURN ADDRESS
*
FG       DSECT    0            COMMON /FG/
KLEVEL   RES      1
IEA      RES      1
         USECT    OVMOVE
         END      OVMOVE
!RADEDIT
:COP (FIL,BT,G0),(FIL,D1,JHR00T)
!ASS (M:B0,D1,JHR00TB)
!FORTRAN NS,B0,S
C
C
C     ******************************************
C     *                                        *
C     *EXTRA ROUTINES FOR BACKGROUND VERSION ROOT*
C     *                                        *
C     ******************************************
C
C
      SUBROUTINE BGDFGD
C     SET INTERRUPT LEVEL FOR BACKGROUND PROGRAM
      COMMON /FG/ KLEVEL,IEA
      KLEVEL = 0
      IEA = 0
      RETURN
      END
C
C
      SUBROUTINE PRINIT
C     INITIALISATION FOR PRINT COMMAND
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,K8IG
      COMMON /PL/ ISPLOT
      COMMON /PP/ ISPOSE,JTERM,ISDBUG,ISHQC,
     * ISGRD1,ISSAG,ISWIDE
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      DIMENSION LMESS(2)
      DATA LMESS/8HPRINTING/
      CALL TMESS(K8,LMESS)
C     SET FLAGS FOR GRAPH-PLOTTING AND HIGH
C     QUALITY DISPLAY SYMBOLS
      ISPLOT = 1
      ISHQD = 1
      RETURN
      END
C
C
      SUBROUTINE PREND
C     END OF PRINT COMMAND.
      DIMENSION LMESS(5)
      DATA LMESS/17HPRINTING COMPLETE/,LPLDEV/104/
      CALL PPAGE
      ENDFILE LPLDEV
      CALL REJECT
      CALL TMESS(17,LMESS)
      RETURN
      END
C
C
      SUBROUTINE PPLOT(NTYPE,NX,NY)
C     POSITION PEN AND/OR DRAW LINE ON PLOTTER.
C     *NTYPE* = 0 POSITION PEN ONLY, 1 DRAW LINE,
C     *NX*, *NY* ARE FINAL X,Y CO-ORDINATES.
      COMMON /P1/ PSCALE
```

```
      COMMON /QL/ IXG,IYG,IXGMIN,IGMAX,JXMAX,JYMAX
      DATA PSCALE/0.009/
      CALL PLOT(PSCALE*(JYMAX-NY),PSCALE*NX,
     *                          3-NTYPE)
      RETURN
      END
C
C
      SUBROUTINE PWT(NLENG,NTEXT)
C     DISPLAY TEXT STRING ON GRAPH PLOTTER,
C     STARTING AT CURRENT POSITION.
C     *NLENG* = NUMBER OF CHARACTERS IN STRING,
C     *NTEXT* CONTAINS THE TEXT.
      COMMON /P1/ PSCALE
      COMMON /QL/ IXG,IYG,IXGMIN,IGMAX,JXMAX,JYMAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      DIMENSION NTEXT(1)
      CALL SYMBOL(PSCALE*(JYMAX-IYT),PSCALE*IXT,
     *   PSCALE*2*IYINC,NTEXT,90.0,NLENG)
      RETURN
      END
C
C
      SUBROUTINE PPAGE
C     POSITION PEN TO NEW PAGE ON GRAPH PLOTTER
      CALL NUPAGE(1.0)
      RETURN
      END
C
C
      SUBROUTINE PHEAD(NX)
C     DISPLAY NOTE HEAD USING HERSHEY SYMBOL
      COMMON /KM/ KZEROS(2),KMRSIZ,MREST(1)
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /N0/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      IXT = NX
      IYT = IYN
      IF (JLENTH-3) 3,4,5
C     OPEN HEAD FOR BREVE AND SEMIBREVE
    3 LID = 2370
      GOTO 7
C     OPEN HEAD FOR MINIM
    4 LID = 2371
      GOTO 7
C     SOLID NOTE HEAD FOR CROTCHET AND SHORTER
    5 LID = 2372
    7 CALL DSYMB(LID,K1,K0,KZEROS)
      RETURN
      END
C
C
      BLOCK DATA
C     THIS SUBPROGRAM IS PROGRAM-GENERATED.
      COMMON /XJ/ MHYTAB(248)
      DIMENSION M1(76)
      EQUIVALENCE(M1,MHYTAB(1))
      DIMENSION M2(76)
      EQUIVALENCE(M2,MHYTAB(77))
      DIMENSION M3(76)
      EQUIVALENCE(M3,MHYTAB(153))
      DIMENSION M4(20)
      EQUIVALENCE(M4,MHYTAB(229))
      DATA M1/
     *8Z00100022,8Z00340052,8Z007000AA,8Z00E4011A,
     *8Z0132014A,8Z0174018C,8Z01A401E0,8Z020802AC,
     *8Z03403FC1,8ZC0BEC2C0,8ZC0C2BEC0,8Z40C0C2BE,
     *8Z40C2BE3E,8Z3BC2C4BE,8ZC3BEC2BE,8ZC1BCC0BE,
     *8ZBFBE8FBF,8Z38CDC6BD,8Z3AC2C3BF,8ZC4BEC2BE,
     *8ZC13E45C7,8ZBFBE8EBE,8ZBCBE8DBF,8Z46C2BABD,
     *8Z48CDC1BF,8ZC1BEC0BE,8ZBFBDBEBE,8ZBDBE8C3E,
     *8Z3EC58D8F,8ZBFBFBFBE,8ZC0BEC1BE,8ZC1BFC3BF,
     *8ZC4C0C3C1,8ZC1C1C1C2,8ZC0C2BFC2,8ZBFC1BDC1,
     *8ZBCC03DBF,8ZBFBEC0BD,8ZC1BEC1BF,8ZC2BF47C1,
     *8ZC1C2C0C3,8ZBFC2BFC1,8ZBE4141C5,8ZBC8FBDBE,
     *8ZBFBEC0BE,8ZC1BEC2BF,8ZC3C0C4C1,8ZC3C2C1C2,
     *8ZC0C2BFC2,8ZBEC1BDC0,8Z45BFBBC1,8Z43C088BF,
```

```
*8Z8BBE43C2,8Z8CBC41BC,8ZC5BF3DC0,8ZC5C1C5C2,
*8Z3DBEC444,8Z41C5BDBF,8Z8EBEBFBE,8ZC0BEC1BE,
*8ZC2BFC2C0,8ZC3C1C2C2,8ZC1C2C0C2,8ZBFC2BEC1,
*8ZBEC03CBD,8ZC6C339BB,8ZC8C437BA,8ZCAC537BA,
*8ZCAC537BA,8ZC8C439BB,8ZC6433DCB,8ZC0A946D8,
*8ZC0A938CF,8ZCAC236BD,8ZCAC236B6,8ZCAC236BD/
 DATA M2/
*8ZCA423CCC,8ZCCAE48CC,8ZC0AE38D0,8ZC8C238BD,
*8ZC8C236B6,8ZC8C238BD,8ZC8423CD0,8ZC0AB40C9,
*8ZC3C2C3C0,8ZC2BFC1BE,8ZC0BEBFBE,8ZBDBEBEBF,
*8ZBDBF40C9,8ZC3C1C3C0,8ZC2BF3FC1,8ZC1BEC0BE,
*8ZBFBEBE3E,8Z36C9C0BD,8Z54C3C0BD,8Z2CC3D4C0,
*8Z2CBFD4C0,8Z2CBFD4C0,8Z2CBFD440,8Z3BC4C0BD,
*8Z4AC3C0BD,8Z36C3CAC0,8Z36BFCAC0,8Z36BFCAC0,
*8Z36BFCA40,8Z3FCFC5B6,8ZBCB9C0BF,8Z43C9BCB9,
*8Z43CAC0BE,8ZBCB9C2BD,8ZC3BC42BD,8ZBEC3BEC1,
*8ZBEC0BEBF,8ZBFBEC0BE,8ZC1BEC3BE,8Z45C5BEC2,
*8ZBEC1BCC0,8ZCCBCC1BE,8ZC2BE41C9,8ZBDBEBE3D,
*8Z3EC3BFC2,8ZCCC2BEC0,8ZC0BEC1BF,8ZC2BFC3C0,
*8ZC2C1C2C2,8Z37C1C0BD,8Z3FC2C2C0,8Z3EBFC6BE,
*8Z3DC0C5C1,8Z42C2BC33,8Z35ACC1C0,8ZC1C1C0C1,
*8ZBFC1BFC0,8ZBFBFC0BE,8ZC1BEC2BF,8ZC2C0C3C1,
*8ZC2C2C1C2,8ZC1C4C0CB,8ZBFDAC0C7,8ZC1C5C1C2,
*8ZC2C1C1C0,8ZC2BFC1BE,8ZC0BCBFBD,8ZBFBEBEBD,
*8ZBBBCBABC,8ZBEBEBEBD,8ZBFBEBFBC,8ZC0BCC1BC,
*8ZC2BDC3BE,8ZC4BFC4C0,8ZC4C1C2C1,8ZC2C3C1C3/
 DATA M3/
*8ZC0C4BFC3,8ZBFC2BEC2,8ZBDC1BCC0,8ZBDBFBEBE,
*8ZBFBDC0BC,8ZC1BDC2BE,8Z3AB3C0BF,8ZC1C0C0C1,
*8ZBFC04EE9,8ZBCBCBBBC,8ZBDBDBEBD,8ZBFBEBFBD,
*8ZC0BCC1BC,8ZC1BEC3BD,8Z48BFC3C1,8ZC2C1C2C3,
*8ZC1C3C0C4,8ZBFC3BFC2,8ZBEC2BE41,8Z3CC1C1C2,
*8ZC2C1C2C0,8ZC2BFC1BE,8ZC0BEBFBE,8ZBEBFBEC0,
*8ZBEC1BFC1,8ZBFC3C0C3,8ZC1C3C2C2,8ZC3C1C4C0,
*8ZC4BFC3BE,8ZC2BDC1BC,8ZC0BBBFBC,8ZBFBEBEBD,
*8ZBDBDBCBD,8ZBBBDBCBE,8Z4AE3C3BF,8ZC3BEC2BD,
*8ZC1BCC0BB,8ZBFBCBFBE,8ZBEBDBDBD,8ZBBBCBDBE,
*8Z3FDAC4C0,8Z3BBFC6C0,8Z39BFC8C0,8Z38BFC8C0,
*8Z38BFC8C0,8Z39BFC6C0,8Z3BBFC4C0,8Z51C9C0BE,
*8ZC2C0C0C2,8ZBEC041C0,8ZC0BE3FC1,8ZC2C03EB7,
      *8ZC0BEC2C0,8ZC0C2BEC0,8Z41C0C0BE,8Z3FC1C240,
      *8Z36D4C098,8Z41E8C098,8Z44E8C098,8Z44E4C2C0,
      *8ZC0BEBEC0,8ZC0C3C1C2,8ZC2C1C3C0,8ZC2BFC2BE,
      *8ZC1BDC0BB,8ZBFBDBEBE,8ZBEBFBEC0,8ZBEC1BFC2,
      *8ZBFBEBEBD,8ZBFBFC1BF,8ZC2BDC1BE,8ZC1C2C2C1,
      *8ZC2C0C2BF,8ZC2BEC1BD,8ZC0BBBFBD,8ZBEBEBEBF/
       DATA M4/
      *8ZBDC0BEC1,8ZBFC2C0C3,8ZC2C0C0BE,8ZBEC041E0,
      *8ZC0BE3FC1,8ZC2C046C4,8ZC1BEC1BD,8ZC0BBBFBD,
      *8ZBFBE39C2,8ZC0BEBEBD,8ZBEBFC2BF,8ZC2BDC0BE,
      *8Z47C2C1BE,8ZC1BDC0BB,8ZBFBDBFBE,8Z39C5C0BE,
      *8Z3FC1C240,8Z00000000,8Z00000000,8Z00000000/
       END
!ALL (FIL,G0),(RSI,27),(FOR,8),(FSI,100)
!FORTRAN G0,S,NS
C
C
C     ******************************************
C     *                                        *
C     *EXTRA ROUTINES FOR FOREGROUND VERSION ROOT*
C     *                                        *
C     ******************************************
C
C
C
      SUBROUTINE BGDFGD
C     SET INTERRUPT LEVEL FOR FOREGROUND PROGRAM
      COMMON /FG/ KLEVEL,IEA
      KLEVEL = 2Z69
      IEA = 6Z800000
      CALL INTLEVEL(KLEVEL)
      RETURN
      END
!MACRSYM SI,G0
*DUMMY GRAPH PLOTTING ROUTINES FOR FOREGROUND
*VERSION TO AVOID UNSATISFIED REFERENCES.
*THESE ARE NEVER CALLED IN FOREGROUND VERSION.
          DEF       PRINIT,PREND,PPLOT,PWT,PPAGE
          DEF       PHEAD,HYPLOT
PRINIT    EQU       *
```

```
PREND    EQU      $
PPLOT    EQU      $
PWT      EQU      $
PPAGE    EQU      $
PHEAD    EQU      $
HYPLOT   EQU      $
         AI,15    14          SKIP ARGUMENTS
         B        *15         RETURN
         END
!RADEDIT
:COP (FIL,BT,G0),(FIL,D1,JHR00TF)
!ALL (FIL,G0),(RSI,27),(FOR,B),(FSI,100)
!FORTRAN GE,S,NS
C     EXTRA ROUTINES FOR FOREGROUND VERSION ROOT
C     ALTERNATIVE VERSION
      SUBROUTINE BGDFGD
C     SET INTERRUPT LEVEL FOR FOREGROUND PROGRAM
      COMMON /FG/ KLEVEL,IEA
      KLEVEL = 2267
      IEA = 6Z800000
      CALL INTLEVEL(KLEVEL)
      RETURN
      END
!MACRSYM SI,G0
*DUMMY GRAPH PLOTTING ROUTINES FOR FOREGROUND
*VERSION TO AVOID UNSATISFIED REFERENCES.
*THESE ARE NEVER CALLED IN FOREGROUND VERSION.
         DEF      PRINIT,PREND,PPLOT,PWT,PPAGE
         DEF      PHEAD,HYPLOT
PRINIT   EQU      $
PREND    EQU      $
PPLOT    EQU      $
PWT      EQU      $
PPAGE    EQU      $
PHEAD    EQU      $
HYPLOT   EQU      $
         AI,15    14          SKIP ARGUMENTS
         B        *15         RETURN
         END
!RADEDIT
```

```
:COP (FIL,BT,G0),(FIL,D1,JHR00TA)
!JOB JH,COMPILE MUSIC PROGRAM SEGMENT 2
!EXTRABGD
!ASS (M:B0,D1,JHR0M2)
!FORTRAN B0,NS
C
C
C     ******************************************
C     *                                        *
C     *        SEGMENT 2 - PROGRAM CONTROL     *
C     *                                        *
C     ******************************************
C
C
      SUBROUTINE CTRL
C     MAIN PROGRAM CONTROL ROUTINE.  FINISH
C     PROCESSING LAST COMMAND IF ANY, SELECT NEXT
C     COMMAND AND PERFORM INITIALISATION FOR IT.
      COMMON MINBUF(90),M0PBUF(90),NINBUF,N0PBUF,
     * JINSEC,J0PSEC,JINBUF,J0PBUF,JINLK,J0PLK,
     * NILENG,MIBAR(64),N0LENG,M0BAR(64),JIBAR,
     * J0BAR,JBBUF,NFLAGS,MFLAGS(3)
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /CB/ MXCOM(4)
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJCY,JX,JY
      COMMON /ED/ NEDBAR,NEDSTV,NELENG,JEDCMC
      COMMON /EE/ JEDIT,JEFREE,JEBACK,ISITEM
      COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSGZ,
     * ISREST,ISAHED,ISXPOS,ISEMB
      COMMON /FG/ KLEVEL,IEA
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /FS/ JCSTAT,JNSTAT,MSTAT(3)
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /JD/ JSTAFF,JDLINE,JDHIGH,JDLOW
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
```

```
      COMMON /L0/ NLOCS,JLOCN
      COMMON /MC/ KBSIZE,KWSIZE,KXMAX,KYMAX,KFSIZE
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /M0/ JIMODE,JOMODE
      COMMON /N0/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /PP/ ISPOSE,JTERM,ISDBUG,ISHOD,
     * ISGRD1,ISSAG,ISWIDE
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /QL/ IXG,IYG,IXGMIN,IGMAX,JXMAX,JYMAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /SE/ JCMD,KSEGS(7)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /WS/ L1,L2,L3,JBYTE
      EQUIVALENCE(JSPEED,MXCOM(1)),(IXGSAV,
     * MXCOM(1)),(IYGSAV,MXCOM(2)),(IHGSAV,
     * MXCOM(3))
      DIMENSION LMCHAR(1)
      DATA LMCHAR(1)/3HPLF/,KBRDEV/119/
C
C     FINISH PROCESSING PREVIOUS COMMAND
    3 CALL LSEG(21)
      CALL CMDEND
      IF (JCMD .EQ. 0) GOTO 34
C     CONTINUE PREVIOUS COMMAND IF UNFINISHED
      IF (JCMD .EQ. 26) GOTO 385
      GOTO 88
C     CHANGE MODE IF REQUIRED
   34 CALL LSEG(21)
      CALL CHMODE
C     GENERAL INITIALISATION FOR NEXT COMMAND
   38 CALL LSEG(21)
      CALL CMDSET
C     REQUEST AND READ NEXT COMMAND
      CALL NXCMD
      IF (JNSTAT .EQ. 0) GOTO 8
  385 IF (JCMD .NE. 10 .AND. JCMD .NE. 22)
     *                          CALL LSEG(22)
C     BRANCH INDEXED BY COMMAND NUMBER
      GOTO (48,44,46,4,5,42,56,64,62,54,6,88,88,
     * 52,58,6,66,68,72,76,77,78,88,483,39,875),
     * JCMD
C     LIST COMMAND OPTIONS
   39 CALL OPLIST
      GOTO 79
C
C     BUILD
    4 CALL ININIT
C     USE ONLY TOP TWO THIRDS OF SCREEN FOR
C     DISPLAY OF MUSIC DURING INPUT
   41 NDL = (2*JYMAX) / (3*IYSTV)
      GOTO 75
C
C     DISPLAY MUSIC
   42 NDL = JYMAX / IYSTV
      GOTO 74
C
C     EDIT MEASURE
   44 ISEDIT = 1
C     REQUEST BAR NUMBER
      CALL QBAR(JBS,K0)
      IF (JBS .EQ. 0) GOTO 45
C     ERROR IF CURRENT SECTION IS EMPTY
      IF (JCSTAT .EQ. 1) GOTO 8
      GOTO 484
C     EDIT HEADER RECORD, ERROR IF CURRENT MODE
C     IS NOT PRINTED MUSIC
   45 IF (JIMODE .NE. 1) GOTO 84
      JSSEL = NLINES
      GOTO 486
C
C     INSERT BAR -BEFORE -
   46 CALL XMESIJ(29)
C     REQUEST BAR NUMBER
      CALL QBAR(JBS,K1)
      JBAR = JBS
      GOTO 482
C
```

```
C      DELETE BARS
   48  CALL QBARS(K1)
  482  JSSEL = 1
       GOTO 486
C
C      CHANGE BRAILLE FLAGS FIELD
  483  IF (JIMODE .EQ. 1) GOTO 84
C      REQUEST BAR NUMBER
       CALL QBAR(JBS,K1)
C      REQUEST STAVE NUMBER
  484  CALL QSTAV(JSSEL)
C      MARK SPECIFIED STAVE AS SELECTED
       CALL STBYTE(K1,MSSEL,JSSEL)
C      LOCATE SPECIFIED MEASURE IN FILE
  486  CALL BFIND(JBS,JSSEL)
C      FILE ERROR IF END OF SECTION ENCOUNTERED
       IF (ISEND .NE. 0) GOTO 488
       NEDBAR = NIB
       NEDSTV = NIS
C      *NELENG* = NUMBER OF BYTES OF CODED MEASURE
C      ACTUALLY OBTAINED FROM THE FILE
       NELENG = (NILENG+NFLAGS) * (1-ISREST) + 1
       JDHIGH = NIS
       JDLOW = NIS
       IF (JCMD .EQ. 3) GOTO 41
       IF (JCMD .EQ. 24) GOTO 49
       GOTO 88
C
C      IN CASE *NBARS* IS SOMEHOW GREATER THAN
C      ACTUAL NUMBER STORED, REDUCE IT BY ONE
  488  IF (JCMD .EQ. 1) NBARS = NBARS - 1
C      OUTPUT MESSAGE -BAR NUMBER ERROR IN FILE-
       CALL XMESIJ(30)
       GOTO 34
C
C      CHANGE FLAGS FIELD
   49  ICFLAG = MFLAGS(1)
       CALL CHVAL(ICFLAG,K0,127)
       CALL BMOVE(MIBAR,MCBAR)
C      SIMULATE EDIT COMMAND FOR FILE UPDATING
       JCMD = 2
       GOTO 79
C
C      ADD
     5 JBS = NBARS
       NOB = NBARS + 1
       NOS = 1
       GOTO 41
C
C      DELETE SECTION
   52  CALL DELETE(JIMODE)
       GOTO 34
C
C      SAVE
   54  DO 550 L=1,3
  550  IF (MSTAT(L) .GE. 2) GOTO 555
C      ERROR, ALL SECTIONS EMPTY
       GOTO 81
  555  CALL SQEEZE
       GOTO 88
C
C      BRAILLE TRANSLATION
   56  IF (JIMODE .NE. 1) GOTO 84
C      ERROR IF UNFORMATTED BRAILLE IS PROTECTED
       IF (MSTAT(2) .EQ. 3) GOTO 83
       CALL GETHDR
       NOLENG = 0
       JOMODE = 2
       JPPCH = 0
       JLINE = 0
       CALL CHSTAV(K1)
       NOB = 0
       NOS = NLINES
       CALL DELETE(K2)
       CALL TNLINE
       MHEAD(2) = JFREE
       CALL OPPOSN(KDATA*MHEAD(2))
       IYS = 500
       GOTO 74
C
```

```
C       CHANGE MODE
   58 CALL NOECHO
C       IDENTIFY SECTION CHARACTER (P,U,F)
       DO 590 L=1,3
  590 IF (ISSAME(NBYTE(MCHAR,K1),NBYTE(LMCHAR,L))
      *                          .NE. 0) GOTO 595
       CALL REJECT
       GOTO 58
  595 CALL ECHO
       JMMODE = L
       GOTO 34
C
C       END OF RUN OR RESTORE
C       ERROR IF ANY SECTIONS OF FILE PROTECTED
    6 DO 610 L=1,3
  610 IF (MSTAT(L) .EQ. 3) GOTO 83
       IF (JCMD .EQ. 11) GOTO 88
C       STOP COMMAND
       IF (ISEMB .EQ. 0) GOTO 615
C       DISPLAY -EMBOSS BRAILLE- MESSAGE
       CALL XMESIJ(37)
       CALL TNLINE
       ENDFILE KBRDEV
C       DISPLAY -PROGRAM RELEASED- MESSAGE
  615 CALL XMESIJ(38)
C       DISPLAY -PRESS TTY BUTTON- IF TEKTRONIX
       IF (JTERM .EQ. 0) CALL XMESIJ(39)
       GOTO 88
C
C       EMBOSS BRAILLE
   62 IF (JIMODE .EQ. 1) GOTO 84
       GOTO 88
C
C       FORMAT
   64 IF (JIMODE .NE. 2) GOTO 84
C       ERROR IF FORMATTED BRAILLE IS PROTECTED
       IF (MSTAT(3) .EQ. 3) GOTO 83
       CALL DELETE(K3)
C       DISPLAY -FORMATTING- MESSAGE
       CALL XMESIJ(111)
       GOTO 88
C
C       DISPLAY CURRENT FILE SUMMARY INFORMATION
   66 CALL SUMARY
C       OMIT HEADER RECORD DISPLAY IN BRAILLE MODES
       IF (JIMODE .NE. 1) GOTO 34
       GOTO 88
C
C       CHANGE HORIZONTAL DISPLAY SCALE
   68 CALL CHVAL(JSCALE,16,128)
       NLOCS = 0
C       STORE NEW DISPLAY SCALE VALUE IN DISC FILE
       CALL OPPOSN(KO)
       CALL OPHW(JSCALE)
       GOTO 88
C
C       PLAY MUSIC
   72 IF (JIMODE .NE. 1) GOTO 84
       JSPEED = 200
C       REQUEST PLAYING SPEED
       L1 = NQUEST(54,KO,KBIG)
C       USE DEFAULT VALUE IF NO SPEED SPECIFIED
       IF (L1 .GT. 0) JSPEED = L1
       NDL = 0
       CALL TNLINE
C
C       REQUEST BAR NUMBER RANGE
   74 CALL QBARS(KO)
C       REQUEST STAVE NUMBERS
   75 CALL RANGE(107,K1,NLINES)
       NDP = NDL / NSSEL
       IF (JCMD .NE. 7) GOTO 88
       IF (NSSEL .GT. 4) GOTO 85
C       DISPLAY -TRANSLATING- MESSAGE
       CALL XMESIJ(112)
       GOTO 88
C
C       CHANGE PROGRAM PARAMETERS
   76 CALL CHANGE
       GOTO 38
```

```
C
C     PRINT ON GRAPH PLOTTER OR LINEPRINTER
   77 IXQSAV = IXQ
      IYQSAV = IYQ
      IHGSAV = ISHQD
C     USE LINEPRINTER IF NOT PRINTED MUSIC MODE
      IF (JIMODE .NE. 1) GOTO 42
C     NOT AVAILABLE IN FOREGROUND PROGRAM VERSION
      IF (KLEVEL .NE. 0) GOTO 86
C     PLOTTER OUTPUT IS ALL ONE PAGE
      JYMAX = KBIG
      GOTO 42
C
C     CANCEL INITIALISATION FLAG
   79 INIT = 0
      GOTO 3
C
C     SQUEEZE DISC FILE
   78 CALL SQEEZE
      GOTO 34
C
C     COMMAND ERROR MESSAGES
    8 L = JCSTAT + 30
      GOTO 87
C     "NO MUSIC STORED" MESSAGE
   81 L = 31
      GOTO 87
C     "FILE PROTECTED" MESSAGE
   83 L = 33
      GOTO 87
C     "WRONG MODE" MESSAGE
   84 L = 34
      GOTO 87
C     "TOO MANY STAVES" MESSAGE
   85 L = 35
      GOTO 87
C     "COMMAND NOT IMPLEMENTED" MESSAGE
   86 L = 36
C     DISPLAY MESSAGE
   87 CALL XMESIJ(L)
      GOTO 38
C     CONTINUATION OF PROGRAM INITIALISATION
  875 CALL SETUPB
   88 INIT = 0
      RETURN
      END
C
C
      SUBROUTINE XMESIJ(N)
C     OUTPUT MESSAGE NUMBER *N* ON TERMINAL.
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /XA/ MMESIJ(544)
C     MAXIMUM MESSAGE LENGTH 40 CHARACTERS
      DIMENSION LMTEXT(10)
C     SET POINTER TO START OF MESSAGE IN TABLE
      LPTR = NHW(MMESIJ,N+1)
C     SET LENGTH OF MESSAGE IN CHARACTERS
      LLENG = NBYTE(MMESIJ,LPTR+1)
C     SPLIT OFF NEW-LINE INDICATOR BIT
      LNL = NSPLIT(LLENG,K0)
C     MOVE MESSAGE FROM TABLE TO MESSAGE BUFFER
      CALL MBS(MMESIJ,LPTR+2,LMTEXT,K1,LLENG)
      IF (LNL .NE. 0) GOTO 7
C     DISPLAY MESSAGE AT CURRENT POSITION
      CALL TWT(LLENG,LMTEXT)
      GOTO 9
C     DISPLAY MESSAGE ON NEW LINE
    7 CALL TMESS(LLENG,LMTEXT)
    9 RETURN
      END
C
C
      SUBROUTINE TLMESS(N)
C     DISPLAY MESSAGE *N* AT LEFT OF SCREEN.
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
C     NO NEW PAGE IF ALREADY AT LEFT HAND SIDE
      IF (IYQ.GT.6 * IYINC .AND. IXQ.EQ.0) GOTO 3
C     FORCE NEW PAGE ON DISPLAY
      IYQ = 0
```

```
      IXQ = JXMAX
C     DISPLAY MESSAGE
    3 CALL XMESIJ(N)
      RETURN
      END
C
C
      SUBROUTINE DPSET
C     SET DISPLAY PARAMETERS FOR TERMINAL TYPE.
C     *JTERM* = 0 FOR TEKTRONIX, 1 FOR GT40.
      COMMON /DP/ IXSHIF,IYSHIF
      COMMON /PP/ MPP1(1),JTERM,MPP2(4),ISWIDE
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      IXSHIF = JTERM
      IYSHIF = 2 * JTERM
      IXINC = 2 * JTERM + 12
      IYINC = 2 * JTERM + 6
      IYSTV = 10 * IYINC * (ISWIDE+2)
      MYSTV(1) = IYSTV
      RETURN
      END
C
C
      BLOCK DATA
C     THIS SUBPROGRAM IS PROGRAM-GENERATED.
      COMMON /XA/ MMESIJ(544)
      DIMENSION M1(76)
      EQUIVALENCE(M1,MMESIJ(1))
      DIMENSION M2(76)
      EQUIVALENCE(M2,MMESIJ(77))
      DIMENSION M3(76)
      EQUIVALENCE(M3,MMESIJ(153))
      DIMENSION M4(76)
      EQUIVALENCE(M4,MMESIJ(229))
      DIMENSION M5(76)
      EQUIVALENCE(M5,MMESIJ(305))
      DIMENSION M6(76)
      EQUIVALENCE(M6,MMESIJ(381))
      DIMENSION M7(76)
      EQUIVALENCE(M7,MMESIJ(457))
      DIMENSION M8(12)
      EQUIVALENCE(M8,MMESIJ(533))
      DATA M1/
     *8Z007000E2,8Z00E300FE,8Z00FF0122,8Z013C0153,
     *8Z0172018D,8Z01B001CB,8Z01E70201,8Z02150233,
     *8Z0253026C,8Z028002A5,8Z02B6C2DA,8Z02F4031B,
     *8Z03440368,8Z038D039D,8Z03BF03C0,8Z03C803E1,
     *8Z03F10404,8Z0413041E,8Z042E0446,8Z04610472,
     *8Z048A0496,8Z04AE04BE,8Z04CD04E8,8Z0505052I,
     *8Z053E0550,8Z056B0584,8Z058B058C,8Z05910597,
     *8Z05A805AE,8Z05B205C0,8Z05D405E6,8Z05F505F8,
     *8Z06040615,8Z062B063C,8Z0649065B,8Z065D067A,
     *8Z06810699,8Z06B806C2,8Z06C706D1,8Z06D506DD,
     *8Z06E306EB,8Z06F106FA,8Z0702070E,8Z07190722,
     *8Z072A0733,8Z074B0758,8Z07600767,8Z076C0773,
     *8Z077F0789,8Z078F0794,8Z079507A5,8Z07AA07AF,
     *8Z07C807F2,8Z0809081A,8Z082B083C,8Z08530860,
     *8Z086B809A,8ZE3C8C540,8ZD6D7E3C9,8ZD6D5E240,
     *8ZC1E5C1C9,8ZD3C1C2D3,8ZC540C1D9,8ZC57A80A2,
     *8ZC1C44040,8Z40C1C4C4,8Z40E3D640,8ZC5D5C440,
     *8ZD6C640C3,8ZE4D9D9C5,8ZD5E340E2,8ZC5C3E3C9,
     *8ZD6D599C2,8ZD9404040,8ZE3D9C1D5,8ZE2D3C1E3/
      DATA M2/
     *8ZC540E3D6,8Z40C2D9C1,8ZC9D3D3C5,8Z96C2E440,
     *8Z4040C2E4,8ZC9D3C440,8ZD5C5E640,8ZE2C5C3E3,
     *8ZC9D6D59E,8ZC3C84040,8Z40C3C8C1,8ZD5C7C540,
     *8ZD7D9D6C7,8ZD9C1D440,8ZD7C1D9C1,8ZD4C5E3C5,
     *8ZD9E29AC3,8ZD3404040,8ZC3D3C5C1,8ZD940C3E4,
     *8ZD9D9C5D5,8ZE340E2C5,8ZC3E3C9D6,8ZD5A2C3D6,
     *8Z404040C3,8ZD6D7E840,8ZC1D9C3C8,8ZC9E5C540,
     *8ZE3C1D7C5,8Z40E3D640,8ZD5C5E640,8ZE3C1D7C5,
     *8Z9AC4C540,8Z4040C4C5,8ZD3C5E3C5,8Z40E2D7C5,
     *8ZC3C9C6C9,8ZC5C440C2,8ZC1D9E29B,8ZC4C94040,
     *8Z40C4C9E2,8ZD7D3C1E8,8Z40E2D7C5,8ZC3C9C6C9,
     *8ZC5C440C2,8ZC1D9E299,8ZC5C44040,8Z40C5C4C9,
     *8ZE340C3E4,8ZD9D9C5D5,8ZE340E2C5,8ZC3E3C9D6,
     *8ZD593C5D4,8Z404040C5,8ZD4C2D6E2,8ZE240C2D9,
     *8ZC1C9D3D3,8ZC59DC5D9,8Z404040C5,8ZD9C1E2C5,
     *8Z40D4C1C7,8ZD9D6E3C9,8ZC340E3C1,8ZD7C540C6,
     *8ZC9D3C69F,8ZC6C94040,8Z40D3C9E2,8ZE340C6C9,
```

AI-SS

```
*8ZD3C5E240,8ZD6D540C1,8ZD9C3C8C9,8ZE5C540E3,
*8ZC1D7C598,8ZC6D34040,8Z40C3C8C1,8ZD5C7C540/
 DATA M3/
*8ZC2D9C1C9,8ZD3D3C540,8ZC6D3C1C7,8Z93C6D640,
*8Z4040C6D6,8ZD9D4C1E3,8Z40C2D9C1,8ZC9D3D3C5,
*8ZA4C9D540,8Z4040C9D5,8ZE2C5D9E3,8Z40C2C1D9,
*8Z40C2C5C6,8ZD6D9C540,8ZE2D7C5C3,8ZC9C6C9C5,
*8ZC440C2C1,8ZD990D4D6,8Z4040C3,8ZC8C1D5C7,
*8ZC540D4D6,8ZC4C5A3D6,8ZD7404040,8ZD3C9E2E3,
*8Z40C1E5C1,8ZC9D3C1C2,8ZD3C540C3,8ZD6D4D4C1,
*8ZD5C440D6,8ZD7E3C9D6,8ZD5E299D7,8ZD3404040,
*8ZD7D3C1E8,8Z4CC3E4D9,8ZD9C5D5E3,8Z40E2C5C3,
*8ZE3C9D6D5,8ZA6D7D940,8Z4040D7D9,8ZC9D5E340,
*8ZD6D540D7,8ZD9C9D5E3,8ZC5D940D6,8ZD940C7D9,
*8ZC1D7C840,8ZD7D3D6E3,8ZE3C5D9A8,8ZD9C54040,
*8Z40D9C5E2,8ZE3D6D9C5,8Z40C4C9E2,8ZC340C6C9,
*8ZD3C540C6,8ZD9D6D440,8ZC1D9C3C8,8ZC9E5C540,
*8ZE3C1D7C5,8ZA3E2C140,8Z4040E2C1,8ZE5C540C4,
*8ZC9E2C340,8ZC6C9D3C5,8Z40D6D540,8ZC1D9C3C8,
*8ZC9E5C540,8ZE3C1D7C5,8ZA4E2C340,8Z4040C3C8,
*8ZC1D5C7C5,8Z4CC8D6D9,8ZC9E9D6D5,8ZE3C1D340,
*8ZC4C9E2D7,8ZD3C1E840,8ZE2C3C1D3,8ZC58FE2E3/
 DATA M4/
*8Z4040C5,8ZD5C440D6,8ZC640D9E4,8ZD5A1E2E4,
*8Z4040C4,8ZC9E2D7D3,8ZC1E840E2,8ZE4D4D4C1,
*8ZD9E840D6,8ZC640C4C9,8ZE2C340C6,8ZC9D3C580,
*8Z07C2C5C6,8ZD6D9C540,8Z98C2C1D9,8Z40D5E4D4,
*8ZC2C5D940,8ZC5D9D9D6,8ZD940C9D5,8Z40C6C9D3,
*8ZC50FD5D6,8Z4CD4E4E2,8ZC9C340E2,8ZE3D6D9C5,
*8ZC412C6C9,8ZD3C540C1,8ZD3D9C5C1,8ZC4E840E2,
*8ZC1E5C5C4,8Z0EC6C9D3,8ZC540D7D9,8ZD6E3C5C3,
*8ZE3C5C40A,8ZE6D9D6D5,8ZC740D4D6,8ZC4C50FE3,
*8ZD6D640D4,8ZC1D5E840,8ZE2E3C1E5,8ZC5E217C3,
*8ZD6D4D4C1,8ZD5C440D5,8ZD6E340C9,8ZD4D7D3C5,
*8ZD4C5D5E3,8ZC5C49AD9,8ZE4D540D3,8ZC5C4F240,
*8ZE3D640C5,8ZD4C2D6E2,8ZE240C2D9,8ZC1C9D3D3,
*8ZC510D7D9,8ZD6C7D9C1,8ZD440D9C5,8ZD3C5C1E2,
*8ZC5C41740,8Z4B4B4B40,8Z07D7D9C5,8ZE2E240E3,
*8ZE3E840C2,8ZE4E3E3D6,8ZD5070B60,8Z60C9D5E3,
*8ZC5D9D9E4,8ZD7E397C5,8ZD9D9D6D9,8Z40C9D540,
*8ZE3C8C9E2,8Z40D6D7C5,8ZD9C1E3C9,8ZD6D58FC4,
*8ZC9E2C340,8ZC6C9D3C5,8Z40C5D9D9,8ZD6D98EC6/
 DATA M5/
*8ZC9D3C540,8ZE4D5C3C8,8ZC1D5C7C5,8ZC49AC9D5,
*8ZC4C5E740,8ZC6E4D3D3,8Z6B40C6C9,8ZD3C540D5,
*8ZD6E340E2,8ZC1E5C5C4,8Z9CE6D9D6,8ZD5C740E3,
*8ZC1D7C540,8ZD6D940D5,8ZD640E3C1,8ZD7C540D3,
*8ZD6C1C4C5,8ZC498E3C8,8ZC540E3C1,8ZD7C540C9,
*8ZD5C4C5E7,8Z40C9E240,8ZC3D6D9D9,8ZE4D7E3C5,
*8ZC49CE3C8,8ZC9E240C6,8ZC9D3C540,8ZC9E240D5,
*8ZD6E340D6,8ZD540E3C8,8ZC540E3C1,8ZD7C591C3,
*8ZD6D4D4C1,8ZD5C440C3,8ZC1D5C3C5,8ZD3D3C5C4,
*8Z1AD6E4E3,8ZD7E4E340,8ZC6C9D3C5,8Z40D5D6E3,
*8Z40C1C3C3,8ZC5E2E2C9,8ZC2D3C598,8ZE3C1D7C5,
*8Z40C9D540,8ZE4E2C568,8Z40D7D3C5,8ZC1E2C540,
*8ZE6C1C9E3,8Z03C2C1D9,8Z03C2C1D9,8Z04D3C9D5,
*8ZC505E2D7,8ZC5C5C410,8ZD5E4D4C2,8ZC5D940D6,
*8ZC640E2E3,8ZC1E5C5E2,8Z05E2E3C1,8ZE5C50340,
*8ZE3D60DD7,8ZD9C9D5E3,8ZC5C440D4,8ZE4E2C9C3,
*8Z13E4D5C6,8ZD6D9D4C1,8ZE3E3C5C4,8Z40C2D9C1,
*8ZC9D3D3C5,8Z11C6D6D9,8ZD4C1E3E3,8ZC5C440C2,
*8ZD9C1C9D3,8ZD3C58ED5,8ZC5E7E340,8ZC3D6D4D4/
 DATA M6/
*8ZC1D5C46F,8Z4085C6C9,8ZD3C54088,8ZD4D6C4C5,
*8Z40C9E240,8Z90D4E4E2,8ZE340C2C5,8Z40C9D540,
*8ZD9C1D5C7,8ZC595E3C1,8ZD7C540C3,8ZD6D7C9C5,
*8ZC440C3D6,8ZD9D9C5C3,8ZE3D3E890,8ZD7D3C1E8,
*8ZC9D5C740,8ZC3D6D4D7,8ZD3C5E3C5,8Z8CC6C9D3,
*8ZC540E4D7,8ZC4C1E3C5,8ZC48EC4C9,8ZE2C340C6,
*8ZC9D3C540,8ZC6E4D3D3,8Z84D7C1D9,8ZE39CE2E4,
*8ZD4D4C1D9,8ZE840D6C6,8Z40C3E4D9,8ZD9C5D5E3,
*8Z40C4C9E2,8ZC340C6C9,8ZD3C586E2,8ZC3C1D3C5,
*8Z4097E3E8,8ZD7C540D3,8Z40C6D6D9,8Z40D6D7E3,
*8ZC9D6D540,8ZD3C9E2E3,8Z409EE3C8,8ZC540C3C8,
*8ZC1D5C7C5,8ZC1C2D3C5,8Z40D7C1D9,8ZC1D4C5E3,
*8ZC5D9E240,8ZC1D9C57A,8Z09E3D9C1,8ZD5E2D7D6,
*8ZE2C504C7,8ZE3F4F009,8ZC4C5C2E4,8ZC7C7C9D5,
*8ZC703C8D8,8ZC407C7D9,8ZC1C4C540,8ZF105E2C1,
*8ZC7C5D407,8ZE2D7C1C3,8ZC9D5C705,8ZE2C1E5C5,
*8ZC408D9C5,8ZE2E3D6D9,8ZC5C407C4,8ZC5D3C5E3,
```

```
     *8ZC5C40B40,8Z4CD5C1D4,8ZC540C9E2,8Z407F0A7F,
     *8Z4040E5C5,8ZD9E2C9D6,8ZD50840D9,8ZC5C3D6D9/
      DATA M7/
     *8ZC4E20740,8ZE2E3C1E5,8ZC5E20840,8ZC4C5D3C5,
     *8ZE3C5C417,8ZD7D3C5C1,8ZE2C540C3,8ZD6D5C6C9,
     *8ZD9D440C4,8ZC5D3C5E3,8ZC9D6D50C,8Z40C3C8C1,
     *8ZD5C7C540,8ZC6D9D6D4,8Z07E2C5C3,8ZE3C9D6D5,
     *8ZC6E2E3C1,8ZE3E4E204,8ZC2C1D9E2,8Z06E2E3C1,
     *8ZE5C5E20B,8ZE4D5D7D9,8ZD6E3C5C3,8ZE3C5C409,
     *8ZD7D9D6E3,8ZC5C3E3C5,8ZC405C6C1,8ZD3E2C504,
     *8ZE3D9E4C5,8Z0C8F07D4,8ZE4E2C9C3,8Z40D7D9D6,
     *8ZC7D9C1D4,8Z4C04C2C1,8ZC3D204C6,8ZD6D9C518,
     *8ZC7D9D6E4,8ZD5C440E5,8ZC5D9E2C9,8ZD6D540C3,
     *8ZD6D4D7C9,8ZD3C5C440,8ZA9D7D9C5,8ZE2E240C1,
     *8ZE2C3C9C9,8Z4CC2E4E3,8ZE3D6D540,8ZC6D6D940,
     *8ZC6E4D3D3,8Z4CC3C8C1,8ZD9C1C3E3,8ZC5D940E2,
     *8ZC5E396C9,8ZD5C9E3C9,8ZC1D3C9E2,8ZC9D5C740,
     *8ZC4C9E2C3,8Z4CC6C9D3,8ZC590E2C5,8ZD3C5C3E3,
     *8ZC9D6D540,8ZD5E4D4C2,8ZC5D91040,8ZE2C5D3C5,
     *8ZC3E340E2,8ZE3C1E5C5,8ZE26F4090,8ZC3C8C1D5,
     *8ZC7C540D5,8ZE4D4C2C5,8ZD9E26F40,8Z1640C2D9,
     *8ZC1C9D3D3,8ZC54CD7C1,8ZC7C5E240,8ZE6D9C9E3/
      DATA M8/
     *8ZE3C5D50C,8ZE2D8E4C5,8ZC5E9C9D5,8ZC7404040,
     *8Z8AC6D6D9,8ZD4C1E3E3,8ZC9D5C788,8ZE3D9C1D5,
     *8ZE2D3C1E3,8ZC9D5C700,8Z00000000,8Z00000000/
      END
!ASS (M:B0,D1,JHR0M21)
!FORTRAN B0,NS
C
C
C     ******************************************
C     *                                        *
C     *     SEGMENT 21 - COMMAND TERMINATION   *
C     *                                        *
C     ******************************************
C
C
      SUBROUTINE NXCMD
C     REQUEST NEXT COMMAND.  READ TWO CHARACTERS
C     FROM TERMINAL AND RETURN *JCMD* = COMMAND
C     NUMBER IF RECOGNISED, OTHERWISE *JCMD* =
C     NUMBER OF =OPTIONS= COMMAND,  *JNSTAT* =
C     CORRESPONDING NEW DISC FILE STATUS.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /FS/ JCSTAT,JNSTAT,MSTAT(3)
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /SE/ JCMD,KSEGS(7)
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION LTCMD(50),LTRANS(25)
      DATA LNCMDS/25/,LLCA/129/,LLCZ/169/
C     COMMAND NAMES
      DATA LTCMD/
     *8HDELETE  ,8HEDIT    ,8HINSERT  ,8HBUILD   ,
     *8HADD     ,8HDISPLAY ,8HBRAILLE ,8HFORMAT  ,
     *8HEMBOSS  ,8HSAVE    ,8HRESTORE ,8HERASE   ,
     *8HFILES   ,8HCLEAR   ,8HMODE*   ,8HSTOP    ,
     *8HSUMMARY ,8HSCALE   ,8HPLAY    ,8HCHANGE  ,
     *8HPRINT   ,8HSQUEEZE ,8HCOPY    ,8HFLAG    ,
     *8HOPTIONS /
C     DISC FILE STATUS TRANSITION TABLE
      DATA LTRANS/
     *8Z00030300,8Z01030300,8Z00030300,8Z03030000,
     *8Z00030300,8Z00020300,8Z00020300,8Z0C020300,
     *8Z00020300,8Z02020200,8Z02020000,8Z01020300,
     *8Z01020300,8Z00010100,8Z01020300,8Z01020000,
     *8Z01020300,8Z01020300,8Z00020300,8Z01020300,
     *8Z00020300,8Z01020300,8Z01020300,8Z01020300,
     *8Z01020300/
C
C     DISPLAY =NEXT COMMAND= QUESTION
    2 CALL TLMESS(61)
      L = 0
    3 L = L + 1
C     READ CHARACTER
      CALL NOECHO
```

```
C     *C/R* GIVES INVALID COMMAND IMMEDIATELY
      IF (MCHAR(1) .EQ. C) GOTO 6
      L1 = NBYTE(MCHAR,K1)
C     CONVERT LOWER CASE LETTER TO UPPER CASE
      IF (L1 .GE. LLCA .AND. L1 .LE. LLCZ)
     *          CALL STBYTE(L1+64,MCHAR,K1)
C     REFLECT CHARACTER ON SCREEN
      CALL ECHO
C     CANCEL COMMAND IF CANCEL CHARACTER WAS TYPED
      IF (MCHAR(1) .EQ. KQERY(1)) GOTO 2
C     SAVE CHARACTER FOR COMMAND IDENTIFICATION
      CALL MBS(MCHAR,K1,MTEXT,L,K1)
C     READ ANOTHER CHARACTER IF ONLY ONE READ YET
      IF (L .LT. 2) GOTO 3
C     SEARCH LIST OF RECOGNISED COMMAND OPTIONS
      DO 50 L=1,LNCMDS
      LT = 8 * (L-1)
      DO 40 LL=1,2
   4C IF (NBYTE(MTEXT,LL) .NE. NBYTE(LTCMD,LT+LL))
     *                          GOTO 5
      GOTO 8
    5 CONTINUE
   5C CONTINUE
C     INVALID COMMAND, SIMULATE OPTIONS COMMAND
    6 JCMD = LNCMDS
      GOTO 85
C     VALID COMMAND
    8 JCMD = L
C     DISPLAY REST OF COMMAND NAME
      CALL MBS(LTCMD,8*JCMD-5,MTEXT,K1,K6)
      CALL TWT(K6,MTEXT)
C     SET DISC FILE STATUS FOR SUCCESSFUL COMMAND
   85 JNSTAT = NBYTE(LTRANS,4*(JCMD-1)+JCSTAT)
      RETURN
      END
C
C
      SUBROUTINE CMDSET
C     GENERAL INITIALISATION FOR NEW COMMAND.
      COMMON MC1(188),NILENG,MC2(64),NOLENG,
     *   MC3(66),JBBUF,NFLAGS,MFLAGS(3)
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /EE/ JEDIT,JEFREE,JEBACK,ISITEM
      COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSQZ,
     * ISREST,ISAHED,ISXPOS,ISEMB
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FI/ ISWRIT,ISSCAN
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /KB/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      DIMENSION MFLAG(6)
      EQUIVALENCE(MFLAG,ISRPT)
      DATA LELIST/3/,LESIZE/50/
      NILENG = 0
      NOLENG = 0
      DO 20 L=1,6
   20 MFLAG(L) = 0
      ITERR = 0
      ISCHEK = 1
      INIT = 1
      ISXPOS = 0
      ISHDR = 0
      ISSCAN = 0
      ISSQZ = 0
      ISAHED = 0
      JBBUF = 0
      JBAR = 0
      MFLAGS(2) = 0
      MFLAGS(3) = 0
      ISJOY = 0
      JBS = 1
      JBE = NBARS
C     CLEAR ALL LISTS OF EDIT TABLE
      DO 30 L=1,LELIST
```

```
   30 CALL SLINK(L,L)
C     SET UP FREE CHAIN IN EDIT TABLE
      JEFREE = LELIST + 1
      LL = LESIZE - 1
      DO 50 L=JEFREE,LL
   50 CALL SLINK(L,L+1)
      CALL SLINK(LESIZE,K0)
C     SET ALL STAVES TO UNSELECTED
      DO 60 L=1,40
   60 CALL STBYTE(K0,MSSEL,L)
      RETURN
      END
C
C
      SUBROUTINE CMDEND
C     FINISH PROCESSING CURRENT COMMAND.
      COMMON MC1(180),NINBUF,NOPBUF,JINSEC,JOPSEC
      COMMON /BY/ NBYTES,MBYTES(3),NSECTS,NVERS
      COMMON /B1/ IFIRST,JOPAGE
      COMMON /CB/ IXQSAV,IYQSAV,IHQSAV,MXCOM(1)
      COMMON /ED/ NEDBAR,NEDSTV,NELENG,JEDCMD
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /FS/ JCSTAT,JNSTAT,MSTAT(3)
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /LO/ NLOCS,JLOCN
      COMMON /MC/ KBSIZE,KWSIZE,KXMAX,KYMAX,KFSIZE
      COMMON /MO/ JIMODE,JOMODE
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /PL/ ISPLOT
      COMMON /PP/ MPP1(3),ISHQD,MPP2(3)
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /SE/ JCMD,KSEGS(7)
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /TL/ NNAME,MNAME(8)
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION LCOLON(1)
      DATA LCOLON/1H:/
C     EXIT IF NO CURRENT COMMAND
      IF (JCMD) 42,88,16
   16 IF (JCMD .NE. 23) GOTO 18
C     COPY COMMAND CORRUPTS *MINBUF* AND *MOPBUF*
C     SO MARK THESE OUT OF ACTION
      NINBUF = -1
      NOPBUF = -1
   18 IF (JCMD .NE. 21) GOTO 2
C     RESTORE TERMINAL PARAMETERS AFTER PLOTTING
      ISPLOT = 0
      JYMAX = KYMAX
      IXQ = IXQSAV
      IYQ = IYQSAV
      ISHQD = IHQSAV
C     ACKNOWLEDGE TERMINAL INTERRUPT IF ANY
C     UNLESS EXECUTING DISPLAY COMMAND
    2 IF (INIT .NE. 0 .AND. JCMD .NE. 6)
     *                          CALL XMESIJ(40)
      IF (ISERR .EQ. 0) GOTO 4
C     DISPLAY ANY MESSAGE RESULTING FROM LAST CMD
      CALL XMESIJ(40+ISERR)
C     INHIBIT COMMAND COMPLETION AFTER TERMINAL
C     INTERRUPT EXCEPT FOR -BRAILLE- AND -FORMAT-
      IF (JCMD .NE. 7 .AND. JCMD .NE. 8) GOTO 88
C     FINISH PROCESSING LAST COMMAND IF NO ERRORS
    4 GOTO (7,6,62,8,8,88,74,73,76,45,45,45,88,
     * 82,82,88,88,84,5,88,55,88,47,88,88,43),JCMD
      GOTO 88
C
C     PART 1 OF PROGRAM INITIALISATION
   42 CALL SETUPA
      JCMD = 26
      GOTO 9
C
C     END OF INITIALISATION UNLESS INTERRUPTED
   43 IF (INIT .NE. 0) GOTO 9
      IF (JCSTAT .EQ. 1) GOTO 88
      JOPSEC = JFREE
```

```
      JTMIN = 512 * JTIME1 / JTIME2
      JTMAX = JTMIN
      GOTO 88
C
C     SAVE, RESTORE, ERASE
   45 IF (INIT .NE. 0) GOTO 88
C     DISPLAY FILE X-D MESSAGE WITH NAME AND
C     VERSION NUMBER
      CALL TLMESS(62)
      CALL XMESIJ(71+JCMD)
      CALL XMESIJ(84)
      CALL TWT(NNAME,MNAME)
      CALL XMESIJ(85)
      CALL TWT1(NVERS)
      IF (JCMD .LT. 10 .OR. JCMD .GT. 11) GOTO 88
C     DISPLAY NUMBER OF RECORDS COPIED
      CALL TWT(K1,LCELON)
      CALL TWT1(NSECTS)
      L1 = 86
      GOTO 87
C
C     COPY CMD - DISPLAY -TAPE COPIED CORRECTLY-
   47 L1 = 65
      GOTO 87
C
C     PLAY CMD - DISPLAY -PLAYING COMPLETE-
    5 L1 = 66
      GOTO 87
C
C     PRINT COMMAND
   55 IF (JIMODE .EQ. 1) CALL PREND
      GOTO 88
C
C     EDIT COMMAND
    6 IF (ISINPT .NE. 0) GOTO 82
C     NO MEASURE REPLACEMENT IF EDITING HEADER
      IF (NIB .EQ. 0) GOTO 72
      GOTO 7
C     INSERT COMMAND
   62 NLOCS = 0
      NOPBUF = -1
      CALL BFIND(NEDBAR,NEDSTV)
C     UPDATE DISC FILE FOR INSERT COMMAND
      CALL FEDIT
      CALL SECURE
      NEDSTV = NEDSTV + 1
C     REPEAT INSERT COMMAND UNLESS LAST STAVE
      IF (NEDSTV .LE. NLINES) GOTO 9
      NBARS = NBARS + 1
      GOTO 72
    7 IF (INIT .NE. 0) GOTO 88
C     UPDATE FILE FOR EDIT AND DELETE COMMANDS
      CALL FEDIT
C     SECURE DISC FILE FOR CMDS WHICH CHANGE IT
   72 CALL SECURE
C     CHECK AMOUNT OF FREE DISC SPACE LEFT
      CALL SCHECK
      GOTO 82
C     FORMAT COMMAND
   73 IF (JOMODE .NE. 3) GOTO 84
C     SET FORMATTED BRAILLE STATUS TO UNPROTECTED
      MSTAT(JOMODE) = 2
      GOTO 75
C     BRAILLE COMMAND, SOUND BELL TO WAKE UP USER
   74 CALL REJECT
C     ADD END-OF-SECTION RECORD
   75 CALL ENDSEC
C     RESTORE ORIGINAL *NBARS* FOR CHANGE OF MODE
      NBARS = MBARS(JIMODE)
      GOTO 84
C     EMBOSS COMMAND
   76 CALL TWT1(JOPAGE)
C     -BRAILLE PAGES WRITTEN- MESSAGE
      L1 = 109
      GOTO 87
C     BUILD AND ADD COMMANDS - NO ACTION IF
C     INTERRUPTED DURING COMMAND INITIALISATION
    8 IF (INIT .NE. 0) GOTO 88
C     ADD END-OF-SECTION MARKER
      CALL ENDSEC
```

```
C     UPDATE DISC FILE STATUS FOR CURRENT SECTION
   82 JCSTAT = JNSTAT
   84 CALL SECURE
      GOTO 88
C     DISPLAY MESSAGE
   87 CALL XMESIJ(L1)
   88 JCMD = 0
    9 RETURN
      END
C
C
      SUBROUTINE SETLPA
C     PART 1 OF PROGRAM INITIALISATION, EXECUTED
C     ONCE ONLY UNLESS INTERRUPTED.
      COMMON MINBUF(90),MC1(90),NINBUF,NOPBUF,
     * MC2(138),JBBUF
      COMMON /B1/ IFIRST,JOPAGE
      COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSQZ,
     * ISREST,ISAHED,ISXPOS,ISEMB
      COMMON /FG/ KLEVEL,IEA
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FI/ ISWRIT,ISSCAN
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /GP/ JGROUP,MGROUP(7)
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /HE/ MCLEF(10),MPART(10)
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MC/ KBSIZE,KWSIZE,KXMAX,KYMAX,KFSIZE
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /M0/ JIMODE,JOMODE
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /PL/ ISPLOT
      COMMON /PP/ MPARAM(7)
      COMMON /QL/ IXG,IYG,IXGMIN,IGMAX,JXMAX,JYMAX
      COMMON /TL/ NNAME,MNAME(8)
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION MHFILE(14),LMH(5),LDATE(5)
      EQUIVALENCE(MHFILE,KNLIST)
C     NULL HEADER RECORD
      DATA LMH(1)/8Z01118008/,LMH(2)/8Z50505050/,
     *     LMH(3)/8Z50505050/,LMH(4)/8Z50230822/,
     *     LMH(5)/8ZE0217A00/
C     COMPILATION DATE
      DATA LDATE/20H28TH SEPTEMBER 1978 /
C
C     INITIALISE VARIABLES
      NINBUF = -1
      NOPBUF = -1
      ISEMB = 0
      ISPLOT = 0
      JXMAX = KXMAX
      JYMAX = KYMAX
      ISWRIT = 0
      NNAME = 0
      JOPAGE = 0
      JGROUP = 1
      JSCALE = 64
      JBBUF = 0
      ISSCAN = 0
      ISCHEK = 1
C
C     DEFAULT PARAMETERS PENDING USER SELECTION
      DO 20 L1=1,7
   20 MPARAM(L1) = 0
      CALL DPSET
C     CLEAR SCREEN AND DISPLAY INITIAL MESSAGES
      CALL TPAGE(K0)
      CALL XMESIJ(100)
      CALL XMESIJ(101+MINO(KLEVEL,K1))
      CALL XMESIJ(103)
      CALL TWT(20,LDATE)
C
C     READ FIRST SECTOR OF FILE AND CHECK IT
      CALL DREAD(K0,K0)
C     LOOK AT FILE TYPE IDENTIFIER FIELD
      CALL MBS(MINBUF,K5,MTEXT,K1,K4)
```

```
C       ASSUME FILE IS OK IF TYPE IDENTIFIER IS
        IF (ISCBS(K4,KTHDR) .NE. 0) GOTO 9
C
C       FILE NOT SET UP, DISPLAY -INITIALISING FILE-
        CALL XMESIJ(105)
C       LINK EACH SECTOR OF DISC FILE TO NEXT ONE
        DO 320 L1=1,KFSIZE
  320   CALL FLINK(L1-1,L1)
C       LINK LAST SECTOR TO OUTER SPACE
        CALL FLINK(KFSIZE,-2)
C       STORE DUMMY FILE HEADER
        L2 = 4 * KNLIST + 2
        DO 330 L1=2,L2
  330   MHFILE(L1) = 0
        NLINES = 1
        DO 340 L1=1,KNLIST
  340   MLINES(L1) = 1
        CALL OPPOSN(K0)
        CALL OPHW(JSCALE)
        DO 350 L1=1,4
  350   CALL OPBYTE(NBYTE(KTHDR,L1))
        CALL STBYTE(K1,MPART,K1)
C       STORE DUMMY PART CODE AND HEADER RECORD
        CALL OPPOSN(14+8*KNLIST)
        DO 360 L1=1,19
  360   CALL OPBYTE(NBYTE(LMH,L1))
        JIMODE = 1
        NBARS = 0
        NLINES = 1
        CALL SECURE
    9   RETURN
        END
C
C
        SUBROUTINE FEDIT
C       UPDATE DISC FILE FOR EDIT, INSERT AND
C       DELETE COMMANDS.
        COMMON MINBUF(90),MOPBUF(90),NINBUF,NOPBUF,
     * JINSEC,JOPSEC,JINBUF,JOPBUF,JINLK,JOPLK,
     * NILENG,MIBAR(64),NOLENG,MOBAR(64),JIBAR,
     * JOBAR,JBBUF,NFLAGS,MFLAGS(3)
        COMMON /AD/ JIADDR,JOADDR
        COMMON /BF/ KBFSIZ,KBFWDS,KDATA
        COMMON /BY/ NBYTES,MBYTES(3),NSECTS,NVERS
        COMMON /B1/ IFIRST,JOPAGE
        COMMON /ED/ NEDBAR,NEDSTV,NELENG,JEDCMD
        COMMON /EE/ JEDIT,JEFREE,JEBACK,ISITEM
        COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSGZ,
     * ISREST,ISAHED,ISXPOS,ISEMB
        COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
        COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *    ISERR,ISNXST,ISINPT
        COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
        COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
        COMMON /LO/ NLOCS,JLOCN
        COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
        COMMON /MO/ JIMODE,JOMODE
        COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
        COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NCP,NDL
        COMMON /SE/ JCMD,KSEGS(7)
        COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
        COMMON /WS/ L1,L2,L3,JBYTE
C       RELEVANT COMMAND NUMBERS
        EQUIVALENCE(KDELET,K1),(KEDIT,K2),
     *                         (KINSRT,K3)
C
C       UPDATE FLAG FIELD OF HEADER RECORD IF THE
C       FIRST MEASURE OF PRINTED MUSIC WAS CHANGED
        IF (JIMODE .NE. 1 .OR. NEDBAR .NE. 1 .OR.
     *                 JLINE .NE. 1) GOTO 2
        IFIRST = 0
        IF (JTIME .GE. 512*JTIME1/JTIME2) IFIRST = 1
C       POSITION FILE TO FLAG FIELD OF HEADER RECORD
        CALL OPPOSN(JHDR+1)
C       STORE FIRST-MEASURE-FULL INDICATOR
        CALL OPBYTE(128+IFIRST)
    2   IF (JCMD .EQ. KEDIT .AND. NILENG+NFLAGS+1
```

```
     *                    .LE. NELENG) GOTO 8
C      ENSURE BACKWARD FILE LINK SET CORRECTLY
       CALL SECURE
       LEIN1 = JIADDR / KDATA
       LNCOPY = JIADDR - KDATA * LEIN1
       CALL LCLEAR(LEIN1)
       CALL BFIND(NEDBAR,NEDSTV)
       LBKPTR = JEBACK
C      SET INPUT ADDRESS TO START OF SECTOR IN
C      WHICH ORIGINAL OF EDITED MEASURE BEGINS
       CALL INPOSN(KDATA*LEIN1)
       LEOUT1 = JFREE
C      SET OUTPUT ADDRESS TO START OF FREE LIST
       CALL OPPOSN(KDATA*LEOUT1)
       IF (LNCOPY .LE. 0) GOTO 4
C      COPY UP TO START OF SPECIFIED MEASURE
       DO 30 L=1,LNCOPY
    30 CALL OPBYTE(INBYTE(KO))
     4 IF (JCMD .EQ. KDELET) GOTO 42
C      STORE INSERTED OR EDITED MEASURE
       CALL STLINE(ICFLAG)
       LL = 1
       GOTO 43
C      DECREMENT BAR COUNT FOR DELETE COMMAND
    42 NBARS = NBARS - (JBE-JBS+1)
       LL = (JBE-JBS+1) * NLINES
C      SKIP OVER DELETED MEASURE OR MEASURES
    43 NIS = 0
       ISCHEK = 0
       DO 440 L=1,LL
   440 CALL GETBAR
       IF (JCMD .NE. KINSRT) GOTO 46
C      RE-STORE MEASURE FOLLOWING INSERT
       CALL BMOVE(MIBAR,MEBAR)
       CALL STLINE(MFLAGS(1))
    46 LEIN2 = JINSEC
       LELINK = JINLK
C      COPY REST OF LAST EDITED SECTOR
    48 LL = INBYTE(K1)
       IF (LL .EQ. 0 .OR. JINBUF .LE. 1) GOTO 5
       CALL OPBYTE(LL)
       GOTO 48
C      PAD OUT LAST OUTPUT SECTOR WITH ZEROS
     5 LNCOPY = KBFSIZ - JOPBUF
       IF (LNCOPY .LE. 0) GOTO 7
       DO 60 L=1,LNCOPY
    60 CALL OPBYTE(KO)
C      CLEAR INVALID ENTRIES IN LOCATION TABLE
     7 IF (JCMD .EQ. KEDIT) GOTO 72
C      CLEAR WHOLE TABLE FOR INSERT AND DELETE CMDS
       NLOCS = 0
       GOTO 75
C      CLEAR SECTOR CONTAINING START OF ORIGINAL
C      MEASURE FOR EDIT COMMAND
    72 CALL LCLEAR(LEIN1)
C      AND SECTOR CONTAINING END OF IT IF DIFFERENT
       IF (LEIN1 .NE. LEIN2) CALL LCLEAR(LEIN2)
C      RESET FILE LINKS
    75 LEOUT2 = NOPBUF
       CALL FLINK(LEOUT2,LELINK)
       IF (LBKPTR .EQ. 0) GOTO 76
       CALL FLINK(LBKPTR,LEOUT1)
       GOTO 77
    76 MHEAD(JIMODE) = LEOUT1
    77 IF (LEIN2 .EQ. MTAIL(JIMODE))
      *                 MTAIL(JIMODE) = LEOUT2
       CALL FLINK(LEIN2,JFREE)
C      SET NEW HEAD OF FREE STORAGE CHAIN
       JFREE = LEIN1
       GOTO 85
C
C      REPLACE EDITED MEASURE IN ORIGINAL POSITION
     8 CALL OPPOSN(JIADDR)
       CALL STLINE(ICFLAG)
C      SET START ADDRESS OF FOLLOWING MEASURE
       CALL GETBAR
C      PAD OUT ANY GAP CREATED WITH -IGNORE- CODES
    82 IF (KDATA*JOPSEC+JOPBUF .EQ. JIADDR) GOTO 85
       CALL OPBYTE(255)
       GOTO 82
```

```
C
C     DISPLAY -FILE UPDATED- MESSAGE
   85 CALL XMESIJ(67)
      RETURN
      END
C
C
      SUBROUTINE LCLEAR(N)
C     DELETE ALL LOCATION TABLE ENTRIES WITH FILE
C     ADDRESSES IN SECTOR *N*.
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /LO/ NLOCS,JLOCN
      COMMON /LT/ MLOCN(10),MCTK(5),MFIND(10)
      COMMON /WS/ L1,L2,L3,JBYTE
C     EXIT IF TABLE ALREADY EMPTY
      IF (NLOCS .LE. 0) GOTO 9
      JLOCN = 0
      DO 50 L1=1,NLOCS
      IF (MLOCN(L1)/KDATA .EQ. N) GOTO 5
C     COPY TABLE ENTRY
      JLOCN = JLOCN + 1
      MFIND(JLOCN) = MFIND(L1)
      MLOCN(JLOCN) = MLOCN(L1)
      CALL STHW(NHW(MCTK,L1),MCTK,JLOCN)
    5 CONTINUE
   50 CONTINUE
      NLOCS = JLOCN
    9 RETURN
      END
C
C
      SUBROUTINE ENDSEC
C     STORE END-OF-SECTION MARKER.
      COMMON MC1(253),NOLENG
      NOLENG = 0
      CALL STLINE(-1)
      RETURN
      END
C
C
      SUBROUTINE SCHECK
C     CHECK IF ENOUGH SPACE LEFT IN FILE, ATTEMPT
C     SQUEEZE TO RECOVER UNUSED SPACE IF NOT.
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /MC/ KBSIZE,KWSIZE,KXMAX,KYMAX,KFSIZE
      ISERR = 0
      IF (JFREE .LT. KFSIZE) GOTO 9
C     SQUEEZE FILE IF NO ROOM LEFT
      CALL SQEEZE
      IF (JFREE .LE. KFSIZE) GOTO 9
C     ERROR IF STILL NO ROOM LEFT
      ISERR = 1
C     DISPLAY -SPACE JAM- MESSAGE
      CALL XMESIJ(68)
    9 RETURN
      END
C
C
      SUBROUTINE SQEEZE
C     SQUEEZE FILE - RECOVER UNUSED SPACE CREATED
C     BY THE EDITOR.
      COMMON MINBUF(90),MOPBUF(90),NINBUF,NOPBUF,
     * JINSEC,JOPSEC,JINBUF,JOPBUF,JINLK,JOPLK,
     * NILENG,MIBAR(64),NOLENG,MOBAR(64),JIBAR,
     * JOBAR,JBBUF,NFLAGS,MFLAGS(3)
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /BY/ NBYTES,MBYTES(3),NSECTS,NVERS
      COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSQZ,
     * ISREST,ISAHED,ISXPOS,ISEMB
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /KB/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /LO/ NLOCS,JLOCN
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /MO/ JIMODE,JOMODE
```

```
      COMMON /SE/ JCMD,KSEGS(7)
C     DISPLAY -SQUEEZING- MESSAGE
      CALL XMESIJ(11C)
C     SET FLAG TO PREVENT UNWANTED ACTION IN
C     *STLINE* AND *CHMODE* ROUTINES
      ISSQZ = 1
C     CLEAR LOCATION TABLE
      NLOCS = 0
      LOMODE = JIMODE
C     SQUEEZE EACH SECTION OF THE DISC FILE
      DO 50 L=1,KNLIST
      MBYTES(L) = 0
C     NO ACTION IF THE SECTION IS EMPTY
      IF (MHEAD(L) .LE. 0) GOTO 5
C     SET OUTPUT POSITION TO START OF SECTION
      LSTART = KDATA * MHEAD(L)
      CALL OPPOSN(LSTART)
      JOMODE = L
      CALL CHMODE
      JBE = NBARS
      CALL GETHDR
C     SKIP BACK OVER HEADER LINES IN MODE U
      IF (JIMODE .EQ. 2) CALL INPOSN(LSTART)
C     FETCH NEXT MEASURE
    3 CALL GETBAR
      NOLENG = 0
C     TRANSFER MEASURE TO OUTPUT BUFFER
      CALL BMOVE(MIBAR,MOBAR)
C     STORE MEASURE BACK IN FILE
      CALL STLINE(MFLAGS(1))
      MBYTES(L) = MBYTES(L) + NBYTES
      IF (ISEND .EQ. 0) GOTO 3
C     ADD END-OF-SECTION MARKER FOR SAFETY
      CALL ENDSEC
      MBYTES(L) = MBYTES(L) + NBYTES
      IF (JCMD .NE. 22) GOTO 4
C     DISPLAY NUMBER OF SECTORS FOR -SQUEEZE- CMD
      LSECTS = (MBYTES(L) - 1) / (KBFSIZ-2) + 1
      CALL TWT1(LSECTS)
C     DISPLAY -RECORDS-
      CALL XMESIJ(86)
    4 LTAIL = JOPSEC
      IF (LTAIL .EQ. MTAIL(L)) GOTO 5
C     RESTORE FREED SECTORS TO FREE CHAIN
      LNFREE = JOPLK
      CALL FLINK(MTAIL(L),JFREE)
      JFREE = LNFREE
      MTAIL(L) = LTAIL
    5 CONTINUE
   50 CONTINUE
C     RESTORE CONTEXT FOR CURRENT MODE
      JOMODE = LOMODE
      CALL CHMODE
      CALL SECURE
      ISSQZ = 0
      RETURN
      END
C
C
      SUBROUTINE CHMODE
C     SETS CURRENT MODE TO *JOMODE* IF DIFFERENT.
C     *JOMODE* = 1 FOR PRINTED MUSIC, 2 FOR
C     UNFORMATTED BRAILLE, 3 FOR FORMATTED BRL.
      COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSQZ,
     * ISREST,ISAHED,ISXPOS,ISEMB
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /FS/ JCSTAT,JNSTAT,MSTAT(3)
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /LO/ NLOCS,JLOCN
      COMMON /MO/ JIMODE,JOMODE
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NCP,NDL
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /WS/ L1,L2,L3,TBYTE
      DIMENSION LSEMIC(1),LCOMMA(1),LLETS(1)
      DATA LSEMIC/1H;/,LCOMMA/1H,/,LLETS/1HS/
C
```

```
C     SAVE CONTEXT FOR CURRENT MODE
      MSTAT(JIMODE) = JCSTAT
      MBARS(JIMODE) = NBARS
      MLINES(JIMODE) = NLINES
C     NO FURTHER ACTION IF ALREADY IN CORRECT MODE
      IF (JIMODE .EQ. JOMODE) GOTO 9
C     CLEAR LOCATION TABLE
      NLOCS = 0
C     SET NEW CURRENT MODE
      JIMODE = JOMODE
C     FETCH CONTEXT FOR NEW MODE
      JCSTAT = MSTAT(JIMODE)
      NBARS = MBARS(JIMODE)
      NLINES = MLINES(JIMODE)
      IYSTV = MYSTV(JIMODE)
      IF (ISSQZ .NE. 0) GOTO 9
C     DISPLAY NEW MODE AND NUMBER OF BARS/STAVES
      CALL TLMESS(63)
      CALL XMESIJ(57+JIMODE)
      CALL TWT(K1,LSEMIC)
      CALL TWT1(NBARS)
      CALL TWT(K1,KSPACS)
      CALL XMESIJ(50+JIMODE)
      IF (NBARS .NE. 1) CALL TWT(K1,LLETS)
      IF (NLINES .LE. 1 .OR. JIMODE .EQ. 3) GOTO 9
      CALL TWT(K1,LCOMMA)
      CALL TWT1(NLINES)
C     DISPLAY -STAVES-
      CALL XMESIJ(87)
    9 RETURN
      END
!ASS (M:B0,D1,JHROM22)
!FORTRAN B0,NS
C
C
C     ******************************************
C     *                                        *
C     *    SEGMENT 22 - COMMAND INITIALISATION *
C     *                                        *
C     ******************************************
C
C
      SUBROUTINE SETUPB
C     PART 2 OF PROGRAM INITIALISATION, EXECUTED
C     ONCE ONLY UNLESS INTERRUPTED.
      COMMON /FG/ KLEVEL,IEA
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /PP/ ISPOSE,JTERM,ISDBUG,ISHQD,
     * ISGRD1,ISSAG,ISWIDE
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION MPARAM(7)
      EQUIVALENCE(MPARAM,ISPOSE)
      CALL DPSET
C     REQUEST SELECTION NUMBER
      JBYTE = NQUEST(106,K0,127)
C     SET SELECTED ADJUSTABLE PROGRAM PARAMETERS
      DO 30 L1=1,7
   30 MPARAM(8-L1) = NBIT(L1)
C     SET PARAMETERS FOR SELECTED TERMINAL TYPE
      CALL DPSET
C     HIGH QUALITY DISPLAY IN BACKGROUND ONLY
      IF (KLEVEL .NE. 0) ISHQD = 0
C     DISPLAY -PRESS ASCII BUTTON- IF TEKTRONIX
      IF (JTERM .EQ. 0) CALL XMESIJ(104)
      RETURN
      END
C
C
      SUBROUTINE ININIT
C     INITIALISATION FOR MUSIC INPUT.
      COMMON MC1(253),NOLENG
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /HE/ MCLEF(10),MPART(10)
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
```

```
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /MC/ KBSIZE,KWSIZE,KXMAX,KYMAX,KFSIZE
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /MO/ JIMODE,JOMODE
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      COMMON /TY/ ISCONT,ISBLOW
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION LTPINS(52),LTPVOC(5),LKSMAX(3)
      DATA LNPINS/52/,LNPVOC/5/,LKSMAX(1)/40/,
     * LKSMAX(2)/4/,LKSMAX(3)/1/
      DATA LTPINS/
     *4HRH  ,4HLH  ,4HPED ,4HPIC ,4HFL  ,4HAFL ,
     *4HOB  ,4HOBA ,4HEH  ,4HCL  ,4HBCL ,4HBN  ,
     *4HDBN ,4HHORN,4HFLH ,4HTRP ,4HBTRP,4HTRB ,
     *4HBTRB,4HTUB ,4HBTUB,4HCOR ,4HEU  ,4HSAX ,
     *4HKD  ,4HBELL,4HGL  ,4HCEL ,4HDUL ,4HXY  ,
     *4HMAR ,4HMG  ,4HSD  ,4HTD  ,4HBD  ,4HTAB ,
     *4HTAM ,4HTRI ,4HCYM ,4HGONG,4HCAST,4HRAT ,
     *4HANV ,4HWB  ,4HVN1 ,4HVN2 ,4HVL  ,4HVC  ,
     *4HDB  ,4HHARP,4HGUIT,4HACC /
      DATA LTPVOC/
     *4HVOC ,4HSOP ,4HALTO,4HTEN ,4HBASS/
C
      INIT = 1
      ISINPT = 1
      CALL TPAGE(KO)
      CALL DELETE(JIMODE)
C     REQUEST NUMBER OF STAVES
      CALL TNLINE
      NLINES = NQUEST(55,K1,LKSMAX(JIMODE))
      MLINES(JIMODE) = NLINES
      IF (JIMODE .NE. 1) GOTO 7
C     REQUEST PART CODES
    3 DO 60 L=1,NLINES
   32 CALL XMESIJ(69)
      CALL TWT1(L)
      CALL TWT(K2,KQERY)
C     READ ABBREVIATION FOR PART
      DO 350 LL=1,31
   33 CALL TRD
      IF (MCHAR(1) .NE. 0) GOTO 34
      IF (LL .GE. 1) GOTO 37
C     ERROR IF PLAIN CARRIAGE-RETURN WAS TYPED
      CALL REJECT
      GOTO 33
   34 CALL MBS(MCHAR(1),K1,MTEXT,LL,K1)
  350 CONTINUE
      LL = 32
   37 ISBLOW = 0
      NCHARS = LL - 1
      IF (NCHARS-4) 43,45,51
C     ADD SPACE TO PREVENT SUBSTRING MATCH
   43 NCHARS = NCHARS + 1
      CALL STBYTE(64,MTEXT,NCHARS)
C     CHECK FOR INSTRUMENTAL PART
   45 DO 460 L1=1,LNPINS
  460 IF (ISCBS(NCHARS,LTPINS(L1)) .NE. 0) GOTO 54
C     CHECK FOR VOCAL PART
      DO 480 L1=1,LNPVOC
  480 IF (ISCBS(NCHARS,LTPVOC(L1)) .NE. 0) GOTO 52
C     PART ABBREVIATION UNRECOGNISED
   51 CALL REJECT
      GOTO 32
   52 L1 = L1 + 64
C     STORE PART CODE
   54 CALL STBYTE(L1,MPART,L)
   60 CONTINUE
      CALL TNLINE
      JHDR = 14 + 8 * KNLIST + NLINES
      NOB = 0
      NOS = NLINES
      GOTO 8
C     BRAILLE MODE
    7 NOB = 1
      NOS = 1
    8 NOLENG = 0
      NIB = 0
      NIS = 0
```

```
      NIBS = 0
      ISEND = 0
      RETURN
      END
C
C
      SUBROUTINE DELETE(N)
C     DELETE SECTION *N* OF DISC FILE.
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FS/ JCSTAT,JNSTAT,MSTAT(3)
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /L0/ NLOCS,JLOCN
      COMMON /M0/ JIMODE,JOMODE
      COMMON /SE/ JCMD,KSEGS(7)
      COMMON /TL/ NNAME,MNAME(8)
C     NO ACTION IF SECTION IS ALREADY EMPTY
      IF (MSTAT(N) .LE. 1) GOTO 9
C     CLEAR LOCATION TABLE
      NLOCS = 0
C     DELAY ACTUAL DELETION FOR PROTECTED SECTION
      IF (JCSTAT .EQ. 3 .AND. JCMD .EQ. 14) GOTO 6
C     SET LINKS TO RETURN FILE SPACE TO FREE CHAIN
      CALL FLINK(MTAIL(N),JFREE)
      JFREE = MHEAD(N)
      MHEAD(N) = 0
      MTAIL(N) = 0
C     SET SECTION *N* EMPTY
      MBARS(N) = 0
      NBARS = MBARS(JIMODE)
      MSTAT(N) = 1
      JCSTAT = MSTAT(JIMODE)
      IF (N .EQ. 1) NNAME = 0
      CALL SECURE
C     DISPLAY -SECTION DELETED- MESSAGE
      CALL TNLINE
      CALL XMESIJ(57+N)
      CALL XMESIJ(88)
      GOTO 9
C     DISPLAY -PLEASE CONFIRM DELETION- MESSAGE
    6 CALL XMESIJ(89)
C       SET CURRENT SECTION STATUS TO UNPROTECTED
        JCSTAT = 2
    9 RETURN
      END
C
C
      SUBROUTINE QBARS(NMIN)
C     SELECT BAR NUMBER RANGE. *C/R* INPUT MEANS
C     SELECT ALL BARS.  RETURNS *JBS* AND *JBE* =
C     FIRST AND LAST BARS SELECTED.
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      DIMENSION LFROM(2),LTO(1)
      DATA LFROM/5HFROM /,LTO/4H TO /
C     OMIT QUESTION IF ONLY ONE BAR TO CHOOSE FROM
      IF (NBARS .LE. 1) GOTO 5
      CALL TWT(K5,LFROM)
      CALL QBAR(JBS,NMIN)
      IF (JBS .EQ. 0) GOTO 3
      IF (JBS .LT. NBARS) CALL TWT(K4,LTO)
      CALL QBAR(JBE,JBS)
      GOTO 9
C     USE WHOLE SECTION IF NO NUMBER SPECIFIED
    3 JBE = NBARS
      GOTO 7
    5 JBE = 1
    7 JBS = 1
    9 RETURN
      END
C
C
      SUBROUTINE QBAR(NNBAR,NMIN)
C     RETURNS *NNBAR* = BAR NUMBER READ FROM
C     TERMINAL.  *NMIN* = MINIMUM VALUE ALLOWED.
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /M0/ JIMODE,JOMODE
```

```
      NNBAR = NQUEST(50+JIMODE,NMIN,NBARS)
      RETURN
      END
C
C
      SUBROUTINE QSTAV(NNLINE)
C     RETURNS *NNLINE* = STAVE OR LINE NUMBER
C     READ FROM TERMINAL.
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      CALL TWT(K3,KSPACS)
      NNLINE = NQUEST(56,K1,NLINES)
      RETURN
      END
C
C
      SUBROUTINE CHVAL(N,NMIN,NMAX)
C     CHANGE VALUE OF *N* TO NEW VALUE SPECIFIED
C     BY USER, IN RANGE *NMIN* TO *NMAX*.
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
C     DISPLAY -CHANGE FROM- MESSAGE
      CALL XMESIJ(90)
C     DISPLAY OLD VALUE
      CALL TWT1(N)
C     INPUT NEW VALUE
      N = NQUEST(57,NMIN,NMAX)
      RETURN
      END
C
C
      SUBROUTINE RANGE(N,NMIN,NMAX)
C     SELECT ANY COMBINATION OF VALUES IN RANGE
C     *NMIN* TO *NMAX*.  *N* IS INITIAL MESSAGE
C     NUMBER.  *C/R* TYPED SELECTS ALL VALUES
C     IN THE RANGE.
C     RETURNS <NSSEL> = NUMBER OF SELECTED VALUES,
C     <JSSEL> = FIRST SELECTED VALUE, <MSSEL> =
C     0 IN NON-SELECTED BYTE POSITIONS ONLY.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      DIMENSION LTO(1)
      DATA LCOMMA/1H,/,LHYPH/1H-/,LTO/3H TO/
C     OMIT QUESTION IF ONLY ONE VALUE AVAILABLE
      IF (NMAX .EQ. 1) GOTO 83
      LISSEL = 0
C     DISPLAY QUESTION
    2 CALL XMESIJ(N)
   24 LMIN = 0
   25 LNDIG = 0
      LNUMB = 0
C     READ NEXT CHARACTER
    3 CALL NOECHO
C     BACKSPACE ALLOWED ONLY AFTER A DIGIT
      IF (MCHAR(1) .EQ. KQERY(1) .AND.
     *                  LNDIG .GT. 0) GOTO 65
      LL = NBYTE(MCHAR,K1) - 240
      IF (LL .LT. 0 .OR. LL .GT. 9) GOTO 4
C     DIGIT READ
      IF (LNDIG .GE. 5) GOTO 75
      LNUMB = 10 * LNUMB + LL
      LNDIG = LNDIG + 1
      GOTO 7
C     ERROR IF INVALID CHARACTER TYPED
    4 IF ((MCHAR(1) .NE. 0 .AND .MCHAR(1) .NE.
     * LCOMMA .AND. MCHAR(1) .NE. LHYPH)) GOTO 75
C     PLAIN CARRIAGE-RETURN SELECTS ALL VALUES
      IF (MCHAR(1) .EQ. 0 .AND. LISSEL .EQ. C
     *              .AND. LNUMB .EQ. 0) GOTO 83
C     ERROR IF NUMBER TOO BIG OR TOO SMALL
      IF (LNUMB.LT.NMIN .OR. LNUMB.GT.NMAX) GOTO 6
      IF (MCHAR(1) .EQ. 0) GOTO 42
      IF (LNDIG .LE. 0) GOTO 75
      IF (MCHAR(1) .EQ. LHYPH) GOTO 5
C     END OF RANGE OR SINGLE NUMBER
   42 IF (LMIN .EQ. 0) LMIN = LNUMB
C     SHUFFLE LIMITS OF SEQUENCE RANGE INTO ORDER
```

```
      LL = MINO(LMIN,LNUMB)
      LM = MAXO(LMIN,LNUMB)
C     SELECT ALL VALUES IN SEQUENCE RANGE
      DO 450 L=LL,LM
  450 CALL STBYTE(K1,MSSEL,L)
C     PREVENT C/R NOW SELECTING THE LOT
      LISSEL = 1
C     FINISHED IF LAST CHARACTER WAS C/R
      IF (MCHAR(1) .EQ. C) GOTO 8
C     ECHO INPUT CHARACTER ON SCREEN
      CALL ECHO
      GOTO 24
C     HYPHEN BETWEEN TWO NUMBERS SPECIFYING RANGE
    5 IF (LMIN .NE. C) GOTO 75
      LMIN = LNUMB
      CALL ECHO
      GOTO 25
C     DISPLAY =MUST BE IN RANGE= MESSAGE
    6 CALL XMESIJ(64)
      CALL TWT1(NMIN)
      CALL TWT(K3,LTO)
      CALL TWT1(NMAX)
      CALL TWT(K4,KDOT)
      GOTO 2
C     BACKSPACE ONE DIGIT
   65 LNUMB = LNUMB / 10
      LNDIG = LNDIG - 1
C     ECHO INPUT CHARACTER
    7 CALL ECHO
      GOTO 3
C     SOUND BELL TO INDICATE ERROR
   75 CALL REJECT
      GOTO 3
C     COUNT NUMBER OF SELECTED VALUES
    8 NSSEL = 0
      DO 820 L=1,NMAX
      IF (NBYTE(MSSEL,L) .EQ. 0) GOTO 82
      NSSEL = NSSEL + 1
      CALL STBYTE(NSSEL,MSSEL,L)
      IF (NSSEL .EQ. 1) JSSEL = L
   82 CONTINUE
  820 CONTINUE
      IF (NSSEL .GT. 0) GOTO 9
C     SELECT ALL VALUES IF NONE EXPLICITLY CHOSEN
   83 JSSEL = 1
      NSSEL = NMAX
      DO 840 L=1,NMAX
  840 CALL STBYTE(L,MSSEL,L)
    9 RETURN
      END
C
C
      FUNCTION NQUEST(N,NMIN,NMAX)
C     RETURNS *NQUEST* = INTEGER READ FROM THE
C     TERMINAL, WHICH MUST BE IN RANGE *NMIN* TO
C     *NMAX*.  *N* IS NUMBER OF PROMPT MESSAGE.
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      DIMENSION LTO(1)
      DATA LTO/3H TO/
C     OMIT QUESTION IF ONLY ONE VALUE AVAILABLE
      IF (NMAX .EQ. NMIN) GOTO 6
C     DISPLAY QUESTION
    2 CALL XMESIJ(N)
      CALL TWT(K2,KQERY)
C     READ NUMBER FROM TERMINAL
      CALL RDNUMB(NQUEST)
      IF (NQUEST .GE. NMIN .AND. NQUEST .LE. NMAX)
     *                              GOTO 9
C     DISPLAY =MUST BE IN RANGE= MESSAGE
      CALL XMESIJ(64)
      CALL TWT1(NMIN)
      CALL TWT(K3,LTO)
      CALL TWT1(NMAX)
      CALL TWT(K4,KDOT)
      GOTO 2
    6 NQUEST = NMIN
    9 RETURN
      END
C
```

```
C
      SUBROUTINE RDNUMB(N)
C     RETURNS *N* = POSITIVE OR ZERO INTEGER READ
C     FROM TERMINAL.  CARRIAGE-RETURN TERMINATES
C     NUMBER, QUESTION MARK IS TREATED AS BACK-
C     SPACE, ALL OTHER NON-DIGITS ARE REJECTED.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      N = 0
C     READ NEXT CHARACTER
    3 CALL NOECHO
C     EXIT IF CARRIAGE-RETURN TYPED
      IF (MCHAR(1) .EQ. 0) GOTO 9
      IF (MCHAR(1) .EQ. KQERY(1)) GOTO 6
      LL = NBYTE(MCHAR,K1) - 240
C     ERROR IF NON-DIGIT TYPED
      IF (LL .LT. 0 .OR. LL .GT. 9) GOTO 7
      LL = 10 * N + LL
C     ERROR IF NUMBER TOO BIG
      IF (LL .GT. KBIG) GOTO 7
      N = LL
C     ECHO INPUT CHARACTER ON SCREEN
    5 CALL ECHO
      GOTO 3
C     BACKSPACE
    6 N = N / 10
      GOTO 5
C     SOUND BELL TO INDICATE ERROR
    7 CALL REJECT
      GOTO 3
    9 RETURN
      END
C
C
      SUBROUTINE SUMARY
C     DISPLAY SUMMARY INFORMATION FOR CURRENT FILE
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FS/ JCSTAT,JNSTAT,MSTAT(3)
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MO/ JIMODE,JOMODE
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /SE/ JCMD,KSEGS(7)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
C     TABULATION COLUMN NUMBERS
      DIMENSION LHTAB(4)
      DATA LHTAB(1)/5/,LHTAB(2)/23/,LHTAB(3)/32/,
     * LHTAB(4)/37/
C
C     ERASE SCREEN WITHOUT WAITING
      CALL TPAGE(K0)
C     DISPLAY -SUMMARY OF CURRENT DISC FILE-
      CALL XMESIJ(70)
      CALL TNLINE
      LISMUS = 0
C     FOR EACH SECTION OF THE DISC FILE...
      DO 30 L=1,KNLIST
C     NO ACTION FOR EMPTY SECTIONS
      IF (MSTAT(L) .LE. 1) GOTO 3
C     SKIP HEADER LINE IF ALREADY DISPLAYED
      IF (LISMUS .NE. 0) GOTO 25
C     DISPLAY SECTION/STATUS/BARS/STAVES HEADER
      CALL TNLINE
      DO 280 LL=1,4
      IXT = LHTAB(LL) * IXINC
      CALL TPOSN
      CALL XMESIJ(90+LL)
  280 CONTINUE
      CALL TNLINE
C     SET HEADER-LINE-DISPLAYED INDICATOR
      LISMUS = 1
   25 CALL TNLINE
C     DISPLAY SECTION NAME
      CALL XMESIJ(57+L)
C     DISPLAY STATUS, PROTECTED OR UNPROTECTED
      IXT = 21 * IXINC
```

```
      CALL TPOSN
      CALL XMESIJ(93+MSTAT(L))
C     DISPLAY NUMBER OF BARS IN SECTION
      IXT = 33 * IXINC
      CALL TPOSN
      CALL TWT1(MBARS(L))
C     DISPLAY NUMBER OF STAVES IN SECTION
      IXT = 39 * IXINC
      CALL TPOSN
      CALL TWT1(MLINES(L))
    3 CONTINUE
   30 CONTINUE
C     DISPLAY =NO MUSIC STORED= IF NOTHING THERE
      IF (LISMUS .EQ. 0) CALL XMESIJ(31)
C     DISPLAY CURRENT MODE NAME
      CALL TNLINE
      CALL XMESIJ(63)
      CALL XMESIJ(57+JIMODE)
      CALL TNLINE
C     DISPLAY HORIZONTAL DISPLAY SCALE FACTOR
      CALL XMESIJ(71)
      CALL TWT1(JSCALE)
      CALL TNLINE
C     OBTAIN HEADER RECORD FOR PRINTED MUSIC MODE
      IF (JIMODE .EQ. 1) CALL GETHDR
C     SET UPPER MARGIN FOR RIGHT-HAND HALF PAGE
      IQMAX = IYQ
      RETURN
      END
C
C
      SUBROUTINE CHANGE
C     CHANGE PROGRAM PARAMETERS.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /FG/ KLEVEL,IEA
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /MO/ JIMODE,JOMODE
      COMMON /PP/ ISPOSE,JTERM,ISDBUG,ISHOD,
     * ISGRD1,ISSAG,ISWIDE
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      DIMENSION MPARAM(7),LEQALS(1)
      EQUIVALENCE(MPARAM,ISPOSE)
      DATA LNMOD/7/,LLETL/1HL/,LEQALS/3H = /
C
C     DISPLAY =TYPE L FOR OPTION LIST=
      CALL XMESIJ(72)
    2 CALL NOECHO
      IF (MCHAR(1) .EQ. 0) GOTO 4
      IF (MCHAR(1) .EQ. LLETL) GOTO 25
      CALL REJECT
      GOTO 2
C     ECHO INPUT CHARACTER
   25 CALL ECHO
C     DISPLAY =THE CHANGEABLE PARAMETERS ARE=
      CALL XMESIJ(73)
      CALL TNLINE
C     LIST THE PARAMETERS AND CURRENT VALUES
      DO 30 L=1,LNMOD
      CALL TNLINE
      CALL TWT1(L)
      IXT = IXQ + 5 * IXINC
      CALL TPOSN
C     DISPLAY PARAMETER NAME
      CALL XMESIJ(73+L)
C     DISPLAY VALUE, =TRUE= OR =FALSE=
      IXT = IXQ + 15 * IXINC
      CALL TPOSN
      CALL XMESIJ(97+MINO(K1,MPARAM(L)))
   30 CONTINUE
C     SELECT PARAMETERS
    4 CALL TNLINE
      CALL RANGE(108,K1,LNMOD)
C     FOR EACH ADJUSTABLE PARAMETER...
      DO 70 L=1,LNMOD
C     NO ACTION IF PARAMETER NOT SELECTED
      IF (NBYTE(MSSEL,L) .EQ. 0) GOTO 7
C     DISPLAY PARAMETER NAME
```

```
      CALL TPOSN
      CALL XMESIJ(93+MSTAT(L))
C     DISPLAY NUMBER OF BARS IN SECTION
      IXT = 33 * IXINC
      CALL TPOSN
      CALL TWT1(MBARS(L))
C     DISPLAY NUMBER OF STAVES IN SECTION
      IXT = 39 * IXINC
      CALL TPOSN
      CALL TWT1(MLINES(L))
    3 CONTINUE
   30 CONTINUE
C     DISPLAY "NO MUSIC STORED" IF NOTHING THERE
      IF (LISMUS .EQ. 0) CALL XMESIJ(31)
C     DISPLAY CURRENT MODE NAME
      CALL TNLINE
      CALL XMESIJ(63)
      CALL XMESIJ(57+JIMODE)
      CALL TNLINE
C     DISPLAY HORIZONTAL DISPLAY SCALE FACTOR
      CALL XMESIJ(71)
      CALL TWT1(JSCALE)
      CALL TNLINE
C     OBTAIN HEADER RECORD FOR PRINTED MUSIC MODE
      IF (JIMODE .EQ. 1) CALL GETHDR
C     SET UPPER MARGIN FOR RIGHT-HAND HALF PAGE
      IQMAX = IYQ
      RETURN
      END
C
C
      SUBROUTINE CHANGE
C     CHANGE PROGRAM PARAMETERS.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /FG/ KLEVEL,IEA
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /MO/ JIMODE,JOMODE
      COMMON /PP/ ISPOSE,JTERM,ISDBUG,ISHOD,
     * ISGRD1,ISSAG,ISWIDE
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      DIMENSION MPARAM(7),LEQALS(1)
      EQUIVALENCE(MPARAM,ISPOSE)
      DATA LNMOD/7/,LLETL/1HL/,LEQALS/3H = /
C
C     DISPLAY "TYPE L FOR OPTION LIST"
      CALL XMESIJ(72)
    2 CALL NOECHO
      IF (MCHAR(1) .EQ. 0) GOTO 4
      IF (MCHAR(1) .EQ. LLETL) GOTO 25
      CALL REJECT
      GOTO 2
C     ECHO INPUT CHARACTER
   25 CALL ECHO
C     DISPLAY "THE CHANGEABLE PARAMETERS ARE"
      CALL XMESIJ(73)
      CALL TNLINE
C     LIST THE PARAMETERS AND CURRENT VALUES
      DO 30 L=1,LNMOD
      CALL TNLINE
      CALL TWT1(L)
      IXT = IXQ + 5 * IXINC
      CALL TPOSN
C     DISPLAY PARAMETER NAME
      CALL XMESIJ(73+L)
C     DISPLAY VALUE, "TRUE" OR "FALSE"
      IXT = IXQ + 15 * IXINC
      CALL TPOSN
      CALL XMESIJ(97+MINO(K1,MPARAM(L)))
   30 CONTINUE
C     SELECT PARAMETERS
    4 CALL TNLINE
      CALL RANGE(108,K1,LNMOD)
C     FOR EACH ADJUSTABLE PARAMETER...
      DO 70 L=1,LNMOD
C     NO ACTION IF PARAMETER NOT SELECTED
      IF (NBYTE(MSSEL,L) .EQ. 0) GOTO 7
C     DISPLAY PARAMETER NAME
```

```
      CALL TNLINE
      CALL XMESIJ(73+L)
      IXT = IX0 + 9 * IXINC
      CALL TWT(K3,LEGALS)
C     BRANCH INDEXED BY PARAMETER NUMBER
      GOTO (52,54,56,58,52,52,59),L
C     TOGGLE BINARY SWITCH
   52 MPARAM(L) = 1 - MPARAM(L)
      GOTO 6
C     TOGGLE TERMINAL TYPE SWITCH
   54 JTERM = 1 - JTERM
      GOTO 595
C     INPUT NEW PROGRAM DEBUGGING LEVEL
   56 ISDBUG = NQUEST(99,K0,KBIG)
C     PLAIN C/R SWITCHES OFF DEBUGGING MODE
      IF (ISDBUG .EQ. 0) GOTO 6
      GOTO 7
C     TOGGLE HIGH-QUALITY-DISPLAY IF ALLOWED
   58 IF (KLEVEL .EQ. 0) ISHQD = 1 - ISHQD
      GOTO 6
C     CHANGE VERTICAL DISPLAY SPACING
   59 ISWIDE = 1 - ISWIDE
  595 CALL DPSET
C     DISPLAY NEW VALUE, -TRUE- OR -FALSE-
    6 CALL XMESIJ(97+MPARAM(L))
    7 CONTINUE
   70 CONTINUE
    9 RETURN
      END
C
C
      SUBROUTINE OPLIST
C     DISPLAY LIST OF AVAILABLE COMMAND OPTIONS.
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      CALL TPAGE(K0)
      DO 30 L=1,28
   30 CALL XMESIJ(L)
      RETURN
      END
!JOB JH,COMPILE MUSIC PROGRAM SEGMENT 3
!EXTRABGD 6800
!ASS (M:B0,D1,JHROM3)
!FORTRAN B0,NS
C
C
C     ***********************************************
C     *                                             *
C     *  SEGMENT 3 - INPUT, EDITING AND DISPLAY     *
C     *                                             *
C     ***********************************************
C
C
      SUBROUTINE INPUT
C     CREATE OR EXTEND MUSIC IN CURRENT SECTION.
      COMMON /AD/ JIADDR,JOADDR
      COMMON /DN/ JSDIR,JFDIR,JLEV,JALEV,JPVLEV,
     * IXSM,IYSM,IYSMAX,IYSMIN,MXTEXT(3)
      COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSGZ,
     * ISREST,ISAHED,ISXPOS,ISEMB
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NOP,NDL
      COMMON /SE/ JCMD,KSEGS(7)
      COMMON /WS/ L1,L2,L3,JBYTE
      IF (JCMD .EQ. 4) GOTO 3
C     ADD COMMAND
      CALL MDISP
C     SET OUTPUT ADDRESS TO END OF CURRENT SECTION
      CALL OPPOSN(JIADDR)
      ISINPT = 1
      GOTO 4
C     BUILD COMMAND
    3 CALL LSEG(31)
      CALL MINIT
C     INITIALISE DISPLAY
      CALL LSEG(32)
```

```
      CALL DINIT
      JLINE = 0
    4 CALL CHSTAV(K1)
      NDP = NDL / NSSEL
C     PREVENT SPURIOUS END CONDITION ON RETRIEVAL
      JBE = KBIG
      ISEND = 0
C     INPUT NEXT BAR
    5 DO 60 L=1,NLINES
      CALL MINPUT
C     EXIT AT STAVE 1 AFTER -END- ITEM IS INPUT
      IF (L .EQ. 1 .AND. ISEND .NE. 0) GOTO 9
C     STORE MEASURE IN DISC FILE
      CALL STLINE(ICFLAG)
   60 CONTINUE
      GOTO 5
    9 RETURN
      END
C
C
      SUBROUTINE INSERT
C     INSERT NEW MEASURE BEFORE BAR *NEDBAR*.
      COMMON /ED/ NEDBAR,NEDSTV,NELENG,JEDCMD
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      IF (NEDSTV .NE. 1) GOTO 2
C     INITIALISE DISPLAY FIRST TIME ONLY
      CALL LSEG(32)
      CALL DINIT
      NOB = NEDBAR
    2 ISEND = 0
      JLINE = NEDSTV
      CALL MINPUT
      RETURN
      END
C
C
      SUBROUTINE MINPUT
C     IF CURRENT LINE IS A SELECTED LINE INPUT
C     ONE MEASURE OTHERWISE RETURN NULL MEASURE.
      COMMON MC1(189),MIBAR(64),NOLENG,MOBAR(64),
     * MC2(4),MFLAGS(3)
      COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSGZ,
     * ISREST,ISAHED,ISXPOS,ISEMB
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /HE/ MCLEF(10),MPART(10)
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /IV/ INTYPE,ISBAR
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      NOLENG = 0
      ICFLAG = 0
C     RETURN NULL MEASURE IF END-OF-MUSIC HAS BEEN
C     INPUT OR CURRENT STAVE IS NOT SELECTED
      IF (NBYTE(MSSEL,NOS) .EQ. 0 .OR. ISEND
     *                .NE. 0) GOTO 88
C     INITIALISE EDIT TABLE FOR NEW MEASURE
      CALL LSEG(32)
      CALL ESETUP
      JECLEF = JCLEF
      JPART = NBYTE(MPART,NOS)
      CALL MMOVE(MCMEM,MSMEM)
      JPN = 0
C     INPUT AND DISPLAY NEXT ITEM
```

```
    3 CALL IINDIS
      IF (ISEND .NE. 0) GOTO 9
      IF (ISBAR .NE. 0) GOTO 7
      IF (ISEDIT .NE. 0) GOTO 4
C     ADD NEW ITEM TO END OF CURRENT MEASURE
      CALL IINSRT(K1)
      IF (ITTYPE .NE. 4) GOTO 3
C     SAVE POINTER TO NON-REST NOTE ITEM
      IF (JPITCH .NE. 0) JPN = NBLINK(K1)
C     INCREMENT CURRENT TIME COUNT
      JTIME = JTIME + JNDUR
      GOTO 3
C
C     ENTER EDITING-DURING-INPUT MODE
    4 LXN = IXN
      LXP = IXP
C     PERFORM EDITING
      CALL EDIT
C     COPY CURRENT MEASURE BACK TO *MOBAR*
      CALL BMOVE(MIBAR,MOBAR)
C     RESTORE SCREEN POSITION AS BEFORE EDITING
      IXN = LXN
      IXP = LXP
      GOTO 3
C
C     END OF MEASURE
    7 IF (ISRPT .NE. 0) GOTO 8
C     SET CORRECT VALUE FOR FLAGS FIELD
      CALL BMOVE(MOBAR,MIBAR)
      CALL LSEG(32)
      CALL LSEG(321)
      CALL SFLAG
C     SORT ITEMS IN *MOBAR* INTO THE CORRECT ORDER
      CALL UPDATE
C     DISPLAY SINGLE BAR LINE FOR END OF MEASURE
      CALL DEBAR
      GOTO 85
C     EDIT AND DISPLAY REPEATED MEASURE
    8 CALL LSEG(314)
      CALL EDIBOB
      CALL BMOVE(MOBAR,MIBAR)
      CALL LSEG(32)
      CALL DIBAR
C     KEEP SAME FLAGS FIELD AS ORIGINAL MEASURE
      ICFLAG = MFLAGS(1)
      ISRPT = ISRPT - 1
   85 ISEND = 0
   88 CALL BAR
    9 RETURN
      END
C
C
      SUBROUTINE IINDIS
C     INPUT AND DISPLAY ITEM.
      COMMON /RP/ ISAUTO,JEDCOD,JESEL,ISDISP
C     INPUT ITEM
      CALL LSEG(31)
      CALL IINPUT
      CALL BREAK(&9)
C     EXIT IF DISPLAY INDICATOR NOT SET
      IF (ISDISP .EQ. 0) GOTO 9
C     DISPLAY CURRENT ITEM
      CALL LSEG(32)
      CALL IDISP
    9 RETURN
      END
C
C
      SUBROUTINE MDISP
C     DISPLAY CURRENT SECTION FROM BAR *JBS*
C     TO BAR *JBE*.
      CALL LSEG(32)
      CALL MDISP1
      RETURN
      END
C
C
      SUBROUTINE EDIT
C     EDIT CURRENT MEASURE.
      COMMON MINBUF(90),MOPBUF(90),NINBUF,NOPBUF,
```

```
     * JINSEC,J0PSEC,JINBUF,J0PBUF,JINLK,J0PLK,
     * NILENG,MIBAR(64),N0LENG,M0BAR(64),JIBAR,
     * J0BAR,JBBUF,NFLAGS,MFLAGS(3)
      C0MM0N /CH/ MCHAR(1),INC0DE,KE0D,ISJ0Y,JX,JY
      C0MM0N /DN/ JSCIR,JFDIR,JLEV,JALEV,JPVLEV,
     * IXSM,IYSM,IYSMAX,IYSMIN,MXTEXT(3)
      C0MM0N /ED/ NEDBAR,NEDSTV,NELENG,JEDCMD
      C0MM0N /EE/ JEDIT,JEFREE,JEBACK,ISITEM
      C0MM0N /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      C0MM0N /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      C0MM0N /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      C0MM0N /IU/ JCLEF,JTYME,JKEY,NGR0UP,NPAGE,
     * NLINE,NBEAM,NSCALE
      C0MM0N /JD/ JSTAFF,JDLINE,JDHIGH,JDL0W
      C0MM0N /J0/ JDIST
      C0MM0N /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      C0MM0N /KT/ KSPACS(2),K0ERY(1),KD0T(1)
      C0MM0N /LK/ MBLINK(14),MFLINK(14),
     * MLIST1(14),MLIST2(14),MLIST3(14),MLIST4(14)
      C0MM0N /MC/ KBSIZE,KWSIZE,KXMAX,KYMAX,KFSIZE
      C0MM0N /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      C0MM0N /M0/ JIM0DE,J0M0DE
      C0MM0N /NB/ JBAR,NIB,N0B,NIS,N0S,NIBS
      C0MM0N /ND/ NXLENG,NSHIFT,JBXL,NEXTX
      C0MM0N /NS/ NLINES,JLINE,JPART,JV0IC,NDP,NDL
      C0MM0N /PV/ JPFCH,JPLEN,NNINCB,IXP,JREPT,JPN
      C0MM0N /QL/ IXG,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      C0MM0N /RP/ ISAUT0,JEDC0D,JESEL,ISDISP
      C0MM0N /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      C0MM0N /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
      C0MM0N /TD/ IXC,IYC,IXB,IXR
      C0MM0N /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      C0MM0N /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NKBTES
      C0MM0N /WS/ L1,L2,L3,JBYTE
C
C     EDIT C0MMANDS
C      A    INSERT ITEM AFTER SELECTED ITEM
C      B    INSERT ITEM BEF0RE SELECTED ITEM
C      C    CHANGE SELECTED ITEM
C      D    DELETE SELECTED ITEM
C      E    EXAMINE CR0SSWIRE STATUS
C      F    FRESH DISPLAY
C      G    SELECT ITEM
C      M    0VERRIDE TIME C0UNT CHECK F0R MEASURE
C      X    CR0SSWIRES SWITCH
C      QUERY              END EDIT, D0NT UPDATE FILE
C      EXCLAMATI0N MARK  END EDIT, UPDATE FILE
       EQUIVALENCE(KDLC0M,K2)
       DIMENSION LECHRS(3),LMBL(3),LSTAVE(2),
      * LBFULL(2),LTCERR(4),LFUPD(5)
       DATA LNECHR/11/,LECHRS/11HGDCBAFE!?MX/
       DATA LSTAVE/7H  STAVE/,LBFULL/8HBAR FULL/,
      * LTCERR/16HTIME C0UNT ERR0R/,
      * LFUPD/19HFILE HEADER UPDATED/
       DATA LMBL/4HBAR ,4HBAR ,4HLINE/
C
       LISED = 0
       IF (ISINPT .NE. 0) G0T0 28
C      SET HEADER INDICAT0R IF EDITING HEADER
       IF (NIB .EQ. 0) ISHDR = 1
       LTIME = -1
C      WAIT F0R USER C/R IF ANY ITEM ERR0RS
     2 LWAIT = ITERR
C      CLEAR SCREEN WITH 0R WITH0UT WAIT
    22 CALL TPAGE(LWAIT)
       IF (ISHDR .EQ. 0) G0T0 23
C      EDIT FILE HEADER
       CALL LSEG(32)
       CALL ESETUP
       G0T0 3
C
C      EDIT CURRENT MEASURE
    23 CALL TMESS(K4,LMBL(JIM0DE))
       CALL TWT1(NEDBAR)
       IF (JIM0DE.NE.1 .0R. NLINES.EQ.1) G0T0 24
C      DISPLAY STAVE NUMBER
       CALL TWT(K7,LSTAVE)
```

```
      CALL TWT1(JLINE)
   24 IYS = 500
      NEXTX = 0
      IXQMIN = 0
      JBAR = NEDBAR
      JCLEF = JECLEF
      JSTAFF = 1
      JFDIR = 3
C     DISPLAY STAVE
      CALL LSEG(32)
      CALL LSEG(322)
      CALL DSTAVE(JLINE)
      ISCHEK = 1
C     SET STATE AS AT START OF MEASURE
      JCLEF = JECLEF
      CALL MMOVE(MSMEM,MCMEM)
C     SET UP EDIT TABLE AND DISPLAY ITEMS
      CALL ESETUP
C     DISPLAY ANY UNCLOSED EXTENDED SIGNS AT END
      CALL LSEG(322)
      CALL ESTAFF
C     SAVE TOTAL TIME COUNT AS AT START OF EDITING
      IF (LTIME .LT. 0) LTIME = JTIME
      JTMIN = LTIME
      JTMAX = LTIME
      GOTO 3
C
C     EDITING DURING INPUT
   28 CALL BMOVE(MOBAR,MIBAR)
      JIBAR = 0
C
C     DISPLAY CROSSWIRES AND INPUT EDIT COMMAND
    3 ISITEM = 0
   31 MCHAR(1) = 0
      CALL JOY(MCHAR,JX,JY)
C     IDENTIFY EDIT-COMMAND NUMBER
      L1 = NBYTE(MCHAR,K1)
      DO 320 JEDCMD=1,LNECHR
  320 IF (ISSAME(L1,NBYTE(LECHRS,JEDCMD))
     *                          .NE. 0) GOTO 4
C     EDIT-COMMAND UNRECOGNISED OR OTHER ERROR
   34 CALL REJECT
      GOTO 31
    4 ISCHEK = 0
      IF (JEDCMD .GE. 6) GOTO 5
C     INSERT AND DELETE NOT ALLOWED IN HEADER
      IF (ISHDR.NE.0 .AND. JEDCMD.NE.3) GOTO 34
C
C     EDIT-COMMAND WHICH REQUIRES SPECIFIED ITEM
      CALL LSEG(321)
      CALL ESCAN
C     CURSOR MUST BE WITHIN 20 UNITS TO RECOGNISE
      IF (JDIST .GT. 400) GOTO 34
C     SET UP SPECIFICATIONS FOR SELECTED ITEM
      JESEL = JEDIT
      IXE = 4 * NBYTE(MLIST3,JESEL)
      IYE = 4 * NBYTE(MLIST4,JESEL)
C     IDENTIFY CORRECT DISPLAYED STAVE ON SCREEN
      IF (ISINPT .EQ. 0) IYS = (IYE-4*IYINC) /
     *                              200 * 200 + 100
C     BRANCH INDEXED BY EDIT-COMMAND NUMBER
    5 GOTO (53,54,54,56,57,58,3,8,87,6,62),JEDCMD
C
C     SELECT ITEM
   53 ISITEM = JESEL
      GOTO 31
C
C     DELETE ITEM
   54 JEDIT = NFLINK(JEDIT)
C     DECODE SELECTED ITEM TO SET CORRECT TYPE
      JIBAR = NBYTE(MLIST1,JESEL)
      CALL IDCODE
C     REMOVE SELECTED ITEM FROM EDIT TABLE
      CALL EDEL(JESEL)
      IF (ITTYPE .NE. 4) GOTO 55
C     ADJUST TIME COUNT FOR DELETED NOTE
      CALL NLENTH
      JTIME = JTIME - JNDUR
      NNINCB = NNINCB - 1
C     DRAW CROSS OVER DELETED ITEM
```

```
   55 CALL LSEG(32)
      CALL DCROSS
      LISED = 1
C     DROP THROUGH FOR =CHANGE= EDIT-COMMAND
      IF (JEDCMD .EQ. KDLCOM) GOTO 3
C
C     INSERT BEFORE SELECTED ITEM
   56 IXN = IXE + (5*ISINPT-5) * IXINC
      IXP = 4 * NBYTE(MLIST3,NBLINK(JESEL))
      GOTO 572
C     INSERT AFTER SELECTED ITEM
   57 JEDIT = NFLINK(JEDIT)
      IXN = IXE + 2 * IXINC
      IXP = IXE
  572 IF (IXN .LT. IXB) IXN = IXB
      IXM = IXN
      NOLENG = 0
C     INPUT AND DISPLAY NEW ITEM
      CALL IINDIS
C     TREAT SUBSEQUENT INTERRUPT AS CANCEL-EDIT
      CALL BREAK(&87)
C     =END= ITEM CANCELS INSERTION
      IF (ISEND .NE. 0) GOTO 3
C     ERROR IF BUFFER OVERFLOW
      IF (NILENG + ITLENG .GT. KBSIZE) GOTO 75
C     MOVE NEW ITEM INTO INPUT BUFFER
      CALL MBS(MOBAR,K1,MIBAR,NILENG+1,ITLENG)
      ITPTR = NILENG
C     INSERT NEW ITEM INTO EDIT TABLE
      CALL IINSRT(JEDIT)
C     UPDATE TIME COUNT FOR NOTE ITEM
      IF (ITTYPE .EQ. 4) JTIME = JTIME + JNDUR
      NILENG = NILENG + ITLENG
      LISED = 1
C     AUTOMATIC REFRESH IF CLEF SIGN WAS INPUT
      IF (ITEM .NE. 33) GOTO 3
C
C     UPDATE AND REDISPLAY
   58 IF (ISINPT .NE. 0) GOTO 34
      CALL UPDATE
      GOTO 22
C
C     OVERRIDE TIME COUNT CHECK
    6 JTMIN = 0
      JTMAX = KBIG
      GOTO 3
C
C     TOGGLE INPUT SWITCH
   62 ISJOY = 1 - ISJOY
      GOTO 3
C
C     DISPLAY =BAR FULL= MESSAGE
   75 CALL TMESS(K8,LBFULL)
      GOTO 87
C     DISPLAY =TIME COUNT ERROR= MESSAGE
   76 CALL TMESS(16,LTCERR)
      CALL TWT1(JTIME)
      CALL TWT1(JTMAX)
      GOTO 34
C
C     END OF EDIT, TREAT AS CANCEL IF NO CHANGE
    8 IF (LISED.EQ.0 .AND. ISINPT.EQ.0) GOTO 87
      IF (ISHDR .EQ. 0) GOTO 81
C     STORE NEW FILE HEADER
      CALL LSEG(31)
      CALL UPDATE
      CALL STHDR
C     DISPLAY =FILE HEADER UPDATED= MESSAGE
      CALL TMESS(19,LFUPD)
      GOTO 86
C
   81 IF (JTIME .GT. JTMAX) GOTO 76
      ISCHEK = 1
      IF (ISINPT .NE. 0) GOTO 865
      IF (JTIME .LT. JTMIN) GOTO 76
      ITERR = 0
      CALL LSEG(321)
      CALL SFLAG
      IF (ITERR .NE. 0) GOTO 34
      CALL UPDATE
```

AI-58

```
   86 ISERR = 0
      GOTO 88
  865 CALL LSEG(321)
      CALL SFLAG
      GOTO 88
C     CANCEL EDIT, NOT ALLOWED IF INPUTTING
   87 IF (ISINPT .NE. 0) GOTO 34
      ISERR = 3
   88 ISEDIT = 0
      RETURN
      END
C
C
      SUBROUTINE IINSRT(N)
C     INSERT NEW ENTRY PRECEDING ENTRY *N* OF
C     LINKED EDIT TABLE.
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      CALL EINSRT(N,ITPTR,ITLENG,IXE/K4,
     *                           (IYE+IYS)/K4)
      IF (ITTYPE .EQ. 4) NNINCB = NNINCB + 1
      RETURN
      END
C
C
      SUBROUTINE DHDR
C     DISPLAY FILE HEADER.
      CALL LSEG(32)
      CALL LSEG(323)
      CALL DHDR1
      RETURN
      END
C
C
      SUBROUTINE BAR
C     END OF MEASURE - ADVANCE TO NEXT STAVE.
      COMMON MC1(320),JBBUF
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /GP/ JGROUP,MGROUP(7)
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
      COMMON /TT/ JTTEMP,JXTEMP
      IF (JLINE .LT. NLINES) GOTO 2
      CALL CHSTAV(K1)
      JBAR = JBAR + 1
      GOTO 4
    2 CALL CHSTAV(JLINE+1)
    4 JECLEF = JCLEF
      CALL MMOVE(MCMEM,MSMEM)
      NNINCB = 0
      JBBUF = 0
      JREPT = 0
      JTTEMP = 0
      JGROUP = 1
      RETURN
      END
!JOB JH,COMPILE MUSIC PROGRAM SEGMENT 31
!EXTRABGD 6800
!ASS (M:B0,D1,JHROM31)
!FORTRAN B0,NS
C
C
C     ******************************************
C     *                                        *
C     *          SEGMENT 31 - INPUT            *
C     *                                        *
C     ******************************************
C
C
      SUBROUTINE MINIT
C     INITIALISATION FOR MUSIC INPUT.
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
```

```
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /M0/ JIMODE,JOMODE
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
C     INPUT HEADER RECORD FOR PRINTED MUSIC ONLY
      IF (JIMODE .EQ. 1) CALL INHDR
      CALL BREAK(&9)
      CALL LSEG(311)
      CALL BAR
      MHEAD(JIMODE) = JFREE
      MTAIL(JIMODE) = JFREE
      CALL OPPOSN(KDATA*MHEAD(JIMODE))
      CALL XRESET
      JTMIN = 0
    9 RETURN
      END
C
C
      SUBROUTINE INHDR
C     INPUT FILE HEADER.
      COMMON MC1(322),MFLAGS(3)
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /HE/ MCLEF(10),MPART(10)
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /KK/ KBAR,KBDOTS,KCLEF,KTIME,KKEY,
     * KTEXUP,KTEXDN,KTEXWD,KFLAT,KNATUR,KSHARP,
     * KTIE,KNDOT,KNREPT,KNSDEL,KSWICH
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      DIMENSION LKMESS(4),LTMESS(4),LCMESS(4)
      DATA LKMESS/15HKEY SIGNATURE? /
      DATA LTMESS/15HTIME SIGNATURE?/
      DATA LCMESS/15HCLEF FOR STAVE /
      ISHDR = 1
      CALL LSEG(313)
      CALL HDRTXT
      CALL BREAK(&2)
    2 CALL REQEST(LKMESS,KKEY,K0)
      CALL REQEST(LTMESS,KTIME,K0)
      DO 30 JLINE=1,NLINES
      CALL REQEST(LCMESS,KCLEF,JLINE)
      CALL STBYTE(JCLEF,MCLEF,JLINE)
   30 CONTINUE
      MFLAGS(1) = 0
      CALL STHDR
      ISHDR = 0
      RETURN
      END
C
C
      SUBROUTINE STHDR
C     STORE IDENTIFIER AND HEADER RECORD IN FILE.
      COMMON MC1(322),MFLAGS(3)
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      CALL OPPOSN(K6)
      DO 50 L=1,4
   50 CALL OPBYTE(NBYTE(MIDENT,L))
      CALL OPPOSN(JHDR)
      CALL STLINE(MFLAGS(1))
      RETURN
      END
C
C
      SUBROUTINE REQEST(NTEXT,NVALUE,N)
C     DISPLAY TEXT STRING IN *NTEXT* AND INTEGER
C     *N* IF N0N=ZERO.  INPUT ITEM OF TYPE
C     SPECIFIED BY INTERNAL CODE *NVALUE*.
      COMMON MC1(253),NOLENG
      COMMON /CT/ JLABEL
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
```

```
      COMMON /QL/ IXG,IYG,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TD/ IXC,IYC,IXB,IXR
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      DIMENSION NTEXT(1)
      LOLENG = NOLENG
C     DISPLAY QUESTION
    2 CALL TMESS(15,NTEXT)
      IF (N .GT. 0) CALL TWT1(N)
      IXC = IXT + IXINC
      IYC = IYQ
C     INPUT ONE ITEM
      CALL IINPUT
C+    SIMULATE INTERRUPT IF -END- ITEM WAS INPUT
      IF (ISEND .NE. 0) GOTO JLABEL
      IF (ITEM+31 .EQ. NVALUE) GOTO 9
C     ERROR, WRONG TYPE OF ITEM
      NOLENG = LOLENG
      CALL REJECT
      GOTO 2
    9 RETURN
      END
C
C
      SUBROUTINE IINPUT
C     INPUT AND ENCODE ITEM.
      COMMON MC1(253),NOLENG
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSOZ,
     * ISREST,ISAHED,ISXPOS,ISEMB
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /IV/ INTYPE,ISBAR
      COMMON /KK/ KBAR,KBDOTS,KCLEF,KTIME,KKEY,
     * KTEXUP,KTEXDN,KTEXWD,KFLAT,KNATUR,KSHARP,
     * KTIE,KNDOT,KNREPT,KNSDEL,KSWICH
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MC/ KBSIZE,KWSIZE,KXMAX,KYMAX,KFSIZE
      COMMON /MO/ JIMODE,JOMODE
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /RP/ ISAUTO,JEDCOD,JESEL,ISDISP
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TD/ IXC,IYC,IXB,IXR
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      DIMENSION LLETX(1)
      DATA LLETX/1HX/
      CALL BREAK(66)
      ISDISP = 0
      ITTYPE = 0
C     TYPE IS NOTE-ITEM FOR AUTOMATIC NOTE-REPEAT
      IF (ISAUTO .GT. 0) ITTYPE = 4
C     OR CONTROL-ITEM FOR AUTOMATIC MEASURE-REPEAT
      IF (ISRPT .GT. 0) ITTYPE = 1
      IF (ITTYPE .NE. 0) GOTO 22
C     OTHERWISE INPUT ITEM FROM TERMINAL
   18 ITEMX = IXC
      ISXPOS = 0
      ISERR = 0
C     READ FIRST CHARACTER OF ITEM
      CALL RDCHAR
C     ERROR IF CHARACTER NOT RECOGNISED
      IF (INCODE .EQ. 0) GOTO 75
      IF (INCODE .NE. KNSDEL) GOTO 2
C     CANCEL-EDIT-COMMAND NOT ALLOWED IN HEADER
      IF (ISHDR .NE. 0) GOTO 75
C     CANCEL CURRENT EDIT-COMMAND IF EDITING
      IF (ISEDIT .NE. 0) GOTO 7
C     ERROR IF NOTHING IN CURRENT MEASURE TO EDIT
      IF (NOLENG .LE. 0) GOTO 75
C     SET FLAG TO START EDITING CURRENT MEASURE
      ISEDIT = 1
      GOTO 8
C
C     IDENTIFY ITEM TYPE
    2 ITTYPE = INCODE / 64 + 1
```

```
C     ALL ITEMS IN BRAILLE MODES ARE BRAILLE ITEMS
   22 IF (JIMODE.NE.1 .AND. INIT.EQ.0) ITTYPE = 5
      INTYPE =ITTYPE
      CALL LSEG(310+ITTYPE)
      GOTO (32,34,36,38,4),ITTYPE
C
C     INPUT CONTROL ITEM
   32 CALL CINPUT
      GOTO 45
C     INPUT SEPARATE-SIGN ITEM
   34 CALL SINPUT
      GOTO 5
C     INPUT TEXT ITEM
   36 CALL TINPUT
      GOTO 5
C     INPUT NOTE ITEM
   38 CALL NINPUT
      GOTO 5
C     INPUT BRAILLE ITEM
    4 CALL BINPUT
   45 IF (ISEND .NE. 0) GOTO 7
      IF (ISBAR .NE. 0) GOTO 8
    5 ISBAR = 0
C     ENCODE ITEM IF NO ERRORS OCCURRED
   52 IF (ISERR .EQ. 0) CALL ICODE
C     SET DISPLAY INDICATOR IF STILL NO ERRORS
C     AND NOT HEADER RECORD OR MEASURE-REPEAT
   54 IF (ISERR .EQ. 0 .AND. INIT .EQ. 0 .AND.
     * ISRPT .EQ. 0 .AND. (ISHDR .EQ. 0 .OR.
     * ISEDIT .EQ. 0)) ISDISP = 1
      IF (ISERR .NE. 2) GOTO 8
C
C     RETURN POINT FOR BREAK DURING INPUT
    6 ISAUTO = 0
      ISRPT = 0
C     OVERWRITE DISPLAYED CHARACTERS WITH XXX
      IXT = ITEMX
      IF (IXT .GT. IXC) IYC = IYC + 20
   64 IF (IXT .LE. KXMAX - 9 * IXINC) GOTO 66
      IXT = IYXMIN
      IYC = IYC - 20
   66 IF (IXT .EQ. IXC) GOTO 18
      CALL TWRITE(IXT,IYC,K1,LLETX)
      GOTO 64
C
    7 ISEND = 1
      IYQ = MIN0(IYQ,IYC-3*IYINC)
      GOTO 85
   75 CALL REJECT
      ISERR = 1
    8 ISEND = 0
C     RECYCLE AFTER ERROR UNLESS AUTOMATIC REPEAT
   85 IF (ISERR.NE.0 .AND. ISAUTO.EQ.0) GOTO 18
      RETURN
      END
C
C
      SUBROUTINE ICODE
C     ENCODE ITEM.
      COMMON MC1(253),NOLENG
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *    ISERR,ISNXST,ISINPT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /IU/ MITEM2(8)
      COMMON /IV/ INTYPE,ISBAR
      DIMENSION LMESS(3)
      DATA LMESS/12HBAR TOO LONG/
      ISERR = 0
      ITPTR = NOLENG
      ITTYPE = INTYPE
      CALL BSTORE(ITEM)
      GOTO (6,2,3,4,6),ITTYPE
C
    2 IF (ITEM.LE.48) CALL BSTORE(MITEM2(ITEM-32))
      GOTO 6
C     ENCODE TEXT ITEM
    3 CALL TCODE
      GOTO 6
C     ENCODE NOTE ITEM
    4 CALL NCODE
```

```
    6 IF (ISERR .EQ. 0) GOTO 8
C     IGNORE NEW ITEM IF BUFFER OVERFLOWS
      NOLENG = ITPTR
C     DISPLAY -BAR TOO LONG- MESSAGE
      CALL TMESS(12,LMESS)
    8 ITLENG = NOLENG - ITPTR
      RETURN
      END
C
C
      SUBROUTINE RDCHAR
C     READ SINGLE CHARACTER FROM TERMINAL IN
C     NO-ECHO MODE. RETURNS *MCHAR(1)* = CHARACTER
C     IN FIRST BYTE AND *INCODE* = CORRESPONDING
C     INTERNAL CODE.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MC/ KBSIZE,KWSIZE,KXMAX,KYMAX,KFSIZE
      COMMON /QL/ IXG,IYG,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TD/ IXC,IYC,IXB,IXR
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /XB/ MCODES(64)
C     DISPLAY CHARACTER FOR CARRIAGE-RETURN
      DATA LSLASH/1H//
      IF (IXC .LE. KXMAX - 9 * IXINC) GOTO 2
C     START NEW LINE OF REFLECTED CHARACTERS
      IXC = IXQMIN
      IYC = IYC - 20
C     AVOID COLLISION OF MESSAGES WITH INPUT CHARS
      IYG = MIN0(IYG,IYC-3*IYINC)
    2 IF (ISJOY .NE. 0) GOTO 3
C     POSITION CURSOR TO INDICATE READY TO READ
      IXT = IXC
      IYT = IYC
      CALL TPOSN
C     READ INPUT CHARACTER WITHOUT DISPLAYING IT
      CALL NOECHO
      GOTO 4
C     USE JOYSTICK INPUT
    3 MCHAR(1) = 0
      CALL JOY(MCHAR,JX,JY)
    4 IF (MCHAR(1) .NE. 0) GOTO 5
C     CARRIAGE-RETURN
      MCHAR(1) = LSLASH
      INCODE = KEOD
      GOTO 9
C     TRANSLATE INPUT CHARACTER TO INTERNAL CODE
    5 INCODE = NBYTE(MCODES,NBYTE(MCHAR,K1)+1)
    9 RETURN
      END
C
C
      SUBROUTINE OKNEXT
C     ECHO LAST INPUT CHARACTER AND READ NEXT ONE.
      CALL ACCEPT
      CALL RDCHAR
      RETURN
      END
C
C
      SUBROUTINE NOGOOD
C     BLIP LAST CHARACTER AND READ NEXT ONE.
      CALL REJECT
      CALL RDCHAR
      RETURN
      END
C
C
      SUBROUTINE ACCEPT
C     ECHO INPUT CHARACTER ON SCREEN.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /TD/ IXC,IYC,IXB,IXR
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      CALL TWRITE(IXC,IYC,K1,MCHAR)
      IXC = IXT
      RETURN
```

```
      END
C
C
      SUBROUTINE NUMBER(N,NMAX,NCR)
C     READ NON-NEGATIVE INTEGER FROM TERMINAL.
C     *NMAX* IS MAXIMUM ALLOWED VALUE.  NEGATIVE
C     *N* ON ENTRY INDICATES ZERO RESULT ALLOWED.
C     *NCR* = 1 IF FINAL CARRIAGE-RETURN NEEDED.
C     RETURNS *N* = INTEGER READ, *MCHAR(1)* AND
C     *INCODE* ARE SET FOR FIRST NON-DIGIT READ.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      L4 = 0
    2 CALL OKNEXT
    3 L6 = NBYTE(MCHAR,K1) - 240
      IF (L6 .LT. 0 .OR. L6 .GT. 9) GOTO 7
      L5 = 10 * L4 + L6
      IF (L5 .LE. NMAX) GOTO 6
    4 CALL NOGOOD
      GOTO 3
    6 L4 = L5
      IF (NMAX .GE. 10) GOTO 2
      CALL OKNEXT
    7 IF (N .GE. 0 .AND. L4 .EQ. 0) GOTO 4
      IF (NCR .NE. 0) CALL REQUIR(KEOD)
      N = L4
      RETURN
      END
C
C
      SUBROUTINE REQUIR(N)
C     TEST IF *INCODE* IS *N*, REREAD IF NOT.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
    2 IF (INCODE .EQ. N) GOTO 9
      CALL NOGOOD
      GOTO 2
    9 RETURN
      END
C
C
      BLOCK DATA
C     THIS SUBPROGRAM IS PROGRAM-GENERATED.
      COMMON /XB/ MCODES(64)
      DATA MCODES/
     *8Z00000000,8Z00000000,8Z00000000,8Z0C000000,
     *8Z00000000,8Z00000000,8Z00000000,8Z00000000,
     *8Z00000000,8Z00000000,8Z00000000,8Z0C000000,
     *8Z00000000,8Z00000000,8Z00000000,8Z00000000,
     *8Z57000000,8Z00000000,8Z000000DA,8Z5550DE04,
     *8ZDC000000,8Z00000000,8Z000001D3,8Z59510200,
     *8ZDF000000,8Z00000000,8Z00008000,8ZD45356DD,
     *8Z00000000,8Z00000000,8Z000005D5,8Z030CD9DB,
     *8Z00C6C7C1,8ZC2C3C4C5,8Z47D60000,8Z00000000,
     *8Z00464245,8Z0643E058,8Z44C00000,8Z0C000000,
     *8Z0000D741,8ZD2818207,8ZD1400000,8Z00000000,
     *8Z00040000,8Z52540000,8Z00000000,8Z00000000,
     *8Z00C6C7C1,8ZC2C3C4C5,8Z47D60000,8Z0C000000,
     *8Z00464245,8Z0643E058,8Z44C00000,8Z00000000,
     *8Z0000D741,8ZD2818207,8ZD1400000,8Z0C000000,
     *8ZC7C8C9CA,8ZCBCCCDCE,8ZCFD00000,8Z0000000000/
      END
!ASS (M:B0,D1,JHR0M311)
!FORTRAN B0,NS
C
C
C     ********************************************
C     *                                          *
C     *  SEGMENT 311 - INPUT OF CONTROL ITEMS    *
C     *                                          *
C     ********************************************
C
C
      SUBROUTINE CINPUT
C     INPUT CONTROL ITEM.
      COMMON MC1(188),NILENG,MIBAR(64),NOLENG,
     * MOBAR(64),JIBAR,JOBAR,JBBUF
      COMMON /B1/ IFIRST,JOPAGE
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
```

```
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *    ISERR,ISNXST,ISINPT
      COMMON /GP/ JGROUP,MGROUP(7)
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /IV/ INTYPE,ISBAR
      COMMON /KK/ KBAR,KBDOTS,KCLEF,KTIME,KKEY,
     * KTEXUP,KTEXDN,KTEXWD,KFLAT,KNATUR,KSHARP,
     * KTIE,KNDOT,KNREPT,KNSDEL,KSWICH
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /MO/ JIMODE,JOMODE
      COMMON /MR/ NNOWN
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /SE/ JCMD,KSEGS(7)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /TT/ JTTEMP,JXTEMP
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION LMATCH(3),LCVAL(5),LPCHAN(4)
      DATA LMATCH/10HSCP+-12=YU/,LSTAR/1H*/
      DATA LCVAL(1)/217/,LCVAL(2)/209/,LCVAL(3)/
     * 210/,LCVAL(4)/222/,LCVAL(5)/223/
      DATA LPCHAN(1)/-7/,LPCHAN(2)/7/,LPCHAN(3)/
     * -1/,LPCHAN(4)/1/
C
      ISEND = 0
C     CONTROL ITEMS NOT ALLOWED IN HEADER RECORD
      IF (ISHDR .NE. 0) GOTO 7
C     CARRY ON IF IN MIDDLE OF MEASURE-REPEAT
      IF (ISRPT .NE. 0) GOTO 33
      ITEM = INCODE
      GOTO (12,14,16,18,2,11,45),ITEM
      GOTO 7
C
C     CANCEL TIME COUNT CHECK
   11 JTMIN = 0
      JTMAX = KBIG
      GOTO 46
C
C     END OF MUSIC
   12 ISEND = 1
      GOTO 8
C
C     IN-ACCORD
   14 CALL OKNEXT
C     NEXT CHARACTER MUST BE CARRIAGE-RETURN
      CALL REQUIR(KEOD)
C     RESTORE TIME COUNT SAVED AT SET-RETURN
      JTIME = JTTEMP
      GOTO 8
C     =AT= CHARACTER
   16 CALL OKNEXT
      IF (INCODE .NE. KEOD) GOTO 6
C
C     SET RETURN POINT FOR IN-ACCORD
      JTTEMP = JTIME
      GOTO 8
C
C     VERTICAL BAR CHARACTER
   18 CALL OKNEXT
C     END-OF-MEASURE AND MEASURE-REPEAT NOT
C     AVAILABLE DURING EDITING
  185 IF (ISEDIT .NE. 0) GOTO 186
C     TIME COUNT MUST BE CORRECT TO END MEASURE
      IF (INCODE .EQ. KEOD .AND. JTIME .GE. JTMIN)
     *                              GOTO 34
C     MEASURE MUST BE EMPTY FOR MEASURE-REPEAT
      IF (INCODE .EQ. KNREPT .AND. NFLINK(K1)
     *                              .EQ. 1) GOTO 26
  186 IF (INCODE .EQ. KNSDEL) GOTO 72
      IF (INCODE .EQ. KBAR) GOTO 19
      IF (INCODE .EQ. KNDOT) GOTO 23
      IF (INCODE .EQ. KTEXDN .AND. JTIME .GE.
```

```
     *                              JTMIN) GOTO 24
C      INVALID CHARACTER FOLLOWING VERTICAL BAR
       CALL NOGOOD
       GOTO 185
C      DOUBLE BARLINE
   19  CALL OKNEXT
  191  IF (INCODE .EQ. KNSDEL) GOTO 72
       IF (INCODE .EQ. KEOD) GOTO 192
       IF (INCODE .EQ. KDOTS) GOTO 193
C      CHEAT - T STANDS FOR THIN HERE, NOT TIME
       IF (INCODE .EQ. KTIME) GOTO 194
C      INVALID CHARACTER AFTER DOUBLE BAR LINE
       CALL NOGOOD
       GOTO 191
C      ORDINARY THICK DOUBLE BARLINE
  192  ITEM = 6
       GOTO 8
C      DOUBLE BARLINE FOLLOWED BY DOTS
  193  ITEM = 7
       GOTO 78
C      THIN DOUBLE BARLINE
  194  ITEM = 9
       GOTO 78
C
C      DOTS PRECEDING BARLINE
    2  DO 220 L=1,2
       CALL OKNEXT
       CALL REQUIR(KBAR)
  220  CONTINUE
       CALL OKNEXT
C      NEXT CHARACTER MUST BE DOTS OR C/R
       IF (INCODE .NE. KDOTS) GOTO 79
C      DOUBLE BARLINE PRECEDED AND FOLLOWED BY DOTS
       ITEM = 8
       GOTO 78
C
C      DOTTED BARLINE
   23  ITEM = 10
       GOTO 78
C.
C      CANCEL TIME COUNT CHECK FOR FOOTNOTE
   24  JTMIN = 0
       JTMAX = KBIG
       ITEM = 11
       GOTO 78
C
C      MEASURE REPEAT - EVALUATE BAR NUMBER
   26  LREQB = -1
C      ANY EXISTING BAR NUMBER ALLOWED FOR -INSERT-
       LMAX = NBARS
C      OTHERWISE MUST NOT EXCEED CURRENT BAR
       IF (JCMD .NE. 3) LMAX = NOB
C      READ BAR NUMBER FROM TERMINAL
       CALL NUMBER(LREQB,LMAX,KO)
       IF (INCODE .EQ. KNDOT) GOTO 262
C      USE PREVIOUS MEASURE IF NO STAVE SPECIFIED
       IF (LREQB .LE. 0) LREQB = NOB - 1
       GOTO 263
C      USE CURRENT MEASURE IF STAVE SPECIFIED
  262  IF (LREQB .LE. 0) LREQB = NOB
C      EVALUATE STAVE NUMBER
       LMAX = NLINES
       IF (LREQB .EQ. NOB) LMAX = JLINE = 1
       LREQL = 0
C      REQUEST MEASURE NUMBER
       CALL NUMBER(LREQL,LMAX,KO)
       GOTO 264
C      USE SAME STAVE IF NO STAVE NUMBER SPECIFIED
  263  LREQL = JLINE
C      EVALUATE NUMBER OF MEASURES TO REPEAT
  264  IF (MCHAR(1) .NE. LSTAR) GOTO 265
C      REQUEST NUMBER OF MEASURES TO REPEAT
       CALL NUMBER(ISRPT,KBIG,KO)
       GOTO 266
C      USE DEFAULT REPEAT COUNT
  265  ISRPT = 1
C      EVALUATE CHANGE IN NOTE PITCH IF ANY
  266  NNDOWN = 0
  268  DO 270 L=1,4
  270  IF (INCODE .EQ. LCVAL(L+1)) GOTO 272
```

```
      IF (INCODE .EQ. KEOD) GOTO 3
C     INVALID PITCH CHANGE CHARACTER
      CALL NOGOOD
      GOTO 268
  272 NNDOWN = NNDOWN + LPCHAN(L)
      CALL OKNEXT
      GOTO 268
C
C
C     OBTAIN MEASURE TO BE REPEATED
    3 LITEM = ITEM
      ISCHEK = 0
      CALL BFIND(LREGB,LREGL)
      ISCHEK = 1
      ITEM = LITEM
      GOTO 75
C     OBTAIN NEXT MEASURE FOR AUTOMATIC REPEAT
   33 ISCHEK = 0
      CALL NXSEL
      ISCHEK = 1
      GOTO 75
C
C     END OF MEASURE
   34 IF (NOB .GT. 1) GOTO 38
      IF (JLINE .NE. 1) GOTO 37
C     FIRST MEASURE OF SECTION
      JTMIN = JTIME
C     SET FLAG IN HEADER BAR IF FIRST MEASURE FULL
      IFIRST = 0
      IF (JTIME .GE. 512*JTIME1/JTIME2) IFIRST = 1
      CALL XBYTE(128+IFIRST,JHDR+1)
   37 IF (JLINE .NE. NSSEL) GOTO 4
C     RESET TIME COUNT LIMITS FOR NEXT MEASURE
   38 IF (JTYME .NE. 255) GOTO 39
C     NO LIMITS IF -UNMEASURED- MUSIC
      JTMIN = 0
      JTMAX = KBIG
      GOTO 75
C     OTHERWISE LIMITS ACCORDING TO TIME SIGNATURE
   39 JTMIN = 512 * JTIME1 / JTIME2
    4 JTMAX = JTMIN
      GOTO 75
C
C     TOGGLE INPUT SWITCH
   45 ISJOY = 1 - ISJOY
      ISBAR = 0
C     PREVENT ATTEMPTED STORAGE OF ITEM
   46 ISERR = 1
      GOTO 85
C
C     CONTROL ITEM SPECIFIED USING THE AT SIGN
    6 JBYTE = NBYTE(MCHAR,K1)
C     IDENTIFY FOLLOWING CHARACTER
      DO 620 L=1,10
  620 IF (ISSAME(JBYTE,NBYTE(LMATCH,L)) .NE. 0)
     *                                GOTO 64
C     INVALID CHARACTER
      CALL NOGOOD
      GOTO 6
   64 ITEM = 11 + L
      GOTO 8
C
    7 CALL REJECT
      ISERR = 1
      GOTO 9
C     CANCEL ITEM
   72 ISERR = 2
      GOTO 9
C     SET END-OF-MEASURE FLAG
   75 ISBAR = 1
      ISEND = 0
      GOTO 82
   78 CALL OKNEXT
   79 CALL REQUIR(KEOD)
    8 ISBAR = 0
   82 ISERR = 0
C     DISPLAY LAST INPUT CHARACTER ON SCREEN
   85 CALL ACCEPT
    9 RETURN
      END
```

```
!ASS (M:B0,D1,JHR0M312)
!F0RTRAN B0,NS
C
C
C     ******************************************
C     *                                        *
C     *SEGMENT 312 - INPUT 0F SEPARATE SIGN ITEMS*
C     *                                        *
C     ******************************************
C
C
      SUBR0UTINE SINPUT
C     INPUT SEPARATE SIGN ITEM.
      C0MM0N /CH/ MCHAR(1),INC0DE,KE0D,ISJ0Y,JX,JY
      C0MM0N /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      C0MM0N /GP/ JGR0UP,MGR0UP(7)
      C0MM0N /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      C0MM0N /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      C0MM0N /IU/ JCLEF,JTYME,JKEY,NGR0UP,NPAGE,
     * NLINE,NBEAM,NSCALE
      C0MM0N /KK/ KBAR,KBD0TS,KCLEF,KTIME,KKEY,
     * KTEXUP,KTEXDN,KTEXWD,KFLAT,KNATUR,KSHARP,
     * KTIE,KND0T,KNREPT,KNSDEL,KSWICH
      C0MM0N /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      C0MM0N /NS/ NLINES,JLINE,JPART,JV0IC,NDP,NDL
      C0MM0N /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JB0UR,JND0R,NN0TES
      C0MM0N /WS/ L1,L2,L3,JBYTE
      DIMENSI0N MITEM2(6)
      EQUIVALENCE(MITEM2,JCLEF)
      LENTRY = INC0DE - 63
      ITEM = LENTRY + 32
C     0NLY CLEF, TIME, KEY ALL0WED IN HEADER
      IF (ISHDR .NE. 0 .AND. LENTRY .GT. 3) G0T0 7
      IF (LENTRY .GT. 16) G0T0 8
C     D0UBLE-BYTE ITEM
      G0T0 (22,24,26,28,3,3,3,3),LENTRY
      G0T0 82
C
C     INPUT CLEF SIGN
   22 CALL ICSIG
      G0T0 8
C     INPUT TIME SIGNATURE
   24 CALL ITSIG
      G0T0 8
C     INPUT KEY SIGNATURE
   26 CALL IKSIG
      G0T0 8
C     INPUT IRREGULAR N0TE GR0UPING
   28 IF (ISEDIT.EQ.0 .AND. JTIME.GE.JTMAX) G0T0 7
      L1 = 0
      CALL NUMBER(L1,K7,K1)
      NGR0UP = L1
      JGR0UP = MGR0UP(NGR0UP)
      G0T0 82
C     INKPRINT PAGE 0R LINE NUMBER, N0TE BEAMING
C     0R H0RIZ0NTAL DISPLAY SCALE
    3 L1 = 0
      CALL NUMBER(L1,255,K1)
C     BEAMING NUMBER MUST BE IN RANGE 2 T0 16
      IF (LENTRY .EQ. 7 .AND. (L1 .LE. 1 .0R.
     *                L1 .GT. 16)) G0T0 7
      MITEM2(LENTRY) = L1
      G0T0 82
C     INPUT ERR0R
    7 CALL REJECT
      ISERR = 1
      G0T0 9
    8 C0NTINUE
C     CHECK MATCHING 0F EXTENDED SIGNS
   82 CALL SSCHK(K1)
      IF (ISERR .NE. 0) G0T0 7
C     DISPLAY LAST INPUT CHARACTER
   84 CALL ACCEPT
    9 RETURN
      END
C
C
```

```
      SUBROUTINE ICSIG
C     INPUT CLEF SIGN.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /WS/ L1,L2,L3,JBYTE
C     INPUT CHARACTERS AND CODES FOR CLEFS
      DIMENSION LCCLEF(2),LXCLEF(2)
      DATA LCCLEF/8HBTFG1348/
      DATA LXCLEF/8Z867A867A,8Z7C808281/
      CALL OKNEXT
    2 L1 = NBYTE(MCHAR,K1)
C     IDENTIFY CLEF TYPE
      DO 40 L=1,8
   40 IF (ISSAME(L1,NBYTE(LCCLEF,L)).NE.0) GOTO 8
C     INVALID CLEF TYPE CHARACTER
      CALL NEGOOD
      GOTO 2
    8 CALL OKNEXT
C     MUST BE FOLLOWED BY CARRIAGE-RETURN
      CALL REQUIR(KEOD)
      JCLEF = NBYTE(LXCLEF,L)
      RETURN
      END
C
C
      SUBROUTINE ITSIG
C     INPUT TIME SIGNATURE.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /KK/ KBAR,KBOOTS,KCLEF,KTIME,KKEY,
     * KTEXUP,KTEXDN,KTEXWD,KFLAT,KNATUR,KSHARP,
     * KTIE,KNOOT,KNREPT,KNSDEL,KSWICH
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /ND/ NXLENG,NSHIFT,JBXL,NEXTX
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION LLETC(1),LLETU(1)
      DATA LLETC/1HC/,LLETU/1HU/
C
      L1 = -1
      L2 = 0
      CALL NUMBER(L1,32,KO)
      LCHAR = NBYTE(MCHAR,K1)
      IF (ISSAME(LCHAR,NBYTE(LLETC,K1)) .NE. 0)
     *                          GOTO 18
      IF (ISSAME(LCHAR,NBYTE(LLETU,K1)) .EQ. 0)
     *             GOTO 3
C     UNMEASURED TIME SIGNATURE
      JTYME = 255
      CALL OKNEXT
      GOTO 21
   18 CALL OKNEXT
      IF (INCODE .NE. KBAR) GOTO 2
C     C BAR (EQUIVALENT TO 2/2)
      JTYME = 192
      CALL OKNEXT
      L1 = 2
      GOTO 22
C     C (EQUIVALENT TO 4/4)
    2 JTYME = 224
   21 L1 = 4
   22 L2 = L1
      CALL REQUIR(KEOD)
      GOTO 5
    3 CALL REQUIR(KEOD)
      L2 = -1
      CALL NUMBER(L2,32,K1)
      IF (L1 .EQ. 0) GOTO 42
C     CHECK THAT LOWER NUMBER IS 1,2,4,8,16 OR 32
      DO 40 L=1,6
   40 IF (L2 .EQ. NPOWER(L-1)) GOTO 44
C     ERROR IF NOT
   42 CALL REQUIR(KO)
   44 JTYME = 32 * L + L1 - 33
    5 JTIME1 = L1
```

```
      JTIME2 = L2
      IF (JTYME .NE. 255) GOTO 6
C     NO TIME COUNT LIMIT FOR UNMEASURED MUSIC
      JTMIN = 0
      JTMAX = KBIG
      GOTO 7
C     TIME COUNT LIMITS ACCORDING TO TIME SIG.
    6 JTMIN = 512 * JTIME1 / JTIME2
      JTMAX = JTMIN
    7 JBXL = JTIME1 * NSCALE * 2
      RETURN
      END
C
C
      SUBROUTINE IKSIG
C     INPUT KEY SIGNATURE.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *    ISERR,ISNXST,ISINPT
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /KK/ KBAR,KBDOTS,KCLEF,KTIME,KKEY,
     * KTEXUP,KTEXDN,KTEXWD,KFLAT,KNATUR,KSHARP,
     * KTIE,KNDOT,KNREPT,KNSDEL,KSWICH
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KY/ JKEYNO,JKEYSG,MKPCH(2),MTKPCH(2)
      COMMON /SD/ MYAXID(3)
      COMMON /WS/ L1,L2,L3,JBYTE
      L1 = -1
      CALL NUMBER(L1,K7,KO)
C     Y CO-ORDINATES FOR NATURALS SAME AS
C     THOSE OF PREVIOUS KEY SIGNATURE
      MYAXID(2) = MYAXID(JKEYSG)
      IF (L1 .GT. 0) GOTO 3
      CALL REQUIR(KEOD)
      JKEYNO = 0
      JKEYSG = 1
      GOTO 5
    3 L2 = INCODE - KFLAT + 1
C     ERROR IF NON-ACCIDENTAL CHARACTER OR NATURAL
C     SPECIFIED IN HEADER RECORD
      IF (L2 .LE. 0 .OR. L2 .GT. 3 .OR. L2 .EQ.
     * 2 .AND. ISHDR .NE. 0) CALL REQUIR(KC)
      JKEYNO = L1
      JKEYSG = L2
    5 JKEY = 8 * JKEYSG + JKEYNO
      RETURN
      END
!ASS (M:BO,D1,JHROM313)
!FORTRAN BO,NS
C
C
C     ***********************************************
C     *                                             *
C     *     SEGMENT 313 - INPUT OF TEXT ITEMS       *
C     *                                             *
C     ***********************************************
C
C
      SUBROUTINE TINPUT
C     INPUT TEXT ITEM.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *    ISERR,ISNXST,ISINPT
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /HE/ MCLEF(10),MPART(10)
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /KK/ KBAR,KBDOTS,KCLEF,KTIME,KKEY,
     * KTEXUP,KTEXDN,KTEXWD,KFLAT,KNATUR,KSHARP,
     * KTIE,KNDOT,KNREPT,KNSDEL,KSWICH
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NCP,NDL
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      COMMON /TY/ ISCONT,ISBLOW
      COMMON /WS/ L1,L2,L3,JBYTE
C
      ISWORD = 0
```

```
      IF (INCODE .NE. KTEXWD) GOTO 3
C     NO SUNG WORDS ALLOWED IN INSTRUMENTAL PARTS
      IF (NBYTE(MPART,JLINE) .LT. 64) GOTO 7
C     SET SUNG-WORD INDICATOR
      ISWORD = 1
    3 IF (ISHDR.EQ.0 .OR. ISEDIT.NE.0) CALL ACCEPT
      NCHARS = 0
      ISCONT = 0
      ISBLOW = 1
      IF (INCODE .EQ. KTEXUP) ISBLOW = 0
      ISERR = 0
C     READ ACTUAL TEXT OF ITEM
      CALL INTEXT
C     EXIT IF SIMULATED INTERRUPT
      IF (ISERR .EQ. 2) GOTO 9
      L1 = ISBLOW
      IF (ISWORD .EQ. 0) GOTO 66
      IF (NBYTE(MTEXT,NCHARS) .NE. 96) GOTO 64
C     REMOVE HYPHEN AND SET CONTINUATION FLAG
      NCHARS = NCHARS - 1
      ISCONT = 1
   64 L1 = ISCONT
   66 ITEM = 64 + 32 * ISWORD + 16 * L1 +
     *                          MINO(NCHARS,15)
C     IGNORE NULL TEXT ITEMS EXCEPT IN HEADER
      IF (NCHARS .LE. 0 .AND. ISHDR .EQ. 0) GOTO 7
      IF (ISHDR .EQ. 0) CALL ACCEPT
      ISERR = 0
      GOTO 9
    7 CALL REJECT
   75 ISERR = 1
    9 RETURN
      END
C
C
      SUBROUTINE INTEXT
C     INPUT TEXT OF TEXT ITEM.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /KK/ MKK1(14),KNSDEL
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      COMMON /WS/ L1,L2,L3,JBYTE
      EQUIVALENCE(KSET,K3)
      DIMENSION LVOWEL(2),LMESC(8)
      DATA LNESC/31/,LVOWEL/5HAEIOU/
      DATA LMESC/16HEFR123456789.SOP,
     *                  15H*X#ACILN/'$~-0?/
C
C     READ NEXT CHARACTER
    2 CALL RDCHAR
C     EXIT IF CARRIAGE-RETURN IS TYPED
      IF (INCODE .EQ. KEOD) GOTO 9
      IF (INCODE .EQ. KNSDEL) GOTO 7
      JBYTE = NBYTE(MCHAR,K1)
      IF (JBYTE .EQ. 0) GOTO 72
      IF (INCODE .NE. KSET) GOTO 6
C     SPECIAL ESCAPE CHARACTER
      CALL OKNEXT
C     ERROR IF CARRIAGE-RETURN INPUT AFTER ESCAPE
   32 IF (INCODE .EQ. KEOD) GOTO 42
      JBYTE = NBYTE(MCHAR,K1)
C     IDENTIFY CHARACTER FOLLOWING ESCAPE
      DO 340 L=1,LNESC
  340 IF (ISSAME(JBYTE,NBYTE(LMESC,L)) .NE. 0)
     *                          GOTO 36
      GOTO 42
   36 IF (L .GT. 25) GOTO 38
C     SPECIAL CODE
      JBYTE = L
      IF (L .GT. 3) JBYTE = L + 29
      GOTO 6
   38 IF (L .GT. 29) GOTO 6
C     ACCENT CHARACTER
      IF (NCHARS .EQ. 0) GOTO 72
      LAX = L - 25
      LPREV = NBYTE(MTEXT,NCHARS)
C     MUST BE PRECEDED BY A VOWEL
      DO 40 L=1,5
```

```
      JBYTE = NBYTE(LVOWEL,L)
      IF (LPREV .EQ. JBYTE) GOTO 5
      IF (LPREV .EQ. JBYTE + 64) GOTO 52
   40 CONTINUE
C     INVALID CHARACTER FOLLOWING ESCAPE
   42 CALL NOGOOD
      GOTO 32
C     UPPER CASE VOWEL
    5 JBYTE = 64
      GOTO 54
C     LOWER CASE VOWEL
   52 JBYTE = 0
C     CONVERT TO CODE FOR ACCENTED LETTER
   54 JBYTE = JBYTE + 121 + 16 * LAX + L
      GOTO 65
C     ERROR IF TEXT ITEM BUFFER IS FULL
    6 IF (NCHARS .GE. 31) GOTO 72
      NCHARS = NCHARS + 1
C     ADD CHARACTER TO TEXT ITEM
   65 CALL STBYTE(JBYTE,MTEXT,NCHARS)
      GOTO 71
C     BACKSPACE, SIMULATE INTERRUPT IF ITEM START
    7 IF (NCHARS .LE. 0) GOTO 74
      NCHARS = NCHARS - 1
C     REFLECT INPUT CHARACTER ON SCREEN
   71 CALL ACCEPT
      GOTO 2
   72 CALL REJECT
      GOTO 2
C     SIMULATE INTERRUPT TO CANCEL THIS ITEM
   74 ISERR = 2
    9 RETURN
      END
C
C
      SUBROUTINE TCODE
C     ENCODE DATA FOR TEXT ITEM.
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      COMMON /WS/ L1,L2,L3,JBYTE
C     EXIT IF NO CHARACTERS IN THIS ITEM
      IF (NCHARS .LE. 0) GOTO 9
C     STORE CHARACTER COUNT FOR LONG FORM
      IF (NCHARS .GE. 15) CALL BSTORE(NCHARS)
      DO 30 L=1,NCHARS
      JBYTE = NBYTE(MTEXT,L)
C     ENSURE NO ZERO BYTES STORED (USE SPACE)
      IF (JBYTE .EQ. 0) JBYTE = 64
      CALL BSTORE(JBYTE)
   30 CONTINUE
    9 RETURN
      END
C
C
      SUBROUTINE HDRTXT
C     INPUT TEXTUAL PARTS OF FILE HEADER.
      COMMON MC1(253),NOLENG
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /IV/ INTYPE,ISBAR
      COMMON /KB/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /TL/ NNAME,MNAME(8)
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      DIMENSION LTLENG(10),LTEXT(30)
      DATA LTLENG/10,4,5,8,8,9,6,5,9,8/
      DATA KNTEXT/8/,LTEXT/
     *10HIDENTIFIER,10HNAME      ,10HTITLE     ,
     *10HSUBTITLE  ,10HCOMPOSER  ,10HPUBLISHER ,
     *10HNUMBER    ,10HTEMPO     ,10HMETRONOME ,
     *10HCOMMENTS  /
      NOLENG = 0
      CALL BSTORE(KNTEXT)
C     IDENTIFIER MAY HAVE TRAILING SPACES
      CALL MBS(KSPACS,K1,MTEXT,K1,K4)
C     MINIMUM LENGTH FOR IDENTIFIER AND NAME
      LMIN = 1
```

```
C     INPUT IDENTIFIER, NAME AND OPTIONAL TEXT
      LL = KNTEXT + 2
      DO 70 L=1,LL
      CALL BREAK(63)
    3 CALL RQTXT(LTLENG(L),LTEXT(3*L-2),LMIN)
      IF (L-2) 4,5,6
C     FILE IDENTIFIER
    4 CALL MBS(MTEXT,K1,MIDENT,K1,K4)
      GOTO 7
C     FILE NAME
    5 CALL BMOVE(MTEXT,MNAME)
C     MAKE REMAINING TEXT ITEMS OPTIONAL
      LMIN = 0
C     OTHER TEXT ITEMS
    6 INTYPE = 3
      CALL ICODE
    7 CONTINUE
   70 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE RQTXT(NLENG,NTEXT,NMIN)
C     DISPLAY MESSAGE IN FIRST *NLENG* CHARACTERS
C     OF *NTEXT* AND INPUT TEXT STRING INTO
C     *MTEXT*.  *NMIN* IS MINIMUM LENGTH ALLOWED.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /KK/ KBAR,KBDOTS,KCLEF,KTIME,KKEY,
     * KTEXUP,KTEXDN,KTEXWD,KFLAT,KNATUR,KSHARP,
     * KTIE,KNDOT,KNREPT,KNSDEL,KSWICH
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TD/ IXC,IYC,IXB,IXR
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      DIMENSION NTEXT(1)
C     DISPLAY QUESTION
    3 CALL TMESS(NLENG,NTEXT)
      CALL TWT(K2,KQERY)
C     SET DISPLAY POSITION TO END OF QUESTION
      IXC = IXT + IXINC
      IYC = IYQ
C     ITEM TYPE IS TEXT BELOW
      INCODE = KTEXDN
      CALL TINPUT
      IF (NCHARS .GE. NMIN) GOTO 9
C     ERROR IF PLAIN C/R TYPED FOR COMPULSORY ITEM
      CALL REJECT
      GOTO 3
    9 RETURN
      END
!ASS (M:BO,D1,JHROM314)
!FORTRAN BO,NS
C
C
C     ***********************************************
C     *                                             *
C     *    SEGMENT 314 - INPUT OF NOTE ITEMS        *
C     *                                             *
C     ***********************************************
C
C
      SUBROUTINE NINPUT
C     INPUT NOTE ITEM.
      COMMON MC1(318),JIBAR,JOBAR,JBBUF
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /ED/ NEDBAR,NEDSTV,NELENG,JEDCMD
      COMMON /EE/ JEDIT,JEFREE,JEBACK,ISITEM
      COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSQZ,
     * ISREST,ISAHED,ISXPOS,ISEMB
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /KK/ KBAR,KBDOTS,KCLEF,KTIME,KKEY,
```

```
     * KTEXUP,KTEXDN,KTEXWD,KFLAT,KNATUR,KSHARP,
     * KTIE,KNDOT,KNREPT,KNSDEL,KSWICH
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /LK/ MBLINK(14),MFLINK(14),
     * MLIST1(14),MLIST2(14),MLIST3(14),MLIST4(14)
      COMMON /ND/ NXLENG,NSHIFT,JBXL,NEXTX
      COMMON /NH/ JNHEDS,KNHMAX
      COMMON /N0/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /PP/ ISPOSE,JTERM,ISDBUG,ISHQD,
     * ISGRD1,ISSAG,ISWIDE
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /RP/ ISAUTO,JEDCOD,JESEL,ISDISP
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION MHSPEC(4),MNSPEC(25)
      EQUIVALENCE(MHSPEC,JAXIDS),(MNSPEC,ISEXTR)
      DATA LSTAR/1H*/,LTREB/122/
C
C     NOTE ITEMS NOT ALLOWED IN HEADER RECORD
      IF (ISHDR .NE. 0) GOTO 7
C     SET INPUT TRANSPOSITION FLAG IF REQUIRED
      ISXPOS = ISPOSE
      IF (ISEDIT .EQ. 0) GOTO 2
C     EDITING - DECODE SELECTED ITEM FOR DEFAULTS
      IF (ISITEM .NE. 0) JESEL = ISITEM
      JIBAR = NBYTE(MLIST1,JESEL)
      ISCHEK = 0
      CALL IDCODE
      ISCHEK = 1
C     -CHANGE- EDIT-COMMAND RETAINS ALL ATTRIBUTES
C     OF SELECTED ITEM
      IF (JEDCMD.EQ.3 .OR. ISITEM.NE.0) GOTO 28
      GOTO 225
C     ERROR IF TIME COUNT EXCEEDED FOR MEASURE
    2 IF (JTIME .GE. JTMAX) GOTO 7
C       DECODE PREVIOUS NOTE ITEM TO SET DEFAULTS
   22 CALL NNFECH(JREPT)
      IF (JREPT .GT. 0) GOTO 26
C       RESET ALL ATTRIBUTES EXCEPT LENGTH AND PITCH
  225 DO 230 L=1,25
  230 MNSPEC(L) = 0
      ISAUTO = 0
      GOTO 28
   26 JREPT = JREPT - 1
   28 LPREV = 0
      JNHEDS = 0
      LREST = 0
      JPPCH = KBIG
C       SIMULATE C/R IF AUTOMATIC REPEAT
      IF (ISAUTO .NE. 0) INCODE = KEOD
C       BEGINNING OF NOTE SPECIFICATION
    3 IF (INCODE .EQ. KEOD) GOTO 54
      LENTRY = INCODE - 191
C       BRANCH INDEXED BY NOTE SPECIFICATION CODE
      GOTO (31,32,32,32,32,32,32,32,34,34,34,34,
     * 34,34,34,34,34,38,4,36,36,36,44,46,7,48,
     * 52,5,535,56,58,6,62),LENTRY
      GOTO 7
C
C       REST SPECIFICATION
   31 NHEDS = 0
      JNHEDS = 0
      LREST = 1
      GOTO 33
C       PITCH SPECIFICATION
   32 IF (ISJOY .NE. 0) GOTO 7
   33 JPITCH = LENTRY - 1
      GOTO 82
C       LENGTH SPECIFICATION
   34 JLENTH = LENTRY - 8
      GOTO 82
C
C       ACCIDENTALS SPECIFICATION
   36 JAXIDS = 8
```

```
      IF (INCODE .NE. KNATUR) GOTO 37
C     SET NATURAL BIT
      JAXIDS = 24
      CALL OKNEXT
      IF (INCODE .NE. KSHARP .AND. INCODE .NE.
     *                    KFLAT) GOTO 375
   37 LINC = 2 * (INCODE-KNATUR)
      JAXIDS = JAXIDS + LINC
      L1 = INCODE
      CALL OKNEXT
      IF (INCODE .NE. L1) GOTO 375
      JAXIDS = JAXIDS + LINC
      CALL OKNEXT
  375 IF (INCODE .EQ. KTEXDN) GOTO 376
      IF (INCODE .EQ. KTEXUP) GOTO 377
      GOTO 84
C     SET BIT FOR PARENTHETICAL ACCIDENTALS
  376 JAXIDS = JAXIDS + 64
      GOTO 82
C     SET BIT FOR ACCIDENTALS ABOVE OR BELOW NOTE
  377 JAXIDS = JAXIDS + 32
      GOTO 82
C
C     INCREMENT OCTAVE
   38 IF (JOCTAV .GE. 8) GOTO 7
      JOCTAV = JOCTAV + 1
      GOTO 82
C     DECREMENT OCTAVE
    4 IF (JOCTAV .LE. 0) GOTO 7
      JOCTAV = JOCTAV - 1
      GOTO 82
C
C     TIE TO FOLLOWING NOTE
   44 ISTIE = 1
      GOTO 82
C
C     STEM SIGN
   46 ISSTEM = 1
      GOTO 82
C
C     INPUT EXTRAS
   48 CALL NEXTRA
      GOTO 84
C
C     NOTE REPEAT
    5 IF (ISEDIT .NE. 0) GOTO 7
      ISAUTO = 0
      JREPT = 0
      CALL OKNEXT
      IF (MCHAR(1) .NE. LSTAR) GOTO 516
C     INPUT REPEAT COUNT FOR AUTOMATIC REPEAT
      CALL NUMBER(JREPT,9,KO)
C     ERROR IF NO NOTES IN CURRENT MEASURE
      IF (NNINCB .EQ. 0) GOTO 7
      IF (INCODE .EQ. KNREPT) ISAUTO = 1
      LNOTES = 0
C     SKIP TO FIRST REPEATED NOTE IN MEASURE
      LSKIP = NNINCB - JREPT
      JEDCOD = 1
      ISCHEK = 0
  512 IF (LNOTES .GE. LSKIP) GOTO 515
  514 JEDCOD = NFLINK(JEDCOD)
      CALL EDCODE
      IF (ITTYPE .NE. 4) GOTO 514
      LNOTES = LNOTES + 1
      GOTO 512
  515 ISCHEK = 1
      GOTO 2
C     REPEAT PREVIOUS NOTE ITEM
  516 CALL NNFECH(KO)
      GOTO 28
C
C     DOTS
   52 CALL OKNEXT
      IF (INCODE .NE. KNDOT) GOTO 53
      JDOTS = 2
      GOTO 82
   53 JDOTS = 1
      GOTO 84
C
```

```
C       CHORD SIGN NOT ALLOWED IF ITEM IS REST
  535 IF (JPITCH .EQ. 0) GOTO 7
C       PROCESS NOTE HEAD
   54 IF (ISJOY .EQ. 0 .OR. LREST .NE. 0) GOTO 545
C       EVALUATE PITCH VALUE FOR JOYSTICK INPUT
       L1 = (JY-IYS+2)/IYINC-JCLEF+(JCLEF-LTREB)
      *                          *ISPOSE-NSHIFT+153
       JOCTAV = (L1-1) / K7
       JPITCH = L1 - 7 * JOCTAV
       GOTO 547
  545 L1 = 7 * JOCTAV + JPITCH
  547 IF (ISCHOR.EQ.C .OR. L1.LT.JPPCH) GOTO 55
C       ENSURE PITCH LOWER THAN PREVIOUS HEAD IF ANY
       IF (JOCTAV .LE. 1) GOTO 7
       JOCTAV = JOCTAV - 1
       GOTO 545
C       SAVE PITCH FOR COMPARISON WITH NEXT HEAD
   55 JPPCH = L1
C       ENCODE DATA FOR NOTE-HEAD
       CALL HCODE
       IF (ISERR .NE. 0) GOTO 7
C       END OF ITEM IF CARRIAGE-RETURN WAS TYPED
       IF (INCODE .EQ. KEOD) GOTO 66
C       SET DEFAULT SPECIFICATIONS FOR NEXT HEAD
       DO 5520 L=1,4
 5520 MHSPEC(L) = 0
       IF (NHEDS .GT. JNHEDS) CALL HDCODE(JNHEDS+1)
       ISCHOR = 1
       GOTO 82
C
C       CANCEL PREVIOUS SPECIFICATION
C       *LPREV* = 20,21,22 ACCIDENTALS, 23 TIE,
C       24 STEM, 27 DOTS, 29 CHORD
   56 IF (LPREV.LT.20 .OR. LPREV.GT.23) GOTO 57
       LPREV = MAX0(LPREV,22)
       MHSPEC(LPREV-21) = 0
       GOTO 82
   57 IF (LPREV .NE. 27) GOTO 573
C       RESET DOTS
       JDOTS = 0
       GOTO 82
  573 IF (LPREV .NE. 24) GOTO 575
C       RESET STEM SIGN
       ISSTEM = 0
       GOTO 82
  575 IF (LPREV .NE. 29) GOTO 577
C       RESET CHORD TO SINGLE-HEAD NOTE
       JNHEDS = 0
       GOTO 82
C
C       SIMULATE INTERRUPT TO CANCEL THIS ITEM
  577 ISERR = 2
       GOTO 88
C
C       INCREMENT NOTE PITCH
   58 IF (JPITCH .EQ. 0) GOTO 7
       JPITCH = JPITCH + 1
       IF (JPITCH .LE. 7) GOTO 82
       JPITCH = 1
       GOTO 38
C       DECREMENT NOTE PITCH
    6 IF (JPITCH .EQ. 0) GOTO 7
       JPITCH = JPITCH - 1
       IF (JPITCH .GT. 0) GOTO 82
       JPITCH = 7
       GOTO 4
C       INPUT OCTAVE NUMBER IN RANGE 0 TO 8
   62 JOCTAV = 0
       CALL NUMBER(JOCTAV,K8,K0)
       GOTO 84
C
C       PROCESS NOTE ITEM
   66 IF (JNHEDS .GT. NHEDS) NHEDS = JNHEDS
       ISEXTR = 0
       DO 670 L=1,4
  670 IF (MEXTRA(L) .NE. 0) ISEXTR = 1
       L1 = ISCHOR
       IF (JPITCH .EQ. 0 .OR. ISSTEM .NE. 0) L1 = 1
       ITEM = 128+64*ISEXTR+32*L1+3*JLENTH+JDOTS
       CALL ACCEPT
```

```
      GOTO 88
C
C     SOUND BELL FOR INPUT ERROR
    7 CALL REJECT
      ISERR = 1
      JREPT = 0
      ISAUTO = 0
      JBBUF = 0
C     EXIT IF FIRST SPECIFICATION FOR THIS ITEM
      IF (LPREV .EQ. 0) GOTO 88
      CALL RDCHAR
      GOTO 3
C
C     END OF NOTE SPECIFICATION
   82 CALL OKNEXT
   84 LPREV = LENTRY
      GOTO 3
   88 IF (JREPT .EQ. 0) ISAUTO = 0
      ISXPOS = 0
      RETURN
      END
C
C
      SUBROUTINE NEXTRA
C     INPUT ORNAMENTS, EXPRESSION MARKS ETC.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /KK/ MKK1(14),KNSDEL
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION LMAXID(2),LMAX(2),LMECOD(7)
      DATA LNECOD/27/
      DATA LMAX(1)/14/,LMAX(2)/4/,LCOMMA/1H,/
      DATA LMECOD/27H¬INZTMEUX12&X&*JFDA=:.>^-SP/
      LIM = LNECOD
      LAXNO = 0
      LMAXID(1) = 0
      LMAXID(2) = 0
      LISBIT = 1
C     READ NEXT CHARACTER
   18 CALL OKNEXT
C     IDENTIFY CHARACTER
    2 L1 = NBYTE(MCHAR,K1)
      DO 220 LENTRY=1,LIM
  220 IF (ISSAME(L1,NBYTE(LMECOD,LENTRY)) .NE. 0)
     *                              GOTO 3
C     INVALID EXTRA CHARACTER
   24 CALL NOGOOD
      GOTO 2
C     CHARACTER IDENTIFIED
    3 GOTO (62,62,62,62,62,62,62,62,62,61,61,55,
     *55,55,32,34,38,42,42,4,4,4,5,5,5,5,5),LENTRY
      GOTO 24
C     NOTE FRACTIONING
   32 L = 0
      LBIT = 0
      GOTO 36
C     TREMOLO
   34 L = -1
      LBIT = 8
   36 IF (LISBIT .NE. 0) GOTO 37
      MEXTRA(2) = 0
      GOTO 8
   37 CALL NUMBER(L,K5,KO)
      MEXTRA(2) = LBIT + L
      GOTO 85
C     FINGERING
   38 L = 0
      LL = 0
      IF (LISBIT .NE. 0) GOTO 39
      MHEXT(1) = 0
      GOTO 8
   39 CALL NUMBER(L,K5,KO)
      IF (MCHAR(1) .EQ. LCOMMA)
     *                    CALL NUMBER(LL,K5,KO)
      MHEXT(1) = 5 * LL + L
```

```
      GOTO 85
C     MEZZO-STACCATO, STACCATISSIMO, STACCATO
C     *IAND* MAKES THESE MUTUALLY EXCLUSIVE
    4 MEXTRA(3) = IAND(MEXTRA(3),56)
C     ARPEGGIOS
   42 LBIT = LENTRY - 14
      LPTR = 3
      GOTO 7
C     ATTACK, VEE, AGOGIC, SWELL, PAUSE
    5 LBIT = LENTRY - 19
      LPTR = 4
      GOTO 7
C     ACCIDENTALS FOR ORNAMENTS
   55 LAXNO = LAXNO + 1
      LMAXID(LAXNO) = LENTRY - 11
      LIM = LMAX(LAXNO)
      GOTO 18
C     ACCIACCATURA, APPOGGIATURA
   61 ISSMAL = LISBIT * (LENTRY-9)
C     TURN, TRILL, MORDENT
   62 MHEXT(2) = LISBIT * LENTRY
      MHEXT(3) = LISBIT * LMAXID(1)
      MHEXT(4) = LISBIT * LMAXID(2)
      GOTO 8
C     SET OR RESET REQUIRED BIT
    7 CALL STBIT(LISBIT,MEXTRA(LPTR),LBIT)
    8 CALL OKNEXT
   85 IF (LISBIT .EQ. 0 .OR. INCODE .NE. KNSDEL)
     *                          GOTO 9
      LISBIT = 0
      GOTO 3
    9 RETURN
      END
C
C
      SUBROUTINE NNFECH(NNO)
C     SETS CURRENT NOTE ATTRIBUTE VALUES EQUAL TO
C     THOSE OF PREVIOUS NOTE IF *NNO* IS 0, OR OF
C     NEXT NOTE ITEM IN *MIBAR* OTHERWISE.
      COMMON MC1(318),JIBAR
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NCP,NDL
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /RP/ ISAUTO,JEDCOD,JESEL,ISCISP
      ISCHEK = 0
      IF (NNO .NE. 0) GOTO 5
      IF (NNINCB .EQ. 0 .OR. JPN .EQ. 0) GOTO 2
C     USE PREVIOUS NON-REST NOTE OF THIS MEASURE
      JEDCOD = JPN
      CALL EDCODE
      GOTO 8
C     SEARCH PREVIOUS MEASURES FOR NON-REST NOTE
    2 LBAR = NOB
   25 LBAR = LBAR - 1
      IF (LBAR .LE. 0) GOTO 4
      CALL BFIND(LBAR,JLINE)
      LPTR = -1
    3 CALL IDCODE
      IF (ISEND .NE. 0) GOTO 35
      IF (ITTYPE .EQ. 4 .AND. JPITCH .NE. C)
     *                          LPTR = ITPTR
      GOTO 3
   35 IF (LPTR .LT. 0) GOTO 25
      JIBAR = LPTR
      CALL IDCODE
      GOTO 8
C     DEFAULT SPECIFICATIONS FOR FIRST NOTE OF
C     STAVE = SEMIQUAVER MIDDLE C
    4 JOCTAV = 4
      JLENTH = 5
      JPITCH = 1
      GOTO 8
C     GET NOTE STRAIGHT FROM *MOBAR*
```

```
    5 JEDCOD = NFLINK(JEDCOD)
      CALL EDCODE
      IF (ITTYPE .NE. 4) GOTO 5
      GOTO 8
    8 ISCHEK = 1
      RETURN
      END
C
C
      SUBROUTINE EDCODE
C     DECODE ITEM OF CURRENT MEASURE WHOSE EDIT
C     TABLE INDEX IS *JEDCOD*.
      COMMON MC1(319),JOBAR,JBBUF
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /LK/ MBLINK(14),MFLINK(14),
     * MLIST1(14),MLIST2(14),MLIST3(14),MLIST4(14)
      COMMON /RP/ ISAUTO,JEDCOD,JESEL,ISDISP
      JBBUF = 1
      JOBAR = NBYTE(MLIST1,JEDCOD)
      CALL IDCODE
      JBBUF = 0
      RETURN
      END
C
C
      SUBROUTINE HCODE
C     ENCODE DATA FOR NOTE HEAD.
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /KM/ KZEROS(2),KMRSIZ,MREST(1)
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /N0/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JCOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /NH/ JNHEDS,KNHMAX
      COMMON /PP/ ISPOSE,JTERM,ISDBUG,ISHQD,
     * ISGRD1,ISSAG,ISWIDE
      COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /WS/ L1,L2,L3,JBYTE
      EQUIVALENCE(NORNAM,MHEXT(2)),(NAX1,MHEXT(3))
      EQUIVALENCE(NAX2,MHEXT(4))
C
C     ERROR IF ALREADY MAXIMUM NUMBER OF HEADS
      IF (JNHEDS .GE. KNHMAX) GOTO 7
C     EVALUATE DURATION OF CURRENT NOTE ITEM
      CALL NLENTH
C     ERROR IF TIME COUNT EXCEEDED UNLESS THIS
C     IS PART OF A MEASURE-REPEAT
      IF (ISEDIT .EQ. 0 .AND. ISRPT .EQ. 0 .AND.
     *       JTIME + JNDUR .GT. JTMAX) GOTO 7
      L3 = KHS * JNHEDS + 1
      IF (JPITCH .EQ. 0) GOTO 75
      LPICH = 7 * JOCTAV + JPITCH
      IF (ISPOSE .EQ. 0) GOTO 4
C     TRANSPOSE FROM TREBLE CLEF TO TRUE PITCH
      L1 = NBYTE(MCMEM,K4)
      IF (L1 .GT. 0) L1 = (L1/128) * 14 - 7
      LPICH = LPICH - (JCLEF - 122 + L1)
    4 IF (LPICH .LE. 0 .OR. LPICH .GT. 63) GOTO 7
C     VALUE FOR CONTINUATION BIT IN FINGERING BYTE
      LCONT = 0
      IF (MHEXT(1) .GT. 0 .AND. MHEXT(2) .GT. 0)
     *                              LCONT = 1
      CALL STBIT(LCONT,MHEXT(1),K2)
      CALL STBYTE(128*MINO(JAXIDS+ISTIE,K1)+
     * 64*MINO(MHEXT(1)+MHEXT(2),K1)+LPICH,MHD,L3)
      CALL STBYTE(128*ISTIE+JAXIDS,MHD,L3+1)
C     ENCODE FINGERING
      CALL STBYTE(MHEXT(1),MHD,L3+2)
C     ENCODE ORNAMENTS
      L1 = NORNAM
      IF (L1 .EQ. 0) GOTO 6
      L1 = L1 + 16 * NAX1 + 4 * NAX2 + 127
      IF (NORNAM .LE. 4) L1 = L1 + 64
    6 CALL STBYTE(L1,MHD,L3+3)
      GOTO 8
```

```
      7 ISERR = 1
        GOTO 9
C       ENCODE REST
     75 CALL MBS(KZEROS,K1,MHD,L3,K4)
      8 JNHEDS = JNHEDS + 1
        ISERR = 0
      9 RETURN
        END
C
C
        SUBROUTINE NCODE
C       ENCODE DATA FOR NOTE ITEM.
        COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
        COMMON /N0/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
       * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
       * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
        COMMON /WS/ L1,L2,L3,JBYTE
        DIMENSION LEXTRA(4)
        ISEXTR = 0
        DO 30 L=1,4
        L1 = 5 - L
        L2 = MEXTRA(L1)
        IF (L2 .EQ. 0) GOTO 3
C       ADD TYPE FIELD
        L2 = L2 + 32 * (L1-1)
C       ADD CONTINUATION BIT IF REQUIRED
        IF (ISEXTR .NE. 0) L2 = L2 + 128
        ISEXTR = 1
      3 LEXTRA(L1) = L2
     30 CONTINUE
C       STORE NON-ZERO EXTRAS
        DO 40 L=1,4
     40 CALL BSTORE(LEXTRA(L))
        IF (JPITCH.EQ.C .OR. ISSTEM.NE.0) GOTO 72
C       NOT REST OR STEM SIGN
        IF (NHEDS .GT. 1) CALL BSTORE(NHEDS)
        DO 70 L=1,NHEDS
        LPTR = (L - 1) * KHS
        DO 60 LL=1,KHS
     60 CALL BSTORE(NBYTE(MHD,LPTR+LL))
     70 CONTINUE
        GOTO 9
C       REST OR STEM SIGN
     72 CALL BSTORE(128*ISTIE+64*ISSTEM+1)
      9 CONTINUE
        END
C
C
        SUBROUTINE EDIBOB
C       COPY INPUT MEASURE TO OUTPUT MEASURE,
C       REDUCING ALL NOTE PITCHES BY *NNDOWN* UNITS.
        COMMON MC1(188),NILENG,MIBAR(64),NOLENG,
       * MOBAR(64)
        COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSQZ,
       * ISREST,ISAHED,ISXPOS,ISEMB
        COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
       *    ISERR,ISNXST,ISINPT
        COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
        COMMON /IV/ INTYPE,ISBAR
        COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
        COMMON /MR/ NNDOWN
        COMMON /NH/ JNHEDS,KNHMAX
        COMMON /N0/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
       * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
       * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
        COMMON /PP/ ISPOSE,JTERM,ISDBUG,ISHOD,
       * ISGRD1,ISSAG,ISWIDE
        ISXPOS = ISPOSE
        NOLENG = 0
        IF (NNDOWN .EQ. 0) GOTO 7
      3 CALL IDCODE
        IF (ISEND .NE. 0) GOTO 8
        IF (ITTYPE .NE. 4 .OR. JPITCH .EQ. 0) GOTO 6
        JNHEDS = 0
        DO 40 L=1,NHEDS
        CALL HDCODE(L)
        LPICH = 7 * JOCTAV + JPITCH - NNDOWN
        IF (JPITCH.GT.0 .AND. LPICH.LT.64) GOTO 36
        CALL REJECT
        LPICH = 1
```

```
   36 JOCTAV = (LPICH-1) / K7
      JPITCH = LPICH - 7 * JOCTAV
      CALL HCODE
   40 CONTINUE
C     RE-ENCODE NON-REST NOTE ITEM
      INTYPE = 4
      CALL ICODE
      GOTO 3
C     COPY ENCODED ITEM DIRECTLY TO OUTPUT MEASURE
    6 CALL MBS(MIBAR,ITPTR+1,MOBAR,NOLENG+1,
     *                          ITLENG)
      NOLENG = NOLENG + ITLENG
      GOTO 3
C     STRAIGHT COPY WITH NO PITCH CHANGE
    7 CALL BMOVE(MIBAR,MOBAR)
    8 ISEND = 0
      ISXPOS = 0
      RETURN
      END
!ASS (M:B0,D1,JHR0M315)
!FORTRAN B0,NS
C
C
C     ***************************************** *
C     *                                         *
C     *     SEGMENT 315 - INPUT OF BRAILLE ITEMS  *
C     *                                         *
C     *****************************************
C
C
      SUBROUTINE BINPUT
C     INPUT BRAILLE ITEM.
      COMMON MC1(253),NOLENG
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /IV/ INTYPE,ISBAR
      COMMON /KK/ KBAR,KBOOTS,KCLEF,KTIME,KKEY,
     * KTEXUP,KTEXDN,KTEXWD,KFLAT,KNATUR,KSHARP,
     * KTIE,KNOOT,KNREPT,KNSDEL,KSWICH
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /M0/ JIMODE,JOMODE
      DIMENSION LISOOT(8),LMBRL(64)
      DATA LMBRL/15*0,64,16*0,
     8Z01030919,8Z110B1B13,8Z0A000000,8Z00000000,
     9Z1A05070D,8Z1D150F1F,8Z17000000,8Z0C000000,
     AZ000E1E25,8Z273A2D3D,8Z35000000,8Z00000000,
     BZ00000000,8Z00000000,8Z00000000,8Z0C000000,
     CZ01030919,8Z110B1B13,8Z0A000000,8Z00000000,
     DZ1A05070D,8Z1D150F1F,8Z17000000,8Z0C000000,
     EZ000E1E25,8Z273A2D3D,8Z35000000,8Z00000000,
     FZ00000000,8Z00000000,8Z00000000,8Z00000000/
      DO 20 L=1,8
   20 LISOOT(L) = 0
    3 LPREV = 0
   32 IF (INCODE .EQ. KEOD) GOTO 5
      IF (INCODE .EQ. KNSDEL) GOTO 34
      IF (INCODE .EQ. KBAR) GOTO 72
      IF (INCODE .EQ. 1) GOTO 74
      L = NBYTE(MCHAR,K1) - 240
      IF (ISEDIT.EQ.0 .AND. NOLENG.GE.40) GOTO 7
C     PERMITTED RANGE 1-6 FOR FORMATTED BRAILLE,
C     1-8 FOR FORMATTED BRAILLE
      IF (L.LE.0 .OR. L.GT.12-2*JIMODE) GOTO 36
      LISOOT(L) = 1
      LPREV = L
      CALL OKNEXT
      GOTO 32
   34 IF (LPREV .EQ. 0) GOTO 36
      LISOOT(LPREV) = 0
      CALL OKNEXT
      GOTO 3
   36 ITEM = NBYTE(LMBRL,NBYTE(MCHAR,K1))
      IF (ITEM .NE. 0) GOTO 8
      CALL NOGOOD
      GOTO 32
C
C     ENCODE BRAILLE ITEM
    5 ITEM = 0
```

```
      DO 60 L=1,8
   60 ITEM = ITEM + NPOWER(L-1) * LISDOT(L)
      IF (ITEM .EQ. 0) ITEM = 64
      GOTO 8
    7 CALL REJECT
      ISERR = 1
      GOTO 9
C     END OF MEASURE OR LINE
   72 ISBAR = 1
      GOTO 82
C     END OF INPUT
   74 ISEND = 1
    8 ISBAR = 0
   82 CALL ACCEPT
      ISERR = 0
    9 RETURN
      END
!JOB JH,COMPILE MUSIC PROGRAM SEGMENT 32
!EXTRABGD 6800
!ASS (M:80,D1,JHROM32)
!FORTRAN 80,NS
C
C
C     ******************************************
C     *                                        *
C     *          SEGMENT 32 - DISPLAY          *
C     *                                        *
C     ******************************************
C
C
      BLOCK DATA
C     THIS SUBPROGRAM IS PROGRAM-GENERATED.
      COMMON /XC/ MCORD1(20)
      DATA MCORD1/
     *8Z0006000E,8Z0C180020,8Z002C003C,8Z004436D2,
     *8Z0A8C9C8,8ZB7C44ABA,8Z40D2C0A8,8ZC9C8B744,
     *8Z40CCC0B0,8ZC5C040B8,8ZC0D0BB40,8Z43CCC0A8,
     *8Z44C0C0D8,8Z39B7CAC2,8Z40B8B63E,8Z40BBCACA,
     *8Z36C0CA36,8Z36CAD4AC,8Z40D4AC2C,8Z00000000/
      END
C
C
      SUBROUTINE MDISP1
C     DISPLAY SELECTED BARS OF CURRENT SECTION.
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,K8IG
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /MO/ JIMODE,JOMODE
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NCP,NDL
      COMMON /SE/ JCMD,KSEGS(7)
      EQUIVALENCE(JADCMD,K5)
C     OBTAIN FIRST MEASURE OF FIRST BAR TO DISPLAY
      CALL BFIND(JBS,K1)
      IF (JCMD.EQ.21 .AND. JIMODE.NE.1) GOTO 65
C     INITIALISE DISPLAY
      JBAR = JBS
      IF (JCMD .EQ. JADCMD .AND. NSSEL .GT. NDL)
     *                          JBAR = JBAR + 1
      CALL DINIT
      IF (JSSEL .EQ. 1) GOTO 6
C     OBTAIN NEXT SELECTED MEASURE
    5 CALL NXSEL
    6 IF (ISEND .NE. 0) GOTO 7
      JBAR = NIB
C     NO DISPLAY FOR ADD COMMAND IF MORE THAN ONE
C     PAGE REQUIRED FOR BAR
      IF (JBAR .GT. 0 .AND. (JCMD .NE. JADCMD .OR.
     *            NSSEL .LE. NDL)) CALL DIBAR
      IF (JLINE .EQ. NLINES) JBAR = JBAR + 1
      GOTO 5
C     PRINT BRAILLE ON LINEPRINTER
   65 CALL LSEG(325)
      CALL BPRINT
      GOTO 9
    7 IF (JCMD .EQ. JADCMD) GOTO 9
C     DISPLAY ANY UNCLOSED SLURS ETC.
      CALL LSEG(322)
      CALL ESTAFF
```

```
    9 RETURN
      END
C
C
      SUBROUTINE ESETUP
C     INITIALISE EDIT TABLE, DISPLAYING CONTENTS
C     OF INPUT MEASURE IF EDITING.
      COMMON MC1(318),JIBAR
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      CALL ECLEAR(K1)
      NNINCB = 0
      JTIME = 0
      IF (ISEDIT .NE. 0) CALL DIBAR
      JIBAR = 0
      RETURN
      END
C
C
      SUBROUTINE DIBAR
C     DISPLAY INPUT MEASURE OR FILE HEADER RECORD.
      COMMON MC1(318),JIBAR
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,ISCALE
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MO/ JIMODE,JOMODE
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      JIBAR = 0
      IF (ISHDR .EQ. 0) GOTO 2
C     DISPLAY FILE HEADER
      CALL DHDR
      GOTO 9
C     DECODE NEXT ITEM
    2 CALL IDCODE
      IF (ISEND .NE. 0) GOTO 7
C     DISPLAY ITEM
      CALL IDISP
      IF (ISEDIT .EQ. 0) GOTO 2
C     ALLOW EXTRA SPACE BETWEEN NOTES OR BRAILLE
C     CELLS IF EDITING
      IF (ITTYPE .LT. 4) GOTO 4
      IXM = IXM + 40
      IXN = IXN + 40
      IF (IXN .GE. JXMAX-50) ISNXST = 2
C     INSERT ITEM AT END OF EDIT TABLE LIST 1
    4 CALL IINSRT(K1)
      GOTO 2
C     DISPLAY END-OF-MEASURE
    7 ISEND = 0
      CALL DEBAR
    9 RETURN
      END
C
C
      SUBROUTINE DINIT
C     INITIALISATION FOR MUSIC DISPLAY.
      COMMON /DI/ MBEAMX(8),MBMYH(8),MBMYL(8),
     * MBEAML(4),JBEAM,ISBEAM
      COMMON /DN/ JSDIR,JFDIR,JLEV,JALEV,JPVLEV,
     * IXSM,IYSM,IYSMAX,IYSMIN,MXTEXT(3)
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /JD/ JSTAFF,JDLINE,JDHIGH,JDLOW
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /MO/ JIMODE,JOMODE
      COMMON /ND/ NXLENG,NSHIFT,JBXL,NEXTX
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /SE/ JCMD,KSEGS(7)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      IF (JCMD .EQ. 21) CALL PRINIT
      JFDIR = 3
```

```
      JSTAFF = NDP
      JDHIGH = 0
      JDLINE = NSSEL
      NSHIFT = 0
      ISBEAM = 0
      CALL CHSTAV(K1)
      ISNXST = 3
C     DELAY NEW PARALLEL UNTIL FIRST ITEM HAS
C     BEEN DECODED IF EXECUTING DISPLAY COMMAND
      IF (JCMD .EQ. 6) GOTO 5
C     DISPLAY NEW PARALLEL
      CALL LSEG(322)
      CALL NXSTAF
    5 IF (JIMODE .EQ. 1) GOTO 9
      IXN = 0
      IYS = 700
    9 RETURN
      END
C
C
      SUBROUTINE DEBAR
C     DISPLAY END-OF-MEASURE.
      COMMON MC1(188),NILENG
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /MO/ JIMODE,JOMODE
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      LITEM = 0
      IF (JIMODE .EQ. 1 .AND. (ISINPT .NE. 0 .OR.
     * ITEM .LT. 5 .OR. ITEM .GT. 10)) LITEM = 1
      ITEM = LITEM
      ITTYPE = 1
      CALL IDISP
      RETURN
      END
C
C
      SUBROUTINE IDISP
C     DISPLAY CURRENT ITEM.
      COMMON /DI/ MBEAMX(8),MBMYH(8),MBMYL(8),
     * MBEAML(4),JBEAM,ISBEAM
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      DIMENSION LMESS(5)
      DATA LMESS/17HITEM UNRECOGNISED/
      IF (ISNXST.EQ.0 .OR. ISINPT.NE.0) GOTO 18
C     NEW PARALLEL PRECEDES ITEM DURING DISPLAY
      CALL LSEG(322)
      CALL NXSTAF
C     LEAVE OUT CLEF SIGN IF FIRST ITEM OF STAVE
      IF (ITEM .EQ. 33) GOTO 9
C     DONT ALLOW SIGNS TO GO OFF RH SIDE OF PAGE
   18 IF (IXN .GT. JXMAX) IXN = JXMAX
      CALL LSEG(320+ITTYPE)
      GOTO (2,3,4,5,6),ITTYPE
C
C     DISPLAY CONTROL ITEM
    2 CALL CDISP
      GOTO 7
C     DISPLAY SEPARATE SIGN ITEM
    3 CALL SDISP
      GOTO 7
C     DISPLAY TEXT ITEM
    4 CALL TDISP
      GOTO 8
C     DISPLAY NOTE ITEM
    5 CALL NDISP
      GOTO 8
C     DISPLAY BRAILLE ITEM
    6 CALL BRDISP
      GOTO 8
    7 IF (ISERR .EQ. 0) GOTO 8
C     DISPLAY -ITEM UNRECOGNISED- MESSAGE
```

```
      CALL TMESS(17,LMESS)
      IXE = IXT
      IYE = IYT = IYS
      CALL TWT1(ITEM)
    8 ISERR = 0
      IF (ISBEAM .EQ. 0) GOTO 85
C     NEW PARALLEL FOLLOWS ITEM DURING INPUT
      CALL LSEG(321)
      CALL DBEAM
   85 IF (ISNXST .EQ. 0 .OR. ISINPT .EQ. 0) GOTO 9
      CALL LSEG(322)
      CALL NXSTAF
    9 RETURN
      END
C
C
      SUBROUTINE DAXID(N)
C     DISPLAY ACCIDENTAL SIGN AT (*IXN*,*IYN*).
C     *N* = 1 DOUBLE FLAT, 2 FLAT, 3 NATURAL,
C           4 SHARP, 5 DOUBLE SHARP.
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /XC/ MCORD1(20)
      DIMENSION LMHYAX(5)
      DATA LMHYAX(1)/0/,LMHYAX(2)/2375/,
     *LMHYAX(3)/2374/,LMHYAX(4)/2373/,LMHYAX(5)/0/
      IXT = IXN
      IYT = IYN
      CALL DSYMB(LMHYAX(N),K1,N,MCORD1)
      RETURN
      END
C
C
      SUBROUTINE TWCH
C     WRITE CHARACTER *MCHAR(1)* AT (*IXT*,*IYT*).
      CALL TPOSN
      CALL ECHO
      RETURN
      END
C
C
      SUBROUTINE TTEXT(NLENG,NTEXT)
C     DISPLAY TEXT STRING IN *NTEXT*.  MAGNITUDE
C     OF *NLENG* IS NUMBER OF CHARACTERS, SIGN IS
C     +VE ABOVE STAVE, -VE BELOW STAVE.
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION NTEXT(1)
      L2 = NLENG
      IF (L2 .GE. 0) GOTO 2
      L3 = -5
      L2 = - L2
      GOTO 4
    2 L3 = 11
    4 CALL TTEXT1(L2,NTEXT,L3)
      RETURN
      END
C
C
      SUBROUTINE TTEXT1(NLENG,NTEXT,NY)
C     WRITE TEXT STRING ABOVE OR BELOW STAVE.
C     *NLENG* IS LENGTH OF TEXT, *NTEXT* IS TEXT,
C     *NY* GIVES VERTICAL POSITION.
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      DIMENSION NTEXT(1)
    4 IXE = MAXO(IXN,IXM)
      CALL TWRITE(IXE+IXINC,IYS+NY*IYINC,
     *                              NLENG,NTEXT)
      IYE = IYT + 6 - IYS
      RETURN
      END
C
C
      SUBROUTINE DDOT(NX,NY)
C     DISPLAY DOT AT *NX*, LEVEL *NY* OF CURRENT
C     STAVE, MOVING IT UP ONE LEVEL IF ON A LINE.
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
```

```
      IYT = IYS + NY * IYINC - 2
      IF (NY/K2 * 2 .EQ. NY) IYT = IYT + IYINC
      CALL TWRITE(NX,IYT,K1,KDOT)
      RETURN
      END
C
C
      SUBROUTINE DCROSS
C     DISPLAY CROSS FOR DELETED ITEM.
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /XC/ MCORD1(20)
      IXT = IXE
      IYT = IYE
      CALL DSYMB(K0,K1,K6,MCORD1)
      RETURN
      END
C
C
      SUBROUTINE DSYMB(NID,NSCALE,N,NSYMB)
C     DISPLAY SYMBOL STARTING AT (*IXT*,*IYT*).
C     *NID* = HERSHEY IDENTIFICATION NUMBER IF ANY
C             OTHERWISE ZERO,
C     *NSCALE* = SCALING FACTOR,
C     *N* = SYMBOL NUMBER IN CO-ORDINATE TABLE,
C     *NSYMB* = CO-ORDINATE TABLE.
      COMMON /DI/ MBEAMX(8),MBMYH(8),MBMYL(8),
     * MBEAML(4),JBEAM,ISBEAM
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /PP/ MPP1(3),ISHOD,MPP2(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /WS/ L1,L2,L3,JBYTE
C+    *XJ* DECLARED WRONG SIZE TO AVOID STORAGE
C+    ALLOCATION IN FOREGROUND VERSION...
      COMMON /XJ/ MHYTAB(1)
      DIMENSION LL(2),IXYT(2),NSYMB(1)
      EQUIVALENCE(IXYT,IXT)
      CALL TPOSN
      IF (NID .NE. 0 .AND. ISHOD .NE. 0) GOTO 25
      LPTR = NHW(NSYMB,N+1)
      LSCALE = NSCALE
      GOTO 3
   25 LPTR = NHW(MHYTAB,NID-2365)
      LSCALE = 1
    3 DO 60 L=1,2
      LPTR = LPTR + 1
      IF (NID .NE. 0 .AND. ISHOD .NE. 0) GOTO 4
C     USE LOW-QUALITY CO-ORDINATE
      JBYTE = NBYTE(NSYMB,LPTR)
      GOTO 5
C     USE HIGH-QUALITY HERSHEY CO-ORDINATE
    4 JBYTE = NBYTE(MHYTAB,LPTR)
    5 LL(L) = NBIT(K0)
      IXYT(L) = IXYT(L) + (JBYTE-64) * LSCALE
   60 CONTINUE
      IF (LL(1) .NE. 0) GOTO 7
      CALL TPOSN
      GOTO 8
    7 CALL TLINE
    8 IF (LL(2) .NE. 0) GOTO 3
      RETURN
      END
C
C
      SUBROUTINE TWRIT1(NX,NY,N)
C     DISPLAY INTEGER *N* AT POSITION (*NX*,*NY*).
      COMMON /DP/ IXSHIF,IYSHIF
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      IXT = NX = 2 * IXINC - IXSHIF
      IYT = NY - IYINC - IYSHIF
      CALL TPOSN
      CALL TWT1(N)
      RETURN
      END
!ASS (M:B0,D1,JHR0M321)
!FORTRAN B0,NS
C
C
```

```
C     ********************************************
C     *                                          *
C     *  SEGMENT 321 - DISPLAY OF CONTROL ITEMS  *
C     *                                          *
C     ********************************************
C
C
      BLOCK DATA
C     THIS SUBPROGRAM IS PROGRAM-GENERATED.
      COMMON /XD/ MCORD2(56)
      DATA MCORD2/
     *8Z00090016,8Z0C2C003A,8Z0014005C,8Z00620078,
     *8Z00A400BE,8Z40CC040D1,8ZC0C2C2C0,8ZC0BEBEC0,
     *8Z40CCC0C2,8ZC2C0C0BE,8ZBEC046A3,8ZC0F045C0,
     *8ZC090C1C0,8ZCCF0C1C0,8ZC010C0F0,8ZC1C0C090,
     *8ZC1C0C0F0,8Z45C0C090,8Z46D1C0C2,8ZC2C0C0BE,
     *8ZBEC040CC,8ZCCC2C2C0,8ZC0BEBE40,8ZC0F046C0,
     *8ZC010C0C3,8Z40C5C0C4,8Z40C5C0C4,8Z40C5C0C4,
     *8Z40C5C0C4,8Z4CC5C044,8Z46CCBAC3,8ZBAC0BABD,
     *8ZC0BAD8B4,8ZC0BABABD,8ZBAC0BAC3,8Z3AC0D8D8,
     *8Z40B5C0BE,8ZC2C0C0C2,8ZBEC028C0,8ZBEC0C0BE,
     *8ZC2C0C042,8Z46C2BEC5,8ZBCC3BCBD,8ZBEBBC2BB,
     *8ZC4BDC4C3,8ZC2C544C0,8ZACC04ACC,8ZC028C1BF,
     *8ZBEC0C0C2,8ZC2C0C0BE,8Z35C0C3C7,8ZC7C3C7BD,
     *8ZC3393CC0,8ZC4CAC436,8Z00000000,8Z00000000/
      END
C
C
      SUBROUTINE CDISP
C     DISPLAY CONTROL ITEM.
C     *ITEM* = 0 FOR END-OF-MEASURE WITHOUT SINGLE
C     BARLINE, 1 FOR SINGLE BARLINE.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /DI/ MBEAMX(8),MBMYH(8),MBMYL(8),
     * MBEAML(4),JBEAM,ISBEAM
      COMMON /DN/ JSDIR,JFDIR,JLEV,JALEV,JPVLEV,
     * IXSM,IYSM,IYSMAX,IYSMIN,MXTEXT(3)
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /JD/ JSTAFF,JDLINE,JDHIGH,JDLOW
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /MO/ JIMODE,JOMODE
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /ND/ NXLENG,NSHIFT,JBXL,NEXTX
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NOP,NDL
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TD/ IXC,IYC,IXB,IXR
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /TT/ JTTEMP,JXTEMP
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      COMMON /XD/ MCORD2(56)
      DIMENSION KTCC(12),LBMERR(8),LFOOT(2)
      DATA KTCC/2H; ,2H@ ,2H0S,2H00,2H0P,
     *     2HRH,2HLH,2H01,2H02,2H0=,2H0Y,2H0U/
      DATA LFOOT/8HFOOTNOTE/,LBMERR/
     * 31HBEAMING ERROR: NOT ENOUGH NOTES/
      EQUIVALENCE(KLH,K2)
C
      ISERR = 0
      IF (ITEM .EQ. 0) GOTO 4
      GOTO (4,2,3,18,45,45,45,45,45,5,6,6,6,
     *     66,66,66,66,62,64,64),ITEM
C     UNRECOGNISED ITEM
   18 ISERR = 1
      GOTO 9
C
C     IN=ACCORD
    2 IXN = IXR
      IXM = JXTEMP
C     CANCEL ANY NEW PARALLEL PENDING
      ISNXST = 0
      GOTO 35
C     SET-RETURN POINT FOR IN=ACCORD
    3 IXR = IXN
```

```
      JXTEMP = IXM
C     FORCE NOTE STEMS TO POINT UP OR DOWN
   35 JFDIR = ITEM - 1
      MCHAR(1) = KTCC(JFDIR)
C     REVERSE STEM DIRECTIONS UNLESS LEFT HAND
      IF (JPART .NE. KLH) JFDIR = 3 - JFDIR
      IF (ISEDIT .NE. 0) GOTO 8
      GOTO 9
C     END OF MEASURE
    4 ISAXST = 0
      IF (JIMODE .NE. 1) GOTO 41
C     PRINTED MUSIC MODE
      ISAXST = 0
      NEXTX = MAXO(IXN,MXTEXT(3),NEXTX)
      IXN = NEXTX + 16
      IF (IXN .GE. JXMAX - 4 * IXINC) IXN = JXMAX
      IF (ITEM .EQ. 0) GOTO 401
C     DISPLAY -BEAMING ERROR- IF TOO FEW NOTES
      IF (JBEAM .NE. 0) CALL TMESS(31,LBMERR)
      JBEAM = 0
C     DISPLAY SINGLE BARLINE
      IXT = IXN
      IYT = IYS
      CALL TPOSN
      IYT = IYS + 8 * IYINC
      CALL TLINE
C     NO STAVE CHANGE IF EDITING
  401 IF (ISEDIT .NE. 0) GOTO 404
      IF (JLINE .LT. JDLOW) GOTO 403
      IYS = JYMAX = ((JSTAFF-1)*NSSEL+1)*IYSTV
      IXB = IXN
      IF (NSSEL .LE. NDL .AND. IXB+JBXL .LT.
     *              JXMAX - 40) GOTO 404
C     SET FLAG TO FORCE NEW PARALLEL WITH WAIT
      ISAXST = 4
      GOTO 9
C     MOVE TO NEXT STAVE OF CURRENT PARALLEL
  403 IYS = IYS - IYSTV
  404 IXN = IXB + 16
      IYN = IYS
      IXM = IXN
      MXTEXT(1) = 0
      MXTEXT(2) = 0
      MXTEXT(3) = 0
      IXR = IXN
      JXTEMP = IXN
      JFDIR = 3
      GOTO 9
C     BRAILLE MODES - DISPLAY BAR NUMBER FOR FIRST
C     STAVE IF ROOM AND RESET *IYS*
   41 IF (NIS .EQ. 1 .AND. IYT .LT. JXMAX-50)
     *       CALL TWRIT1(JXMAX-IXINC,IYS,NIB)
C     MOVE TO START OF NEW LINE
      IXN = 0
      IYS = IYS - 50
      IYQ = MINO(IYQ,IYS-5*IYINC)
      IF (IYS .GT. 0) GOTO 9
C     NEW PAGE WITH WAIT IF SCREEN IS FULL
      CALL TPAGE(K1)
      IYS = 700
      GOTO 9
C     DISPLAY SPECIAL BARLINE
   45 IXE = IXN + IXINC
      IYE = 4 * IYINC
      IXT = IXN
      IYT = IYS
   46 CALL DSYMB(K0,K1,ITEM-4,MCORD2)
      IF (ITEM .NE. 8) GOTO 54
      ITEM = 7
      GOTO 46
C
C     START OF FOOTNOTE - DISPLAY OBLIQUE BARLINE
    5 IXT = IXN + IXINC
      IYT = IYS
      CALL TPOSN
      IXT = IXT + IXINC
      IYT = IYT + 8 * IYINC
      CALL TLINE
      IYT = IYT + 4 * IYINC
      IXE = IXT
```

```
      IYE = 12 * IYINC
      CALL TPOSN
      CALL TWT(K8,LFCOT)
   54 IXN = IXE + 2 * IXINC
      GOTO 9
C
C     DISPLAY SEGNO, ENCIRCLED CROSS, PAUSE SIGN
    6 IXE = IXN
      IYE = 11 * IYINC
      IXT = IXE
      IYT = IYE + IYS
      CALL DSYMB(K0,K1,ITEM-5,MCORD2)
      GOTO 9
C
C     CANCEL FORCE-NOTE-STEMS
   62 JFDIR = 3
      GOTO 65
C     FORCE NOTE STEMS TO POINT UP OR DOWN
   64 JFDIR = ITEM - 12
   65 IF (ISEDIT .EQ. 0) GOTO 9
C     SPECIAL DISPLAY CHARACTERS FOR EDITING
   66 MCHAR(1) = KTCC(ITEM-9)
C     DISPLAY CONTROL ITEM
    8 IXE = IXN
      IYE = -3 * ITEM * IYINC
      IF (ITEM .GE. 15) IYE = 11 * IYINC
      CALL TWRITE(IXE-IXINC,IYE+IYS-IYINC,K2,
     *                          MCHAR)
    9 RETURN
      END
C
C
      SUBROUTINE ESCAN
C     SCAN CURRENT MEASURE TO IDENTIFY NEAREST
C     DISPLAYED ITEM TO POSITION (*JX*,*JY*) ON
C     SCREEN.  IF AN ITEM IS IDENTIFIED, RETURNS
C     *JEDIT* = POINTER TO SELECTED ITEM,
C     *IXN* = X CO-ORDINATE OF ITEM IF NOTE ITEM,
C     OTHERWISE RETURNS
C     *JEDIT* = 1, *IXN* = X CO-ORDINATE OF LAST
C     NOTE IN MEASURE.
      COMMON MC1(318),JIBAR
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /EE/ JEDIT,JEFREE,JEBACK,ISITEM
      COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSGZ,
     * ISREST,ISAHED,ISXPOS,ISEMB
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /JO/ JDIST
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /LK/ MBLINK(14),MFLINK(14),
     * MLIST1(14),MLIST2(14),MLIST3(14),MLIST4(14)
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /WS/ L1,L2,L3,JBYTE
C
      IF (ISCHEK .EQ. 0) GOTO 2
      JTIME = 0
C     SET STATE FOR START OF MEASURE
      CALL MMOVE(MSMEM,MCMEM)
    2 JEDIT = 1
      LPTR = 1
      JPN = 0
      JCLEF = JECLEF
      ISCTK = 0
      JDIST = KBIG
C     OBTAIN NEXT ENTRY IN EDIT TABLE LIST 1
    3 JEDIT = NFLINK(JEDIT)
      IF (JEDIT .EQ. 1) GOTO 7
      JIBAR = NBYTE(MLIST1,JEDIT)
      LIXT = 4 * NBYTE(MLIST3,JEDIT) - JX
```

```
      LIYT = 4 * NBYTE(MLIST4,JEDIT) - JY
      LDIST = LIXT * LIXT + LIYT * LIYT
      CALL IDCODE
      IF (ITTYPE .NE. 4 .OR. JPITCH .EQ. 0) GOTO 5
      JPN = JEDIT
      IXN = LIXT + JX
    5 IF (LDIST .GT. JDIST) GOTO 3
C     SAVE INDEX OF NEAREST ITEM SO FAR
      LPTR = JEDIT
C     ALSO SQUARE OF DISTANCE
      JDIST = LDIST
      GOTO 3
    7 JEDIT = LPTR
      RETURN
      END
C
C
      SUBROUTINE SFLAG
C     SETS *ICFLAG*=FLAG FIELD FOR INPUT MEASURE.
      COMMON MC1(322),MFLAGS(3)
      COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSQZ,
     * ISKEST,ISAHED,ISXPOS,ISEMB
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /M0/ JIMODE,JOMODE
      COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      DIMENSION LCMEM(2)
      LOGICAL CBS
      IF (JIMODE .NE. 1) GOTO 8
C     SAVE CURRENT STATE VALUES
      CALL MMOVE(MCMEM,LCMEM)
      LIXN = IXN
      CALL ESCAN
      IXN = LIXN
      IF (ISCTK .EQ. 0 .AND.
     *  CBS(MSMEM,K1,MCMEM,K1,KMSIZE)) GOTO 5
C     SET FLAG FOR OVERALL STATE CHANGE
      ICFLAG = 1
      GOTO 6
C     SET FLAG FOR NO OVERALL STATE CHANGE
    5 ICFLAG = 0
C     RESTORE CURRENT STATE VALUES
    6 CALL MMOVE(LCMEM,MCMEM)
      GOTO 9
    8 CONTINUE
    9 RETURN
      END
C
C
      SUBROUTINE DBEAM
C     DISPLAY NOTE BEAMING.
      COMMON /DI/ MBEAMX(8),MBMYH(8),MBMYL(8),
     * MBEAML(4),JBEAM,ISBEAM
      COMMON /DN/ JSDIR,JFDIR,JLEV,JALEV,JPVLEV,
     * IXSM,IYSM,IYSMAX,IYSMIN,MXTEXT(3)
      COMMON /DX/ JXBEAM,JYBEAM,JPVAX
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /PP/ MPP1(3),ISHQD,MPP2(3)
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /WS/ L1,L2,L3,JBYTE
      EQUIVALENCE(MBEAMY,MBMYH)
      DIMENSION MBEAMY(16),LMESS(7)
      DATA LMESS/28HBEAMING ERROR: NOTE TOO LONG/
C     DETERMINE STEM DIRECTION
      LYMAX = 0
      LYMIN = KBIG
      DO 30 L1=1,NBEAM
      LYMAX = MAXO(NHW(MBMYH,L1),LYMAX)
      LYMIN = MINO(NHW(MBMYL,L1),LYMIN)
   30 CONTINUE
      IF (JFDIR-2) 32,34,31
   31 IF ((LYMAX+LYMIN)/K2.GT.IYS+4*IYINC) GOTO 34
C     STEMS POINTING UPWARDS
   32 JSDIR = -1
      L2 = 16
      JXBEAM = 8 + 5 * ISHQD
      JYBEAM = MINO(LYMAX,IYS+8*IYINC) + 6 * IYINC
```

```
      GOTO 36
C     STEMS POINTING DOWNWARDS
   34 JSDIR = 1
      L2 = 0
      JXBEAM = 0
      JYBEAM = MAXO(LYMIN,IYS) - 6 * IYINC
C     DISPLAY STEMS
   36 DO 40 L1=1,NBEAM
      IXT = NHW(MBEAMX,L1) + JXBEAM
      IYT = NHW(MBEAMY,L2+L1)
      CALL TPOSN
      IYT = JYBEAM
      CALL TLINE
   40 CONTINUE
C     DETERMINE LENGTHS OF LONGEST AND SHORTEST
C     NOTES IN GROUP
      LYMAX = 0
      LYMIN = KBIG
      DO 450 L1=1,NBEAM
      L2 = NBYTE(MBEAML,L1)
      LYMAX = MAXO(LYMAX,L2)
      LYMIN = MINO(LYMIN,L2)
  450 CONTINUE
C     ERROR IF GROUP HAS CROTCHET OR LONGER NOTE
      IF (LYMIN .LE. 4) GOTO 7
C     DISPLAY CROSSBARS
      DO 60 L1=5,LYMAX
      IF (L1 .LE. LYMIN) GOTO 55
C     DISPLAY PARTIAL CROSSBARS FOR NOTES OF
C     DIFFERENT LENGTH FROM NEIGHBOURS
      DO 50 L2=1,NBEAM
      IF (NBYTE(MBEAML,L2) .LT. L1) GOTO 5
      LRHS = NHW(MBEAMX,L2)
      IF (L2 .GT. 1 .AND. NBYTE(MBEAML,L2-1)
     *                          .GE. L1) GOTO 47
C     NO SHORT BAR IF THIS NOTE IS BEAMED TO
C     NEXT NOTE AT CURRENT LEVEL
      IF (L2 .LT. NBEAM .AND. NBYTE(MBEAML,L2+1)
     *                          .GE. L1) GOTO 5
C     SHORT BEAM - SHIFT RIGHT OF STEM IF 1ST NOTE
      IF (L2 .EQ. 1) LRHS = LRHS + 8
      LLHS = LRHS - 8
      GOTO 48
C     DISPLAY CROSSBAR TO PREVIOUS NOTE
   47 LLHS = NHW(MBEAMX,L2-1)
   48 CALL DXBAR(L1,LLHS,LRHS)
    5 CONTINUE
   50 CONTINUE
      GOTO 6
   55 CALL DXBAR(L1,NHW(MBEAMX,K1),
     *                          NHW(MBEAMX,NBEAM))
    6 CONTINUE
   60 CONTINUE
      GOTO 8
C     DISPLAY -BEAMING ERROR NOTE TOO LONG-
    7 CALL TMESS(28,LMESS)
    8 ISBEAM = 0
      JBEAM = 0
      RETURN
      END
C
C
      SUBROUTINE DXBAR(NLENG,NLHS,NRHS)
C     DISPLAY CROSSBAR FOR NOTE-BEAMING.
C     *NLENG* = DEPTH OF BEAMING,
C     *NLHS* = LEFT HAND X CO-ORDINATE,
C     *NRHS* = RIGHT HAND X CO-ORDINATE.
      COMMON /DN/ JSDIR,JFDIR,JLEV,JALEV,JPVLEV,
     * IXSM,IYSM,IYSMAX,IYSMIN,MXTEXT(3)
      COMMON /DX/ JXBEAM,JYBEAM,JPVAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      IF (NRHS .LE. NLHS) NLHS = 0
C     USE THREE PARALLEL LINES TO GIVE SOLID BAR
      DO 50 L=1,3
      IXT = NRHS + JXBEAM
      IYT = JYBEAM + (NLENG-5)*IYINC*JSDIR - (L-1)
      CALL TPOSN
      IXT = NLHS + JXBEAM
      CALL TLINE
```

```
   50 CONTINUE
      RETURN
      END
!ASS (M:B0,D1,JHR0M322)
!FORTRAN B0,NS
C
C
C     ******************************************
C     *                                        *
C     * SEGMENT 322 - DISPLAY SEPARATE SIGN ITEMS*
C     *                                        *
C     ******************************************
C
C
      BLOCK DATA
C     THIS SUBPROGRAM IS PROGRAM-GENERATED.
      COMMON /XE/ MCORD3(12)
      DATA MCORD3/
     *8Z00030008,8Z00180020,8Z40BCC0CC,8ZC28EBCBC,
     *8ZC0BCC4C0,8ZCCC48D40,8Z40C2C3C0,8ZC0BCBD3E,
     *8Z3FC4C0B8,8Z42C0C0C8,8Z40BDBEC0,8Z40BEC240/
      END
C
C
      SUBROUTINE SDISP
C     DISPLAY SEPARATE SIGN ITEM.
      COMMON /DI/ MBEAMX(8),MBMYH(8),MBMYL(8),
     * MBEAML(4),JBEAM,ISBEAM
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *    ISERR,ISNXST,ISINPT
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /MM/ IXST,JMEM
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /ND/ NXLENG,NSHIFT,JBXL,NEXTX
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /SE/ JCMD,KSEGS(7)
      COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION MITEM2(6),KTPL(2),KTPED(1),
     * KT8VA(1),KTSTAR(1),LETJ(1),LCHAR(1),
     * LSCALE(2)
      EQUIVALENCE(MITEM2,JCLEF)
      DATA KTPL/4HPAGE,4HLINE /,LETJ(1)/1HJ/
      DATA KTPED(1)/3HPED/,KT8VA(1)/3H8VA/
      DATA KTSTAR(1)/1H*/,LSCALE/5HSCALE/
      LENTRY = ITEM - 32
      IXE = IXST * 8
      LBIT = 0
C     BRANCH INDEXED BY ITEM CODE
      IF (LENTRY .GT. 16) GOTO 2
C     DOUBLE-BYTE ITEM
      GOTO (22,24,26,28,5,5,51,54),LENTRY
      GOTO 7
C     SINGLE-BYTE ITEM
    2 LENTRY = LENTRY - 16
      GOTO (34,36,38,42,4,46,44,48,3,32),LENTRY
      GOTO 7
C     DISPLAY CLEF SIGN
   22 CALL DCLEF
      GOTO 8
C     DISPLAY TIME SIGNATURE
   24 CALL DTSIG
      GOTO 8
C     DISPLAY KEY SIGNATURE
   26 CALL DKSIG
      GOTO 8
C     DISPLAY IRREGULAR NOTE-GROUPING
   28 NBEAM = NGROUP
      LCHAR(1) = KSPACS(1)
      GOTO 52
C
C     DISPLAY PEDALLING SIGN
```

```
      3 CALL TTEXT(-3,KTPED)
        GOTO 8
C       DISPLAY END-OF-PEDAL SIGN
     32 CALL TTEXT(-1,KTSTAR)
        GOTO 8
C       START-OF-SLUR, NO DISPLAY TILL END
     34 IYE = (JMEM+11)*IYINC
        GOTO 6
C       END-OF-SLUR, DISPLAY SLUR
     36 CALL DSLUR
        GOTO 8
C       OTTAVA BELOW SIGN
     38 LBIT = 128
        L1 = -3
        GOTO 41
C       OTTAVA ABOVE SIGN
      4 L1 = 3
     41 CALL TTEXT(3*LENTRY-12,KTOVA)
        GOTO 6
C       END-OF-OTTAVA SIGN
     42 CALL DOVA
        GOTO 8
C       START-OF-DECRESCENDO SIGN
     44 LBIT = 128
C       START OF CRESCENDO SIGN
     46 IYE = -5 * IYINC
        GOTO 6
C       END-OF-CRESCENDO-OR-DECRESCENDO SIGN
     48 CALL DCRESC
        GOTO 8
C       DISPLAY INKPRINT PAGE OR LINE NUMBER
      5 CALL TTEXT1(K4,KTPL(LENTRY-4),
     *              (3+ISEDIT)*(9-LENTRY))
        GOTO 56
C       DISPLAY NOTE BEAMING
     51 LCHAR(1) = LETJ(1)
     52 JBEAM = 1
        IXE = IXN + 4 * IXINC + 20 * ISEDIT
        IYE = (12 + 6 * ISEDIT) * IYINC
        IF (LENTRY .EQ. 4) GOTO 53
C       NO BEAMING NUMBERS SHOWN FOR DISPLAY COMMAND
        IF (JCMD .EQ. 6 .OR. JCMD .EQ. 21) GOTO 535
     53 CALL TWRITE(IXE-2*IXINC,IYS+IYE,K1,LCHAR)
        CALL TWT1(NBEAM)
C       START NEW PARALLEL IF CLOSE TO RHS OF SCREEN
    535 IF (IXN .GE. JXMAX-NBEAM*JSCALE) ISNXST = 2
        GOTO 8
C       DISPLAY SCALE FACTOR
     54 IF (ISEDIT .EQ. 0) GOTO 8
        CALL TTEXT1(K5,LSCALE,20)
     56 CALL TWT1(MITEM2(LENTRY))
        GOTO 8
C
C       X CO-ORDINATE FOR START OF EXTENDED SIGN
      6 IXST = IXN + IXINC
        IXE = IXST
        CALL STBYTE(IXST/K8+LBIT,MCMEM,JMEM)
        GOTO 8
C       UNRECOGNISED ITEM
      7 ISERR = 1
        GOTO 9
      8 ISERR = 0
      9 RETURN
        END
C
C
        SUBROUTINE DCLEF
C       DISPLAY CLEF SIGN.
        COMMON /DI/ MBEAMX(8),MBMYH(8),MBMYL(8),
     * MBEAML(4),JBEAM,ISBEAM
        COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
        COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,K8IG
        COMMON /PP/ MPP1(3),ISHOD,MPP2(3)
        COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
        COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
        COMMON /WS/ L1,L2,L3,JBYTE
        COMMON /XE/ MCORD3(12)
        IXE = IXN + 2 * IYINC
        IYE = IYS + 5 * IYINC
```

```
      IF (JCLEF .EQ. 122) GOTO 42
      IF (JCLEF .EQ. 134) GOTO 5
      IF (JCLEF .EQ. 129) GOTO 4
C     C CLEF AT APPROPRIATE POSITION
      IYE = IYE + (JCLEF-128) * IYINC
      LYINC = 0
      LTYPE = 3
      GOTO 7
C     DISPLAY 8 BELOW TREBLE CLEF
    4 CALL TWRIT1(IXN,IYS-4*IYINC,K8)
C     DISPLAY TREBLE CLEF
   42 LYINC = -2
      LTYPE = 1
      GOTO 7
C     DISPLAY BASS CLEF
    5 IF (ISHOD .NE. 0) GOTO 56
      DO 540 L=5,7,2
  540 CALL DOOT(IXN+20,L)
   56 LYINC = 2
      LTYPE = 2
      IXT = IXE - IXINC
      GOTO 72
C     DISPLAY SIGN
    7 IXT = IXE
   72 IYT = IYE + LYINC * IYINC - 3 * ISHOD - 3
      CALL DSYMB(2379+LTYPE,IYINC,LTYPE,MCORD3)
      IXN = IXN + 5 * IYINC
      IYE = IYE - IYS
      RETURN
      END
C
C
      SUBROUTINE DKSIG
C     DISPLAY KEY SIGNATURE AT (*IXN*,*IYS*).
C     *JKEYNO* = NUMBER OF ACCIDENTALS TO DISPLAY,
C     *JKEYSG* = TYPE OF ACCIDENTAL.  1 = FLAT,
C                2 = NATURAL, 3 = SHARP.
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KY/ JKEYNO,JKEYSG,MKPCH(2),MTKPCH(2)
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /SD/ MYAXID(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /WS/ L1,L2,L3,JBYTE
      IF (JKEYNO .EQ. 0) GOTO 8
      IXE = IXN + 20
      IYE = 5 * IYINC
      L1 = MYAXID(JKEYSG)
      DO 60 L=1,JKEYNO
      L2 = L1 / 10
      IYN = IYS+((L1-10*L2)+(128-JCLEF)/K6)*IYINC
      L1 = L2
      CALL DAXID(JKEYSG+1)
      IXN = IXN + 9
   60 CONTINUE
    8 IXN = IXN + 9
      RETURN
      END
C
C
      SUBROUTINE DTSIG
C     DISPLAY TIME SIGNATURE.
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      DIMENSION LCHAR(1)
      DATA LLETC/1HC/,LLETU/1HU/
      IXE = IXN + IXINC
      IYE = 5 * IYINC
      IF (JTYME = 192) 6,2,4
C     C BAR
    2 IXT = IXE + 4
      IYT = IYS + 6 * IYINC
      CALL TPOSN
```

```
      IYT = IYS + 2 * IYINC
      CALL TLINE
C     C OR UNMEASURED
    4 LCHAR(1) = LLETC
      IF (JTYME .EQ. 255) LCHAR(1) = LLETU
      CALL TWRITE(IXE,IYS+3*IYINC,K1,LCHAR)
      GOTO 8
C     TWO NUMBERS
    6 CALL TWRIT1(IXE,IYS+6*IYINC,JTIME1)
      CALL TWRIT1(IXE,IYS+2*IYINC,JTIME2)
    8 IXN = IXN + 3 * IXINC
      RETURN
      END
C
C
      SUBROUTINE DCRESC
C     DISPLAY CRESCENDO OR DECRESCENDO.
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *    ISERR,ISNXST,ISINPT
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /WS/ L1,L2,L3,JBYTE
      L1 = IXE
      IXE = IXP
      IYE = -4 * (ISEDIT+1) * IYINC
      IF (L1 .GT. 1024) GOTO 3
C     DECRESCENDO
      LX1 = IXE
      LX2 = L1
      GOTO 5
C     CRESCENDO
    3 LX1 = L1 - 1024
      LX2 = IXE
    5 IXT = LX1
      IYT = IYS + IYE + IYINC
      CALL TPOSN
      IXT = LX2
      IYT = IYT - IYINC
      CALL TLINE
      IXT = LX1
      IYT = IYT - IYINC
      CALL TLINE
      RETURN
      END
C
C
      SUBROUTINE DSLUR
C     DISPLAY SLUR SIGN ABOVE CURRENT STAVE.
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *    ISERR,ISNXST,ISINPT
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /MM/ IXST,JMEM
      IXT = IXE
      IXE = IXP
      IYT = MINO(MAXO(IYS+(JMEM+10)*IYINC,IYN),
     *                              JYMAX - 15)
      IYE = IYT - IYS
C     POSITION TO START OF SLUR
      CALL TPOSN
      IXT = IXT + 12
      IYT = IYT + IYINC
C     DRAW UPWARD SLOPING LINE
      IF (IXT .GT. 28) CALL TLINE
      CALL TPOSN
      IXT = IXP - 12
C     DRAW HORIZONTAL LINE
      CALL TLINE
      IF (IXE .GE. JXMAX) GOTO 9
      IXT = IXP
      IYT = IYT - IYINC
C     DRAW DOWNWARD SLOPING LINE
      CALL TLINE
    9 RETURN
      END
C
C
```

```
      SUBROUTINE D8VA
C     DISPLAY DASHES FOR DURATION OF 8VA.
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /QL/ IXG,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /WS/ L1,L2,L3,JBYTE
      JBYTE = IXE + 4 * IXINC
      LBELOW = NBIT(-3)
      IXE = IXP
      IYE = (13-22*LBELOW) * IYINC
      IYT = IYE + IYS
      DO 20 L=JBYTE,IXE,24
      IXT = L
      CALL TPOSN
      IXT = IXT + 8
      CALL TLINE
   20 CONTINUE
      IF (IXE .GE. JXMAX) GOTO 9
      IYT = IYT + 12 * LBELOW - 6
      CALL TLINE
    9 RETURN
      END
C
C
      SUBROUTINE ESTAFF
C     DISPLAY ANY CURRENTLY-OPEN EXTENDED SIGNS
C     FOR END OF PARALLEL.
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /JD/ JSTAFF,JDLINE,JDHIGH,JDLOW
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /MM/ IXST,JMEM
      COMMON /MO/ JIMODE,JOMODE
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /QL/ IXG,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /WS/ L1,L2,L3,JBYTE
C     EXIT IF BRAILLE MODE OR NO PREVIOUS PARALLEL
      IF (JIMODE .NE. 1 .OR. JDHIGH .EQ. 0) GOTO 9
      LXST = IXST
      LMEM = JMEM
      LIYS = IYS
      LIYN = IYN
      IYN = 0
      LL = 0
C     FOR EACH STAVE OF CURRENT PARALLEL...
      DO 60 L=JDHIGH,JDLOW
C     NO ACTION IF STAVE NOT SELECTED
      IF (NBYTE(MSSEL,L) .EQ. 0) GOTO 6
      IF (ISEDIT .NE. 0) GOTO 3
      LL = LL + 1
      IYS = JYMAX - (JJSTAFF-1)*NSSEL+LL)*IYSTV
      CALL CHSTAV(L)
    3 IXP = JXMAX
      DO 40 JMEM=1,4
      JBYTE = NBYTE(MCMEM,JMEM)
      IXST = 128 * NBIT(K0) + 2
      IF (JBYTE .LE. 0) GOTO 4
      IXE = 8 * (IXST + JBYTE - 2)
      GOTO (32,32,34,36),JMEM
   32 CALL DSLUR
      GOTO 38
   34 CALL DCRESC
      GOTO 38
   36 CALL D8VA
C     MODIFY X CO-ORDINATE FOR START OF SIGN
   38 CALL STBYTE(IXST,MCMEM,JMEM)
    4 CONTINUE
   40 CONTINUE
C     ONLY ONE STAVE IF EDITING
      IF (ISEDIT .NE. 0) GOTO 7
    6 CONTINUE
   60 CONTINUE
    7 CONTINUE
```

```
      IXSL = TXST
      JMEM = LMEM
      IYN = LIYN
      IYS = LIYS
    9 RETURN
      END
C
C
      SUBROUTINE NXSTAF
C     START NEW PARALLEL ON DISPLAY.
C     *ISNXST* = 1 FOR NO WAIT, 2 FOR WAIT,
C     3 FORCE NEW PARALLEL WITH NO WAIT,
C     4 FORCE NEW PARALLEL WITH WAIT FOR C/R.
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /JD/ JSTAFF,JDLINE,JDHIGH,JDLOW
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /MO/ JIMODE,JOMODE
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /QL/ IXG,IYG,IXGMIN,IGMAX,JXMAX,JYMAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /TT/ JTTEMP,JXTEMP
      DIMENSION LMIG1(2),LMIG2(3)
      DATA LNIG1/5/,LNIG2/12/
      DATA LMIG1/8Z31333536,8Z37000000/,
     * LMIG2/8Z00010305,8Z06090A32,8Z34383A04/
C     CANCEL NEW PARALLEL IF EDITING DURING INPUT
      IF (ISEDIT.NE.0 .AND. ISINPT.NE.0) GOTO 8
C     DELAY NEW PARALLEL FOR START EXTENDED SIGNS
      DO 1950 L=1,LNIG1
 1950 IF (ITEM .EQ. NBYTE(LMIG1,L)) GOTO 9
      LBAR = JBAR
      IF (ISNXST-3) 19,21,205
   19 IF (JIMODE .NE. 1) GOTO 22
C     DELAY NEW STAFF FOR SOME NON-BRAILLE ITEMS
      DO 20 L=1,LNIG2
   20 IF (ITEM .EQ. NBYTE(LMIG2,L)) GOTO 9
      GOTO 22
  205 IF (ISINPT .NE. 0) LBAR = JBAR + 1
   21 ISNXST = ISNXST - 2
C     SAVE STAVE STATUS
   22 LLINE = JLINE
      CALL CHSTAV(LLINE)
      CALL ESTAFF
      IF (ISEDIT .EQ. 0) GOTO 24
C     EDITING
      IYS = IYS - 200
      CALL DSTAVE(JLINE)
      GOTO 8
C     NOT EDITING
   24 LLEFT = NSSEL - JDLINE
      IF (LLEFT .GT. 0) GOTO 25
C     NEW PARALLEL
      LLEFT = NSSEL
      JDLINE = 0
      JDLOW = 0
      JSTAFF = JSTAFF + 1
      IF (JSTAFF .LE. NDP) GOTO 3
      JSTAFF = 1
C     NEW PAGE ON TERMINAL
   25 CALL TPAGE(MAXO(ISNXST-1,ITERR))
      IF (ISEND .NE. 0) GOTO 8
    3 LDISP = MINO(NDL,LLEFT)
      JDLINE = JDLINE + LDISP
      IF (JIMODE .NE. 1) GOTO 5
C     PRINTED MUSIC MODE
      LIYS = JYMAX - (JSTAFF-1) * LDISP * IYSTV
      IYS = LIYS
      DO 40 L=1,LDISP
      IYS = IYS - IYSTV
      JDLOW = NXLINE(JDLOW)
      IF (L .EQ. 1) JDHIGH = JDLOW
      CALL DSTAVE(JDLOW)
   40 CONTINUE
```

```
      IXT = 0
      LIYT = LIYS = IYSTV * LDISP
      IYT = LIYT
      CALL TPOSN
      IYT = LIYS = IYSTV + 8 * IYINC
      CALL TLINE
      IYT = LIYT
      CALL TLINE
C     RESTORE CURRENT STAVE AND STAVE STATUS
      CALL CHSTAV(LLINE)
      IYS = LIYS = IYSTV
C     DISPLAY BAR NUMBER ABOVE TOP OF PARALLEL
      IF (ISEDIT .EQ. 0) CALL TWRIT1(KO,
     *                  IYS+11*IYINC,LBAR)
      GOTO 8
C     BRAILLE MODE
    5 IXN = 0
      IYS = IYS - 50
    8 ISNXST = 0
    9 RETURN
      END
C
C
      FUNCTION NXLINE(N)
C     RETURNS *NXLINE* = NEXT SELECTED LINE AFTER
C     LINE *N*.
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      NXLINE = N
    3 NXLINE = NXLINE + 1
      IF (NXLINE .GT. NLINES) NXLINE = 1
      IF (NBYTE(MSSEL,NXLINE) .EQ. 0) GOTO 3
      RETURN
      END
C
C
      SUBROUTINE DSTAVE(N)
C     DISPLAY STAVE FOR LINE *N*.
C     *IYS* = CO-ORDINATE OF BOTTOM LINE OF STAVE
      COMMON /DN/ JSDIR,JFDIR,JLEV,JALEV,JPVLEV,
     * IXSM,IYSM,IYSMAX,IYSMIN,MXTEXT(3)
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /JD/ JSTAFF,JDLINE,JDHIGH,JDLOW
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KY/ JKEYNO,JKEYSG,MKPCH(2),MTKPCH(2)
      COMMON /MO/ JIMODE,JOMODE
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /ND/ NXLENG,NSHIFT,JBXL,NEXTX
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TD/ IXC,IYC,IXB,IXR
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /TT/ JTTEMP,JXTEMP
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      IYQ = MINO(IYQ,IYS-8*IYINC)
      IXB = 8
      IXN = IXB
      IXP = IXN
      IXM = IXN
      IF (JIMODE .NE. 1) GOTO 5
C     DISPLAY FIVE HORIZONTAL STAFF LINES
      DO 30 L=1,5
      IYT = IYS + (10 - 2 * L) * IYINC
      IXT = 0
      CALL TPOSN
      IXT = JXMAX
      CALL TLINE
      IXT = 0
      CALL TLINE
   30 CONTINUE
      IF (ISEDIT .EQ. 0) CALL CHSTAV(N)
      CALL DCLEF
C     DISPLAY KEY SIGNATURE UNLESS NATURALS
      IF (JKEYSG .NE. 2) CALL DKSIG
C     DISPLAY TIME SIGNATURE FOR FIRST PARALLEL
      IF (JSTAFF .EQ. 1) CALL DTSIG
C     DISPLAY STAVE NUMBER UNLESS ALL STAVES SHOWN
```

```
      IF (ISEDIT .EQ. 0 .AND. NLINES .GT. NDL)
     *    CALL TWRIT1(5*IXINC,IYS+11*IYINC,JLINE)
    5 IXB = IXN
      IXM = IXB + 16
      IXN = IXM
      NEXTX = IXM
      MXTEXT(1) = IXM
      MXTEXT(2) = IXM
      MXTEXT(3) = IXM
      IXR = IXM
      JXTEMP = IXM
      RETURN
      END
!ASS (M:B0,D1,JHR0M323)
!FORTRAN B0,NS
C
C
C     ******************************************
C     *                                        *
C     *    SEGMENT 323 - DISPLAY OF TEXT ITEMS *
C     *                                        *
C     ******************************************
C
C
      SUBROUTINE TDISP
C     DISPLAY TEXT ITEM.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /DN/ JSCIR,JFDIR,JLEV,JALEV,JPVLEV,
     * IXSM,IYSM,IYSMAX,IYSMIN,MXTEXT(3)
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      COMMON /TY/ ISCONT,ISBLOW
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION LTEXT(8),LMESC(7),LVOWEL(2)
      DIMENSION LHYPH(1),LAXENT(1)
      DATA LAXENT/4H'¢^-/,LUNDER/1H-/
      DATA LMESC/25HEFR123456789.SOP$%#ACILN//
      DATA LVOWEL/5HAEIOU/,LHYPH/2H =/
      IF (ISHDR .NE. 0) GOTO 2
C     SET CURSOR POSITION TO DISPLAY TEXT ITEM
      IXN = MAXO(IXN,MXTEXT(ISWORD+ISBLOW+1))
      IXT = IXN
      IYT = (10-(16+2*ISEDIT)*ISBLOW)*IYINC + IYS
    2 IXE = IXT + 6
      IYE = IYT - IYS + 8
      IF (NCHARS .EQ. 0) GOTO 9
      LIXT = IXT
C     DISPLAY ACCENTS AND UNDERLINES AND SET UP
C     TEXT IN *LTEXT* FOR DISPLAY AS ONE STRING
      DO 70 L=1,NCHARS
      LCHAR = NBYTE(MTEXT,L)
      JBYTE = LCHAR
      IF (JBYTE .LT. 128) GOTO 4
      LL = NBIT(K3)
      IF (JBYTE .GT. 91) GOTO 25
      JBYTE = LCHAR
      GOTO 6
C     ACCENTED LETTER
   25 LADD = 64
      IF (LL .LT. 12) GOTO 3
      LL = LL - 4
      LADD = 0
    3 CALL MBS(LAXENT,LL-7,MCHAR,K1,K1)
      L1 = 10
C     RAISE UMLAUT A BIT FURTHER
      IF (LL .EQ. 11) L1 = 20
      IYT = IYT + L1
      CALL TWCH
      IYT = IYT - L1
      JBYTE = NBYTE(LVOWEL,JBYTE-9) - LADD
      GOTO 65
    4 IF (JBYTE .GE. 64) GOTO 6
      MCHAR(1) = LUNDER
      CALL TWCH
```

```
      IF (JBYTE .GT. 3) JBYTE = JBYTE - 29
      JBYTE = NBYTE(LMESC,JBYTE)
      GOTO 65
    6 IXT = IXT + IXINC
   65 CALL STBYTE(JBYTE,LTEXT,L)
   70 CONTINUE
C     DISPLAY MAIN TEXT STRING
      IXT = LIXT
      CALL TPOSN
      CALL TWT(NCHARS,LTEXT)
C     ADD HYPHEN FOR CONTINUED WORD
      IF (ISCONT .NE. 0) CALL TWT(K2,LHYPH)
      MXTEXT(ISWORD+ISBLOW+1) =
     *      IXE + (NCHARS+2*ISCONT) * IXINC
    9 RETURN
      END
C
C
      SUBROUTINE DHDR1
C     DISPLAY FILE HEADER.
      COMMON MINBUF(90),MOPBUF(90),NINBUF,NOPBUF,
     * JINSEC,JOPSEC,JINBUF,JOPBUF,JINLK,JOPLK,
     * NILENG,MIBAR(64),NOLENG,MOBAR(64),JIBAR,
     * JOBAR,JBBUF,NFLAGS,MFLAGS(3)
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /HE/ MCLEF(10),MPART(10)
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /K8/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KY/ JKEYNO,JKEYSG,MKPCH(2),MTKPCH(2)
      COMMON /ND/ NXLENG,NSHIFT,JBXL,NEXTX
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION KTINST(156),M1(114),M2(42),
     * KTVOC(15),KTEXT(30),LCCLEF(6),LTCLEF(12),
     * KYTEXT(6),LLETS(1),LCOLON(1),LCLEF(2),
     * LKEY(1),LTIME(1),LPART(1)
      EQUIVALENCE(M1,KTINST),(M2,KTINST(115))
      DATA KNPINS/52/,KNPVOC/5/
      DATA M1/
     * 12HRIGHT HAND  ,12HLEFT HAND   ,
     * 12HPEDALS      ,12HPICCOLO     ,
     * 12HFLUTE       ,12HALTO FLUTE  ,
     * 12HOBOE        ,12HOBOE D'AMORE,
     * 12HENGLISH HORN,12HCLARINET    ,
     * 12HB. CLARINET ,12HBASSOON     ,
     * 12HDBLE BASSOON,12HHORN        ,
     * 12HFLUGEL HORN ,12HTRUMPET     ,
     * 12HBASS TRUMPET,12HTROMBONE    ,
     * 12HB. TROMBONE ,12HTUBA        ,
     * 12HBASS TUBA   ,12HCORNET      ,
     * 12HEUPHONIUM   ,12HSAXOPHONE   ,
     * 12HKETTLEDRUM  ,12HBELLS       ,
     * 12HGLOCKENSPIEL,12HCELESTA     ,
     * 12HDULCITONE   ,12HXYLOPHONE   ,
     * 12HMARIMBA     ,12HMARIMBA GONG,
     * 12HSIDE DRUM   ,12HTENOR DRUM  ,
     * 12HBASS DRUM   ,12HTABOR       ,
     * 12HTAMBOURINE  ,12HTRIANGLE    /
      DATA M2/
     * 12HCYMBALS     ,12HGONG        ,
     * 12HCASTANETS   ,12HRATTLE      ,
     * 12HANVIL       ,12HWIRE BRUSHES,
     * 12HVIOLIN 1    ,12HVIOLIN 2    ,
     * 12HVIOLA       ,12HCELLO       ,
     * 12HDOUBLE BASS ,12HHARP        ,
     * 12HGUITAR      ,12HACCORDION   /
      DATA KTVOC/12HVOCAL       ,
     *  12HSOPRANO     ,12HALTO VOICE  ,
     *  12HTENOR VOICE ,12HBASS VOICE  /
      DATA KTEXT/
     *  12HIDENTIFIER; ,12HNAME:       ,
     *  12HTITLE:      ,12HSUBTITLE:   ,
     *  12HCOMPOSER:   ,12HPUBLISHER:  ,
     *  12HNUMBER:     ,12HTEMPO:      ,
```

```
     *  12HMETRONOME:  ,12HCOMMENTS:    /
      DATA LCOLON/2H: /,LCLEF/8H  CLEF: /
      DATA LNCLEF/6/,LLETS(1)/1HS/,LKEY/3HKEY/,
     * LTIME/4HTIME/,LPART/4HPART/
      DATA LCCLEF/122,124,128,129,130,134/
      DATA LTCLEF/
     *8HTREBLE  ,8HSOPRANO ,8HALTO    ,
     *8HTREBLE 8,8HTENOR   ,8HBASS    /
      DATA KYTEXT/
     * 8H    FLAT,8H NATURAL,8H   SHARP /
C
      CALL TMESS(12,KTEXT)
      CALL TWT(K4,MIDENT)
      ISHDR = 1
      JBAR = 1
      LL = NWTEXT + 1
      DO 50 L=1,LL
      CALL IDCODE
      CALL TMESS(12,KTEXT(3*L+1))
      CALL TDISP
      CALL IINSRT(K1)
   50 CONTINUE
C     KEY SIGNATURE
      CALL HINSRT(K3,LKEY)
      CALL TWT1(JKEYNO)
      CALL TWT(K8,KYTEXT(2*JKEYSG-1))
C     ADD LETTER S FOR PLURAL
      IF (JKEYNO .NE. 1) CALL TWT(K1,LLETS)
C     TIME SIGNATURE
      CALL HINSRT(K4,LTIME)
      CALL TWT1(JTIME1)
      CALL TWT1(JTIME2)
C     DISPLAY PART AND CLEF NAMES FOR EACH STAVE
      DO 70 L=1,NLINES
      CALL HINSRT(K4,LPART)
      CALL TWT1(L)
      CALL TWT(K2,LCOLON)
      L1 = NBYTE(MPART,L)
      IF (L1 .GT. 64) GOTO 54
C     INSTRUMENTAL PART
      IF (L1 .GT. KNPINS) GOTO 58
      CALL TWT(12,KTINST(3*L1-2))
      GOTO 6
C     VOCAL PART
   54 L1 = L1 - 64
      IF (L1 .GT. KNPVOC) GOTO 58
      CALL TWT(12,KTVOC(3*L1-2))
      GOTO 6
C     UNRECOGNISED PART CODE
   58 CALL TWT1(L1)
C     CLEF SIGN
    6 CALL TWT(K8,LCLEF)
      JCLEF = NBYTE(MCLEF,L)
      DO 620 LL=1,LNCLEF
  620 IF (JCLEF .EQ. LCCLEF(LL)) GOTO 64
C     UNRECOGNISED CLEF CODE
      CALL TWT1(JCLEF)
      GOTO 7
C     DISPLAY CLEF NAME
   64 CALL TWT(K8,LTCLEF(2*LL-1))
    7 CONTINUE
   70 CONTINUE
      CALL TNLINE
      RETURN
      END
C
C
      SUBROUTINE HINSRT(NLENG,NTEXT)
C     INSERT HEADER RECORD ITEM INTO EDIT TABLE.
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,K9IG
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      DIMENSION NTEXT(1)
      CALL IDCODE
      CALL TMESS(NLENG,NTEXT)
      IXE = IXT + IXINC
      IYE = IYT - IYS + IYINC
      CALL IINSRT(K1)
      RETURN
      END
```

```
!ASS (M:R0,D1,JHROM324)
!FORTRAN B0,NS,S
C
C
C     ********************************************
C     *                                          *
C     *   SEGMENT 324 - DISPLAY OF NOTE ITEMS    *
C     *                                          *
C     ********************************************
C
C
      BLOCK DATA
C     THIS SUBPROGRAM IS PROGRAM-GENERATED.
      COMMON /XF/ MCORD4(76)
      DATA MCORD4/
     *8Z001DC03C,8Z0C4C0C4E,8Z0056005E,8Z007C0076,
     *8Z00700068,8Z0C6400A6,8Z00B400A4,8Z00B2018D,
     *8Z00800090,8Z0C840094,8Z01040106,8Z00C000D2,
     *8Z00C600E0,8Z0CFE00E6,8Z00D800EA,8Z40C8C0B0,
     *8Z4CC0C0D0,8Z4CBBB4C0,8Z40BACC40,8Z40C0CCC0,
     *8Z40BFB4C0,8Z4CBFCCC0,8Z40BFB4C0,8Z46424CC0,
     *8ZB4C0CC28,8ZCCC0BAB4,8Z34C0CCC0,8ZBAB434C0,
     *8ZCCC0BAB4,8Z34C0CCC0,8ZBAB434C0,8ZCCC0B428,
     *8ZC0D0C0B0,8Z36C4C4C8,8ZC4B8C4C8,8ZC4B8C448,
     *8ZC0D0C0B0,8Z32C4C4C8,8ZC4B8C4C8,8ZC4B8C4C8,
     *8ZC4B8C448,8Z5AC036C2,8ZC4C8C4C0,8ZC4B8C4C0,
     *8ZC4485AC0,8Z36CAC4B8,8ZC4C0C4C8,8ZC4C0C438,
     *8Z3AC8CCC0,8Z3ABA41C2,8ZBFC0C0C1,8ZC1C0C0BF,
     *8Z3F3EC2C4,8ZBCC0C23C,8Z3DC0B4C4,8ZCCC44CB8,
     *8Z3AC0CCC4,8ZB4443AC4,8ZCC40C1BF,8ZBEC0C0C2,
     *8ZC2C0C0BE,8Z35C0C3C7,8ZC7C3C7BD,8ZC3393CC0,
     *8ZC4CAC436,8Z40C030B4,8ZC3BDBBBB,8ZC5BBBBBB,
     *8ZC5BBBBBB,8ZC5BBBBBB,8ZC5BBBBBB,8ZC5BBBBBB,
     *8ZC53B0000,8Z0C000000,8Z00000000,8Z00000000/
      END
C
C
      SUBROUTINE NDISP
C     DISPLAY NOTE ITEM.
      COMMON /DI/ MBEAMX(8),MBMYH(8),MBMYL(8),
     * MBEAML(4),JBEAM,ISBEAM
      COMMON /DN/ JSDIR,JFDIR,JLEV,JALEV,JPVLEV,
     * IXSM,IYSM,IYSMAX,IYSMIN,MXTEXT(3)
      COMMON /DX/ JXBEAM,JYBEAM,JPVAX
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /ND/ NXLENG,NSHIFT,JBXL,NEXTX
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NCP,NDL
      COMMON /PP/ MPP1(3),ISHOD,MPP2(3)
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREFT,JPN
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /SE/ JCMD,KSEGS(7)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION LFING(7),LTIE(1)
      DATA LTIE/4H TIE/
      NXLENG = NSCALE * JNDUR * JTIME2 / 256
      IF (NSSEL .EQ. 1 .AND. JPART .GE. 64)
     *           NXLENG = MIN0(NXLENG,NWIDTH)
      IF (ISSTEM .EQ. 0) GOTO 18
C     SAME X COORDINATE AS PREVIOUS NOTE ITEM
      IXN = IXP + 32
      IXM = IXN
   18 JPVAX = 0
      IXP = IXN
      JPVLEV = KBIG
      NFING = 0
C
      DO 40 L=1,NHEDS
      CALL HDCODE(L)
```

```
      IXN = IXP
      JLEV = 7*JOCTAV+JPITCH+JCLEF+NSHIFT-153
      IYN = IYS + IYINC * JLEV
      IF (JAXIDS .EQ. 0) GOTO 2
      CALL DAXIDS
      GOTO 22
    2 JPVAX = 0
   22 IXN = IXP + IXINC
C     SET *IXSM* = X CO-ORD OF LHS OF NOTE-HEADS
      IXSM = IXN - 6 * ISHOD
      LXSM = IXSM
      IF (JPITCH .EQ. 0) GOTO 28
C     DISPLAY NOTE HEAD
      LXI = IXN
      IF (JPVLEV .GT. JLEV + 1) GOTO 23
C     MOVE HEAD RIGHT IF IT OVERLAPS PREVIOUS HEAD
      LXI = LXI + IXINC
      GOTO 24
   23 JPVLEV = JLEV
   24 CALL DHEAD(LXI)
      IF (ISTIE .EQ. 0) GOTO 245
C     DISPLAY TIE
      CALL TWT(K4,LTIE)
  245 IF (MHEXT(1) .EQ. 0) GOTO 27
C     SAVE FINGERING
      NFING = NFING + 1
      LFING(NFING) = 0
      JBYTE = MHEXT(1)
      L1 = NBIT(K2)
      L1 = (JBYTE-1) / K5
      IF (L1 .GT. 0)
     *  CALL STBYTE(L1+240,LFING(NFING),K2)
      L1 = JBYTE - 5 * L1
      IF (L1 .GT. 0)
     *  CALL STBYTE(L1+240,LFING(NFING),K1)
   27 IF (L .NE. 1) GOTO 3
      IXE = IXN + IXINC / K2
      IF (JLEV .GE. 10) CALL DLEGER
      JALEV = JLEV
C     SET *IYSMAX* = Y CO-ORD OF UPPERMOST HEAD
      IYSMAX = IYS + JLEV * IYINC
      LIYMAX = IYSMAX
      GOTO 3
   28 CALL DREST
    3 IXN = IXN + 16
      IF (JDOTS .EQ. 0) GOTO 4
C     DISPLAY DOTS FOLLOWING NOTE HEAD OR REST
      DO 350 LL=1,JDOTS
      CALL DDOT(IXN,JLEV)
      IXN = IXN + 8
  350 CONTINUE
    4 CONTINUE
   40 CONTINUE
C
      IF (JPITCH .EQ. 0) GOTO 6
      IF (JLEV .LE. -2) CALL DLEGER
C     SET *JALEV* = AVERAGE LEVEL OF TOP AND
C     BOTTOM NOTE-HEADS
      JALEV = (JALEV+JLEV) / K2
C     SET *IYSMIN* = Y CO-ORD OF LOWERMOST HEAD
      IYSMIN = IYS + JLEV * IYINC
      LIYMIN = MAXO(IYSMIN,KO)
C     NO STEM REQUIRED FOR SEMIBREVE OR BREVE
      IF (JLENTH .LE. 2) GOTO 55
C     DETERMINE CO-ORDINATES OF NOTE STEM
      LLENG = (7-3*MINO(ISSMAL,K1))*IYINC
      IF (ISSTEM .NE. 0) GOTO 43
      GOTO (44,46,42,43),JFDIR
C     STEM DIRECTION ACCORDING TO PITCH
   42 IF (JALEV .GE. 4) GOTO 46
      GOTO 44
C     STEM DIRECTION OPPOSITE TO NORMAL DIRECTION
   43 IF (JALEV .LT. 4) GOTO 46
C     STEM POINTING UPWARDS
   44 JSDIR = -1
      IXSM = IXSM + 8 + 5 * ISHOD
C     SET *IYSMAX* = Y CO-ORDINATE OF TOP OF STEM
      IYSMAX = IYSMAX + LLENG
      IYSM = IYSMAX
      IYT = IYSMIN
```

```
      GOTO 48
C     STEM POINTING DOWNWARDS
   46 JSDIR = 1
C     SET *IYSMIN* = Y CO-ORD OF BOTTOM OF STEM
      IYSMIN = IYSMIN - LLENG
      IYSM = IYSMIN
      IYT = IYSMAX
   48 LLENG = IYSMAX - IYSMIN
C     NO BEAMING FOR SMALL NOTE OR STEM SIGN
C     OR DURING INPUT OF MUSIC
      IF (ISINPT+ISSMAL+ISSTEM .NE. 0 .OR.
     * JBEAM .LE. 0 .OR. JBEAM .GT. 16) GOTO 488
C
C     SAVE STEM CO-ORDINATES FOR BEAMING
      CALL STHW(LXSM,MBEAMX,JBEAM)
      CALL STHW(LIYMAX,MBMYH,JBEAM)
      CALL STHW(LIYMIN,MBMYL,JBEAM)
      CALL STBYTE(JLENTH,MBEAML,JBEAM)
      JBEAM = JBEAM + 1
      IF (JBEAM .GT. NBEAM) ISBEAM = 1
      GOTO 55
C
C     DISPLAY NOTE STEM
  488 IXT = IXSM
      CALL TPOSN
      IYT = IYSM
      CALL TLINE
      LLENG = IYSMAX - IYSMIN
      IYE = IYSM + LLENG * JSDIR - IYS
      IF (JLENTH .LT. 5) GOTO 6
C     DISPLAY HOOKS ON NOTE STEM
      DO 50 L=5,JLENTH
      IYT = IYSM + IYINC * JSDIR * (L-5)
      IXT = IXSM
      CALL TPOSN
      IXT = IXT + 8
      IYT = IYT + 8 * JSDIR
      CALL TLINE
   50 CONTINUE
      GOTO 6
C
C     NO STEM
   55 IYE = IYN - IYS
C
C     SET *IXN* AND *IYN* FOR EXTRAS AND FINGERING
    6 IYN = MAXO(IYSMAX,IYS+8*IYINC) + 2 * IYINC
      CALL DEXTRA
      IF (NFING .EQ. 0) GOTO 65
C     DISPLAY FINGERING
      DO 620 L=1,NFING
  620 CALL DCHARS(LFING(NFING+1-L))
   65 IXM = IXM + NXLENG
      IXN = MAXO(IXN,IXM)
      NEXTX = MAXO(NEXTX,IXN)
      IXP = IXP + 32
C     IF END OF DISPLAY STAVE, SET FLAG TO DISPLAY
C     NEW PARALLEL UNLESS ONLY BARLINE TO GO
      IF (IXN .GE. JXMAX-NSCALE .AND.
     *    JTIME+JNDUR .NE. JTMAX) ISNXST = 2
      IF (JFDIR .EQ. 4) JFDIR = 3
      RETURN
      END
C
C
      SUBROUTINE DHEAD(NX)
C     DISPLAY NOTE HEAD CENTRED ON (*NX*,*IYN*).
      COMMON /DN/ JSDIR,JFDIR,JLEV,JALEV,JPVLEV,
     * IXSM,IYSM,IYSMAX,IYSMIN,MXTEXT(3)
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /PP/ MPP1(3),ISHOD,MPP2(3)
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /XF/ MCORD4(76)
      DIMENSION LHCHAR(6),LYINC(6)
      DATA LHCHAR(1)/1HO/,LHCHAR(2)/1HO/,
     * LHCHAR(3)/1HO/,LHCHAR(4)/1H*/,
     * LHCHAR(5)/1H./,LHCHAR(6)/1H+/,
```

```
     * LYINC(1)/1/,LYINC(2)/2/,LYINC(3)/2/,
     * LYINC(4)/1/,LYINC(5)/4/,LYINC(6)/0/
      IF (JLENTH .GT. 1) GOTO 3
C     DISPLAY BREVE HEAD
      IXT = IXN
      IYT = IYN
      CALL DSYMB(K0,K1,K1,MCORD4)
      GOTO 9
    3 IF (ISHQD .NE. 0) GOTO 5
C     DISPLAY NOTE HEAD USING CHARACTERS
      DO 40 L=1,4,3
   40 IF (JLENTH .GT. L-1) CALL TWRITE(NX,IYN-
     * IYINC+LYINC(ISSMAL+L),K1,LHCHAR(ISSMAL+L))
      GOTO 9
C     DISPLAY NOTE HEAD USING HERSHEY SYMBOL
    5 CALL PHEAD(NX)
    9 RETURN
      END
C
C
      SUBROUTINE DREST
C     DISPLAY REST SYMBOL.
      COMMON /DN/ JSDIR,JFDIR,JLEV,JALEV,JPVLEV,
     * IXSM,IYSM,IYSMAX,IYSMIN,MXTEXT(3)
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /ND/ NXLENG,NSHIFT,JBXL,NEXTX
      COMMON /N0/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /XF/ MCORD4(76)
      DIMENSION LIMP(1)
      DATA LIMP/4H IMP/
C
C     IF REST IS MINIM OR LONGER THEN MOVE SYMBOL
C     TO CENTRE OF SCOPE
      IF (JLENTH .LE. 3) IXN = IXP + NXLENG/K2
      IXT = IXN
      JLEV = 5
C     ADJUST VERTICAL POSITION IF IN-ACCORD
      IF (JFDIR .LE. 2) JLEV = 17 - 8 * JFDIR
      LIYT = IYS + JLEV * IYINC
      IF (JLENTH-3) 3,4,5
C     BREVE, SEMIBREVE OR WHOLE BAR REST
    3 IYT = LIYT - IYINC - 2
      GOTO 45
C     MINIM REST
    4 IYT = LIYT - IYINC + 4
C     DRAW SEMIBREVE OR MINIM REST
   45 CALL DSYMB(K0,K1,JLENTH+1,MCORD4)
      IXE = IXT
      IYN = IYT
      GOTO 8
C     CROTCHET REST OR SHORTER
    5 LID = 0
      IF (JLENTH .EQ. 4) LID = 2378
      IYT = LIYT
      IXE = IXT + IXINC / K2
      CALL DSYMB(LID,K1,JLENTH+1,MCORD4)
      IYN = LIYT - 2 * IYINC
    8 IYE = IYN - IYS
C     DISPLAY -IMP- FOR IMPLIED REST
      IF (ISTIE .NE. 0) CALL TWT(K4,LIMP)
      RETURN
      END
C
C
      SUBROUTINE DAXIDS
C     DISPLAY ACCIDENTALS FOR CURRENT NOTE ITEM.
      COMMON /DN/ JSDIR,JFDIR,JLEV,JALEV,JPVLEV,
     * IXSM,IYSM,IYSMAX,IYSMIN,MXTEXT(3)
      COMMON /DX/ JXBEAM,JYBEAM,JPVAX
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /N0/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
```

```
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /WS/ L1,L2,L3,JBYTE
      LIYN = IYN
      JBYTE = JAXIDS
      L1 = NBIT(K2)
      IF (L1-1) 4,3,2
C     PARENTHETICAL ACCIDENTALS
    2 IYN = IYS = 4 * IYINC
      GOTO 4
C     ACCIDENTALS ABOVE OR BELOW NOTE
    3 IYN = IYS + 14 * IYINC
    4 L1 = NBIT(K3)
      L2 = JBYTE / K2
      IF (L2 .EQ. 4) L1 = 0
      IXN = IXN - L1 * IXINC
      IF (JPVAX .EQ. 0) GOTO 5
C     SHIFT LEFT TO AVOID PREVIOUS ACCIDENTALS
      IF (JPVLEV-JLEV .LE. 4) IXN = IXN - IXINC
      JPVAX = 0
      GOTO 6
    5 JPVAX = 1
    6 IF (L1 .EQ. 0) GOTO 7
C     DISPLAY NATURAL SIGN BEFORE SHARP OR FLAT
      CALL DAXID(K3)
      IXN = IXN + IXINC
C     DISPLAY REMAINING ACCIDENTALS
    7 IF (L2.GE.2 .AND. L2.LE.6) CALL DAXID(L2-1)
      IXN = IXN + IXINC
      IYN = LIYN
      RETURN
      END
C
C
      SUBROUTINE DLEGER
C     DISPLAY LEGER LINES.
      COMMON /DN/ JSDIR,JFDIR,JLEV,JALEV,JPVLEV,
     * IXSM,IYSM,IYSMAX,IYSMIN,MXTEXT(3)
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      IF (JLEV .LT. 0) GOTO 2
C     ABOVE THE STAVE
      LSIGN = 1
      LSTART = 10
      LEND = JLEV
      GOTO 4
C     BELOW THE STAVE
    2 LSIGN = -1
      LSTART = 2
      LEND = -JLEV
    4 DO 50 L=LSTART,LEND,2
      IXT = IXSM = 5
      IYT = IYS + L * IYINC * LSIGN
      CALL TPOSN
      IXT = IXSM + 15
      CALL TLINE
   50 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE DEXTRA
C     DISPLAY ORNAMENTS AND EXPRESSION MARKS.
      COMMON /DN/ JSDIR,JFDIR,JLEV,JALEV,JPVLEV,
     * IXSM,IYSM,IYSMAX,IYSMIN,MXTEXT(3)
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /WS/ L1,L2,L3,JBYTE
      COMMON /XF/ MCORD4(76)
      DIMENSION LTEXT(2),LTRILL(1)
      DATA LTEXT(1)/2HFN/,LTEXT(2)/2HTM/
      DATA LTRILL/2HTR/
      EQUIVALENCE(NORNAM,MHEXT(2)),(NAX1,MHEXT(3))
      EQUIVALENCE(NAX2,MHEXT(4))
      LINC = 2 * IYINC
```

```
      IF (NORNAM .LE. 0 .OR. NORNAM .GT. 9) GOTO 3
      IF (NORNAM .EQ. 5) GOTO 25
      IF (NAX1 .EQ. 0) GOTO 22
C     DISPLAY ACCIDENTAL BELOW ORNAMENT
      CALL DAXID(NAX1+1)
      IYN = IYN + LINC
C     DISPLAY ORNAMENT
   22 IXT = IXE - 3
      IYT = IYN
      CALL DSYMB(K0,K1,NORNAM+10,MCORD4)
      IYN = IYN + LINC
      IF (NAX2 .EQ. 0) GOTO 3
C     DISPLAY ACCIDENTAL ABOVE ORNAMENT
      CALL DAXID(NAX2+1)
      IYN = IYN + LINC
      GOTO 3
C     DISPLAY TRILL
   25 CALL DCHARS(LTRILL)
    3 IF (ISEXTR .EQ. 0) GOTO 9
      IF (MEXTRA(2) .EQ. 0) GOTO 4
      JBYTE = MEXTRA(2)
      L1 = NBIT(K4)
C     DISPLAY FRACTIONING OR TREMOLO
      CALL DCHARS(LTEXT(L1+1))
      CALL TWT1(JBYTE)
    4 LL = 32 * MEXTRA(3) + MEXTRA(4)
      IF (LL .EQ. 0) GOTO 9
      DO 60 L=1,10
      IF (NSPLIT(LL,L-3) .EQ. 0) GOTO 6
C     DISPLAY EXPRESSION MARK OR ARPEGGIO
      IXT = IXE - 2
      IYT = IYN
      CALL DSYMB(K0,K1,L+19,MCORD4)
C     INCREMENT Y CO-ORDINATE EXCEPT FOR ARPEGGIOS
      IF (L .GT. 2) IYN = IYN + LINC
    6 CONTINUE
   60 CONTINUE
    9 RETURN
      END
C
C
      SUBROUTINE DCHARS(NTEXT)
C     WRITE FIRST TWO CHARACTERS OF *NTEXT*
C     ABOVE STAVE TO REPRESENT -EXTRA- SYMBOL.
      COMMON /DN/ JSDIR,JFDIR,JLEV,JALEV,JPVLEV,
     * IXSM,IYSM,IYSMAX,IYSMIN,MXTEXT(3)
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      DIMENSION NTEXT(1)
      CALL TWRITE(IXE-IXINC,IYN,K2,NTEXT)
      IYN = IYN + 2 * IYINC
      RETURN
      END
!ALL (FIL,GO),(RSI,27),(FSI,200),(FOR,B)
!FORTRAN GO,NS,S
C
C
C     ********************************************
C     *                                          *
C     *  SEGMENT 325 - DISPLAY OF BRAILLE ITEMS  *
C     *                                          *
C     ********************************************
C
C
      SUBROUTINE BRDISP
C     DISPLAY BRAILLE ITEM.
C     (*IXN*,*IYS*) IS CO-ORDINATE OF DOT 3.
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /WS/ L1,L2,L3,JBYTE
      JBYTE = ITEM
      LFLAG = NBIT(K1)
      IXN = IXN + 25
      IXE = IXN + 5
```

```
      IF (NORNAM .LE. 0 .OR. NORNAM .GT. 9) GOTO 3
      IF (NORNAM .EQ. 5) GOTO 25
      IF (NAX1 .EQ. 0) GOTO 22
C     DISPLAY ACCIDENTAL BELOW ORNAMENT
      CALL DAXID(NAX1+1)
      IYN = IYN + LINC
C     DISPLAY ORNAMENT
   22 IXT = IXE - 3
      IYT = IYN
      CALL DSYMB(K0,K1,NORNAM+10,MCORD4)
      IYN = IYN + LINC
      IF (NAX2 .EQ. 0) GOTO 3
C     DISPLAY ACCIDENTAL ABOVE ORNAMENT
      CALL DAXID(NAX2+1)
      IYN = IYN + LINC
      GOTO 3
C     DISPLAY TRILL
   25 CALL DCHARS(LTRILL)
    3 IF (ISEXTR .EQ. 0) GOTO 9
      IF (MEXTRA(2) .EQ. 0) GOTO 4
      JBYTE = MEXTRA(2)
      L1 = NBIT(K4)
C     DISPLAY FRACTIONING OR TREMOLO
      CALL DCHARS(LTEXT(L1+1))
      CALL TWT1(JBYTE)
    4 LL = 32 * MEXTRA(3) + MEXTRA(4)
      IF (LL .EQ. 0) GOTO 9
      DO 60 L=1,10
      IF (NSPLIT(LL,L-3) .EQ. 0) GOTO 6
C     DISPLAY EXPRESSION MARK OR ARPEGGIO
      IXT = IXE - 2
      IYT = IYN
      CALL DSYMB(K0,K1,L+19,MCORD4)
C     INCREMENT Y CO-ORDINATE EXCEPT FOR ARPEGGIOS
      IF (L .GT. 2) IYN = IYN + LINC
    6 CONTINUE
   60 CONTINUE
    9 RETURN
      END
C
C
      SUBROUTINE DCHARS(NTEXT)
C     WRITE FIRST TWO CHARACTERS OF *NTEXT*
C     ABOVE STAVE TO REPRESENT -EXTRA- SYMBOL.
      COMMON /DN/ JSDIR,JFDIR,JLEV,JALEV,JPVLEV,
     * IXSM,IYSM,IYSMAX,IYSMIN,MXTEXT(3)
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      DIMENSION NTEXT(1)
      CALL TWRITE(IXE-IXINC,IYN,K2,NTEXT)
      IYN = IYN + 2 * IYINC
      RETURN
      END
!ALL (FIL,GO),(RSI,27),(FSI,200),(FOR,8)
!FORTRAN GO,NS,S
C
C
C     ********************************************
C     *                                          *
C     *  SEGMENT 325 - DISPLAY OF BRAILLE ITEMS  *
C     *                                          *
C     ********************************************
C
C
      SUBROUTINE BRDISP
C     DISPLAY BRAILLE ITEM.
C     (*IXN*,*IYS*) IS CO-ORDINATE OF DOT 3.
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /QL/ IXQ,IYQ,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /WS/ L1,L2,L3,JBYTE
      JBYTE = ITEM
      LFLAG = NBIT(K1)
      IXN = IXN + 25
      IXE = IXN + 5
```

```
      IYE = 10
      DO 30 L=1,2
      DO 30 LL=1,3
   30 IF (NBIT(3*L+LL-2) .GT. 0) CALL TWRITE
     * (IXN+10*(2-L),IYS+10*(LL-1),K1,KDOT)
      IF (LFLAG .EQ. 0) GOTO 8
C     DISPLAY DASH FOR BLANK OR SPECIAL CELL
      IXT = IXN + 3
      IYT = IYS - 5
      CALL TPOSN
      IXT = IXT + 13
      CALL TLINE
    8 IF (IXE .GE. JXMAX - 25) ISNXST = 2
      RETURN
      END
C
C
      SUBROUTINE BPRINT
C     PRINT BRAILLE ON LINEPRINTER.
      COMMON MC1(188),NILENG,MIBAR(64),NOLENG,
     * MOBAR(64)
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MC/ KBSIZE,KWSIZE,KXMAX,KYMAX,KFSIZE
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION LTEXT(2)
C     USE OUTPUT MEASURE AS PRINT BUFFER
      EQUIVALENCE(MPRBUF,MOBAR)
      DATA LPRDEV/108/,LPRBUF/128/,LPRLPP/62/
      DATA LEONE/241/
      LPRWDS = (LPRBUF-1) / KWSIZE + 1
      LLINES = LPRLPP
      CALL GETHDR
    2 CALL NXSEL
      IF (ISEND .NE. 0) GOTO 8
C     TRUNCATE BRAILLE LINE TO WIDTH OF PRINTER
      LENG = MIN0(NILENG,40)
C     FIVE PRINT LINES FOR EACH BRAILLE LINE
      DO 60 L1=1,5
C     RESET PRINT BUFFER TO BLANKS
      DO 30 L2=1,LPRBUF
   30 CALL STBYTE(64,MPRBUF,L2)
      GOTO (32,35,33,35,5),L1
C     FIRST LINE IS BLANK OR PAGE FEED
   32 LLINES = LLINES + 5
      IF (LLINES .LE. LPRLPP) GOTO 5
C    INSERT PAGE FEED VERTICAL CONTROL
      CALL STBYTE(LEONE,MPRBUF,K1)
      LLINES = 5
      GOTO 5
C     CONVERT LINE NUMBER TO CHARACTER FORMAT
   33 CALL ICCONV(NIB,LCHARS,LTEXT)
C     PUT LINE NUMBER AT START OF THIRD LINE
      CALL MBS(LTEXT,K1,MPRBUF,K1,LCHARS)
C     SET UP BRAILLE CELL IMAGE USING DOTS
   35 DO 40 L2=1,LENG
      DO 40 L3=1,2
      IF (ISBIT(MIBAR,8*L2-L1-3*L3+4) .NE. 0)
     *     CALL STBYTE(75,MPRBUF,6+3*L2+L3)
   40 CONTINUE
C     PRINT CONTENTS OF PRINT BUFFER
    5 CALL SY
      CALL BUFFEROUT(LPRDEV,K1,MPRBUF,LPRWDS,
     *                          LSTAT)
      CALL SY
      IF (LSTAT .NE. 2) GOTO 75
   60 CONTINUE
      GOTO 2
   75 ISERR = 1
      GOTO 9
    8 ENDFILE LPRDEV
    9 RETURN
      END
C
C
      FUNCTION ISBIT(NARRAY,NPTR)
C     RETURNS *NPTR*TH BIT OF *NARRAY*.
C     HIGH ORDER BIT OF FIRST WORD IS BIT ZERO.
```

```
S       LI,2    0
S       LW,3    *NPTR
S       DW,2    -32             R3=WORD, R2=BIT POSN
S       AI,2    -31             GIVING SHIFT COUNT
S       LW,9    *NARRAY,3       R9=REQUIRED WORD
S       SLS,9   0,2             SHIFT REQD BIT TO RHS
S       AND,9   =1              MASK TO BOTTOM BIT
S       STW,9   ISBIT           STORE RESULT
        RETURN
        END
!MACRSYM SI,GO
*EXCHANGES SYSTEM SY FLAG WITH *SYFLAG* TO ALLOW
*FOREGROUND PROGRAM TO WRITE TO BACKGROUND FILE.
        DEF     SY
K:SY    EQU     X'152'          ADDR OF SYSTEM FLAG
SY      EQU     $               SET SKELETON WRITE
        XPSD,0  PSD             KEY AND GOTO SETSY
        B       *15             RETURN
SETSY   LW,1    SYFLAG          LOAD STORED VALUE
        XW,1    K:SY            EXCHANGE
        STW,1   SYFLAG          SAVE SYSTEM FLAG
        LPSD,0  PSD             EXIT SKELETON MODE
PSD     DATA    0,0,SETSY,0 P.S. DOUBLEWORDS
SYFLAG  DATA    1               SAVED WRITE-PERMIT
        END
!RADEDIT
:COP (FIL,BT,GO),(FIL,D1,JHROM325)
!JOB JH,COMPILE MUSIC PROGRAM SEGMENT 4
!EXTRABGD 6800
!ASS (M:BO,D1,JHROM4)
!FORTRAN BO,S,NS
C
C
C     ***********************************************
C     *                                             *
C     *       SEGMENT 4 - BRAILLE ROUTINES          *
C     *                                             *
C     ***********************************************
C
C
```

```
      SUBROUTINE BRAILE
C     MAIN CONTROL ROUTINE FOR BRAILLE ROUTINES.
      COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /DB/ ISDUB,NDBL,ISSIG,NSLURS,JSLUR(2)
      COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /KC/ KWHYPH,KTOUT,KSPLIT,KBREAK,KRBAR
      COMMON /SE/ JCMD,KSEGS(7)
      COMMON /TA/ MPITCH(4),MREPT(4),MRESTS(4),
     * MCXMEM(3),MPCONT(4),ISPCON,JBNO,JISNO,
     * JOSNO,ISCANC,ISLINR,JLINR
      COMMON /TJ/ JCKMAX,MCHECK(100)
      COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISGUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
      COMMON /XG/ MCELLS(112)
      IF (JCMD .NE. 7) GOTO 2
      CALL LSEG(41)
      CALL BSETUP
    2 CALL LSEG(35+JCMD)
      IF (JCMD-8) 3,4,5
C
    3 CALL TRANS
      GOTO 9
    4 CALL FORMAT
      GOTO 9
    5 CALL EMBOSS
    9 RETURN
      END
C
C
      BLOCK DATA
C     THIS SUBPROGRAM IS PROGRAM-GENERATED.
      COMMON /XG/ MCELLS(112)
      DIMENSION M1(76)
      EQUIVALENCE(M1,MCELLS(1))
      DIMENSION M2(36)
      EQUIVALENCE(M2,MCELLS(77))
      DATA M1/
     *8Z09099026,8ZA0262600,8ZA826B026,8ZB826A104,
     *8Z1C0C2C3C,8Z1434224,8Z01030702,8Z05060400,
```

```
     *8Z3D3D1D39,8Z193D1D39,8Z19042C2C,8Z23232129,
     *8Z08202830,8Z3C008808,8Z08001800,8Z38001040,
     *8Z28003000,8Z2C00AC20,8Z003F3200,8Z3600001F,
     *8Z01030702,8Z05041232,8Z00160000,8Z36060024,
     *8Z00040501,8Z03070201,8Z04000200,8Z000026A3,
     *8Z23230021,8Z0C2900A9,8Z29123C00,8Z04362600,
     *8Z01030919,8Z110B1B13,8Z0A003F00,8Z00009022,
     *8Z1E05070D,8Z1D150F1F,8Z17372E00,8Z3F3E8404,
     *8Z000E1E25,8Z273A2D3D,8Z35212329,8Z39310000,
     *8Z00000000,8Z0C00909C,8Z05002B3B,8Z2A330000,
     *8Z01030919,8Z110B1B13,8Z0A003F00,8Z00000000,
     *8Z1A05070D,8Z1D150F1F,8Z17372E00,8Z003E0000,
     *8Z000E1E25,8Z273A2D3D,8Z35212329,8Z0000001A,
     *8Z01030919,8Z110B1B13,8Z0A002B3B,8Z2A338D8D,
     *8Z0D0D2527,8Z2C0D2527,8Z2D3D3D1D,8Z39193D1D,
     *8Z39193535,8Z15311135,8Z1531112F,8Z2F0F2B0B,
     *8Z2F0F2B0B,8Z3F3F1F3B,8Z1B3F1F3B,8Z1B373717/
      DATA M2/
     *8Z33133717,8Z33132E2E,8Z0E2A0A2E,8Z0E2A0A3E,
     *8Z3E1E3A1A,8Z3E1E3A1A,8ZA31C00A8,8Z05009002,
     *8Z00A30600,8ZA30500A3,8Z3600A336,8Z00A38504,
     *8Z05000000,8Z0C00AC40,8Z008C0700,8ZB8A307A8,
     *8Z1C00B81C,8Z0CBC0200,8ZBC06A032,8Z00A0B207,
     *8Z32B003B2,8Z07001698,8Z06909607,8ZB0960790,
     *8Z1600B016,8ZA309A109,8Z90A309A8,8Z09B80900,
     *8Z00008809,8ZA8099040,8ZA00C0000,8ZB89C84A4,
     *8Z2490923C,8ZA81CB81C,8Z981C0000,8Z00000000/
      END
C
C
      SUBROUTINE OPBRL(N)
C     OUTPUT BRAILLE MUSIC SIGN *N*.
C     *ISOPT* = 0 FOR COMPULSORY CELLS, 1 FOR
C     CELLS IN FIRST MEASURE OF LINE ONLY.
      COMMON /DB/ ISDUB,NDBL,ISSIG,NSLURS,JSLUR(2)
      COMMON /KB/ KBCOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG   C
      COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),  C
     * ISQUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
      COMMON /XG/ MCELLS(112)
      IF (IDOUB .NE. 0) GOTO 8
      LPTR = N
    3 LCELL = NBYTE(MCELLS,LPTR)
      IF (LCELL .EQ. 0) GOTO 9
      LCONT = NSPLIT(LCELL,K0)
      IF (INCELL .EQ. 0) GOTO 5
C     INSERT *INCELL* IF DOTS 1,2 OR 3 PRESENT
C     UNLESS CURRENT CELL IS WORD SIGN
      IF (LCELL .NE. 28 .AND. LCELL/K8*8 .NE.
     *  LCELL) CALL BCELL(INCELL+64*ISOPT)
      INCELL = 0
    5 IF (LCELL .GT. 0) CALL BCELL(LCELL+64*ISOPT)
      IF (LCONT .EQ. 0) GOTO 8
      LPTR = LPTR + 1
      GOTO 3
    8 ISOPT = 0
    9 RETURN
      END


      SUBROUTINE BCELL(N)
C     STORE BRAILLE CODE *N* IN OUTPUT MEASURE.
      COMMON MC1(253),NOLENG
      COMMON /DB/ ISDUB,NDBL,ISSIG,NSLURS,JSLUR(2)
      COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISQUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
      IF (IDOUB .NE. 0) GOTO 9
      IF (INDENT .NE. -1) GOTO 6
      INDENT = NOLENG = NIPL
      INDTX = INDENT
    6 CALL BSTORE(N)
      IF (N .GT. 128) NIPL = NIPL + 1
    9 RETURN
      END
```

```
      SUBROUTINE BNUMB(N,NLOW)
C     TRANSLATE INTEGER *N* TO BRAILLE.
C     *NLOW* = 1 FOR UPPER PART OF CELL,
C              2 FOR LOWER PART OF CELL.
      COMMON /XG/ MCELLS(112)
      DIMENSION LTEXT(2)
C     CONVERT *N* TO CHARACTER FORM IN *LTEXT*
      CALL ICCONV(N,LLENG,LTEXT)
      DO 60 L=2,LLENG
   60 CALL BSTORE(NBYTE(MCELLS,NBYTE(LTEXT,L))
     *                          *NLOW)
      RETURN
      END
C
C
      BLOCK DATA
C     CONSTANTS FOR SEGMENT 4.
      COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /KC/ KWHYPH,KTOUT,KSPLIT,KBREAK,KRBAR
      DATA KBDOT/4/,KBHYPH/36/,KBNUMB/60/,KBRPT/
     *54/,KBSPAC/64/,KBLENG/40/,KBHSIG(1)/40/,
     *KBHSIG(2)/56/,KBHSIG(3)/24/,KWHYPH/252/,
     *KTOUT/252/,KSPLIT/253/,KBREAK/254/,KRBAR/16/
      END
!ASS (M:B0,D1,JHR0M41)
!FORTRAN B0,NS
C
C
C     ***********************************************
C     *                                             *
C     *   SEGMENT 41 - BRAILLE INITIALISATION       *
C     *                                             *
C     ***********************************************
C
C
      SUBROUTINE BSETUP
C     INITIALISATION FOR BRAILLE COMMAND.
      COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /DB/ ISDUB,NDBL,ISSIG,NSLURS,JSLUR(2)
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FS/ JCSTAT,JNSTAT,MSTAT(3)
      COMMON /HE/ MCLEF(10),MPART(10)
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NCP,NDL
      COMMON /TA/ MPITCH(4),MREPT(4),MRESTS(4),
     * MCXMEM(3),MPCONT(4),ISPCON,JBNO,JISNO,
     * JOSNO,ISCANC,ISLINR,JLINR
      COMMON /TJ/ JCKMAX,MCHECK(100)
      COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISQUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
      COMMON /WS/ L1,L2,L3,JBYTE
      EQUIVALENCE(NLNB,MLINES(1)),(NLUB,MLINES(2))
     *,(NLFB,MLINES(3)),(KRH,K1),(KLH,K2)
C
      IDOUB = 0
      ISTERM = 0
      JSLUR(1) = 0
      JSLUR(2) = 0
      NSLURS = 0
      ISLINR = 0
      JLINR = 0
C     IDENTIFY BRAILLE PARTS AND NUMBER OF
C     BRAILLE LINES
      L1 = 0
      ISQUOT = 0
      ISOPT = 0
      ISKEYB = 0
      DO 640 L=1,NLNB
C     IGNORE UNSELECTED STAVES
      IF (NBYTE(MSSEL,L) .EQ. 0) GOTO 64
      L1 = L1 + 1
      JPART = NBYTE(MPART,L)
      IF (JPART .GE. 64) GOTO 61
      IF (JPART .NE. KRH) GOTO 62
C     RIGHT HAND + LEFT HAND = KEYBOARD
      ISKEYB = 0
```

```
      IF (L .LT. NLNB .AND. NBYTE(MPART,L+1)
     *              .EQ. KLH) ISKEYB = 1
      GOTO 62
C     EXTRA BRAILLE PART CODE FOR WORDS LINE
   61 CALL STBYTE(64,MBPART,L1)
      L1 = L1 + 1
C     STORE BRAILLE PART CODE FOR MUSIC
   62 CALL STBYTE(JPART,MBPART,L1)
   64 CONTINUE
  640 CONTINUE
      NLUB = L1
C     INITIALISE CONTEXT FOR ALL OUTPUT STAVES
      DO 660 JOSNO=1,NSSEL
      MPITCH(JOSNO) = 0
      MREPT(JOSNO) = 0
      MRESTS(JOSNO) = 0
      MPCONT(JOSNO) = 0
      DO 650 L=1,3
  650 CALL STBYTE(KO,MCXMEM,3*(JOSNO-1)+L)
  660 CONTINUE
C     INITIALISE MEASURE-REPEAT CHECKING TABLE
      JCKMAX = 400 / NLUB
      DO 680 L=1,400
  680 CALL STBYTE(KO,MCHECK,L)
      ISPCON = 0
C     SET UNFORMATTED BRAILLE STATUS TO PROTECTED
      MSTAT(2) = 2
    9 RETURN
      END
!JOB JH,COMPILE MUSIC PROGRAM SEGMENT 42
!EXTRABGD 6800
!ASS (M:BO,D1,JHROM42)
!FORTRAN BO,NS
C
C
C     ********************************************
C     *                                          *
C     *       SEGMENT 42 - BRAILLE TRANSLATOR    *
C     *                                          *
C     ********************************************
C
C
      SUBROUTINE TRANS
C     TRANSLATE TO BRAILLE.
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /JS/ JSUM,JMATCH,JNLAST,JLASTN
      COMMON /KC/ KWHYPH,KTOUT,KSPLIT,KBREAK,KRBAR
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /TK/ NRLENG,MRBAR(4),MTKEY(4),
     * ISGR1,ISBEAT,ISPVBT,ISCMPD,ISPAUS,JXPOSE
      DATA LTSEG1/421/,LTSEG2/422/,LTSEG3/423/
C
C     TRANSLATE HEADER
      CALL LSEG(LTSEG2)
      CALL HTRAN1
      CALL LSEG(LTSEG3)
C     TRANSLATE KEY SIGNATURE
      CALL IDCODE
      CALL ITRANS
C     TRANSLATE TIME SIGNATURE
      CALL IDCODE
      CALL ITRANS
      CALL LSEG(LTSEG2)
      CALL HTRAN2
      GOTO 31
C
C     OBTAIN NEXT MEASURE FOR TRANSLATION
    3 CALL NXSEL
   31 IF (ISEND .NE. 0) GOTO 9
C     TEST FOR REPEATS ETC.
      CALL LSEG(LTSEG1)
      CALL MTEST
C     SET CONTEXT FOR LINE AND TRANSLATE WORDS
      CALL LSEG(LTSEG2)
      CALL MTRAN1
C     TRANSLATE MUSIC
      CALL LSEG(LTSEG3)
      CALL MTRAN2
C     SAVE CONTEXT FOR CURRENT LINE
```

```
      CALL LSEG(LTSEG1)
      CALL MTRAN3
      GOTO 3
    9 RETURN
      END
!ASS (M:B0,D1,JHR0M421)
!FORTRAN B0,NS
C
C
C     ***********************************************
C     *                                             *
C     *     SEGMENT 421 - PATTERN RECOGNITION       *
C     *                                             *
C     ***********************************************
C
C
      SUBROUTINE MTEST
C     IDENTIFY MEASURE REPEATS, WHOLE-MEASURE
C     RESTS, PARALLEL MOVEMENT OF HANDS AND
C     PART-MEASURE REPEATS.
      COMMON MC1(183),J0PSEC,MC2(1),J0PBUF,MC3(3),
     * MIBAR(64),N0LENG,M0BAR(64),JIBAR,J0BAR
      COMMON /B1/ IFIRST,J0PAGE
      COMMON /FF/ ISCTK,ICFLAG,ISV0C,ISSQZ,
     * ISREST,ISAHED,ISXP0S,ISEMB
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IS/ ICITEM,ICTYPE,MCSAVE(9),
     * MCXTRA(4),ISPMT
      COMMON /JS/ JSUM,JMATCH,JNLAST,JLASTN
      COMMON /KB/ KBC0T,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /KC/ KWHYPH,KT0UT,KSPLIT,KBREAK,KRBAR
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KY/ JKEYN0,JKEYSG,MKPCH(2),MTKPCH(2)
      COMMON /MD/ J0S,J0E,JSSEL,NSSEL,MSSEL(10)
      COMMON /NB/ JBAR,NIB,N0B,NIS,N0S,NIBS
      COMMON /NS/ NLINES,JLINE,JPART,JV0IC,NDP,NDL
      COMMON /TA/ MPITCH(4),MREPT(4),MRESTS(4),
     * MCXMEM(3),MPCONT(4),ISPCON,JBN0,JISN0,
     * J0SN0,ISCANC,ISLINR,JLINR
      COMMON /TJ/ JCKMAX,MCHECK(100)
      COMMON /TK/ NRLENG,MRBAR(4),MTKEY(4),
     * ISGR1,ISBEAT,ISPVBT,ISCMPD,ISPAUS,JXP0SE
      COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISQU0T,ISFEND,NIPL,ID0UB,IND1,INDENT,INDTX,
     * IS0CV,ISKEYB,ISTERM,IS0PT,JADDR1
      EQUIVALENCE(KRH,K1),(KLH,K2)
      DIMENSION LMESS(1)
      DATA LBREST/13/,LMESS/3HBAR/
C
      ISPMT = 0
      LMATCH = 0
C     SAVE KEY SIGNATURE AT START OF MEASURE
      CALL MBS(MKPCH,K1,MTKEY,K1,16)
C     SET INPUT MEASURE-NUMBER FOR THIS MEASURE
      JBN0 = NIB
      JISN0 = NIS
      J0SN0 = NBYTE(MSSEL,JISN0)
C     DISPLAY MEASURE NUMBER IF MORE THAN 100
      IF (JBN0 .LE. 100) GOTO 2
      CALL TMESS(K3,LMESS)
      CALL TWT1(JBN0)
    2 CONTINUE
C     SCAN MEASURE TO FIND CHECKSUM
      CALL TSCAN
      JLASTN = JNLAST
C     SAVE CHECKSUM FOR COMPARISON OF LATER
C     MEASURES WITH THIS ONE
      CALL STBYTE(JSUM,MCHECK,(JBN0-
     * JBN0/JCKMAX*JCKMAX-1)*MLINES(2)+J0SN0)
      IF (ISKEYB .EQ. 0) GOTO 4
C     TEST FOR LEFT AND RIGHT HANDS PARALLEL
      IF (JPART .NE. KRH) GOTO 3
C     RIGHT HAND PART
      IF (JLINR .GT. 0) GOTO 4
      CALL LHTEST
      CALL LOCATE
      GOTO 4
```

```
    3 IF (JPART .NE. KLH) GOTO 4
C     LEFT HAND PART
      IF (JLINR .LE. 0) GOTO 4
      JLINR = JLINR - 1
      LMATCH = 5
      GOTO 7
C
C     TEST FOR MEASURE RESTS
    4 NOLENG = 0
      LNREST = MRESTS(JOSNO)
      IF (LNREST .LE. 0) GOTO 41
      GOTO 47
   41 ISCHEK = 0
   42 IF (ISREST .EQ. 0) GOTO 44
      LNREST = LNREST + 1
      DO 430 L=1,NLINES
  430 CALL GETBAR
      IF (ISEND .EQ. 0) GOTO 42
   44 IF (LNREST .LE. 0) GOTO 48
      CALL LOCATE
      MRESTS(JOSNO) = LNREST
C     OUTPUT SINGLE OR MULTIPLE MEASURE REST
      IF (LNREST .LE. 3) GOTO 45
      CALL OPBRL(20)
      CALL BNUMB(LNREST,K1)
      CALL OPBRL(257)
      GOTO 47
C     ONE, TWO OR THREE MEASURES REST
   45 CALL OPBRL(258-LNREST)
   47 MRESTS(JOSNO) = LNREST - 1
      LMATCH = 1
      GOTO 7
C
C     TEST FOR MEASURE REPEATS
   48 IF (JBNO .LE. 1) GOTO 72
      NOLENG = 0
      LNREPT = MREPT(JOSNO)
      IF (LNREPT) 67,49,665
C     NO REPEAT IF NO NOTES IN CURRENT MEASURE
   49 IF (JSUM .EQ. C) GOTO 72
      LSTART = 0
      LNEW = JBNO
      ISCHEK = 0
C     SAVE CURRENT MEASURE FOR COMPARISON
      CALL BMOVE(MIBAR,MOBAR)
      IF (JPART .LT. 64) GOTO 52
C     NUMERAL REPEATS NOT ALLOWED
      LOLD = LNEW - 1
      GOTO 54
   52 LOLD = MAXO(KO,LNEW-JCKMAX)
   53 LOLD = LOLD + 1
      IF (LOLD .GE. JBNO .AND. LSTART .LT. JBNO-1)
     *                          GOTO 6
C     COMPARE CHECKSUMS
   54 IF (NBYTE(MCHECK,(LOLD-LOLD/JCKMAX*JCKMAX-1)
     *    *MLINES(2)+JOSNO).NE. JSUM) GOTO 57
C     FETCH EARLIER MEASURE FOR COMPARISON
      CALL BFIND(LOLD,JISNO)
C     COMPARE CURRENT MEASURE WITH EARLIER ONE
      JOBAR = 0
      CALL CMPARE(KO,LSTART)
      IF (JMATCH .NE. 1) GOTO 57
C     MATCH
      LNEW = LNEW + 1
C     FIND NEXT MEASURE AND SAVE FOR COMPARISON
      CALL BFIND(LNEW,JISNO)
      CALL BMOVE(MIBAR,MOBAR)
      CALL TSCAN
C     SET STARTING MEASURE NUMBER FOR REPEAT SEQ
      IF (LSTART .EQ. 0) LSTART = LOLD
      GOTO 53
C     NO MATCH
   57 IF (LSTART .EQ. 0) GOTO 53
      IF (LSTART .LT. JBNO-1) GOTO 62
C     OUTPUT BRAILLE REPEAT SIGN FOR A SEQUENCE
C     OF IDENTICAL MEASURES
      LMATCH = 2
      NOLENG = 0
      CALL BSTORE(KBRPT)
      LNREPT = LOLD - LSTART
```

```
      IF (LNREPT-2) 66,59,58
C     THREE OR MORE IDENTICAL MEASURE REPEATS
   58 CALL BSTORE(KBNUMB)
      CALL BNUMB(LNREPT,K1)
      GOTO 66
C     SET FLAG FOR SECOND MEASURE REPEAT TO FOLLOW
   59 LNREPT = 0
      GOTO 66
C
    6 IF (LSTART .EQ. 0) GOTO 72
   62 LNREPT = LOLD - LSTART
C     OUTPUT NUMERAL REPEAT SIGN
      NOLENG = 0
      CALL BSTORE(KBNUMB)
      CALL BNUMB(LSTART-1+IFIRST,K2)
      IF (LNREPT .EQ. 1) GOTO 65
C     OUTPUT HYPHEN AND SECOND NUMBER
      CALL BSTORE(KBHYPH)
      CALL BNUMB(LOLD-2+IFIRST,K2)
      LMATCH = 4
      GOTO 66
C     IGNORE SINGLE MEASURE REPEAT IF MORE THAN
C     10 BARS BACK
   65 IF (JBNO - LSTART .GT. 10) GOTO 72
      LMATCH = 3
   66 CALL LOCATE
      GOTO 68
C     NULL MEASURE FOLLOWING NUMERAL REPEAT
  665 LMATCH = 5
      GOTO 68
C     STORE SECOND OF TWO MEASURE REPEATS
   67 LMATCH = 2
      CALL BSTORE(KBRPT)
      LNREPT = 1
C     SAVE EXTENT OF REPEAT SEQUENCE
   68 MREPT(JOSNO) = LNREPT = 1
C     SAVE REPEAT OR NULL MEASURE
    7 NOLENG = MINO(NOLENG,KRBAR)
      CALL BMOVE(MOBAR,MRBAR)
C     TEST FOR PART-MEASURE REPEATS
   72 CALL LOCATE
      CALL PMTEST
      JMATCH = LMATCH
      RETURN
      END
C
C
      SUBROUTINE TSCAN
C     IF NO PLAYED NOTES IN CURRENT MEASURE,
C     RETURNS *JSUM* = *JNLAST* = 0 OTHERWISE
C     *JSUM* = CHECKSUM VALUE FOR MEASURE,
C     *JNLAST* = POINTER TO LAST PLAYED NOTE.
      COMMON MC1(318),JIBAR
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /JS/ JSUM,JMATCH,JNLAST,JLASTN
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      JSUM = 0
      JNLAST = 0
      JIBAR = 0
      ISCHEK = 0
    3 CALL IDCODE
      IF (ISEND .NE. 0) GOTO 8
      IF (ITTYPE .NE. 4 .OR. JPITCH .EQ. 0) GOTO 3
      JSUM = JSUM + ITEM + 128 + JPITCH
      JNLAST = ITPTR
      GOTO 3
C     MASK CHECKSUM TO LEAST SIGNIFICANT BYTE
    8 IF (JSUM .GT. 0) JSUM = JSUM-JSUM/255*255+1
      RETURN
      END
C
C
      SUBROUTINE LOCATE
C     POSITION FILE TO CURRENT TRANSLATION MEASURE
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
```

```
      COMMON /NB/ JBAR,NIB,NBB,NIS,NBS,NIBS
      COMMON /TA/ MPITCH(4),MREPT(4),MRESTS(4),
     * MCXMEM(3),MPCENT(4),ISPCBN,JBNB,JISNB,
     * JBSNB,ISCANC,ISLINR,JLINR
      ISCHEK = 1
      IF (NIB .NE. JBNB .BR. NIS .NE. JISNB)
     *             CALL BFIND(JBNB,JISNB)
      RETURN
      END
C
C
      SUBROUTINE LHTEST
C     RETURNS *JLINR* = NUMBER OF BARS WITH LEFT
C     HAND ONE BCTAVE BELOW RIGHT HAND INCLUDING
C     CURRENT BAR.
      COMMON MC1(189),MIBAR(64),NBLENG,MBBAR(64),
     * JIBAR,JBBAR
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IS/ ICITEM,ICTYPE,MCSAVE(9),
     * MCXTRA(4),ISPMT
      COMMON /JS/ JSUM,JMATCH,JNLAST,JLASTN
      COMMON /KB/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /TA/ MPITCH(4),MREPT(4),MRESTS(4),
     * MCXMEM(3),MPCENT(4),ISPCBN,JBNB,JISNB,
     * JBSNB,ISCANC,ISLINR,JLINR
      ISCHEK = 0
    5 CALL BMBVE(MIBAR,MBBAR)
C     GET LEFT-HAND MEASURE
      CALL GETBAR
C     COMPARE RIGHT AND LEFT HAND PARTS
      JBBAR = 0
      CALL CMPARE(K1,KO)
      IF (JMATCH .NE. 2) GBTB 8
      ISLINR = 1
      JLINR = JLINR + 1
C     GET NEXT RIGHT HAND MEASURE
      CALL GETBAR
      IF (ISEND .EQ. 0) GBTB 5
    8 IF (JLINR .NE. 1) GBTB 9
C     DBNT BBTHER FBR BNE MEASURE BNLY
      ISLINR = 0
      JLINR = 0
      JMATCH = 0
    9 RETURN
      END
C
C
      SUBROUTINE PMTEST
C     IDENTIFY BEATS AND PART-MEASURE REPEATS.
      COMMON MC1(188),NILENG,MIBAR(64),NBLENG,
     * MBBAR(64),JIBAR,JBBAR,JBBUF
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IS/ ICITEM,ICTYPE,MCSAVE(9),
     * MCXTRA(4),ISPMT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /JS/ JSUM,JMATCH,JNLAST,JLASTN
      COMMON /KB/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNBTES
      COMMON /TK/ NRLENG,MRBAR(4),MTKEY(4),
     * ISGR1,ISBEAT,ISPVBT,ISCMPD,ISPAUS,JXPBSE
      COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISQUBT,ISFEND,NIPL,IDBUB,IND1,INDENT,INDTX,
     * ISBCV,ISKEYB,ISTERM,ISBPT,JADDR1
      DIMENSION LMKEY(4),LMEES(4)
      EQUIVALENCE(LNEES,K4)
      DATA LMEES(1)/50/,LMEES(2)/52/,LMEES(3)/56/,
     * LMEES(4)/58/
      ISPMT = 1
C     IDENTIFY START AND END BF EACH BEAT
      JTIME = 0
      JIBAR = 0
      NBEATS = 0
      DB 20 L=1,3
   20 CALL STBYTE(KO,MBRPT,L)
C     FETCH NEXT ITEM
    3 CALL GITEM(KO)
```

```
      IF (ISEND .NE. 0) GOTO 4
      IF (JTIME-NBEATS*JBDUR) 3,35,4
C     TEST FOR -END- ITEMS
   35 IF (ITTYPE .EQ. 4) GOTO 3
      DO 370 L=1,LNEES
  370 IF (ITEM .EQ. LMEES(L)) GOTO 3
C     START OF BEAT
    4 NBEATS = NBEATS + 1
C     SAVE KEY SIGNATURE AT START OF FIRST AND
C     SECOND BEATS
      IF (NBEATS .LE. 2) CALL MBS(MTKEY,K1,LMKEY,
     *                            17-8*NBEATS,K8)
C     SAVE STARTING POSITION OF CURRENT BEAT
      CALL STBYTE(ITPTR,MBEAT,NBEATS)
      IF (ISEND .EQ. 0) GOTO 3
      NBEATS = NBEATS - 1
C     MAKE 2ND COPY OF MEASURE FOR BEAT COMPARISON
      CALL BMOVE(MIBAR,MOBAR)
      LILENG = NILENG
      CALL MBS(LMKEY,K1,MTKEY,K1,16)
      ISCHEK = 0
C     IDENTIFY REPEATED BEATS
      DO 60 L=1,NBEATS
      IF (L .EQ. 1) GOTO 6
      JOBAR = NBYTE(MBEAT,L-1)
      JIBAR = NBYTE(MBEAT,L)
      NOLENG = JIBAR
      NILENG = NBYTE(MBEAT,L+1)
      CALL CMPARE(KO,KO)
      IF (JMATCH .NE. 1 .OR. NILENG-NOLENG .LE. 3)
     *                          GOTO 6
C     CURRENT BEAT IS REPEAT OF PREVIOUS BEAT
      CALL STBYTE(K1,MBRPT,L)
    6 CONTINUE
   60 CONTINUE
      NILENG = LILENG
C     RESTORE MEMORY AS AT START OF MEASURE
      CALL MMOVE(MSMEM,MCMEM)
      ISCHEK = 1
      RETURN
      END
C
C
      SUBROUTINE CMPARE(N,NSTART)
C     COMPARES PRIMARY MEASURE WITH SECONDARY
C     MEASURE, STARTING AT *JIBAR* AND *JOBAR*.
C     IF *N* = 0 *MIBAR* IS PRIMARY
C     IF *N* = 1 *MOBAR* IS PRIMARY,
C     *NSTART* = 0 TO IGNORE LEADING TEXT ITEMS,
C                NON-ZERO OTHERWISE,
C     RETURNS *JMATCH* = 0 DIFFERENT, 1 SAME,
C     2 SECONDARY MEASURE OCTAVE BELOW PRIMARY.
      COMMON MC1(318),JIBAR,JOBAR
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IS/ ICITEM,ICTYPE,MCSAVE(9),
     * MCXTRA(4),ISPMT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /IU/ MITEM2(8)
      COMMON /JS/ JSUM,JMATCH,JNLAST,JLASTN
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,K8IG
      COMMON /N0/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      DIMENSION JBBAR(2),LISNT(2)
      EQUIVALENCE(MHSAVE,MCSAVE(2)),(JBBAR,JIBAR)
      LOGICAL CBS
      LDIFF = 0
      LISNT(1) = NSTART
      LISNT(2) = NSTART
C     NEXT PRIMARY MEASURE ITEM
    2 CALL GITEM(N)
      IF (ISEND .NE. 0) GOTO 5
C     SAVE ITEM
      ICITEM = ITEM
      ICTYPE = ITTYPE
      GOTO (4,32,34,36),ITTYPE
      GOTO 4
C     SAVE SEPARATE SIGN ITEM
```

```
   32 IF (ITEM.LE.38) MCSAVE(1) = MITEM2(ITEM-32)
      GOTO 4
C     IGNORE LEADING TEXT ITEMS FOR FIRST MEASURE
C     OF REPEAT SEQUENCE
   34 IF (LISNT(N+1) .EQ. 0) GOTO 2
C     SAVE TEXT ITEM
      CALL MBS(MTEXT,K1,MCSAVE,K1,NCHARS)
      GOTO 4
C     SAVE NOTE ITEM
   36 MCSAVE(1) = NHEDS
      CALL MBS(MHD,K1,MHSAVE,K1,KHS*NHEDS)
      DO 380 L=1,4
  380 MCXTRA(L) = MEXTRA(L)
C     CANCEL FIRST-ITEM INDICATOR
    4 LISNT(N+1) = 1
C     FETCH NEXT SECONDARY MEASURE ITEM
   41 CALL GITEM(1-N)
      IF (ISEND .NE. 0) GOTO 6
C     IGNORE LEADING TEXT ITEMS
      IF (ITTYPE .EQ. 3 .AND. LISNT(2-N) .EQ. 0)
     *                          GOTO 41
      LISNT(2-N) = 1
C     COMPARE PRIMARY AND SECONDARY ITEMS
      IF (ITEM .NE. ICITEM) GOTO 6
      GOTO (7,42,44,46),ITTYPE
C     SEPARATE SIGN ITEM
   42 IF (ITEM .LE. 38 .AND. MCSAVE(1) .NE.
     *              MITEM2(ITEM-32)) GOTO 6
      GOTO 7
C     TEXT ITEM
   44 IF (CBS(MTEXT,K1,MCSAVE,K1,NCHARS)) GOTO 7
      GOTO 6
C     NOTE ITEM
   46 CALL NCOMP(L)
      IF (L .LT. 0) GOTO 7
      IF (LDIFF .GT. 0 .AND. LDIFF .NE. L .OR.
     *                       L .EQ. 0) GOTO 6
      IF (LDIFF .LE. 0) LDIFF = L
      GOTO 7
C
C     BOTH MEASURES MUST END TOGETHER TO MATCH
    5 CALL GITEM(N)
      IF (ISEND .NE. 0) GOTO 8
C     NO MATCH
    6 JMATCH = 0
      GOTO 85
C     MATCH ON CURRENT ITEM
    7 GOTO 2
C     MATCH ON WHOLE BAR
    8 JMATCH = LDIFF
   85 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE GITEM(N)
C     OBTAIN NEXT ITEM FROM MEASURE, IGNORING
C     ITEMS IRRELEVANT TO BRAILLE REPEATS.
C     *N* = 0 FOR MEASURE IN *MIBAR*,
C           1 FOR MEASURE IN *MOBAR*.
      COMMON MC1(318),JIBAR,JOBAR,JBBUF
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IS/ ICITEM,ICTYPE,MCSAVE(9),
     * MCXTRA(4),ISPMT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /TK/ NRLENG,MRBAR(4),MTKEY(4),
     * ISGR1,ISBEAT,ISPVBT,ISCMPD,ISPAUS,JXPOSE
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION LMIG(3)
      DATA LNIG/11/
      DATA LMIG/8Z05070C0D,8Z0E0F1025,8Z26272800/
      JBBUF = N
    2 CALL IDCODE
      IF (ISEND .NE. 0) GOTO 8
      IF (ITTYPE .EQ. 4 .AND. ISPMT .NE. 0) GOTO 4
```

```
      DO 30 L=1,LNIG
   30 IF (ITEM .EQ. NBYTE(LMIG,L)) GOTO 2
      GOTO 8
    4 IF (JPITCH .EQ. 0) GOTO 8
C     REPLACE ACCIDENTALS BY ACTUAL VALUES AND
C     UPDATE KEY SIGNATURE
      DO 70 L=1,NHEDS
      IF (JAXIDS .LE. 0) GOTO 5
      L1 = NSPLIT(JAXIDS,K3)
      JAXIDS = JAXIDS/K2 - 1
      CALL STBYTE(JAXIDS,MTKEY,8*N+JPITCH)
      GOTO 6
    5 JAXIDS = NBYTE(MTKEY,8*N+JPITCH)
    6 CALL STBYTE(JAXIDS,MHD,KHS*(L-1)+2)
   70 CONTINUE
    8 JBBUF = 0
      RETURN
      END
C
C
      SUBROUTINE NCOMP(N)
C     COMPARE CURRENT AND SAVED NOTE ITEMS.
C     RETURNS *N* = -1 MATCHING REST, 0 NO MATCH,
C     1 MATCHING NON-REST, 2 OCTAVE APART.
      COMMON /IS/ ICITEM,ICTYPE,MCSAVE(9),
     * MCXTRA(4),ISPMT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION LHEAD(2),LOCV(2),LPCH(2)
      EQUIVALENCE(MHSAVE,MCSAVE(2))
      IF (ISEXTR .EQ. 0) GOTO 3
C     COMPARE EXTRAS
      DO 20 L=1,4
   20 IF (MEXTRA(L) .NE. MCXTRA(L)) GOTO 6
    3 IF (NHEDS .NE. MCSAVE(1)) GOTO 6
      DO 50 L=1,NHEDS
      LPTR = KHS * (L-1)
      DO 350 LL=2,KHS
  350 IF (NBYTE(MHD,LPTR+LL) .NE.
     *            NBYTE(MHSAVE,LPTR+LL)) GOTO 6
      LHEAD(1) = NBYTE(MHD,LPTR+1)
      LHEAD(2) = NBYTE(MHSAVE,LPTR+1)
      DO 370 LL=1,2
      JBYTE = LHEAD(LL)
C     THROW AWAY FLAGS
      LIG = NBIT(K1)
      LOCV(LL) = (JBYTE - 1) / K7
      LPCH(LL) = JBYTE - 7 * LOCV(LL)
  370 CONTINUE
      IF (LPCH(1) .NE. LPCH(2)) GOTO 6
      IF (L .NE. 1) GOTO 4
      LDIFF = LOCV(1) - LOCV(2)
      GOTO 5
    4 IF ((LOCV(1) - LOCV(2)) .NE. LDIFF) GOTO 6
    5 CONTINUE
   50 CONTINUE
      IF (LDIFF .NE. 0) GOTO 8
      IF (JPITCH .GT. 0) GOTO 7
      N = -1
      GOTO 9
    6 N = 0
      GOTO 9
    7 N = 1
      GOTO 9
    8 N = 2
    9 RETURN
      END
C
C
      SUBROUTINE MTRAN3
C     STORE BRAILLE MEASURE AND SAVE BRAILLE
C     CONTEXT FOR CURRENT OUTPUT LINE.
      COMMON MC1(253),NOLENG,MOBAR(64)
      COMMON /AD/ JIADDR,JOADDR
      COMMON /DB/ ISDUB,NDBL,ISSIG,NSLURS,JSLUR(2)
      COMMON /JS/ JSUM,JMATCH,JNLAST,JLASTN
```

```
      COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /PV/ JPPCH,JPLEN,NNINCR,IXP,JREPT,JPN
      COMMON /TA/ MPITCH(4),MREPT(4),MRESTS(4),
     * MCXMEM(3),MPCENT(4),ISPCON,JBNO,JISNO,
     * JOSNO,ISCANC,ISLINR,JLINR
      COMMON /TK/ NRLENG,MRBAR(4),MTKEY(4),
     * ISGR1,ISBEAT,ISPVOT,ISCMPD,ISPAUS,JXPOSE
      COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISQUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
      EQUIVALENCE(MCX,NSLURS)
      DIMENSION MCX(3)
      IF (JOSNO .EQ. 1) IND1 = INDENT
      ISOCV = 1
      DO 20 L=1,3
   20 CALL STBYTE(MCX(L),MCXMEM,3*(JOSNO-1)+L)
      MPITCH(JOSNO) = JPPCH
      IF (JMATCH .EQ. 0) GOTO 5
      IF (JMATCH .NE. 3) GOTO 3
C     USE BRAILLE REPEAT ONLY IF CURRENT MEASURE
C     WOULD OTHERWISE OCCUPY AT LEAST 8 CELLS
      IF (NOLENG .LT. 8) GOTO 5
    3 IF (JPART .LT. 64) GOTO 4
C     STORE NULL MEASURE FOR WORDS IF VOCAL PART
      L = NOLENG
      NOLENG = 0
      CALL STLINE(KO)
      NOLENG = L
C     ADD REPEAT SIGNS SET UP BY *MTEST* TO
C     LEADING TEXT IF ANY
    4 IF (NRLENG .EQ. 0) INDTX = 0
C     INSERT DOT 3 BETWEEN TEXT AND REPEAT SIGN
      IF (INDTX .EQ. 0 .OR. NBYTE(MOBAR,INDTX)
     *                    .EQ. KBDOT) GOTO 41
      INDTX = INDTX + 1
      CALL STBYTE(KBDOT,MOBAR,INDTX)
   41 CALL MBS(MRBAR,K1,MOBAR,INDTX+1,NRLENG)
      NOLENG = INDTX + NRLENG
      IF (JMATCH .NE. 2 .OR. JPITCH .EQ. 0) GOTO 5
C     TIE OR CHORD TIE AT END OF MEASURE REPEAT
      IF (ISCTIE .NE. 0) GOTO 42
      IF (ISTIE .NE. 0) GOTO 44
      GOTO 5
C     CHORD TIE
   42 CALL OPBRL(421)
      GOTO 5
C     ORDINARY TIE
   44 CALL OPBRL(419)
C     *LL*=NUMBER OF CELLS TO INDENT THIS MEASURE
    5 LL = IND1 - INDENT
      IF (LL .GE. 0) GOTO 6
C     INDENT FIRST STAVE INSTEAD
      CALL XBYTE(128-LL,JADDR1)
      LL = 0
C     STORE BRAILLE MEASURE
    6 CALL STLINE(LL)
      IF (JOSNO .EQ. 1) JADDR1 = JOADDR
      RETURN
      END
!ASS (M:B0,D1,JHROM422)
!FORTRAN B0,NS
C
C
C     ***********************************************
C     *                                             *
C     *     SEGMENT 422 - TRANSLATION OF WORDS      *
C     *                                             *
C     ***********************************************
C
C
      SUBROUTINE HTRAN1
C     TRANSLATE TEXTUAL PARTS OF HEADER.
      COMMON MC1(253),NOLENG,MC2(64),JIBAR
      COMMON /AD/ JIADDR,JOADDR
```

```
      COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /PP/ MPP1(4),ISGRD1,ISSAG,ISWIDE
      COMMON /TK/ NRLENG,MRBAR(4),MTKEY(4),
     * ISGR1,ISBEAT,ISPVBT,ISCMPD,ISPAUS,JXPOSE
      COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISQUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      EQUIVALENCE(NLLB,MLINES(2))
      DIMENSION LMPART(12)
      DATA LMPART/7,7HSOPRANO ,
     *    4,8HALTO    ,5,8HTENOR   ,4,8HBASS    /
C
      ISHDR = 1
      JBAR = 1
      ISGR1 = ISGRD1
C     TRANSLATE NAME AND SAVE OUTPUT POSITION
      CALL IDCODE
      CALL WTRANS
      CALL STLINE(K0)
      LHP = JOADDR
      LHL = 1
C     TRANSLATE TITLE, SUBTITLE, COMPOSER,
C     PUBLISHER, OPUS NUMBER
      LL = NHTEXT - 3
      DO 220 L=1,LL
      CALL IDCODE
      CALL WTRANS
      IF (NOLENG .LE. 0) GOTO 22
      CALL STLINE(K0)
      LHL = LHL + 1
   22 CONTINUE
  220 CONTINUE
      IF (NLUB .NE. 2) GOTO 3
      LPART = NBYTE(MBPART,K2)
      IF (LPART .LT. 66 .OR. LPART .GT. 69) GOTO 3
C     PART NAME
      CALL BMOVE(LMPART(3*(LPART-65)-1),MTEXT)
      CALL WTRANS
      CALL STLINE(K0)
      LHL = LHL + 1
    3 ISGR1 = 1
      INCELL = 0
C     TEMPO
      CALL IDCODE
      IF (NCHARS .EQ. 0) GOTO 32
      CALL WTRANS
C     ADD FULL STOP TO TEMPO IF NOT ALREADY THERE
      IF (NBYTE(MTEXT,NCHARS) .NE. 75)
     *                          CALL OPBRL(88)
C     METRONOME MARK
   32 CALL IDCODE
      IF (NCHARS .EQ. 0) GOTO 36
      IF (NOLENG .LE. 12) GOTO 33
C     PUT LONG TEMPO ON SEPARATE LINE IF METRONOME
C     MARK PRESENT
      CALL STLINE(K0)
      LHL = LHL + 1
      GOTO 34
   33 CALL OPBRL(64)
   34 CALL WTRANS
C     ADD FULL STOP
      CALL OPBRL(88)
   36 ISGR1 = ISGRD1
C     STORE NUMBER OF BRAILLE HEADER LINES
      CALL XBYTE(LHL+128,LHP)
C     SKIP COMMENTS ITEM
      CALL IDCODE
      RETURN
      END
C
C
```

```
      SUBROUTINE HTRAN2
C     STORE FINAL BRAILLE HEADER LINE + PART CODES
      COMMON MC1(253),NOLENG
      COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,KR,KBIG
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /TA/ MPITCH(4),MREPT(4),MRESTS(4),
     * MCXMEM(3),MPCONT(4),ISPCON,JBNO,JISNO,
     * JOSNO,ISCANC,ISLINR,JLINR
      EQUIVALENCE(NLLB,MLINES(2))
C     REMOVE TRAILING BLANK AFTER TIME SIGNATURE
      NOLENG = NOLENG - 1
C     STORE SIGNATURE LINE
      CALL STLINE(KO)
C     SET UP BRAILLE PART CODES
      DO 40 L=1,NLUB
   40 CALL BSTORE(NBYTE(MBPART,L))
C     STORE BRAILLE PART CODES
      CALL STLINE(KO)
C     CANCEL HEADER-TRANSLATION FLAG
      ISHDR = 0
C     SET PARAMETERS FOR AND FIND FIRST MEASURE
C     TO BE TRANSLATED
      NOB = 1
      NOS = 1
      JBNO = JBS
      JISNO = JSSEL
      JOSNO = 1
      CALL BFIND(JBNO,JISNO)
      RETURN
      END
C
C
      SUBROUTINE MTRAN1
C     SET CONTEXT FOR MEASURE AND TRANSLATE WORDS.
      COMMON MC1(253),NOLENG,MOBAR(64),JIBAR,JOBAR
      COMMON /DB/ ISDUB,NDBL,ISSIG,NSLURS,JSLUR(2)
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /KC/ KWHYPH,KTOUT,KSPLIT,KBREAK,KRBAR
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /TA/ MPITCH(4),MREPT(4),MRESTS(4),
     * MCXMEM(3),MPCONT(4),ISPCON,JBNO,JISNO,
     * JOSNO,ISCANC,ISLINR,JLINR
      COMMON /TK/ NRLENG,MRBAR(4),MTKEY(4),
     * ISGR1,ISBEAT,ISPVBT,ISCMPD,ISPAUS,JXPOSE
      COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISQUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      COMMON /TY/ ISCONT,ISBLOW
      DIMENSION MCX(3)
      EQUIVALENCE(MCX,NSLURS)
      DATA LTENOR/68/,LTREB/122/
      ISCANC = 1
C     RESTORE SLUR CONTEXT
      DO 30 L=1,3
   30 MCX(L) = NBYTE(MCXMEM,3*(JOSNO-1)+L)
      JPPCH = MPITCH(JOSNO)
      JIBAR = 0
      NOLENG = 0
      INDENT = -2
      INDTX = 0
      ISCONT = 0
      NIPL = 0
      INCELL = 0
```

```
      ISPVBT = 0
      ISFEND = 0
      JLINE = JISNO
C     SET TRANSPOSE FLAG IF TENOR IN TREBLE CLEF
      JXPOSE = 0
      IF (JPART .EQ. LTENOR .AND. JCLEF .EQ.
     *                LTREB) JXPOSE = 1
C     STORE INKPRINT PAGE-NUMBER CODE IF PENDING
C     FROM PREVIOUS UNSTORED MEASURE
      IF (NPAGE .GT. 0 .AND. JOSNO .EQ. 1)
     *                CALL BCELL(128+NPAGE)
      IF (JPART .LT. 64) GOTO 8
C     TRANSLATE WORDS
      ISPCON = MPCONT(JISNO)
    5 CALL IDCODE
      IF (ISEND .NE. 0) GOTO 7
      IF (ITTYPE .EQ. 3 .AND. ISWORD .NE. 0)
     *                          CALL WTRANS
      GOTO 5
C     ADD OPTIONAL HYPHEN IF LAST WORD CONTINUED
    7 IF (ISCONT .NE. 0) CALL BCELL(KWHYPH)
C     STORE BRAILLE WORDS
      CALL STLINE(KO)
      MPCONT(JISNO) = ISPCON
    8 ISCHEK = 1
      NBEATS = 0
      RETURN
      END
C
C
      SUBROUTINE WTRANS
C     TRANSLATE SUNG WORD OR SYLLABLE TO BRAILLE.
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /KB/ KBCOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /TA/ MPITCH(4),MREPT(4),MRESTS(4),
     * MCXMEM(3),MPCONT(4),ISPCON,JBNO,JISNO,
     * JOSNO,ISCANC,ISLINR,JLINR
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /TK/ NRLENG,MRBAR(4),MTKEY(4),
     * ISGR1,ISBEAT,ISPVBT,ISCMPD,ISPAUS,JXPOSE
      COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISQUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      COMMON /TY/ ISCONT,ISBLOW
      COMMON /XH/ MTABLE(220)
      LOGICAL CBS
      DIMENSION LMTYPE(64),LMXX(9)
C     TABLE OF CHARACTER TYPES INDEXED BY EBCDIC
      DATA LMTYPE/8Z05040000,7*8Z0000000C,
     28Z02020202,8Z02020202,8Z02020202,8Z02020202,
     38Z02020202,8Z02020000,8Z00000000,8Z0C000001,
     48Z00000000,8Z00000000,8Z00000200,8Z02000002,
     58Z00000000,8Z00000000,8Z00020002,8Z02020002,
     68Z00000000,8Z00000000,8Z00000202,8Z00020002,
     78Z00000000,8Z00000000,8Z00000202,8Z00020307,
     88Z08080808,8Z08080808,8Z08080808,8Z08080000,
     98Z08080808,8Z08080808,8Z08080808,8Z08080000,
     A8Z00080808,8Z08080808,8Z08080808,8Z08080000,
     B8Z00020202,8Z02000000,8Z00080808,8Z08080000,
     C8Z07070707,8Z07070707,8Z07070707,8Z07070000,
     D8Z07070707,8Z07070707,8Z07070707,8Z07070000,
     E8Z00070707,8Z07070707,8Z07070707,8Z07070006,
     F8Z06060606,8Z06060606,8Z06070707,8Z07070000/
      DATA LMXX(1)/4/,LMXX(2)/2/,LMXX(3)/4/,
     *     LMXX(4)/4/,LMXX(5)/4/,LMXX(6)/4/,
     *     LMXX(7)/12/,LMXX(8)/1/,LMXX(9)/1/
C
C     IGNORE EMPTY WORD ITEMS
      IF (NCHARS .LE. 0) GOTO 9
C     CONVERT TO UPPER CASE ONLY FOR COMPARISON
      DO 20 L=1,NCHARS
      LCHAR = NBYTE(MTEXT,L)
      IF (NBYTE(LMTYPE,LCHAR) .EQ. 8)
     *              CALL STBYTE(LCHAR+64,MTEXT,L)
   20 CONTINUE
```

```
C      INITIAL LEFT CONTEXT IS SPACE IF *ISPCON*
C      IS ZERO, LETTER OTHERWISE
       LLXX = 2 - ISPCON
C      IF *ISCONT*=0, APPEND SPACE TO ITEM,
C      OTHERWISE APPEND HEX 80
       CALL STBYTE(64*(ISCONT+1),MTEXT,NCHARS+1)
       LTEXT = 1
     3 LLEFT = NCHARS - LTEXT
       IF (LLEFT .LT. 0) GOTO 8
       LCHAR = NBYTE(MTEXT,LTEXT)
       LTYPE = NBYTE(LMTYPE,LCHAR) + 1
C      BRANCH INDEXED BY CHARACTER TYPE + 1
       GOTO (71,7,7,58,35,36,6,4),LTYPE
       GOTO 9
C      SET GRADE 1 TRANSLATION
    35 ISGR1 = 1
       GOTO 71
C      SET GRADE 2 TRANSLATION
    36 ISGR1 = 0
       GOTO 71
C      ALPHABETIC CHARACTER
     4 IF (ISGR1 .NE. 0 .OR. LLEFT .EQ. 0) GOTO 7
C      SEARCH CONTRACTION TABLE
       LENTRY = 0
       LSIZE = NHW(MTABLE,K1)
     5 LENTRY = LENTRY + 1
       IF (LENTRY .GT. LSIZE) GOTO 7
       LPTR = NHW(MTABLE,LENTRY+1)
       LLENG = NBYTE(MTABLE,LPTR+1)
C      COMPARE CHARACTER STRING
       IF (.NOT. CBS(MTABLE,LPTR+3,MTEXT,LTEXT,
     *                            LLENG)) GOTO 5
C      EVALUATE RIGHT CONTEXT CODE
       LRXX = LMXX(NBYTE(LMTYPE,NBYTE(MTEXT,
     *                            LTEXT+LLENG))+1)
C      CHECK LEFT AND RIGHT CONTEXT
       LRXXT = NBYTE(MTABLE,LPTR+2)
       LLXXT = NSPLIT(LRXXT,K4)
       IF (IAND(LLXX,LLXXT) .EQ. 0 .OR.
     *       IAND(LRXX,LRXXT) .EQ. 0) GOTO 5
C      OUTPUT SINGLE OR DOUBLE CELL CONTRACTION
       CALL OPWBRL(LPTR+LLENG+3)
       LTEXT = LTEXT + LLENG
       LCHAR = NBYTE(MTEXT,LTEXT-1)
       GOTO 72
C      QUOTATION MARK
    58 ISQUOT = 1 - ISQUOT
       IF (ISQUOT .NE. 0) GOTO 7
C      CLOSE QUOTATION MARK
       CALL OPBRL(22)
       GOTO 71
C      OUTPUT NUMBER SIGN BEFORE FIRST DIGIT
     6 IF (LLXX .LT. 8) CALL OPBRL(20)
C      OUTPUT SINGLE CHARACTER BRAILLE CELL
     7 CALL OPBRL(LCHAR)
    71 LTEXT = LTEXT + 1
C      SET NEW LEFT CONTEXT
    72 LLXX = LMXX(NBYTE(LMTYPE,LCHAR)+1)
       GOTO 3
     8 IF (ISCONT .EQ. 0 .AND. ISHDR .EQ. 0)
      *                              CALL OPBRL(64)
       ISPCON = ISCONT
     9 RETURN
       END
C
C
       SUBROUTINE OPWBRL(N)
C      OUTPUT BRAILLE CONTRACTION *N*.
       COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
       COMMON /XH/ MTABLE(220)
       LPTR = N
     3 LCELL = NBYTE(MTABLE,LPTR)
       LCONT = NSPLIT(LCELL,K0)
       CALL BCELL(LCELL)
       IF (LCONT .EQ. 0) GOTO 9
       LPTR = LPTR + 1
       GOTO 3
     9 RETURN
       END
C
```

```
C
      BLOCK DATA
C     THIS SUBPROGRAM IS PROGRAM-GENERATED.
      COMMON /XH/ MTABLE(220)
      DIMENSION M1(76)
      EQUIVALENCE(M1,MTABLE(1))
      DIMENSION M2(76)
      EQUIVALENCE(M2,MTABLE(77))
      DIMENSION M3(68)
      EQUIVALENCE(M3,MTABLE(153))
      DATA M1/
     *8Z006200C6,8Z00CB00D0,8Z00D800DD,8Z00E200E7,
     *8Z00ED00F2,8Z00FA00FF,8Z01070111,8Z0116011E,
     *8Z0126012E,8Z0133013B,8Z0144014D,8Z0156015C,
     *8Z0163016A,8Z0175017A,8Z0182018A,8Z01900195,
     *8Z019A01A1,8Z01A601AB,8Z01B301BB,8Z01C401C9,
     *8Z01CE01D5,8Z01DC01E3,8Z01EB01F4,8Z01FA0202,
     *8Z020A0212,8Z0217021D,8Z0226022C,8Z02310236,
     *8Z023C0241,8Z02490251,8Z02590261,8Z0266026C,
     *8Z02710276,8Z027C0282,8Z028C0293,8Z029C02A3,
     *8Z02A802AD,8Z02B202B7,8Z02BE02C5,8Z02CD02D6,
     *8Z02DF02E4,8Z02EB02F1,8Z02FA0302,8Z030A0313,
     *8Z031C0324,8Z032A0331,8Z0338033F,8Z0345034D,
     *8Z0355035C,8Z0365023F,8ZC5C42B02,8Z3FC5D938,
     *8Z040FC5D5,8ZC3C5B011,8Z023FC5D5,8Z220209C5,
     *8ZC102023F,8ZC1D91C03,8Z3FC1D5C4,8Z2F0236C1,
     *8ZE2350536,8ZE2E3C9D3,8ZD30C023F,8ZE2E30C05,
     *8Z36E2C8C1,8ZD3D32906,8Z3FE2C8D6,8ZE4D3C4A9,
     *8Z190231E2,8ZC829040F,8ZE2C9D6D5,8ZA81D043F,
     *8ZE2C1C9C4,8Z8E190431,8ZE2D6D4C5,8Z900E0236/
      DATA M2/
     *8ZE2D60E04,8Z3FE2E4C3,8ZC88E2105,8Z31E3C8C5,
     *8ZD9C5902E,8Z053FE3C8,8ZC5C9D988,8Z2E0536E3,
     *8ZC8C5E2C5,8Z982E033F,8ZE3C8C52E,8Z0436E3C8,
     *8ZC1E31E04,8Z36E3C8C9,8ZE239073F,8ZE3C8D9D6,
     *8ZE4C7C890,8Z39023FE3,8ZC839040F,8ZE3C9D6D5,
     *8ZB01D043F,8ZE3C9D4C5,8Z901E030F,8ZC9D5C72C,
     *8Z0236C9D5,8Z140209C9,8ZD5140336,8ZC9E3E2AD,
     *8Z0E0236C9,8ZE32D023F,8ZD6C63704,8Z0FD6E4D5,
     *8ZC4A81904,8Z0FD6E4D5,8ZE3A81E05,8Z3FD6E4C7,
     *8ZC8E39033,8Z023FD6E4,8Z33023FD6,8ZE62A030F,
     *8ZD6D5C7B0,8Z1B033FD6,8ZD5C59015,8Z0436D3C9,
     *8ZD2C50704,8Z3FD3D6D9,8ZC4900705,8Z3FD9C9C7,
     *8ZC8E39017,8Z0336D5D6,8ZE31D043F,8ZD5C1D4C5,
     *8Z901D040F,8ZD5C5E2E2,8ZB00E0536,8ZC3C8C9D3,
     *8ZC421023F,8ZC3C82103,8Z31C3D6D5,8Z12053FC3,
     *8ZD6E4D3C4,8Z89190336,8ZC3C1D509,8Z0209C3C3,
     *8Z120236C4,8ZD6190331,8ZC4C9E232,8Z0209C4C4,
     *8Z32043FD7,8ZC1D9E390,8Z0F040FD4,8ZC5D5E3B0,
     *8Z1E043FD4,8ZE4C3C88D,8Z21040ED4,8ZE4E2E38D/
      DATA M3/
     *8Z0C0236E4,8ZE2250336,8ZC2E4E303,8Z0212C2C5,
     *8Z060209C2,8ZC206030F,8ZC2D3C53C,8Z033FC6D6,
     *8ZD93F063F,8ZC6D9C9C5,8ZD5C48B17,8Z0436C6D9,
     *8ZD6D40B05,8Z36C6C9D9,8ZE2E38B0C,8Z030FC6E4,
     *8ZD3B00702,8Z09C6C616,8Z023FC7C8,8Z230236C7,
     *8ZD61B0209,8ZC7C73604,8Z3FE6C9E3,8ZC83E0436,
     *8ZE6C9D3D3,8Z3A0536E6,8ZC8C9C3C8,8Z31053FE6,
     *8ZC8C5D9C5,8Z9031053F,8ZE6C8D6E2,8ZC5983102,
     *8Z3FE6C831,8Z0412E6C5,8ZD9C53603,8Z12E6C1E2,
     *8Z34053FE6,8ZD6E4D3C4,8ZBA19043F,8ZE6D6D9D2,
     *8Z903A043F,8ZE6D6D9C4,8Z983A053F,8ZE6D6D9D3,
     *8ZC4B83A05,8Z3FE8D6E4,8ZD5C7903D,8Z043FE8D6,
     *8ZE4D9BD17,8Z0336E8D6,8ZE43D0336,8ZC8C1C488,
     *8Z130436C8,8ZC1E5C513,8Z0336C8C9,8ZD4930D03,
     *8Z12C8C9E2,8Z26043FC8,8ZC5D9C590,8Z13043FD2,
     *8ZD5D6E690,8Z050436D1,8ZE4E2E31A,8Z053FD8E4,
     *8ZC9C3D29F,8Z050536D8,8ZE4C9E3C5,8Z1F000000/
      END
!ASS (M:B0,D1,JHR0M4231)
!FORTRAN B0,NS
C
C     ******************************************
C     *                                        *
C     *    SEGMENT 423 - TRANSLATION OF MUSIC  *
C     *                                        *
C     ******************************************
C
```

```
      SUBROUTINE MTRAN2
C     TRANSLATE MUSIC FOR CURRENT MEASURE.
      COMMON MC1(253),NOLENG,MC2(64),JIBAR
      COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /KC/ KWHYPH,KTOUT,KSPLIT,KBREAK,KRBAR
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /N0/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /TK/ NRLENG,MRBAR(4),MTKEY(4),
     * ISGR1,ISBEAT,ISPVBT,ISCMPD,ISPAUS,JXPOSE
      COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISQUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
      JIBAR = 0
      JTIME = 0
      NNOTES = 0
      NOLENG = 0
      LSTART = NBYTE(MBEAT,K1)
    3 IF (JIBAR .LT. LSTART) GOTO 5
C     START OF NEW BEAT
      ISBEAT = 2
      IF (JTIME/K3*3 .EQ. JTIME) ISBEAT = 3
      NBEATS = NBEATS + 1
      LSTART = NBYTE(MBEAT,NBEATS+1)
      IF (NBYTE(MBRPT,NBEATS) .EQ. 0) GOTO 4
C     BRAILLE REPEAT SIGN
      CALL OPBRL(77)
      JTIME = JTIME + JBDUR
      JIBAR = LSTART
      GOTO 3
C     MARK POTENTIAL BREAK POINT IF REQUIRED
    4 IF (NOLENG.GE.12 .AND. JTIME.EQ.JTIME/JBDUR*
     *JBDUR.AND.JTIME.NE.JTMAX) CALL BCELL(KSPLIT)
      GOTO 6
C     TEST FOR START OF PART-BEAT
    5 IF (JTIME.EQ.NBEATS*JBDUR-JBDUR/K2) ISBEAT=1
    6 CALL IDCODE
      IF (ISEND .NE. 0) GOTO 8
      CALL ITRANS
      GOTO 3
C     CHECK FOR GROUPING OF LAST NON-REPEAT BEAT
    8 ISBEAT = 3
      JLENTH = 0
      CALL NGPING
      RETURN
      END
C
C
      SUBROUTINE ITRANS
C     TRANSLATE ITEM TO BRAILLE.
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /N0/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISQUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
C     INDENT UNLESS TEXT-TYPE ITEMS OR PEDALLING
      IF (ITEM .LT. 32 .OR. ITEM .GE. 128)
     *              INDENT = MAXO(INDENT,-1)
      GOTO (2,3,4,5),ITTYPE
      GOTO 9
    2 CALL CTRANS
      GOTO 9
    3 CALL STRANS
      GOTO 9
    4 CALL TTRANS
      GOTO 9
    5 CALL NTRANS
    9 RETURN
      END
C
```

```
C
      SUBROUTINE CTRANS
C     TRANSLATE CONTROL ITEM TO BRAILLE.
      COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /KC/ KWHYPH,KTOUT,KSPLIT,KBREAK,KRBAR
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISQUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
      COMMON /TT/ JTTEMP,JXTEMP
      GOTO (9,3,4,9,7,7,7,6,7,7,9,7,7,7,7,7,65,7,
     * 9,9,9,9),ITEM
      GOTO 9
C     IN-ACCORD
    3 IF (JTIME.GE.JTMAX .AND. JTTEMP.EQ.0) GOTO 7
C     PART-MEASURE IN-ACCORD
      CALL OPBRL(335)
      JPPCH = 0
      GOTO 9
C     SET-RETURN = BRAILLE MEASURE DIVISION
    4 JTTEMP = JTIME
      IF (JTIME .GT. 0) GOTO 7
      GOTO 9
C     DOUBLE BARLINE PRECEDED AND FOLLOWED BY DOTS
    6 CALL OPBRL(338)
      CALL BCELL(KBREAK)
      GOTO 7
C     FIRST ENDING
   65 ISFEND = 1
C     OUTPUT BRAILLE SIGN
    7 CALL OPBRL(323+3*ITEM)
C     FORCE NEW LINE AFTER DOUBLE BARLINE UNLESS
C     MEASURE IS FIRST-ENDING
      IF ((ITEM.EQ.5 .OR. ITEM.EQ.6 .OR. ITEM.EQ.9
     * ) .AND. ISFEND .EQ. 0) CALL BCELL(KBREAK)
C     POSSIBLE DOT AFTER HAND SIGNS
      IF (ITEM.EQ.15 .OR. ITEM.EQ.16) INCELL=KBDOT
    8 ISOCV = 1
    9 RETURN
      END
C
C
      SUBROUTINE TTRANS
C     TRANSLATE TEXT ITEMS TO BRAILLE.
      COMMON MC1(253),NOLENG,MOBAR(64)
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /KC/ KWHYPH,KTOUT,KSPLIT,KBREAK,KRBAR
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISQUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      COMMON /TY/ ISCONT,ISBLOW
      DIMENSION LWLENG(3),LWTEXT(3)
      DATA LECOMA/107/
      DATA LWLENG(1)/4/,LWLENG(2)/3/,LWLENG(3)/4/
      DATA LWTEXT/4HCRES,3HDIM,4HDECR/
      EQUIVALENCE(KBDRAK,KBRPT)
C
C     IGNORE EMPTY TEXT ITEMS AND SUNG WORDS
      IF (NCHARS .LE. 0 .OR. ISWORD .NE. 0) GOTO 9
      INCELL = 0
      LNXCH = NBYTE(MTEXT,K1)
C     REHEARSAL NUMBER IS PARENTHETICAL EXPRESSION
      IF (LNXCH .EQ. 31 GOTO 23
      IF (LNXCH .EQ. LECOMA) GOTO 6
      LWORD = 0
      IF (NCHARS .LE. 2) GOTO 3
```

```
C     TEST FOR SPECIAL WORDS
      DO 220 L=1,3
  220 IF (ISCBS(MINO(NCHARS,LWLENG(L)),LWTEXT(L))
     *              .NE. 0) GOTO 26
C     PARENTHETICAL EXPRESSION
C     INSERT BRAILLE MUSIC HYPHEN IF REQUIRED
   23 IF (JTIME .GT. 0 .AND. NBYTE(MOBAR,NOLENG)
     *         .NE. KBSPAC) CALL OPBRL(423)
C     OPEN PARENTHESIS SIGN
      CALL OPBRL(77)
      LWORD = 1
      GOTO 32
C     WORD WITH SPECIAL BRAILLE ABBREVIATION
   26 LWORD = -1
      NCHARS = L + 1
C     WORD SIGN FOR ABBREVIATION OR SPECIAL WORD
    3 CALL OPBRL(17)
   32 LPCH = 0
      IF (NCHARS .GT. 1) LPCH = 64
      CALL STBYTE(64,MTEXT,NCHARS+1)
      DO 390 L=1,NCHARS
      LCH = LNXCH
      LNXCH = NBYTE(MTEXT,L+1)
C     IGNORE SPECIAL CONTROL CHARACTERS
      IF (LCH .LE. 3) GOTO 38
      IF (LCH .LT. 240 .OR. LPCH .GE. 240) GOTO 35
C     BRAILLE NUMBER SIGN
      CALL OPBRL(20)
   35 IF (LCH .EQ. 75) GOTO 37
      CALL OPBRL(LCH)
      GOTO 38
C     BRAILLE DOT 3
   37 CALL OPBRL(31)
   38 LPCH = LCH
  390 CONTINUE
C     *LWORD* = -VE SPECIAL WORD, 0 ABBREVIATION,
C     +VE OTHER WORDS
      IF (LWORD) 4,7,5
C     BRAILLE DOT 3 AFTER SPECIAL WORD
    4 CALL OPBRL(31)
      GOTO 8
C     BRAILLE CLOSE PARENTHESIS SIGN
    5 CALL OPBRL(77)
      IF (NCHARS .LE. 12) GOTO 56
      IF (JTIME .GT. 0) GOTO 54
      CALL BCELL(KTOUT)
      GOTO 8
   54 CALL BCELL(KSPLIT)
C     BRAILLE SPACE
   56 CALL OPBRL(64)
      GOTO 8
C     BREATH MARK
    6 CALL OPBRL(425)
      GOTO 9
C     POSSIBLE DOT AFTER ABBREVIATION OTHER THAN
C     (DE)CRESCENDO TERMINATION SIGN
    7 IF(LCH .GT. 2) INCELL = KBDOT
C     FORCE OCTAVE SIGN AFTER TEXT ITEM
    8 JPPCH = 0
    9 RETURN
      END
C
C
      SUBROUTINE STRANS
C     TRANSLATE SEPARATE SIGN ITEMS TO BRAILLE.
      COMMON MC1(253),NOLENG,MOBAR(64)
      COMMON /DB/ ISDUB,NDBL,ISSIG,NSLURS,JSLUR(2)
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KY/ JKEYNO,JKEYSG,MKPCH(2),MTKPCH(2)
      COMMON /MM/ IXST,JMEM
      COMMON /NG/ JNGRP,MJNOTE(4)
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NCP,NDL
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREFT,JPN
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /TK/ NRLENG,MRBAR(4),MTKEY(4),
```

```
     * ISGR1,ISBEAT,ISPVBT,ISCMPD,ISPAUS,JXPOSE
      COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISQUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION MITEM2(6),JTIMES(2)
      EQUIVALENCE(MITEM2,JCLEF),(JTIMES,JTIME1)
      DATA LELETC/195/
C
      LENTRY = ITEM - 32
      IF (LENTRY .GT. 16) GOTO 6
      IF (LENTRY .GT. 3) GOTO 5
C     IGNORE CLEF SIGNS
      IF (LENTRY .EQ. 1) GOTO 9
      IF (NOLENG .LE. 0) GOTO 3
      IF (NBYTE(MOBAR,NOLENG) .EQ. KBSPAC) GOTO 2
C     SPACE BEFORE TIME OR KEY
      CALL OPBRL(64)
      GOTO 3
C     BUT IF PREVIOUS ITEM WAS CLEF, TIME OR KEY
C     THEN REMOVE EXISTING SPACE INSTEAD
    2 NOLENG = NOLENG - 1
C     CANCEL NOTE-GROUPING
    3 JNGRP = 0
      IF (LENTRY-2) 32,34,4
C
C     CLEF SIGN
   32 GOTO 9
C
C     TIME SIGNATURE
   34 IF (JTYME-192) 35,38,37
C     BRAILLE NUMBER SIGN
   35 CALL OPBRL(20)
      DO 360 L=1,2
  360 CALL BNUMB(JTIMES(L),L)
      GOTO 82
   37 LSIGN = 412
      GOTO 8
   38 IF (JTYME .EQ. 255) GOTO 82
      LSIGN = 414
      GOTO 8
C
C     KEY SIGNATURE
    4 IF (JKEYNO .EQ. 0) GOTO 82
      LSIGN = 112 + 2 * JKEYSG
      IF (JKEYNO .GT. 3) GOTO 42
      DO 410 L=1,JKEYNO
  410 CALL OPBRL(LSIGN)
      GOTO 82
   42 CALL OPBRL(20)
      CALL OPBRL(192+JKEYNO)
      GOTO 8
C
C     DOUBLE-BYTE ITEM OTHER THAN CLEF TIME OR KEY
    5 L1 = LENTRY - 3
      GOTO (52,54,54,9,9),L1
      GOTO 9
C     IRREGULAR NOTE GROUPING
   52 IF (NGROUP .NE. 3) GOTO 53
C     TRIPLET
      LSIGN = 30
      GOTO 8
C     BRAILLE DOTS 456 SIGN
   53 CALL OPBRL(61)
      CALL BNUMB(NGROUP,K2)
C     BRAILLE DOT 3
      LSIGN = 31
      GOTO 8
C     INKPRINT PAGE OR LINE NUMBER
   54 CALL BCELL(64*(LENTRY-3)+
     *           MINO(MITEM2(LENTRY),63))
      GOTO 9
C     SINGLE-BYTE ITEM
    6 LENTRY = LENTRY - 16
      GOTO (62,64,9,9,9,66,66,68,7,72),LENTRY
      GOTO 9
C     START OF SLUR
   62 IF (NSLURS .GE. 2) GOTO 9
      NSLURS = NSLURS + 1
```

```
      JSLUR(NSLURS) = 2
      GOTO 9
C     END OF SLUR
   64 IF (NSLURS .LE. 0) GOTO 9
C     OUTPUT CLOSE BRACKET SLUR IF REQUIRED
      IF (JSLUR(NSLURS) .EQ. 1) CALL OPBRL(392)
      JSLUR(NSLURS) = 0
      NSLURS = NSLURS - 1
      GOTO 9
C     START OF CRESCENDO OR DECRESCENDO
   66 L1 = LELETC + LENTRY - 6
      GOTO 69
C     END OF CRESCENDO OR DECRESCENDO
   68 LXST = IXST
      CALL LKAHED(-1)
      IF (NDBL .NE. 0) GOTO 9
      L1 = 87 + NSPLIT(LXST,K0)
C     OUTPUT SIGN FOR START/END OF (DE)CRESCENDO
   69 NCHARS = 1
      CALL STBYTE(L1,MTEXT,K1)
      CALL TTRANS
      GOTO 9
C     PEDALLING SIGN
    7 CALL OPBRL(405)
      GOTO 9
C     END OF PEDALLING SIGN
   72 CALL OPBRL(407)
      GOTO 9
    8 CALL OPBRL(LSIGN)
   82 IF (LENTRY .GT. 3) GOTO 9
C     SPACE AFTER TIME OR KEY SIGNATURE
      CALL OPBRL(64)
      JPPCH = 0
    9 RETURN
      END
C
C
      SUBROUTINE NTRANS
C     TRANSLATE NOTE ITEM.
      COMMON /DB/ ISDUB,NDBL,ISSIG,NSLURS,JSLUR(2)
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KY/ JKEYNO,JKEYSG,MKPCH(2),MTKPCH(2)
      COMMON /LK/ MBLINK(14),MFLINK(14),
     * MLIST1(14),MLIST2(14),MLIST3(14),MLIST4(14)
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISQUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
      DIMENSION LMKEY(2)
      IF (JPITCH .EQ. 0) GOTO 7
C     IDENTIFY DOUBLE-ABLE SIGNS FOR CURRENT ITEM
      CALL MMOVE(MTKPCH,LMKEY)
      IDOUB = -1
      CALL NTRANA
      CALL MMOVE(LMKEY,MTKPCH)
C     EVALUATE EXTENT OF DOUBLE-ABLE SIGNS
      IDOUB = 2
    3 IDOUB = NFLINK(IDOUB)
      IF (IDOUB .EQ. 2) GOTO 7
      IF (JVOIC .NE. NBYTE(MLIST2,IDOUB)) GOTO 3
      ISDUB = NBYTE(MLIST3,IDOUB)
      NDBL = NBYTE(MLIST4,IDOUB)
      IF (NDBL .EQ. 0)
     *      CALL LKAHED(NBYTE(MLIST1,IDOUB))
      NDBL = NDBL - 1
      CALL STBYTE(ISDUB,MLIST3,IDOUB)
      CALL STBYTE(NDBL,MLIST4,IDOUB)
      GOTO 3
C     TRANSLATE CURRENT ITEM
    7 IDOUB = 0
      CALL NTRANA
      RETURN
      END
C
C
```

```
      SUBROUTINE NTRANA
C     TRANSLATE NOTE ITEM.
      COMMON MC1(188),NILENG,MIBAR(64),NOLENG,
     * MC2(64),JIBAR
      COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /DB/ ISDUB,NDBL,ISSIG,NSLURS,JSLUR(2)
      COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSQZ,
     * ISREST,ISAHED,ISXPOS,ISEMB
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /JS/ JSUM,JMATCH,JNLAST,JLASTN
      COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KY/ JKEYNO,JKEYSG,MKPCH(2),MTKPCH(2)
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /TA/ MPITCH(4),MREPT(4),MRESTS(4),
     * MCXMEM(3),MPCENT(4),ISPCON,JBNO,JISNO,
     * JBSNO,ISCANC,ISLINR,JLINR
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /TK/ NRLENG,MRBAR(4),MTKEY(4),
     * ISGR1,ISBEAT,ISPVBT,ISCMPD,ISPAUS,JXPOSE
      COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISQUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
      COMMON /WS/ L1,L2,L3,JBYTE
      EQUIVALENCE(KLH,K2),(NU1,MEXTRA(3)),
     * (NU2,MEXTRA(4)),(NORNAM,MHEXT(2)),
     * (NAX1,MHEXT(3)),(NAX2,MHEXT(4))
C
      IF (ISSTEM .NE. 0) GOTO 8
      LPITCH = 7 * JOCTAV + JPITCH
      LPP = JPPCH
      IF (JPITCH .EQ. 0) GOTO 25
C     MUST TEST SLUR HERE IN CASE BRACKET SLUR
      DO 210 L=1,2
      IF (JSLUR(L) .GE. 2) CALL DOUBLE(L)
  210 CONTINUE
C     APPOGGIATURA OR ACCIACCATURA
      IF (ISSMAL .NE. 0) CALL DOUBLE(145-ISSMAL)
   25 IF (JPART .NE. KLH) GOTO 27
C     INTERVALS READ UPWARDS
      LINC = NHEDS + 1
      LCL = -1
      GOTO 3
C     INTERVALS READ DOWNWARDS
   27 LINC = 0
      LCL = 1
C
    3 DO 60 LHEAD=1,NHEDS
      IF (JPITCH .EQ. 0) GOTO 46
      CALL HDCODE(LHEAD*LCL+LINC)
C     ORNAMENTS
      IF (NORNAM .GT. 0 .AND. NORNAM .LE. 9)
     *            CALL OPBRL(376+3*NORNAM)
      IF (LHEAD .NE. 1) GOTO 42
C     EXPRESSION MARKS
      JBYTE = 32 * NU1 + NU2
C     DOUBLE AND SINGLE STAVE ARPEGGIOS
      CALL OPBRL(182+NBIT(-1))
C     STACCATO, ACCENTS ETC.
      LL = JBYTE
      LPTR = 0
  402 IF (LL .EQ. 0) GOTO 404
      L1 = NSPLIT(LL,LPTR)
      LPTR = LPTR + 1
      IF (L1 .NE. 0) CALL DOUBLE(1+2*LPTR)
      IF (LPTR .LE. 5) GOTO 402
C     SWELL
      CONTINUE
  404 ISPAUS = LL
C     TRANSLATE ACCIDENTALS
   42 IF (JAXIDS .EQ. 0) GOTO 43
      JBYTE = JAXIDS
      L1 = NBIT(K2)
      L1 = NBIT(K3)
      JBYTE = JBYTE / K2 - 1
```

```
C     NO NEED TO BRAILLE SUPERFLUOUS ACCIDENTALS
      IF (JBYTE.EQ.NBYTE(MTKPCH,JPITCH)) GOTO 425
C     NATURAL SIGN IF PRESENT
      IF (L1 .NE. 0 .AND. JBYTE .NE. 3)
     *                          CALL OPBRL(116)
      CALL OPBRL(110+2*JBYTE)
C     UPDATE CURRENT KEY SIGNATURE VALUE
      CALL STBYTE(JBYTE,MTKPCH,JPITCH)
      GOTO 43
  425 JAXIDS = 0
   43 IF (LHEAD .NE. 1) GOTO 52
C     MAIN NOTE
      LPICH = 7 * JOCTAV + JPITCH
C     INSERT OCTAVE SIGN IF REQUIRED
      LPOCT = (LPP-1) / K7
      LDIFF = IABS(LPICH - LPP)
      IF (LDIFF .GT. 4 .OR. (LDIFF .GT. 2 .AND.
     *      LPOCT .NE. JOCTAV)) GOTO 44
C     OCTAVE SIGN IF FIRST BAR OF BRAILLE LINE
      IF (ISOCV .EQ. 0) GOTO 45
      ISOPT = 1
      GOTO 441
C     OCTAVE SIGN
   44 ISOPT = 0
  441 CALL OPBRL(55+2*(JOCTAV-JXPOSE))
   45 LPP = LPICH
      GOTO 47
C
C     REST
   46 ISPAUS = NU2
      L1 = NSPLIT(K6)
      IF (ISTIE .NE. 0) CALL OPBRL(63)
C     TRANSLATE NOTE OR REST
   47 CALL OPBRL(256+9*JPITCH+JLENTH)
C     IF BREVE ADD DOTS 1,3 TO NOTE OR REST
C     TEST FOR REGULAR BRAILLE NOTE-GROUPING
      CALL NGPING
C     ADD DOTS 1,3 TO BREVE
      IF (JLENTH .EQ. 1) CALL OPBRL(29)
C     DOTS
      CALL OPBRL(161-JDOTS)
      GOTO 54
C
C     INTERVALS
   52 LLPICH = 7 * JOCTAV + JPITCH
      LINT = (LPICH - LLPICH) * LCL
      IF (IABS(LPP-LLPICH) .LE. 8) GOTO 53
C     OUTPUT OCTAVE SIGN AND REDUCE INTERVAL TO
C     LESS THAN ONE OCTAVE
      CALL OPBRL(55+2*JOCTAV)
      LINT = LINT - (LINT-2)/K7 * 7
C     NO DOUBLING IF MARKED ACCIDENTALS
   53 IF (JAXIDS .NE. 0) GOTO 534
      CALL DOUBLE(17+LINT)
      GOTO 538
  534 CALL OPBRL(17+LINT)
  538 LPP = LLPICH
   54 IF (MHEXT(1) .EQ. 0) GOTO 56
C     FINGERING
      JBYTE = MHEXT(1)
      L1 = NBIT(K1)
      L1 = NBIT(K2)
      L2 = (JBYTE-1) / K5
      L3 = JBYTE - 5 * L2
      IF (L3 .GT. 0) CALL OPBRL(24+L3)
      IF (L1 .NE. 0) CALL OPBRL(K1)
      IF (L2 .GT. 0) CALL OPBRL(24+L2)
C     PAUSE
   56 CALL OPBRL(364+2*ISPAUS)
C     SLURS AND END OF PARALLEL MOVT ON TOP HEAD
      IF (LHEAD*LCL+LINC .NE. 1) GOTO 58
      IF (IDOUB .NE. 0) GOTO 57
C     MARK START OR END OF PARALLEL MOVEMENT
      IF (ISLINR .EQ. 0) GOTO 566
      ISLINR = 0
      CALL OPBRL(429)
      GOTO 57
C     SINGLE OCTAVE INTERVAL SIGN AFTER LAST NOTE
  566 IF (ITPTR .EQ. JLASTN .AND. JLINR .EQ. 1)
     *                           CALL OPBRL(24)
```

```
C       SLUR
   57 DO 5790 L=1,2
      IF (JSLUR(L)-5) 579,572,574
C       SET FLAG TO OUTPUT CLOSE BRACKET SLUR
  572 JSLUR(L) = 1
      GOTO 579
  574 IF (JSLUR(L)-7) 576,577,578
C       DOUBLE SLUR SIGN
  576 JSLUR(L) = 4
      CALL OPBRL(K1)
      GOTO 578
C       FINAL SINGLE SLUR SIGN
  577 JSLUR(L) = 0
C       SINGLE SLUR SIGN
  578 CALL OPBRL(K1)
  579 CONTINUE
 5790 CONTINUE
C       TIE
   58 IF (ISTIE .EQ. 0 .OR. JPITCH .EQ. 0) GOTO 6
      IF (ISCTIE .NE. 0) GOTO 59
C       ORDINARY TIE
      CALL OPBRL(419)
      GOTO 6
C       CHORD TIE
   59 IF (LHEAD .EQ. NHEDS) CALL OPBRL(421)
    6 CONTINUE
   60 CONTINUE
C
      IF (IDOUB .LT. 0) GOTO 9
      JPLEN = JLENTH
      IF (JPITCH .EQ. 0) GOTO 9
      ISECV = 0
      JPPCH = LPICH
      GOTO 9
C
C       STEM SIGN
    8 CALL OPBRL(61)
      CALL OPBRL(96+JLENTH)
    9 RETURN
      END
C
C
      SUBROUTINE NGPING
C     BRAILLE REGULAR NOTE-GROUPING.
      COMMON MC1(253),NOLENG,MOBAR(64)
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /NG/ JNGRP,MJNOTE(4)
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /TK/ NRLENG,MRBAR(4),MTKEY(4),
     * ISGR1,ISBEAT,ISPVBT,ISCMPD,ISPAUS,JXPOSE
      COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISQUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
      DIMENSION LMESS(3)
      DATA LMESS/11HGROUP ERROR/
C     IGNORE NOTE GROUPING EXCEPT ON TRANSLATION
      IF (IDOUB .NE. 0) GOTO 9
      IF (ISBEAT .EQ. 0) GOTO 5
C     BEAT OR HALF-BEAT = QUAVER CANCELS GROUPING
      IF (JLENTH .EQ. 5) GOTO 6
C     MUST HAVE AT LEAST THREE NOTES TO GROUP
      IF (JNGRP .LT. 3) GOTO 4
      IF (ISPVBT .EQ. 3 .AND. ISBEAT .NE. 3 .AND.
     *              ISCMPD .EQ. 0) GOTO 5
C     PERFORM GROUPING
      DO 30 L=2,JNGRP
      LPOS = NBYTE(MJNOTE,L)
      CALL STBYTE(IAND(NBYTE(MOBAR,LPOS),27),
     *                          MOBAR,LPOS)
   30 CONTINUE
      GOTO 45
    4 IF (ISBEAT .LT. ISPVBT) GOTO 5
   45 IF (JLENTH .LE. 5) GOTO 6
C     START NEW GROUPING
      JNGRP = 1
      ISPVBT = ISBEAT
      GOTO 8
```

```
C       SLUR
   57   DO 5790 L=1,2
        IF (JSLUR(L)-5) 579,572,574
C       SET FLAG TO OUTPUT CLOSE BRACKET SLUR
  572   JSLUR(L) = 1
        GOTO 579
  574   IF (JSLUR(L)-7) 576,577,578
C       DOUBLE SLUR SIGN
  576   JSLUR(L) = 4
        CALL OPBRL(K1)
        GOTO 578
C       FINAL SINGLE SLUR SIGN
  577   JSLUR(L) = 0
C       SINGLE SLUR SIGN
  578   CALL OPBRL(K1)
  579   CONTINUE
 5790   CONTINUE
C       TIE
   58   IF (ISTIE .EQ. 0 .OR. JPITCH .EQ. 0) GOTO 6
        IF (ISCTIE .NE. 0) GOTO 59
C       ORDINARY TIE
        CALL OPBRL(419)
        GOTO 6
C       CHORD TIE
   59   IF (LHEAD .EQ. NHEDS) CALL OPBRL(421)
    6   CONTINUE
   60   CONTINUE
C
        IF (IDOUB .LT. 0) GOTO 9
        JPLEN = JLENTH
        IF (JPITCH .EQ. 0) GOTO 9
        ISOCV = 0
        JPPCH = LPICH
        GOTO 9
C
C       STEM SIGN
    8   CALL OPBRL(61)
        CALL OPBRL(96+JLENTH)
    9   RETURN
        END
```

```
C
C
      SUBROUTINE NGPING
C     BRAILLE REGULAR NOTE-GROUPING.
      COMMON MC1(253),NOLENG,MOBAR(64)
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /NG/ JNGRP,MJNOTE(4)
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /PV/ JPPCH,JPLEN,NNINCB,IXP,JREPT,JPN
      COMMON /TK/ NRLENG,MRBAR(4),MTKEY(4),
     * ISGR1,ISBEAT,ISPVBT,ISCMPD,ISPAUS,JXPOSE
      COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISQUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
      DIMENSION LMESS(3)
      DATA LMESS/11HGROUP ERROR/
C     IGNORE NOTE GROUPING EXCEPT ON TRANSLATION
      IF (IDOUB .NE. 0) GOTO 9
      IF (ISBEAT .EQ. 0) GOTO 5
C     BEAT OR HALF-BEAT = QUAVER CANCELS GROUPING
      IF (JLENTH .EQ. 5) GOTO 6
C     MUST HAVE AT LEAST THREE NOTES TO GROUP
      IF (JNGRP .LT. 3) GOTO 4
      IF (ISPVBT .EQ. 3 .AND. ISBEAT .NE. 3 .AND.
     *             ISCMPD .EQ. 0) GOTO 5
C     PERFORM GROUPING
      DO 30 L=2,JNGRP
      LPOS = NBYTE(MJNOTE,L)
      CALL STBYTE(IAND(NBYTE(MOBAR,LPOS),27),
     *                          MOBAR,LPOS)
   30 CONTINUE
      GOTO 45
    4 IF (ISBEAT .LT. ISPVBT) GOTO 5
   45 IF (JLENTH .LE. 5) GOTO 6
C     START NEW GROUPING
      JNGRP = 1
      ISPVBT = ISBEAT
      GOTO 8
```

```
C       NOT START OF BEAT
      5 IF (JNGRP .EQ. 0) GOTO 8
C       REST OR DOTTED NOTE OR DIFFERENT LENGTH
C       CANCELS GROUPING
        IF (JPITCH .EQ. 0 .OR. JDOTS .NE. 0 .OR.
     *       JLENTH .NE. JPLEN) GOTO 6
        JNGRP = JNGRP + 1
        IF (JNGRP .GT. 16) CALL TMESS(11,LMESS)
        CALL STBYTE(NOLENG,MJNOTE,JNGRP)
        GOTO 8
      6 JNGRP = 0
      8 IF (ISBEAT .GE. 2) ISCMPD = ISBEAT - 2
        ISBEAT = 0
      9 RETURN
        END
C
C
        SUBROUTINE DOUBLE(N)
C       DOUBLING FOR BRAILLE SIGN *N*.
C       *IDOUB*= -1 ENTER SIGN, 0 TRANSLATE,
C       1 LOOKING AHEAD.
        COMMON /DB/ ISDUB,NDBL,ISSIG,NSLURS,JSLUR(2)
        COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSQZ,
     * ISREST,ISAHED,ISXPOS,ISEMB
        COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
        COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
        COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
        COMMON /LK/ MBLINK(14),MFLINK(14),
     * MLIST1(14),MLIST2(14),MLIST3(14),MLIST4(14)
        COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
        COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
        COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISQUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
        DIMENSION LMESS(3)
        DATA LMESS/12HDOUBLE ERROR/
C
        LSDOUB = N
        IF (LSDOUB .GT. 2) GOTO 2
C       DOUBLING SIGN IS A SLUR
        LSLUR = JSLUR(LSDOUB)
        GOTO 205
C       DOUBLING SIGN IS NOT A SLUR
      2 LSDOUB = 0
C       SCAN DOUBLE TABLE FOR SPECIFIED SIGN
    205 LDOUB = 2
     21 LDOUB = MFLINK(LDOUB)
        IF (LDOUB .EQ. 2) GOTO 22
C       COMPARE FIRST TWO BYTES
        IF (N .NE. NBYTE(MLIST1,LDOUB) .OR.
     *  JVOIC .NE. NBYTE(MLIST2,LDOUB)) GOTO 21
        GOTO 23
     22 LDOUB = 0
C       *LDOUB* IS TABLE ENTRY INDEX IF FOUND,
C       OTHERWISE 0
     23 IF (IDOUB) 24,3,25
C
C       ENTER SIGN IN TABLE - EXIT IF ALREADY IN
     24 IF (LDOUB .GT. 0) GOTO 9
C       ENTER SIGN AT END OF DOUBLE TABLE
        CALL EINSRT(K2,N,JVOIC,KO,KO)
        GOTO 9
C
C       LOOKING AHEAD
     25 IF (IDOUB .EQ. LDOUB) ISSIG = 1
        GOTO 9
C
C       TRANSLATE SIGN (PUT IN TABLE BY *LKAHED*)
      3 IF (LDOUB .LE. 0) GOTO 64
        ISDUB = NBYTE(MLIST3,LDOUB)
        NDBL = NBYTE(MLIST4,LDOUB)
        IF (NDBL .EQ. 0) GOTO 5
        IF (ISDUB-1) 305,32,38
    305 IF (NDBL .LT. 3) GOTO 36
C       SET IS=DOUBLING FLAG
        CALL STBYTE(K1,MLIST3,LDOUB)
        IF (LSDOUB-1) 4,308,31
    308 IF (ISVOC .NE. 0) GOTO 31
C       OPEN BRACKET SLUR
```

```
      CALL OPBRL(386)
      LSLUR = 3
      GOTO 9
C     START OF DOUBLED SLUR
   31 LSLUR = 6
      GOTO 9
C
C     CHECK CONDITIONAL SIGNS AT START OF MEASURE
   32 IF (LSDOUB .NE. 0 .AND. LSLUR .EQ. 3) GOTO 9
      IF (NNOTES .GT. 1) GOTO 355
      IF (NDBL-2) 39,34,33
C     OUTPUT CONDITIONAL DOUBLE SIGN
   33 ISOPT = 1
      CALL OPBRL(N)
      GOTO 39
C     OUTPUT CONDITIONAL SINGLE SIGN WITH ANOTHER
C     TO FOLLOW
   34 ISDUB = 2
      GOTO 39
C
  355 IF (NDBL .GE. 3) GOTO 9
C     LESS THAN THREE SIGNS TO GO
   36 IF (ISDUB .EQ. 0) GOTO 6
      IF (LSDOUB.NE.C .AND. LSLUR.GE.4) LSLUR = 4
      GOTO 9
   38 ISDUB = 1
C     OUTPUT CONDITIONAL SIGN
   39 CALL STBYTE(ISDUB,MLIST3,LDOUB)
      ISOPT = 1
      GOTO 65
C
C     OUTPUT DOUBLE SIGN
    4 CALL OPBRL(N)
      GOTO 65
C
C     REMOVE SIGN FROM DOUBLE TABLE
    5 CALL EDEL(LDOUB)
      IF (LSDOUB .EQ. 0) GOTO 65
      IF (LSLUR .EQ. 3) GOTO 52
      LSLUR = 7
      GOTO 9
   52 LSLUR = 5
      GOTO 9
    6 IF (LSDOUB .EQ. 0) GOTO 65
C     SET FLAG FOR SINGLE SLUR AFTER CURRENT NOTE
      LSLUR = 8
      GOTO 9
C     OUTPUT =DOUBLE ERROR= MESSAGE
   64 CALL TMESS(12,LMESS)
      GOTO 9
C     OUTPUT SINGLE SIGN
   65 CALL OPBRL(N)
    9 IF (LSDOUB .GT. 0) JSLUR(LSDOUB) = LSLUR
      RETURN
      END
C
C
      SUBROUTINE LKAHED(N)
C     LOOK AHEAD OF CURRENT TRANSLATION POSITION
C     TO DETERMINE EXTENT OF DOUBLE-ABLE SIGNS AND
C     EXTENDED SIGNS.
C     IF *N* = -1, RETURNS *NDBL* = 0 TO MARK
C     TERMINATION SIGN, -1 TO OMIT SIGN.
C     IF *N* IS POSITIVE, RETURNS *NDBL* = NUMBER
C     OF CONSECUTIVE NON-REST NOTE ITEMS INCLUDING
C     CURRENT NOTE ITEM HAVING SIGN *N*.
      COMMON MC1(253),NOLENG,MC2(64),JIBAR
      COMMON /DB/ ISDUB,NDBL,ISSIG,NSLURS,JSLUR(2)
      COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSQZ,
     * ISREST,ISAHED,ISXPOS,ISEMB
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *    ISERR,ISNXST,ISINPT
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KY/ JKEYNO,JKEYSG,MKPCH(2),MTKPCH(2)
      COMMON /LK/ MBLINK(14),MFLINK(14),
     * MLIST1(14),MLIST2(14),MLIST3(14),MLIST4(14)
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
```

```
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
       COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
       COMMON /PV/ JPPCH,JPLEN,NNINCR,IXP,JREPT,JPN
       COMMON /ST/ KMSIZE,MCMEM(2),MSMEM(2),JECLEF
       COMMON /TA/ MPITCH(4),MREPT(4),MRESTS(4),
     * MCXMEM(3),MPCENT(4),ISPCON,JBNO,JISNO,
     * JESNO,ISCANC,ISLINR,JLINR
       COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
       COMMON /TK/ NRLENG,MRBAR(4),MTKEY(4),
     * ISGR1,ISBEAT,ISPVBT,ISCMPD,ISPAUS,JXPOSE
       COMMON /TR/ INCELL,NBEATS,MBEAT(8),MBRPT(8),
     * ISQUOT,ISFEND,NIPL,IDOUB,IND1,INDENT,INDTX,
     * ISOCV,ISKEYB,ISTERM,ISOPT,JADDR1
       COMMON /TT/ JTTEMP,JXTEMP
       COMMON /TX/ NCHARS,MTEXT(8),ISWORD
       DIMENSION LMEM(2),MKEY(4)
       EQUIVALENCE(MKEY,MKPCH)
       ISAHED = 1
       CALL MMOVE(MCMEM,LMEM)
C      SAVE VOLATILE VARIABLES
       LNOTES = NNOTES
       JXTEMP = JTIME
       LPLEN = JPLEN
       LPPCH = JPPCH
       LISOCV = ISOCV
       LSTEM = ISSTEM
       LSLURS = NSLURS
       LNSLUR = NSLURS
       LSLUR1 = JSLUR(1)
       LSLUR2 = JSLUR(2)
       LITPTR = ITPTR
       LOLENG = NOLENG
       LVOICE = JVOIC
       CALL MBS(MKEY,K1,MTKEY,K1,16)
       NDBL = 0
       IF (N .GE. 0) GOTO 4
       LITPTR = JIBAR
C
     3 ISSIG = 0
       CALL IDCODE
       IF (ISEND .EQ. 0) GOTO 35
C      GET NEXT MEASURE OF SAME STAVE
       ISCHEK = 0
       DO 320 L=1,NLINES
       IF (L .EQ. NLINES) ISCHEK = 1
       CALL GETBAR
   320 CONTINUE
       IF (ISEND .EQ. 0) GOTO 3
       IF (N .GE. 0) GOTO 7
       GOTO 6
    35 IF (JVOIC .NE. LVOICE) GOTO 3
       GOTO (3,36,38,4),ITTYPE
       GOTO 3
    36 IF (N .LE. 0 .OR. N .GE. 3) GOTO 37
       IF (ITEM .EQ. 49) LSLURS = LSLURS + 1
       IF (ITEM .NE. 50) GOTO 3
       LSLURS = LSLURS - 1
       IF (LSLURS .LT. NSLURS) GOTO 6
       GOTO 3
    37 IF (ITEM .NE. 54 .AND. ITEM .NE. 55) GOTO 3
C      TEXT ITEM
    38 IF (N .GE. 0 .OR. NCHARS .GT. 3) GOTO 3
       GOTO 6
C      NOTE ITEM
     4 IF (JPITCH .EQ. 0) GOTO 3
       IF (N .LT. 0) GOTO 7
       CALL NTRANA
       IF (ISSIG .EQ. 0) GOTO 7
C      SIGN PRESENT
       NDBL = NDBL + 1
       GOTO 3
C      DISCARD LAST NOTE IF SLUR
     6 NDBL = NDBL - 1
C      STORE COUNT
     7 NDBL = MIN0(NDBL,255)
C      RESTORE ORIGINAL MEASURE IF CHANGED
       IF (NIB.EQ.JBNO .AND. NIS.EQ.JISNO) GOTO 75
       ISAHED = 0
```

```
      CALL BFIND(JBNC,JISNO)
      ISAHED = 1
   75 JLINE = JISNO
      JPLEN = LPLEN
      JPPCH = LPPCH
      ISSTEM = LSTEM
      NSLURS = LNSLUR
      JSLUR(1) = LSLLR1
      JSLUR(2) = LSLLR2
      ISECV = LISECV
      CALL MMOVE(LMEM,MCMEM)
      JIBAR = LITPTR
      NOLENG = LOLENG
C     RE-DECODE CURRENT NOTE ITEM TO SET SPECS
      IF (N .GE. 0) CALL IDCODE
      NNOTES = LNOTES
      JVOIC = LVOICE
      JTIME = JXTEMP
      CALL MBS(MTKEY,K1,MKEY,K1,16)
      ISAHED = 0
      RETURN
      END
!JOB JH,COMPILE MUSIC PROGRAM SEGMENT 43
!EXTRABGD 6800
!ASS (M:B0,D1,JHROM43)
!FORTRAN B0,NS
C     ***********************************************
C     *                                             *
C     *       SEGMENT 43 - BRAILLE FORMATTER        *
C     *                                             *
C     ***********************************************
C
C
      SUBROUTINE FORMAT
C     FORMAT UNFORMATTED BRAILLE.
      COMMON MC1(188),NILENG,MIBAR(64),NOLENG,
     * MOBAR(64),MC2(3),NFLAGS,MFLAGS(3)
      COMMON /AD/ JIADDR,JOADDR
      COMMON /B1/ IFIRST,JOPAGE
      COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *    ISERR,ISNXST,ISINPT
      COMMON /F0/ MEND(4),MSPLIT(3),ISSPLT,ISDOT,
     * ISHYPH,IPAGE,ILINE,JSTART,NSLENG,MSBAR(10)
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,K8IG
      COMMON /M0/ JIMODE,JOMODE
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION LMST(16)
      EQUIVALENCE(KRH,K1),(KLH,K2),
     *                          (NLUB,MLINES(2))
      DATA LBNMAX/64/
C
C     FIRST PASS - EVALUATE BAR LENGTHS WITH AND
C     WITHOUT OPTIONAL CELLS AND STORE IN EXTRA
C     FLAGS FIELDS OF FIRST STAVE OF UNFORMATTED
C     BRAILLE.  ALSO SET BIT FOR DOT AFTER HAND
C     SIGNS.
      ISCHEK = 0
      DO 180 L=1,NLINES
  180 MEND(L) = 0
      CALL GETHDR
      DO 30 L=1,NBARS
      ISDOT = 0
      CALL GETBAR
      IF (ISEND .NE. 0) GOTO 31
      CALL BLENTH(LIND1,LMAX1,LMAX2)
      LPOSN = JIADDR + 1
      IF (NLINES .LE. 1) GOTO 25
      DO 20 LL=2,NLINES
      CALL GETBAR
```

```
      IF (ISEND .NE. 0) GOTO 31
      CALL BLENTH(LINDX,LLENG1,LLENG2)
      LMAX1 = MAXO(LLENG1,LMAX1)
      LMAX2 = MAXO(LLENG2,LMAX2)
   20 CONTINUE
   25 CALL OPPOSN(LPOSN)
      CALL OPBYTE(128+64*ISDOT+LIND1)
      CALL OPBYTE(128+LMAX1)
      CALL OPBYTE(128+LMAX2)
   30 CONTINUE
C
C     PASS 2 = CREATE FORMATTED BRAILLE SECTION
   31 JOMODE = 3
      MLINES(3) = 1
      NOB = 1
      MHEAD(3) = JFREE
      CALL OPPOSN(KDATA+MHEAD(3))
      ILINE = 0
      IPAGE = 0
      NSLENG = 0
      LISEND = 0
      LISHS = 1
C     FORMAT HEADER LINES
      CALL GETHDR
      ISHDR = 1
      CALL INPOSN(KDATA+MHEAD(2))
C     PAGE HEADER LINE
      CALL GETBAR
      LHL = MFLAGS(1)
      CALL BMOVE(MIBAR,MOBAR)
      CALL STLINE(KO)
      IF (LHL .LE. 0) GOTO 34
      LL = 127
      DO 320 L=1,LHL
      CALL GETBAR
      CALL CENTRE(LL)
      LL = 0
  320 CONTINUE
C     GET BRAILLE PART CODES
   34 CALL GETBAR
      LISVOC = 0
      DO 350 L=1,NLOB
      JBYTE = NFETCH(KO)
      IF (JBYTE .EQ. 64) LISVOC = 1
      CALL STBYTE(JBYTE,MBPART,L)
  350 CONTINUE
      ISHDR = 0
      JBAR = 0
C
C     START OF BRAILLE PARALLEL
   36 IF (LISVOC .NE. 0) GOTO 364
C     OUTPUT BAR NUMBER
      CALL BNUMB(JBAR+IFIRST,K1)
C     START OF HAND SIGN OR FIRST MEASURE OF LINE
      JSTART = NOLENG + 2
      LMAX = NOLENG + 1
C     DOT 3 AFTER BAR NUMBER FOR CONTINUATION LINE
      IF (MEND(NIS) .GT. 0) CALL OPBRL(31)
      GOTO 366
C     WORD PART
  364 JSTART = 0
      LMAX = 2
  366 LSTART = JSTART
      NLINE = ILINE
C     TRANSFER PENDING PAGE NUMBER IF NO CURRENT
C     PAGE NUMBER WAITING
      IF (NPAGE .EQ. 0) NPAGE = IPAGE
      ILINE = 0
      IPAGE = 0
      LFIRST = JBAR + 1
      LBN = 0
C     GET NEXT MEASURE ON FIRST STAVE
   38 JBAR = JBAR + 1
      LBN = LBN + 1
      CALL BFIND(JBAR,K1)
      IF (ISEND .NE. 0) GOTO 42
      L1 = NSPLIT(MFLAGS(1),K1)
      IF (LBN .EQ. 1) ISDOT = L1
      LMAX1 = MFLAGS(MINO(LBN,K2)+1)
      ISSPLT = 0
```

```
C       IGNORE DUMMY BARS
        IF (LMAX1 .LE. 0) GOTO 415
C       SKIP EMPTY MEASURES
        IF (NILENG .LE. 0) GOTO 4
        JPART = NBYTE(MBPART,K1)
        IF (LBN .NE. 1) GOTO 4
        CALL FILL(JSTART)
        IF (JPART .NE. KRH) GOTO 4
C       RIGHT HAND SIGN
        LRHST = JSTART
        IF (LISHS .EQ. 0) GOTO 39
        CALL BHSIGN(KRH)
        GOTO 395
C       TWO BRAILLE SPACES
   39   CALL OPBRL(64)
        CALL OPBRL(64)
  395   LMAX = NOLENG
        JSTART = NOLENG + 1
    4   LMAX = LMAX + LMAX1 + 1
        IF (LMAX = MAXC(MEND(NIS),KO) .LE. KBLENG)
       *                              GOTO 41
C       BRAILLE LINE LENGTH EXCEEDED
        IF (LBN .GT. 1) GOTO 44
C       MUST SPLIT MEASURE IF THIS IS THE FIRST ONE
C       ON THIS BRAILLE LINE
        ISSPLT = 1
C       SET STARTING POSITION ON BRAILLE LINE,
C       ALLOWING FOR INDENTATION
   41   LSTART = JSTART + LMAX1 + 1
        IF (LISVOC .NE. 0) GOTO 412
C       INDENT AS REQUIRED UNLESS LINE IS SPLIT
        LST = JSTART + MFLAGS(1) * (1-ISSPLT)
C       ALIGN START OF MEASURE FOR NON-VOCAL MUSIC
        CALL FILL(LST)
C       COPY MEASURE TO BRAILLE LINE
  412   CALL IBOB(LBN)
  415   IF (NLINES .EQ. 1) GOTO 417
        IF (LBN .GT. LBNMAX) GO TO 419
C       SAVE STARTING POSITION OF THIS MEASURE
C       FOR ALIGNMENT OF THE NEXT LINE
        CALL STBYTE(JSTART,LMST,LBN)
        GOTO 418
C       ALIGNMENT FOR START OF NEXT MEASURE UNLESS
C       VOCAL MUSIC
  417   LSTART = NOLENG + 2
  418   IF (LISVOC .EQ. 0) JSTART = LSTART
  419   IF (ISSPLT .NE. 0) GOTO 446
        IF (MEND(1) .EQ. 0) GOTO 38
        GOTO 445
C
C       END OF SECTION
   42   LISEND = 1
        IF (LBN .LE. 1) GOTO 9
C
C       OUTPUT BRAILLE LINE FOR FIRST STAVE
   44   JBAR = JBAR - 1
        LBN = LBN - 1
  445   IF (ISSPLT .NE. 0) GOTO 446
        CALL STFBRL(NLUB-1)
        GOTO 448
  446   CALL BSPLIT
  448   IF (NLINES .LE. 1) GOTO 62
C
C       OUTPUT BRAILLE LINES FOR REMAINING STAVES
        DO 60 L=2,NLINES
        LISHS = 0
C       INDENT TWO SPACES FOR VOCAL MUSIC
C       (*FILL* PUTS THE SECOND ONE IN)
        IF (NBYTE(MBPART,L) .GT. 64) CALL OPBRL(64)
        DO 560 LL=1,LBN
        CALL BFIND(LFIRST+LL-1,L)
C       IGNORE DUMMY BARS
        IF (NILENG .EQ. 0) GOTO 56
C       ALIGN MEASURE, LEAVING AT LEAST ONE SPACE
        JSTART = MAXO(NOLENG+2,NBYTE(LMST,LL))
C       INDENTATION UNLESS FIRST STAVE WAS SPLIT
        IF (ISSPLT .EQ. 0 .AND. LISVOC .EQ. 0)
       *          JSTART = JSTART + MFLAGS(1)
        JPART = NBYTE(MBPART,L)
        IF (LISHS .NE. 0 .OR. JPART .GT. 3) GOTO 46
```

```
      CALL FILL(LRHST-1)
      IF (L .EQ. 2 .AND. NLINE .GT. 0) GOTO 45
      CALL BSPACE
      GOTO 455
C     PUT INKPRINT LINE NUMBER ON SECOND LINE
C     OF BRAILLE PARALLEL
   45 CALL BNUMB(NLINE,K1)
C     HAND SIGN
  455 CALL BHSIGN(JPART)
      LISHS = 1
   46 CALL FILL(JSTART)
      CALL IBOB(LL)
      IF (ISSPLT .NE. 0) GOTO 58
   56 CONTINUE
  560 CONTINUE

      CALL STFBRL(KO)
      GOTO 6
   58 CALL BFIND(LFIRST,L)
      CALL BSPLIT
    6 CONTINUE
   60 CONTINUE
C
   62 IF (LISEND .NE. 0) GOTO 9
      IF (MEND(1) .GT. 0) JBAR = JBAR = 1
      GOTO 36
    9 RETURN
      END
C
C
      SUBROUTINE CENTRE(N)
C     CENTRE *MIBAR* IN BRAILLE OUTPUT LINE AND
C     WRITE TO FILE WITH FLAGS FIELD *N*.
      COMMON MC1(188),NILENG,MC2(64),NOLENG
      COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /F0/ MEND(4),MSPLIT(3),ISSPLT,ISDOT,
     * ISHYPH,IPAGE,ILINE,JSTART,NSLENG,MSBAR(10)
      COMMON /K0/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      LSKIP = (KBLENG-NILENG)/K2
      IF (LSKIP .LE. 0) GOTO 6
      NOLENG = 0
      DO 40 L=1,LSKIP
   40 CALL BSPACE
    6 CALL IBOB(KO)
      CALL STLINE(N)
      RETURN
      END
C
C
      SUBROUTINE FILL(NEND)
C     FILL OUTPUT LINE WITH TRACKER DOTS OR SPACES
C     UP TO *NEND* - 1.
      COMMON MC1(253),NOLENG
      COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /K0/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      LTRACK = NEND - NOLENG - 3
      IF (LTRACK .LT. 5) GOTO 4
      CALL BSPACE
      DO 30 L=1,LTRACK
   30 CALL BSTORE(KBDOT)
    4 IF (NOLENG .GE. NEND-1) GOTO 9
      CALL BSPACE
      GOTO 4
    9 RETURN
      END
C
C
      SUBROUTINE IBOB(N)
C     ADD CONTENTS OF INPUT MEASURE TO OUTPUT LINE
C     DEALING WITH OPTIONAL CELLS AND FORMAT CODES
C     AND INKPRINT PAGE AND LINE NUMBER CODES.
C     *N* = POSITION OF MEASURE ON BRAILLE LINE.
      COMMON MC1(188),NILENG,MIBAR(64),NOLENG,
     * MOBAR(64),JIBAR,JOBAR
      COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /F0/ MEND(4),MSPLIT(3),ISSPLT,ISDOT,
     * ISHYPH,IPAGE,ILINE,JSTART,NSLENG,MSBAR(10)
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
```

```
      CALL FILL(LRHST-1)
      IF (L .EQ. 2 .AND. NLINE .GT. 0) GOTO 45
      CALL BSPACE
      GOTO 455
C     PUT INKPRINT LINE NUMBER ON SECOND LINE
C     OF BRAILLE PARALLEL
   45 CALL BNUMB(NLINE,K1)
C     HAND SIGN
  455 CALL BHSIGN(JPART)
      LISHS = 1
   46 CALL FILL(JSTART)
      CALL IBOB(LL)
      IF (ISSPLT .NE. 0) GOTO 58
   56 CONTINUE
  560 CONTINUE

      CALL STFBRL(KO)
      GOTO 6
   58 CALL BFIND(LFIRST,L)
      CALL BSPLIT
    6 CONTINUE
   60 CONTINUE
C
   62 IF (LISEND .NE. 0) GOTO 9
      IF (MEND(1) .GT. 0) JBAR = JBAR + 1
      GOTO 36
    9 RETURN
      END
C
C
      SUBROUTINE CENTRE(N)
C     CENTRE *MIBAR* IN BRAILLE OUTPUT LINE AND
C     WRITE TO FILE WITH FLAGS FIELD *N*.
      COMMON MC1(188),NILENG,MC2(64),NOLENG
      COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /FO/ MEND(4),MSPLIT(3),ISSPLT,ISDOT,
     * ISHYPH,IPAGE,ILINE,JSTART,NSLENG,MSBAR(10)
      COMMON /KB/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      LSKIP = (KBLENG-NILENG)/K2
      IF (LSKIP .LE. 0) GOTO 6
      NOLENG = 0
      DO 40 L=1,LSKIP
   40 CALL BSPACE
    6 CALL IBOB(KO)
      CALL STLINE(N)
      RETURN
      END
C
C
      SUBROUTINE FILL(NEND)
C     FILL OUTPUT LINE WITH TRACKER DOTS OR SPACES
C     UP TO *NEND* - 1.
      COMMON MC1(253),NOLENG
      COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      LTRACK = NEND - NOLENG - 3
      IF (LTRACK .LT. 5) GOTO 4
      CALL BSPACE
      DO 30 L=1,LTRACK
   30 CALL BSTORE(KBDOT)
    4 IF (NOLENG .GE. NEND-1) GOTO 9
      CALL BSPACE
      GOTO 4
    9 RETURN
      END
C
C
      SUBROUTINE IBOB(N)
C     ADD CONTENTS OF INPUT MEASURE TO OUTPUT LINE
C     DEALING WITH OPTIONAL CELLS AND FORMAT CODES
C     AND INKPRINT PAGE AND LINE NUMBER CODES.
C     *N* = POSITION OF MEASURE ON BRAILLE LINE.
      COMMON MC1(188),NILENG,MIBAR(64),NOLENG,
     * MOBAR(64),JIBAR,JOBAR
      COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /FO/ MEND(4),MSPLIT(3),ISSPLT,ISDOT,
     * ISHYPH,IPAGE,ILINE,JSTART,NSLENG,MSBAR(10)
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
```

```
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /N8/ JBAR,NIB,N0B,NIS,N0S,NIBS
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION IPL(2),NPL(2)
      EQUIVALENCE(IPL,IPAGE),(NPL,NPAGE)
      DO 20 L=1,3
   20 MSPLIT(L) = KBIG
      ISHYPH = 0
      J0BAR = N0LENG
      LL = MAX0(MEND(NIS),K0) + 1
      IF (LL .GT. NILENG) G0T0 8
      D0 60 L=LL,NILENG
      JBYTE = NBYTE(MIBAR,L)
      LFLAG = NBIT(K1) + 1
      IF (JBYTE .GT. 0) G0T0 (25,3,4,32),LFLAG
C     BRAILLE SPACE
      JBYTE = 64
C     NORMAL BRAILLE CELL
   25 CALL BSTORE(JBYTE)
      G0T0 6
C     CELL OMITTED UNLESS FIRST BAR OF LINE
    3 IF (N .EQ. 1) G0T0 25
      G0T0 6
   32 L1 = JBYTE - 58
      IF (L1 .LE. 0) G0T0 4
C     SPECIAL FORMATTER CONTROLS
      G0T0 (33,34,35,36,6),L1
C     SET FLAG TO ADD BRAILLE HYPHEN
   33 ISHYPH = 1
      G0T0 6
C     PUT TEXT ON SEPARATE LINE IF FIRST MEASURE
   34 IF (N .NE. 1) G0T0 6
      NSLENG = MINO(N0LENG-J0BAR,KBLENG-5)
      CALL MBS(M0BAR,J0BAR+1,MSBAR,K1,NSLENG)
      N0LENG = J0BAR
      G0T0 6
C     BREAK POINT
   35 LS = (N0LENG-5) / (KBLENG-5) + 1
      IF (LS .LE. 3) MSPLIT(LS) = N0LENG
      G0T0 6
C     FORCE NEW BRAILLE LINE
   36 MEND(NIS) = L
      IF (L .EQ. NILENG) MEND(NIS) = -1
      G0T0 9
C     INKPRINT PAGE OR LINE NUMBER
    4 IF (N .EQ. 1) G0T0 5
      IPL(LFLAG-2) = JBYTE
      G0T0 6
    5 NPL(LFLAG-2) = JBYTE
    6 CONTINUE
   60 CONTINUE
    8 MEND(NIS) = 0
    9 RETURN
      END
C
C
      SUBROUTINE STFBRL(N)
C     STORE FORMATTED BRAILLE LINE IN FILE WITH
C     FLAGS FIELD *N*, INSERTING INKPRINT PAGE
C     NUMBER LINE IF NECESSARY.
      COMMON MC1(189),MIBAR(64),N0LENG,M0BAR(64)
      COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /F0/ MEND(4),MSPLIT(3),ISSPLT,ISJ0T,
     * ISHYPH,IPAGE,ILINE,JSTART,NSLENG,MSBAR(10)
      COMMON /IU/ JCLEF,JTYME,JKEY,NGR0UP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /KB/ KBD0T,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
C     EXIT IF CURRENT LINE IS EMPTY
      IF (N0LENG .LE. 0) G0T0 9
      IF (NPAGE .EQ. 0 .AND. NSLENG .EQ. 0) G0T0 8
C     SAVE CURRENT BRAILLE LINE
      CALL BMOVE(M0BAR,MIBAR)
      IF (NPAGE .EQ. 0) G0T0 6
C     SET UP INKPRINT PAGE NUMBER LINE
      N0LENG = 0
C     TABULATE TO CENTRE OF LINE
      LSKIP = (KBLENG-(3+NDIGIT(NPAGE))) / K2
      D0 30 L=1,LSKIP
   30 CALL BSPACE
```

```
C       ADD INKPRINT PAGE NUMBER
        CALL OPBRL(434)
        CALL BNUMB(NPAGE,K1)
        NPAGE = 0
        IF (NSLENG .EQ. 0) GOTO 7
C       STORE INKPRINT PAGE LINE
        CALL STLINE(N+2)
C       INSERT SEPARATE TEXT LINE
    6   NOLENG = 0
        DO 620 L=1,5
  620   CALL BSPACE
        CALL MBS(MSBAR,K1,MOBAR,K6,NSLENG)
        NOLENG = NOLENG + NSLENG
        NSLENG = 0
C       STORE INKPRINT PAGE OR SEPARATE TEXT LINE
    7   CALL STLINE(N+1)
C       RESTORE CURRENT BRAILLE OUTPUT LINE
        CALL BMOVE(MIBAR,MOBAR)
C       ADD BRAILLE HYPHEN IF REQUIRED
    8   IF (ISHYPH .NE. 0) CALL OPBRL(24)
C       STORE CURRENT BRAILLE OUTPUT LINE
        CALL STLINE(N)
    9   RETURN
        END
C
C
        SUBROUTINE BSPACE
C       OUTPUT BRAILLE SPACE.
        COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
        CALL OPBRL(64)
        RETURN
        END
C
C
        SUBROUTINE BLENTH(NIND,NLENG1,NLENG2)
C       DETERMINE LENGTH OF CURRENT MEASURE WITH AND
C       WITHOUT FIRST-MEASURE-ONLY CELLS.  RETURNS
C       *NLENG1* = TOTAL NUMBER OF CELLS,
C       *NLENG2* = NUMBER OF NON-OPTIONAL CELLS.
C       BOTH THESE INCLUDE ANY INDENTATION.
C       *ISDOT* IS SET IF DOT NEEDED AFTER HAND SIGN
        COMMON MC1(188),NILENG,MIBAR(64),MC2(65),
       * JIBAR,MC3(3),MFLAGS(3)
        COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
       *    ISERR,ISNXST,ISINPT
        COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
        COMMON /F0/ MEND(4),MSPLIT(3),ISSPLT,ISDOT,
       * ISHYPH,IPAGE,ILINE,JSTART,NSLENG,MSBAR(10)
        COMMON /WS/ L1,L2,L3,JBYTE
        DATA KTOUT/252/
        NLENG1 = 0
        NLENG2 = 0
        NIND = MFLAGS(1)
        L1 = NSPLIT(NIND,K1)
        IF (NILENG .EQ. 0) GOTO 9
        LISDOT = -1
        DO 60 JIBAR=1,NILENG
        JBYTE = NBYTE(MIBAR,JIBAR)
        IF (JBYTE .GE. 128) GOTO 4
C       BRAILLE CELL
        NLENG1 = NLENG1 + 1
        IF (JBYTE .LE. 64) NLENG2 = NLENG2 + 1
        IF (LISDOT .GE. 0) GOTO 6
        L1 = NBIT(K4)
        LISDOT = 0
        IF (JBYTE .NE. 0) LISDOT = 1
        GOTO 6
C       FORMAT CONTROL
    4   IF (JBYTE .EQ. KTOUT) LISDOT = -1
    6   CONTINUE
   60   CONTINUE
        IF (LISDOT .GT. 0) ISDOT = 1
C       ADD INDENTATION IF ANYTHING IN MEASURE
        IF (NLENG1 .GT. 0) NLENG1 = NLENG1 + NIND
        IF (NLENG2 .GT. 0) NLENG2 = NLENG2 + NIND
    9   RETURN
        END
C
C
```

```
      SUBROUTINE BSPLIT
C     SPLIT MEASURE INTO TWO OR MORE LINES.
      COMMON MC1(188),NILENG,MIBAR(64),NOLENG,
     * MOBAR(64)
      COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /FO/ MEND(4),MSPLIT(3),ISSPLT,ISDOT,
     * ISHYPH,IPAGE,ILINE,JSTART,NSLENG,MSBAR(10)
      COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MT/ NTLENG,MTBAR(64)
      COMMON /NB/ JBAR,NIB,NOB,NIS,NOS,NIBS
      DIMENSION LMESS(4)
      DATA LMESS/16HOVERFLOW AT LINE/
C     NO SPLIT IF BRAILLE LINE LENGTH NOT EXCEEDED
      IF (NOLENG .LE. KBLENG) GOTO 7
C     SAVE WHOLE MEASURE IN *MTBAR*
      CALL BMOVE(MOBAR,MTBAR)
      LSKIP = 0
      LTBAR = 0
      LLINE = 0
      LSPLIT = 0
C     NEXT FORMATTED OUTPUT LINE
    2 LLINE = LLINE + 1
      LMAX = KBLENG - LSKIP - 1
      NOLENG = 0
   25 LSPLIT = LSPLIT + 1
      IF (LSPLIT .GT. 3) GOTO 3
      L = MSPLIT(LSPLIT)
      IF (L .EQ. KBIG) GOTO 3
      LENG = L - LTBAR
      IF (LENG .GT. LMAX) GOTO 4
      NOLENG = LENG
      GOTO 25
C     USE END OF MEASURE IF NO BREAK POINTS REMAIN
    3 LENG = NTLENG - LTBAR
      IF (LENG .LE. LMAX) NOLENG = LENG
    4 LSPLIT = LSPLIT + 1
      IF (NOLENG .GT. 0) GOTO 5
C     LINE OVERFLOW, FORCE DIVISION AT END OF LINE
      NOLENG = LMAX
      CALL TMESS(16,LMESS)
      CALL TWT1(NOB)
    5 LTBAR = LTBAR + NOLENG
      NOLENG = NOLENG + LSKIP
      IF (LTBAR .GE. NTLENG) GOTO 7
C     ADD BRAILLE MUSIC HYPHEN TO CONTINUED LINE
      CALL OPBRL(63)
      CALL STFBRL(K2)
      IF (LLINE .NE. 1) GOTO 65
C     SET INDENTATION FOR CONTINUATION LINES
      LSKIP = 5
      DO 60 L=1,LSKIP
   60 CALL STBYTE(KBSPAC,MOBAR,L)
C     MOVE REMAINING PART OF MEASURE TO OUTPUT BAR
   65 CALL MBS(MTBAR,LTBAR+1,MOBAR,LSKIP+1,
     *              NTLENG-LTBAR)
      GOTO 2
C     STORE FINAL PART OF MEASURE
    7 CALL STFBRL(KO)
      RETURN
      END
C
C
      SUBROUTINE BHSIGN(N)
C     OUTPUT HAND SIGN FOR PART *N* FOLLOWED BY
C     DOT 3 IF NECESSARY.
      COMMON MC1(318),JIBAR,MC2(3),MFLAGS(3)
      COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /FO/ MEND(4),MSPLIT(3),ISSPLT,ISDOT,
     * ISHYPH,IPAGE,ILINE,JSTART,NSLENG,MSBAR(10)
      COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /WS/ L1,L2,L3,JBYTE
      CALL OPBRL(435+2*N)
      IF (ISDOT .EQ. 0) GOTO 9
      IF (MFLAGS(1).NE.0 .AND. ISSPLT.EQ.0) GOTO 8
      JIBAR = 0
    2 JBYTE = NFETCH(KO)
```

```
      IF (ISEND .NE. 0) GOTO 7
      IF (JBYTE=64) 7,8,2
C     BRAILLE DOT 3
    7 CALL OPBRL(31)
      GOTO 9
C     BRAILLE SPACE
    8 CALL OPBRL(64)
    9 RETURN
      END
C
C
      FUNCTION NDIGIT(N)
C     RETURNS *NDIGIT* = NUMBER OF DECIMAL DIGITS
C     IN NON-NEGATIVE INTEGER *N*.
      NDIGIT = 1
      L = 10
    3 IF (N .LT. L) GOTO 9
      NDIGIT = NDIGIT + 1
      L = L + 1
      GOTO 3
    9 RETURN
      END
!JOB JH,COMPILE MUSIC PROGRAM SEGMENT 44
!EXTRABGD 6800
!ASS (M:B0,D1,JHR0M44)
!FORTRAN B0,NS
C     ******************************************
C     *                                        *
C     *      SEGMENT 44 - BRAILLE EMBOSSER     *
C     *                                        *
C     ******************************************
C
C
      SUBROUTINE EMBOSS
C     COPY BRAILLE TO BRAILLE FILE FOR EMBOSSING.
      COMMON MINBUF(90),MOPBUF(90),NINBUF,NOPBUF,
     * JINSEC,JOPSEC,JINBUF,JOPBUF,JINLK,JOPLK,
     * NILENG,MIBAR(64),NOLENG,MOBAR(64),JIBAR,
     * JOBAR,JOBUF,NFLAGS,MFLAGS(3)
      COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /B1/ IFIRST,JOPAGE
      COMMON /EK/ KBRDEV,KMLPP(2)
      COMMON /EM/ JOUT,JOLINE,NBHDR,MBHDR(10)
      COMMON /FF/ ISCTK,ICFLAG,ISVOC,ISSGZ,
     * ISREST,ISAHED,ISXPOS,ISEMB
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *    ISERR,ISNXST,ISINPT
      COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MO/ JIMODE,JOMODE
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NDP,NDL
      COMMON /PP/ MPP1(5),ISSAG,MPP2(1)
      EQUIVALENCE(KERROR,K4),(NLUB,MLINES(2))
      DATA KBRDEV/119/,KMLPP/24,24/
      NINBUF = -1
      NOPBUF = -1
      IF (ISEMB .NE. 0) GOTO 25
C     TRY TO READ OUTPUT FILE FIRST TIME ROUND TO
C     MAKE SURE IT IS AVAILABLE
      CALL BUFFERIN(KBRDEV,K1,MINBUF,KBFWDS+1,
     *                    LSTAT,LWORDS)
      IF (LSTAT .EQ. KERROR .OR. LWORDS .NE.
     *                         KBFWDS) GOTO 7
      REWIND KBRDEV
      ISEMB = 1
   25 JOUT = 0
      JOLINE = 0
      DO 30 L=1,KBFSIZ
   30 CALL STBYTE(64,MOPBUF,L)
      CALL GETHDR
      IF (JIMODE .NE. 3) GOTO 4
C     FORMATTED BRAILLE = OBTAIN AND CENTRE PAGE
C     HEADER LINE
      CALL GETBAR
      IF (NILENG .GT. 0) GOTO 33
      NBHDR = 0
      GOTO 5
```

```
   33 NBHDR = KBLENG
      DO 350 L=1,NBHDR
  350 CALL STBYTE(KBSPAC,MBHDR,L)
C     RESTRICT LENGTH OF RUNNING TITLE TO ALLOW
C     FOR BRAILLE PAGE NUMBER
   36 NILENG = MIN0(NILENG,KBLENG-4)
      CALL MBS(MIBAR,K1,MBHDR,
     *             (KBLENG-NILENG)/K2+1,NILENG)
      GOTO 5
    4 IF (JOLINE + MFLAGS(1) .GE. KMLPP(ISSAG+1))
     *                          CALL NEWPAG
      CALL BWRITE(NILENG,MIBAR)
    5 CALL GETBAR
      IF (ISEND .EQ. 0) GOTO 4
      CALL NEWPAG
      CALL BWRITE(NILENG,MIBAR)
      JOPAGE = JOPAGE + 1
      ISERR = 0
      GOTO 9
C     BRAILLE OUTPUT FILE NOT ACCESSIBLE
    7 ISERR = 9
    9 RETURN
      END
C
C
      SUBROUTINE BWRITE(NLENG,NBAR)                          C
C     CONVERT *NBAR* TO BRAILLE CODES FOR SPECIFIC           C
C     EMBOSSING DEVICE AND WRITE LINE TO FILE
      COMMON MINBUF(90),MOPBUF(90),MC1(8),NILENG,            C
     * MIBAR(64)
      COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /EM/ JOUT,JOLINE,NBHDR,MBHDR(10)
      COMMON /KB/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /PP/ MPP1(5),ISSAG,MPP2(1)
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION LMEM(32),LMLED(16),LMSAG(16)
      EQUIVALENCE(LMLED,LMEM(1)),(LMSAG,LMEM(17))
C     TABLE OF LED EMBOSSER CHARACTERS INDEXED
C     BY INTERNAL BRAILLE CODE
      DATA LMLED/
     * 16HA1B'K2L@CIF/MSP",16HE3H9O6R-DJG>NTG,,
     * 16H*5<-U8V.%4$+X!&;,16H:4IOZ7(-?W5#Y)= /
C     TABLE OF CHARACTERS FOR SAGEM EMBOSSER
C     INDEXED BY INTERNAL BRAILLE CODE
      DATA LMSAG/
     * 16HA,B'K;L.CIF*MSP-,16HE:H)O!R%DJG&NTG(,
     * 16H1?2=U<V#396-X|@+,16H5.8>Z=4$4W7OY5/ /
      IF (JOUT .LT. KBFSIZ) GOTO 2
      CALL BCLEAR
      JOUT = 0
C     CHOP LINE IF TOO LONG
    2 LLENG = MIN0(NLENG,KBLENG)
      IF (LLENG .LE. 0) GOTO 6
      DO 40 L=1,LLENG
      JBYTE = NBYTE(NBAR,L)
C     MASK TO BOTTOM 6 BITS
      IF (JBYTE .GT. 64) LL = NBIT(K1)
      CALL STBYTE(NBYTE(LMEM,64*ISSAG+JBYTE),
     *                              MOPBUF,JOUT+L)
   40 CONTINUE
    6 JOUT = JOUT + 40
      JOLINE = JOLINE + 1
      RETURN
      END
C
      SUBROUTINE NEWPAG                                      C
C     START NEW PAGE ON BRAILLE OUTPUT.
      COMMON MINBUF(90),MOPBUF(90),NINBUF,NOPBUF,
     * JINSEC,JOPSEC,JINBUF,JOPBUF,JINLK,JOPLK,
     * NILENG,MIBAR(64),NOLENG,MOBAR(64),JIBAR,
     * JOBAR,JBBUF,NFLAGS,MFLAGS(3)
      COMMON /BE/ KBLENG,MBPART(2),KBHSIG(3)
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /B1/ IFIRST,JOPAGE
      COMMON /EM/ JOUT,JOLINE,NBHDR,MBHDR(10)
      COMMON /KB/ KBDOT,KBHYPH,KBNUMB,KBRPT,KBSPAC
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /PP/ MPP1(5),ISSAG,MPP2(1)
```

```
      DIMENSION LPAGE(2)
      DATA LENAK/10/,LPAGE(1)/11/,LPAGE(2)/12/
      IF (JOUT .EQ. C) GOTO 25
      IF (JOUT .GE. KBFSIZ) GOTO 22
C     ENTER END-OF-BLOCK MARKER
      DO 20 L=5,8
   20 CALL STBYTE(KO,MOPBUF,JOUT+L)
   22 CALL BCLEAR
C     PAGE-FEED CONTROL LINE
   25 DO 30 L=1,40
   30 CALL STBYTE(LENAK,MOPBUF,L)
      CALL STBYTE(LPAGE(ISSAG+1),MOPBUF,K3)
      JOLINE = 0
      JOUT = 40
C     BRAILLE HEADER LINE INCLUDING PAGE NUMBER
      JOPAGE = JOPAGE + 1
C     NO HEADER LINE ON FIRST PAGE
      IF (JOPAGE .EQ. 1 .OR. NBHDR .EQ. 0) GOTO 9
      NOLENG = 0
      CALL OPBRL(20)
      CALL BNUMB(JOPAGE,K1)
C     TRANSFER BRAILLE PAGE NUMBER TO HEADER LINE
      CALL MBS(MOBAR,K1,MBHDR,KBLENG-NOLENG+1,
     *                        NOLENG)
C     OUTPUT HEADER LINE
      CALL BWRITE(NBHDR,MBHDR)
    9 RETURN
      END
C
C
      SUBROUTINE BCLEAR
C     OUTPUT AND CLEAR BRAILLE BUFFER.
      COMMON MINBUF(90),MOPBUF(90)
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /EK/ KBRDEV,KMLPP(2)
      DIMENSION LMESS(5)
      DATA LMESS/18HBRAILLE FILE ERROR/
      CALL BUFFEROUT(KBRDEV,1,MOPBUF,KBFWDS,LSTAT)
      IF (LSTAT .NE. 2) CALL TMESS(18,LMESS)
      DO 20 L=1,KBFSIZ
   20 CALL STBYTE(64,MOPBUF,L)
      RETURN
      END
!JOB JH,COMPILE MUSIC PROGRAM SEGMENT 5
!EXTRABGD 6800
!ASS (M:BO,D1,JHROM5)
!FORTRAN BO,NS,S
C
C
C     ********************************************
C     *                                          *
C     *     SEGMENT 5 - ARCHIVE TAPE HANDLING    *
C     *                                          *
C     ********************************************
C
C
      SUBROUTINE ARCHIV
C     ARCHIVE TAPE HANDLING ROUTINE.
C     *LPROG* = -1 FILES AND COPY, 0 SAVE,
C                  1 RESTORE, 2 ERASE.
      COMMON MC1(90),MOPBUF(90),NINBUF,NOPBUF,
     * JINSEC,JOPSEC,JINBUF,JOPBUF,JINLK,JOPLK
      COMMON /AR/ KATDEV,KISIZE,KIBASE,KSBASE,
     * KVBASE,KNBASE,KMTMES(3),MINDEX(540),NFNAME,
     * MFNAME(8)
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /BY/ NBYTES,MBYTES(3),NSECTS,NVERS
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /FG/ KLEVEL,IEA
      COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
     * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
      COMMON /FI/ ISWRIT,ISSCAN
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /FS/ JCSTAT,JNSTAT,MSTAT(3)
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /MC/ KBSIZE,KWSIZE,KXMAX,KYMAX,KFSIZE
```

```
      COMMON /MO/ JIMODE,JOMODE
      COMMON /QL/ IXG,IYG,IXQMIN,IQMAX,JXMAX,JYMAX
      COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
      COMMON /SE/ JCMD,KSEGS(7)
      COMMON /TE/ IXE,IYE,IXM,IXN,IYN,IYS,IXT,IYT
      COMMON /TL/ NNAME,MNAME(8)
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION LTITLE(2),LMVOW(2),LVERS(3),
     * LIDMES(3),LNMMES(2),LSUMES(8),LNSMES(6),
     * LNEMES(6),LIDMS(2),LREMES(7),LSAMES(7)
      EQUIVALENCE(NFOT,MINDEX(4)),
     * (NDELS,MINDEX(5)),(NUSED,MINDEX(6))
      LOGICAL CBS
      DATA LTITLE/6HNAME: /,LMVOW/5HAEIOU/,
     *LVERS/12H   VERSION? /,LIDMS/5HIDENT/,
     *LIDMES/12HIDENTIFIER? /,LNMMES/6HNAME? /,
     *LSUMES/32HSUMMARY OF FILES ON ARCHIVE TAPE/,
     *LNSMES/24HNUMBER OF FILES STORED =/,
     *LNEMES/24HNUMBER OF FILES ERASED =/,
     *LREMES/25HCOPYING DISC FILE TO TAPE/,
     *LSAMES/25HCOPYING TAPE FILE TO DISC/
      DATA KATDEV/121/,KISIZE/2160/,LFMAX/200/
      DATA KIBASE/40/,KSBASE/260/,KVBASE/760/,
     * KNBASE/880/,KMTMES/12HTAPE NAME = /
C
      IF (JCMD .NE. 26) GOTO 2
C     PROGRAM INITIALISATION
      CALL LSEG(51)
      CALL RSETUP
      GOTO 9
C
    2 IF (KLEVEL .EQ. 0) GOTO 205
C     CHECK TAPE AVAILABLE AND SET INDICATOR
      L1 = 1
      CALL TAPE(L1)
      IF (L1 .NE. 0) GOTO 71
  205 LPROG = JCMD - 10
      IF (LPROG .GE. 3) LPROG = -1
      CALL BREAK(&79)
      CALL REWIND(KATDEV)
C     READ INDEX FROM TAPE
      LISREW = 1
      CALL AREAD(KATDEV,MINDEX,KISIZE)
      NSECTS = 0
      IF (LPROG .GE. 0) GOTO 21
C     REWIND TAPE IF ONLY LISTING INDEX
      CALL REWIND(KATDEV)
      LISREW = 0
   21 CALL MBS(MINDEX,9,MTEXT,K1,K4)
      IF (ISCBS(K4,KTHDR) .EQ. 0) GOTO 72
      IF (JCMD .EQ. 23) GOTO 215
      LNFILS = NFOT - NDELS
C     CLEAR SCREEN IF LISTING FILES
      IF (LPROG .LT. 0) CALL TPAGE(KO)
C     WRITE TAPE NAME ON TERMINAL
  215 CALL TWT(20,KMTMES)
      IF (JCMD .EQ. 23) GOTO 69
      LVNO = 0
      LIDENT = 0
      LNAME = 0
      CALL LSEG(51)
      IF (LPROG) 3,22,24
C     SAVE
   22 IF (NNAME .LE. 0) GOTO 78
C     REMOVE ANY FANCY BITS FROM FILE NAME
      NCHARS = 0
      DO 230 L=1,NNAME
      JBYTE = NBYTE(MNAME,L)
C     IGNORE SPECIAL CONTROL CODES AND QUERY
      IF (JBYTE.LT.64 .OR. JBYTE.EQ.111) GOTO 23
C     CHECK FOR ACCENTED LETTERS
      DO 2240 LL=1,8
      LBYTE = 121 + 16 * LL
      IF (JBYTE .GT. LBYTE .AND. JBYTE .LE.
     *             LBYTE+5) GOTO 226
 2240 CONTINUE
      GOTO 228
C     REMOVE ACCENTS FROM ACCENTED LETTERS
  226 JBYTE = NBYTE(LMVOW,JBYTE-LBYTE)
```

```
C       USE LOWER CASE IF ACCENTED LETTER WAS LC
        IF (LBYTE .LT. 192) JBYTE = JBYTE - 64
C       STORE CHARACTER IN MODIFIED FILE NAME
  228   NCHARS = NCHARS + 1
        CALL STBYTE(JBYTE,MTEXT,NCHARS)
   23   CONTINUE
  230   CONTINUE
        CALL BMOVE(MTEXT,MFNAME)
        GOTO 31
C
C       RESTORE AND ERASE COMMANDS
   24   IF (LPROG .EQ. 2) GOTO 25
C       REQUEST IDENTIFIER
        CALL MBS(KSPACS,K1,MTEXT,K1,K4)
        CALL TMESS(12,LIDMES)
        CALL INTEX
        IF (NCHARS .EQ. 0) GOTO 25
        CALL MBS(MTEXT,K1,LMIDEN,K1,K4)
        LIDENT = 1
        GOTO 28
C       REQUEST NAME IF EXECUTING ERASE COMMAND OR
C       NO IDENTIFIER WAS SPECIFIED
   25   CALL TMESS(K6,LNMMES)
        CALL INTEX
        IF (NCHARS .GT. 0) GOTO 28
        IF (LPROG .EQ. 1) GOTO 31
   26   CALL REJECT
        GOTO 25
C       REQUEST VERSION NUMBER
   28   CALL TWT(12,LVERS)
        CALL RDNUMA(LVNO)
        IF (LVNO.GT.0 .AND. LVNO.LE.255) GOTO 31
        IF (LPROG .EQ. 2) GOTO 26
        LVNO = 0
        GOTO 31
C       LIST CONTENTS OF ARCHIVE TAPE
    3   CALL TNLINE
        CALL TMESS(32,LSUMES)
        CALL TNLINE
        CALL TMESS(24,LNSMES)
        CALL TWT1(LNFILS)
        IF (NDELS .EQ. 0) GOTO 305
        CALL TMESS(24,LNEMES)
        CALL TWT1(NDELS)
  305   CALL TNLINE
   31   IF (LNFILS .EQ. 0) GOTO 35
        IF (LPROG .GE. 0) GOTO 32
        CALL TMESS(K5,LIDMS)
        IXT = 14 * IXINC
        CALL TPOSN
        CALL TWT(K4,LTITLE)
        IXT = 30 * IXINC
        CALL TPOSN
        CALL TWT(10,LVERS)
        CALL TNLINE
C
C       SEARCH TAPE INDEX BACKWARDS
   32   DO 340 L=1,NFOT
        LL = NFOT + 1 - L
        LSTART = NHW(MINDEX,KSBASE+LL)
C       IGNORE DELETED FILE
        IF (LSTART .LE. 0) GOTO 34
        NVERS = NBYTE(MINDEX,KVBASE+LL)
        IF (LPROG .LT. 0) GOTO 33
        IF (LIDENT .EQ. 0) GOTO 322
C       TEST FOR MATCH WITH IDENTIFIER
        IF (CBS(MINDEX,KIBASE+4*(LL-1)+1,
     *              LMIDEN,K1,K4)) GOTO 325
        GOTO 34
  322   IF (NCHARS .EQ. 0) GOTO 38
C       TEST FOR MATCH WITH NAME
        IF (.NOT. CBS(MINDEX,LSTART+1,NCHARS,K4,
     *              NCHARS+1)) GOTO 34
C       SAVE STARTING LOCATION OF NAME IN INDEX
        LNAME = LSTART
  325   IF (LVNO.EQ.0 .OR. LVNO.EQ.NVERS) GOTO 37
        GOTO 34
C       DISPLAY IDENTIFIER, NAME AND VERSION NUMBER
   33   CALL MBS(MINDEX,KIBASE+4*(LL-1)+1,MTEXT,
     *              K1,K4)
```

```
      CALL TMESS(K4,MTEXT)
      CALL MBS(MINDEX,LSTART+1,MTEXT,K0,
     *              NBYTE(MINDEX,LSTART+1)+1)
      CALL TWT(K1,KSPACS)
      CALL TWT(NCHARS,MTEXT)
      IXT = IXQ + 37 * IXINC
      CALL TPOSN
      CALL TWT1(NVERS)
   34 CONTINUE
  340 CONTINUE
C
C     FILE NOT FOUND OR END OF LISTING
   35 IF (LPROG) 8,36,76
C     SAVE
   36 NVERS = 0
      IF (NFOT .EQ. C) GOTO 53
      GOTO 5
C
   37 IF (LIDENT .EQ. 0) GOTO 39
      CALL TWT(K6,LTITLE)
C     NO NAME SPECIFIED - OBTAIN AND DISPLAY NAME
C     OF THE LAST FILE ON THE TAPE
   38 CALL MBS(MINDEX,LSTART+1,MTEXT,K0,
     *              NBYTE(MINDEX,LSTART+1)+1)
      CALL TWT(NCHARS,MTEXT)
C
C     FILE FOUND
   39 LFILNO = LL
      IF (LPROG-1) 5,4,6
C
C     RESTORE
    4 CALL TMESS(25,LREMES)
      CALL BMOVE(MTEXT,MFNAME)
      CALL SKIPFILE(KATDEV,LFILNO)
      CALL AREAD(KATDEV,MOPBUF,KBFSIZ)
      NOPBUF = 0
      ISWRIT = 1
      CALL FUPDAT
      CALL RSETUP
      LFREE = 1
      DO 440 L=1,3
      IF (MSTAT(L) .GT. 1) GOTO 41
      MHEAD(L) = 0
      MTAIL(L) = 0
      GOTO 44
   41 LLINK = MHEAD(L)
      MHEAD(L) = LFREE
   42 CALL AREAD(KATDEV,MOPBUF,KBFSIZ)
      JOPLK = NHW(MOPBUF,K1)
      LL = LFREE + 1
      CALL MBS(LL,K3,MOPBUF,K1,K2)
      NOPBUF = LFREE
      ISWRIT = 1
      CALL FUPDAT
      LFREE = LFREE + 1
      IF (LLINK .EQ. MTAIL(L)) GOTO 43
      LLINK = JOPLK
      GOTO 42
   43 MTAIL(L) = LFREE - 1
   44 CONTINUE
  440 CONTINUE
      CALL REWIND(KATDEV)
      LISREW = 1
C     SET UP FREE CHAIN TO FILL REST OF DISC FILE
      DO 460 L=LFREE,KFSIZE
  460 CALL FCHAIN(L,L+1)
      CALL FCHAIN(KFSIZE,-2)
      JFREE = LFREE
      CALL SECURE
      GOTO 68
C
C     SAVE. CHECK FOR DELETED FILES AT END OF TAPE
    5 DO 510 L=1,NFOT
      LL = NFOT + 1 - L
      LSTART = NHW(MINDEX,KSBASE+LL)
      IF (LSTART .NE. 0) GOTO 52
  510 CONTINUE
C     TAPE CONTAINS ONLY DELETED FILES
      NFOT = 0
      GOTO 53
```

```
   52 NFOT = LL
      GOTO 54
   53 NUSED = 0
C     ERROR IF TOO MANY FILES ON TAPE ALREADY
   54 IF (NFOT .GE. LFMAX) GOTO 7
      IF (NVERS .GT. 0) GOTO 545
      LNAME = KNBASE + NUSED
C     ERROR IF NO ROOM FOR NEW NAME IN INDEX
      IF (LNAME + NNAME .GE. KISIZE) GOTO 7
C     COPY NEW FILE NAME TO TAPE INDEX
      CALL MBS(MFNAME,K0,MINDEX,LNAME+1,NFNAME+1)
      NUSED = NUSED + NFNAME + 1
C     COPY DISC FILE TO END OF ARCHIVE TAPE
  545 CALL TMESS(25,LSAMES)
      NFOT = NFOT + 1
      NVERS = NVERS + 1
      LFILNO = NFOT
C     COPY FILE IDENTIFIER TO TAPE INDEX
      CALL MBS(MIDENT,K1,
     *              MINDEX,KIBASE+4*NFOT-3,K4)
C     RECYCLE VERSION NUMBER IF TOO BIG
      IF (NVERS .GE. 256) NVERS = 1
C     COPY VERSION NUMBER TO TAPE INDEX
      CALL STBYTE(NVERS,MINDEX,KVBASE+NFOT)
C     POSITION TAPE TO END OF EXISTING FILES
      CALL SKIPFILE(KATDEV,NFOT)
C     SET LIST POINTERS FOR TAPE FILE
      JFREE = 1
      DO 5470 L=1,KNLIST
      IF (MHEAD(L) .EQ. 0) GOTO 547
      MHEAD(L) = JFREE
      MTAIL(L) = JFREE + (MBYTES(L)-1)/(KBFSIZ-2)
      JFREE = MTAIL(L) + 1
  547 CONTINUE
 5470 CONTINUE
C     OBTAIN HEADER BLOCK IN *MOPBUF* AND INSERT
C     NEW LIST POINTERS
      CALL HDRSET
C     COPY HEADER BLOCK TO TAPE
      CALL AWRITE(KATDEV,MOPBUF,KBFSIZ)
C     RESTORE LIST POINTERS FOR DISC FILE
      CALL RSETUP
C     NO POINT IN TRYING TO SAVE FAULTY FILE
      IF (ISERR .NE. 0) GOTO 9
      MSTAT(JIMODE) = JCSTAT
C     COPY NON-EMPTY SECTIONS OF DISC FILE TO TAPE
      LOLINK = 1
      DO 560 L=1,KNLIST
      IF (MSTAT(L) .LE. 1) GOTO 56
      JOPLK = MHEAD(L)
   55 LLINK = JOPLK
      CALL DREAD(LLINK,K1)
C     LINK TO NEXT SECTOR
      LOLINK = LOLINK + 1
      CALL STHW(LOLINK,MOPBUF,K1)
      CALL AWRITE(KATDEV,MOPBUF,KBFSIZ)
      IF (NOPBUF .NE. MTAIL(L)) GOTO 55
   56 CONTINUE
  560 CONTINUE
      ENDFILE KATDEV
      ENDFILE KATDEV
      GOTO 62
C
C     DELETE TAPE FILE ENTRY FROM INDEX
    6 CALL TNLINE
      CALL REQCR
      CALL TNLINE
      IF (ISEND .NE. 0) GOTO 77
      CALL BMOVE(MTEXT,MFNAME)
      IF (LFILNO .LT. NFOT) GOTO 61
C     DELETE LAST FILE ON TAPE BY DECREMENTING
C     TOTAL FILE COUNT
      NFOT = NFOT - 1
      GOTO 65
C     ZERO TITLE ADDRESS FIELD FOR DELETED FILE
   61 LNAME = 0
C
C     UPDATE INDEX
   62 CALL MBS(LNAME,K3,
     *              MINDEX,2*(KSBASE+LFILNO)-1,K2)
```

```
C     COUNT NUMBER OF HOLES IN INDEX REPRESENTING
C     DELETED FILES NOT YET REMOVED FROM TAPE
      NDELS = 0
      IF (NFOT .EQ. 0) GOTO 65
      DO 630 L=1,NFOT
  630 IF (NHW(MINDEX,KSBASE+L) .EQ. 0)
     *            NDELS = NDELS + 1
   65 CALL REWIND(KATDEV)
      CALL AWRITE(KATDEV,MINDEX,KISIZE)
      CALL REWIND(KATDEV)
      LISREW = 0
      NSECTS = NSECTS - 1
      IF (LPROG .NE. 0) GOTO 68
C     REMOVE DISC FILE PROTECTION FOR SAVE COMMAND
      DO 670 L=1,3
  670 IF (MSTAT(L) .EQ. 3) MSTAT(L) = 2
      JCSTAT = MSTAT(JIMODE)
C
   68 CALL BMOVE(MFNAME,MNAME)
      GOTO 8
C     COPY ARCHIVE TAPE TO NEW TAPE
   69 CALL LSEG(52)
      CALL TCOPY
      GOTO 85
C
C     INDEX FULL
    7 ISERR = 4
      GOTO 8
C     TAPE IN USE BY OTHER VERSION OF PROGRAM
   71 ISERR = 10
      GOTO 9
C     WRONG TAPE LOADED
   72 ISERR = 5
      GOTO 8
C     INDEX FORMAT ERROR
   74 ISERR = 6
      GOTO 8
C     FILE NOT ON TAPE
   76 ISERR = 7
      GOTO 8
C     ERASE COMMAND CANCELLED
   77 ISERR = 8
      GOTO 8
   78 ISERR = 1
      GOTO 8
C     SET INTERRUPT FLAG
   79 INIT = 1
    8 IF (LISREW .EQ. 0) GOTO 88
   85 CALL REWIND(KATDEV)
   88 IF (KLEVEL .EQ. 0) GOTO 9
C     CANCEL TAPE-IN-USE INDICATOR
      L1 = 0
      CALL TAPE(L1)
    9 RETURN
      END
C
C
      SUBROUTINE INTEX
C     INPUT TEXT STRING.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /KO/ KO,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
C
      NCHARS = 0
C     READ NEXT CHARACTER
    2 CALL NOECHO
      IF (MCHAR(1) .EQ. 0) GOTO 9
      IF (MCHAR(1) .EQ. KQERY(1)) GOTO 6
      IF (NCHARS .GE. 31) GOTO 72
C     STORE CHARACTER
      NCHARS = NCHARS + 1
      CALL MBS(MCHAR,K1,MTEXT,NCHARS,K1)
      GOTO 7
C     BACKSPACE
    6 IF (NCHARS .LE. 0) GOTO 72
      NCHARS = NCHARS - 1
C     DISPLAY INPUT CHARACTER
    7 CALL ECHO
      GOTO 2
```

```
   72 CALL REJECT
      GOTO 2
    9 RETURN
      END
C
C
      SUBROUTINE AWRITE(NUNIT,NBUFF,NLENG)
C     WRITE RECORD OF LENGTH *NLENG* BYTES FROM
C     INTERNAL BUFFER *NBUFF* TO ARCHIVE TAPE.
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MC/ KBSIZE,KWSIZE,KXMAX,KYMAX,KFSIZE
      DIMENSION NBUFF(1)
      CALL BUFFOUT(NUNIT,K1,NBUFF,NLENG/KWSIZE)
      CALL ACHECK(NUNIT,NLENG)
      RETURN
      END
C
C
      SUBROUTINE AREAD(NUNIT,NBUFF,NLENG)
C     READ NEXT RECORD FROM TAPE INTO FIRST
C     *NLENG* BYTES OF INTERNAL BUFFER *NBUFF*.
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /MC/ KBSIZE,KWSIZE,KXMAX,KYMAX,KFSIZE
      CALL BUFFIN(NUNIT,K1,NBUFF,NLENG/KWSIZE)
      CALL ACHECK(NUNIT,NLENG)
      RETURN
      END
C
C
      SUBROUTINE ACHECK(NUNIT,NLENG)
C     CHECK STATUS OF TAPE I/O OPERATION.
C     *NUNIT* = UNIT NUMBER,
C     *NLENG* = NUMBER OF BYTES EXPECTED.
      COMMON /BY/ NBYTES,MBYTES(3),NSECTS,NVERS
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /MC/ KBSIZE,KWSIZE,KXMAX,KYMAX,KFSIZE
      COMMON /SE/ JCMD,KSEGS(7)
      DIMENSION LMESS(3)
      DATA LMESS/10HTAPE ERROR/
      GO TO 3
    2 CALL WAIT
    3 CALL ICHECK(NUNIT,LSTAT,LWORDS)
      GOTO (2,8,4,5),LSTAT
C     END-OF-FILE READ, ERROR UNLESS COPY COMMAND
    4 IF (JCMD .NE. 23) GOTO 5
      ISEND = 1
      GOTO 9
    5 ISERR = 1
      CALL TMESS(10,LMESS)
      GOTO 9
    8 IF (LWORDS * KWSIZE .NE. NLENG) GOTO 5
      NSECTS = NSECTS + 1
    9 RETURN
      END
C
C
      SUBROUTINE REWIND(N)
C     REWIND MAGNETIC TAPE ON UNIT *N*.
      IF (N .EQ. 121) GOTO 3
      IF (N .EQ. 123) GOTO 5
      GOTO 9
    3 CONTINUE
S        LI,1       F:121
S        AW,1       =8Z01000000
S        CAL1,1     1            REWIND
      GOTO 9
    5 CONTINUE
S        LI,1       F:123
S        AW,1       =8Z01000000
S        CAL1,1     1            REWIND
    9 RETURN
      END
C
C
      SUBROUTINE TAPE(N)
C     IF *N* = 0, CANCELS TAPE-IN-USE INDICATOR,
C     OTHERWISE RETURNS *N* = CURRENT VALUE OF
C     TAPE INDICATOR, AND SETS INDICATOR = *N* IF
C     IF WAS PREVIOUSLY ZERO.
```

```
C      TAPE INDICATOR IS WORD 9 OF FGD MAILBOX.
S         LI,1       9            FMBOX INDEX
S         LW,2       *N
S         BEZ        5S           B IF ZERO ARGUMENT
S         LW,3       *X'147',1    FETCH INDICATOR
S         BNEZ       $+2
S         STW,2      *X'147',1    SET INDICATOR IF 0
S         STW,3      *N           RETURN ORIGINAL
S         B          9S
S    5    STW,2      *X'147',1    CANCEL INDICATOR
     9 RETURN
       END
!ASS (M:B0,D1,JHR0M51)
!FORTRAN B0,NS,S
C
C
C      ******************************************
C      *                                        *
C      *  SEGMENT 51 - ARCHIVE TAPE FILE COPYING  *
C      *                                        *
C      ******************************************
C
C
       SUBROUTINE FCHAIN(NSECT1,NSECT2)
C      LINK SECTOR *NSECT1* OF DISC FILE TO SECTOR
C      *NSECT2* AND FILL SECTOR *NSECT1* WITH ZEROS
       COMMON MC1(90),M0PBUF(90)
       COMMON /BF/ KBFSIZ,KBFWDS,KDATA
       COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
       CALL FLINK(NSECT1,NSECT2)
       DO 20 L=3,KBFSIZ
    2C CALL STBYTE(K0,M0PBUF,L)
       RETURN
       END
C
C
       SUBROUTINE RSETUP
C      INITIALISE PARAMETERS FOR CURRENT DISC FILE
C      FROM HEADER SECTOR.
       COMMON MINBUF(90),M0PBUF(90),NINBUF,N0PBUF
       COMMON /FH/ KNLIST,MBARS(3),MLINES(3),
      * MHEAD(3),MTAIL(3),JFREE,INBACK,NBARS
       COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
      *   ISERR,ISNXST,ISINPT
       COMMON /FS/ JCSTAT,JNSTAT,MSTAT(3)
       COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
      * NHTEXT,JSCALE
       COMMON /HE/ MCLEF(10),MPART(10)
       COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
       COMMON /M0/ JIM0DE,J0M0DE
       COMMON /NS/ NLINES,JLINE,JPART,JV0IC,NDP,NDL
       COMMON /SC/ IXINC,IYINC,IYSTV,MYSTV(3)
       COMMON /SE/ JCMD,KSEGS(7)
       COMMON /WS/ L1,L2,L3,JBYTE
       DIMENSION MHFILE(14),LMESS(5)
       EQUIVALENCE(MHFILE,KNLIST),(NLNB,MLINES(1))
       DATA LKSMAX/40/,LMESS/17HFILE FORMAT ERROR/
       NINBUF = -1
       N0PBUF = -1
       DO 180 L=1,KNLIST
  180  MSTAT(L) = 1
C      GET HORIZONTAL DISPLAY SCALE
       CALL INP0SN(K0)
       CALL INHW(JSCALE)
       CALL MBS(MINBUF,9,MIDENT,K1,K4)
       CALL INP0SN(10)
       CALL INHW(LNLIST)
       IF (LNLIST .NE. KNLIST) GOTO 8
       L1 = 4 * KNLIST + 2
       DO 20 L=2,L1
   20  CALL INHW(MHFILE(L))
C      FILE FORMAT CHECK
       DO 30 L=1,KNLIST
       IF (MBARS(L) .NE. 0 .AND. (MHEAD(L) .EQ. 0
      * .OR. MTAIL(L) .EQ. 0) .OR. MHEAD(L) .EQ.
      * JFREE) GOTO 8
   30  CONTINUE
       IF (NLNB.LE.0 .OR. NLNB.GT.LKSMAX) GOTO 8
       NDP = 4 / NLNB
       JHDR = 14 + 8 * KNLIST + NLNB
```

```
      JCSTAT = 2
      DO 40 L=1,KNLIST
   40 IF (MBARS(L) .GT. 0) MSTAT(L) = JCSTAT
C     GET PART CODES
      DO 50 L=1,NLNB
   50 CALL STBYTE(INBYTE(K0),MPART,L)
      JIMODE = 1
      CALL GETHDR
C     CHECK CORRECT NUMBER OF TEXT ITEMS IN HEADER
      IF (NHTEXT .EQ. 8) GOTO 85
C     ERROR IN HEADER BLOCK
    8 ISERR = 2
C     DISPLAY -FILE FORMAT ERROR- MESSAGE
      CALL TMESS(17,LMESS)
C     SET FORMATTED BRAILLE MODE TO FORCE CHANGE
C     TO PRINTED MUSIC, SETTING PARAMETERS
   85 NLINES = MLINES(3)
      NBARS = MBARS(3)
      JCSTAT = MSTAT(3)
      JIMODE = 3
      JOMODE = 1
      RETURN
      END
C
C
      SUBROUTINE INHW(N)
C     RETURNS *N* = 16-BIT NON-NEGATIVE INTEGER
C     READ FROM FILE AT CURRENT INPUT ADDRESS.
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      N = INBYTE(K1)
      N = 256 * N + INBYTE(K1)
      RETURN
      END
C
C
      SUBROUTINE RDNUMA(N)
C     READ INTEGER *N* FROM THE TERMINAL.
      COMMON /CH/ MCHAR(1),INCODE,KEOD,ISJOY,JX,JY
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      N = 0
    3 CALL NOECHO
      IF (MCHAR(1) .EQ. 0) GOTO 9
      LL = NBYTE(MCHAR,K1) - 240
      IF (LL .LT. 0 .OR. LL .GT. 9) GOTO 7
      N = 10 * N + LL
      CALL ECHO
      GOTO 3
    7 CALL REJECT
      GOTO 3
    9 RETURN
      END
!ASS (M:B0,D1,JHR0M52)
!FORTRAN B0,NS
C
C
C     ******************************************
C     *                                        *
C     *     SEGMENT 52 - ARCHIVE TAPE COPYING  *
C     *                                        *
C     ******************************************
C
C
      SUBROUTINE TCOPY
C     COPY ARCHIVE TAPE TO NEW TAPE REMOVING
C     DELETED FILES
      COMMON MNAMES(320)
      COMMON /AR/ KATDEV,KISIZE,KIBASE,KSBASE,
     * KVBASE,KNBASE,KMTMES(3),MINDEX(540),NFNAME,
     * MFNAME(8)
      COMMON /BF/ KBFSIZ,KBFWDS,KDATA
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /HD/ KTHDR(1),MIDENT(1),ISHDR,JHDR,
     * NHTEXT,JSCALE
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KT/ KSPACS(2),KQERY(1),KDOT(1)
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      EQUIVALENCE(NFOP,MINDEX(4)),
     * (NDELS,MINDEX(5)),(NUSED,MINDEX(6))
      LOGICAL CBS
```

AI-154

```
      DIMENSION MBUFF(90),LLTMES(7),LNTMES(4),
     * LCOMES(2)
      DATA KNTDEV/123/,KTITS/1280/
      DATA LLTMES/26HLOAD OUTPUT TAPE ON UNIT 1/,
     *LNTMES/15HNEW TAPE NAME? /LCOMES/7HCOPYING/
C
      CALL TMESS(26,LLTMES)
      CALL REQCR
C     REQUEST NEW TAPE NAME
      CALL MBS(KSPACS,K1,MTEXT,K1,K8)
   21 CALL TMESS(15,LNTMES)
      CALL INTEX
      IF (NCHARS .LE. 0) GOTO 21
      CALL TMESS(K7,LCOMES)
      CALL REWIND(KNTDEV)
C     WRITE DUMMY INDEX TO RESERVE INDEX SPACE
C     ON NEW TAPE
      DO 220 L=1,KISIZE
  220 CALL STBYTE(KO,MINDEX,L)
      CALL AWRITE(KNTDEV,MINDEX,KISIZE)
      IF (ISERR .NE. 0 .OR. ISEND .NE. 0) GOTO 72
      ENDFILE KNTDEV
C     READ OLD TAPE INDEX AGAIN
      CALL AREAD(KATDEV,MINDEX,KISIZE)
C     ENTER NEW TAPE NAME IN INDEX
      CALL MBS(MTEXT,K1,MINDEX,K1,K8)
      LFOT = NFOP
      LDELS = NDELS
C     SKIP TO START OF FIRST MUSIC FILE
      CALL SKIPFILE(KATDEV,K1)
C     SAVE NAMES
      CALL MBS(MINDEX,KNBASE+1,MNAMES,K1,KTITS)
C
      NUSED = 0
      NFOP = 0
      DO 70 L=1,LFOT
      LIST = NHW(MINDEX,KSBASE+L)
      IF (LIST .EQ. 0) GOTO 6
C     COPY FILE
      ISEND = 0
    3 CALL AREAD(KATDEV,MBUFF,KBFSIZ)
      IF (ISEND .NE. 0) GOTO 4
      CALL AWRITE(KNTDEV,MBUFF,KBFSIZ)
      GOTO 3
    4 ENDFILE KNTDEV
C     COPY IDENTIFIER
      CALL MBS(MINDEX,KIBASE+4*(L-1)+1,
     *              MINDEX,KIBASE+4*NFOP+1,K4)
C     COPY VERSION NUMBER
      CALL MBS(MINDEX,KVBASE+L,
     *              MINDEX,KVBASE+NFOP+1,K1)
C     SEARCH OUTPUT INDEX FOR NAME
      LLENG = NBYTE(MNAMES,LIST-KNBASE+1) + 1
      IF (NFOP .LE. 0) GOTO 52
      DO 50 LL=1,NFOP
      LOST = NHW(MINDEX,KSBASE+LL)
      IF (CBS(MNAMES,LIST-KNBASE+1,
     *              MINDEX,LOST+1,LLENG)) GOTO 55
   50 CONTINUE
C     NAME NOT ALREADY IN OUTPUT INDEX
   52 LOST = KNBASE + NUSED
      NUSED = NUSED + LLENG
      IF (NUSED .GT. KTITS) GOTO 74
      CALL MBS(MNAMES,LIST-KNBASE+1,
     *                 MINDEX,LOST+1,LLENG)
C     INSERT NEW START ADDRESS OF NAME
   55 NFOP = NFOP + 1
      CALL MBS(LOST,K3,
     *              MINDEX,2*(KSBASE+NFOP)-1,K2)
      GOTO 7
C     SKIP DELETED FILE ON INPUT TAPE
    6 CALL SKIPFILE(KATDEV,K1)
    7 CONTINUE
   70 CONTINUE
      IF (NFOP .NE. LFOT - LDELS) GOTO 74
C     RESET NUMBER OF DELETIONS TO ZERO
      NDELS = 0
C     OUTPUT NEW INDEX ON OUTPUT TAPE
      CALL REWIND(KNTDEV)
      CALL AWRITE(KNTDEV,MINDEX,KISIZE)
```

```
      CALL REWIND(KNTDEV)
      GOTO 9
C
C     WRONG TAPE OR NO TAPE
   72 ISERR = 5
      GOTO 9
C     INDEX FORMAT ERROR
   74 ISERR = 6
    9 RETURN
      END
!JOB JH,COMPILE MUSIC PROGRAM SEGMENT 6
!EXTRABGD 6800
!ASS (M;B0,D1,JHROM6)
!FORTRAN B0,NS
C
C
C     ******************************************
C     *                                        *
C     *          SEGMENT 6 - PLAY MUSIC        *
C     *                                        *
C     ******************************************
C
C
      SUBROUTINE PLAY
C     PLAY MUSIC USING ANALOGUE OUTPUTS.
      COMMON /CB/ JSPEED,MXCOM(3)
      COMMON /FG/ KLEVEL,IEA
      COMMON /FL/ INIT,ISCHEK,ISRPT,ISEDIT,ISEND,
     *   ISERR,ISNXST,ISINPT
      COMMON /GP/ JGROUP,MGROUP(7)
      COMMON /IT/ ITEM,ITTYPE,ITPTR,ITLENG,ITERR
      COMMON /IU/ JCLEF,JTYME,JKEY,NGROUP,NPAGE,
     * NLINE,NBEAM,NSCALE
      COMMON /KM/ KZEROS(2),KMRSIZ,MREST(1)
      COMMON /K0/ K0,K1,K2,K3,K4,K5,K6,K7,K8,KBIG
      COMMON /KY/ JKEYNO,JKEYSG,MKPCH(2),MTKPCH(2)
      COMMON /MD/ JBS,JBE,JSSEL,NSSEL,MSSEL(10)
      COMMON /NO/ KHS,JOCTAV,JPITCH,JLENTH,ISEXTR,
     * JDOTS,NHEDS,ISCHOR,ISSTEM,MHD(8),MEXTRA(4),
     * JAXIDS,ISTIE,ISSMAL,ISCTIE,MHEXT(4)
      COMMON /NS/ NLINES,JLINE,JPART,JVOIC,NCP,NDL
      COMMON /TI/ JTIME1,JTIME2,JTIME,JTMIN,JTMAX,
     * JBDUR,JNDUR,NNOTES
      COMMON /TX/ NCHARS,MTEXT(8),ISWORD
      COMMON /WS/ L1,L2,L3,JBYTE
      DIMENSION LMPICH(3),LITEXT(8),LMILEN(8),
     * LSTEXT(15),LMSLEN(5),LSPEED(5),LMNP(4),
     * LINTEN(8),LSPCH(7),LMPCH(108),LMORN(18),
     * LPLMES(2),LNFMES(6)
      EQUIVALENCE(KSET,K3),(NU1,MEXTRA(3)),
     * (NU2,MEXTRA(4)),(NORNAM,MHEXT(1)),
     * (KINACC,K2)
C
C     TEXT ITEMS INDICATING INTENSITY
      DATA LNITEX/8/,LITEXT/
     * 2HMP,2HMF,1HP,1HF,2HPP,2HFF,3HPPP,3HFFF/
C     LENGTHS OF TEXT ITEMS INDICATING INTENSITY
      DATA LMILEN/2,2,1,1,2,2,3,3/
C
C     TEXT ITEMS INDICATING SPEED
      DATA LNSTEX/5/,LSTEXT/
     *10HLENTO     ,10HADAGIO    ,10HALLEGRO   ,
     *10HPRESTO    ,11HPRESTISSIMO/
      DATA LMSLEN/5,6,7,6,11/
      DATA LSPEED/40,70,100,150,200/
C     PERCENTAGES OF TOTAL DURATION FOR STACCATOS
      DATA LMNP/50,25,90,75/
C     OUTPUT VALUES FOR INTENSITY CONTROL
      DATA LINTEN/200,235,180,265,160,300,
     *            140,512/
C     PITCH INDEX INCREMENTS FOR C,D,E,F,G,A,B
      DATA LSPCH/1,3,5,6,8,10,12/
C     OUTPUT VALUES FOR FREQUENCY CONTROL
      DATA LPSIZE/108/,LMPCH/
     *     62,  63,  64,  66,  68,  70,  71,  72,
     *     75,  77,  79,  81,  84,  86,  89,  92,
     *     95,  98, 102, 106, 109, 113, 118, 122,
     *    127, 132, 137, 143, 149, 155, 162, 170,
     *    177, 186, 194, 203, 213, 223, 233, 245,
     *    257, 270, 284, 298, 313, 330, 346, 364,
```

```
     *    384, 404, 426, 449, 473, 503,-101,-105,
     *   -109,-114,-118,-123,-128,-133,-139,-145,
     *   -152,-159,-166,-173,-181,-190,-199,-209,
     *   -219,-230,-241,-253,-266,-280,-294,-310,
     *   -326,-343,-361,-380,-401,-422,-445,-469,
     *   -495,19*0/
C
C     PITCH PATTERNS FOR ORNAMENTS - BYTE TABLE OF
C     INDICES TO *LMPICH*,  1 = NOTE ABOVE,
C     2 = WRITTEN NOTE, 3 = NOTE BELOW
      DATA LNORNS/9/,LMORN/
     18Z01010202,8Z03030202,8Z03030202,8Z01010202,
     38Z02020202,8Z01020302,8Z02020202,8Z03020102,
     58Z01020102,8Z01020102,
     68Z02030202,8Z02020202,8Z02030202,8Z02020202,
     88Z02010202,8Z02020202,8Z02010202,8Z02020202/
C
      DATA LPLMES/7HPLAYING/,LNFMES/
     * 22HNOT FOREGROUND PROGRAM/
      IF (KLEVEL .LE. 0) GOTO 7
      CALL BREAK(&78)
      CALL TMESS(K7,LPLMES)
      LISLUR = 0
      LPVTIE = 0
      LOPICH = 0
      LINT = LINTEN(1)
      LSPD = 100
      LPAUSE = 0
      LIDEC = 0
      LORNAM = 0
      LOVOL = 0
      LPLENG = 0
      LSLENG = 0
C     START CLOCK = *JSPEED* IS NUMBER OF BEATS
C     PER MINUTE
      CALL TIMER(MAXC(1500*JTIME2/JSPEED,K1))
C     POSITION FILE TO START OF FIRST BAR
      CALL BFIND(JBS,K0)
C
C     GET NEXT MEASURE
    2 CALL NXSEL
      IF (ISEND .NE. 0) GOTO 4
C
C     DECODE NEXT ITEM
    3 CALL IDCODE
      IF (ISEND .NE. 0) GOTO 2
C     IGNORE IN-ACCORD PARTS
      IF (JVOIC .NE. JLINE) GOTO 3
      GOTO (32,34,36,41,ITTYPE
      GOTO 3
C
C     CONTROL ITEM
   32 IF (ITEM .NE. KINACC) GOTO 3
C     SKIP TO END OF MEASURE OR SET RETURN
  321 CALL IDCODE
      IF (ISEND .NE. 0) GOTO 2
      IF (ITEM .NE. KSET) GOTO 321
      GOTO 3
C
C     SEPARATE SIGN ITEM
   34 LENTRY = ITEM - 32
      IF (LENTRY .GT. 16) GOTO 35
      IF (LENTRY .NE. 4) GOTO 3
C     IRREGULAR NOTE GROUPING
      LGRP = (NGROUP+JGROUP-1) * 100 / NGROUP
      LGROUP = NGROUP = 1
      GOTO 3
   35 LENTRY = LENTRY - 16
      GOTO (351,352,3,3,3,353,353,353,3,3),LENTRY
      GOTO 3
C     START OF SLUR
  351 LISLUR = 1
      GOTO 3
C     END OF SLUR
  352 LISLUR = 0
      LPLENG = 9 * JNDUR / 10
      LSLENG = JNDUR - LPLENG
      GOTO 3
C     START OR END OF CRESCENDO OR DECRESCENDO
  353 LIDEC = LENTRY - 7
```

```
      GOTO 3
C
C     TEXT ITEM
   36 IF (ISWORD .NE. 0) GOTO 3
C     TEST FOR TEXT ITEM INDICATING INTENSITY
      DO 380 L=1,LNITEX
      IF (NCHARS .NE. LMILEN(L)) GOTO 38
      LT = LITEXT(L)
      DO 370 LL=1,NCHARS
  370 IF (ISSAME(NBYTE(MTEXT,LL),NBYTE(LT,LL))
     *              .EQ. 0) GOTO 38
      GOTO 399
   38 CONTINUE
  380 CONTINUE
C     TEST FOR WORD INDICATING SPEED
      DO 3950 L=1,LNSTEX
      IF (NCHARS .NE. LMSLEN(L)) GOTO 395
      DO 3930 LL=1,NCHARS
 3930 IF (ISSAME(NBYTE(MTEXT,LL),NBYTE(LSTEXT,12*
     *  (L-1)+LL)) .NE. 0) GOTO 398
  395 CONTINUE
 3950 CONTINUE
      GOTO 3
  398 LSPD = LSPEED(L)
      GOTO 3
  399 LINT = LINTEN(L)
      GOTO 3
C
C     NOTE ITEM - OUTPUT PREVIOUS NOTE
    4 IF (LORNAM .EQ. 0 .OR. LORNAM .GT. LNORNS)
     *                          GOTO 405
      LNOTES = 8
      IF (LORNAM .EQ. 5 .AND. LPLENG .GT. 128)
     *            LNOTES = LPLENG /32 * 2
      LPLENG = LPLENG / LNOTES
      GOTO 407
  405 LNOTES = 1
      L1 = 2
      GOTO 408
  407 L1 = LNOTES = (LNOTES-1)/K8 * 8
      L1 = NBYTE(LMORN,8*LORNAM-L1+1)
  408 LOPICH = LMPCH(LMPICH(L1))
      CALL IDAC(K1,LOPICH)
      CALL IDAC(K2,LOVOL)
C     WAIT FOR SPECIFIED TIME DURATION
      LTIME = 0
      IF (LNOTES .EQ. 1 .AND. LPAUSE .NE. 0)
     *                  LPLENG = LPLENG * 4
   41 IF (LTIME .GT. LPLENG) GOTO 42
      CALL WAIT
      LTIME = LTIME + 4
      GOTO 41
   42 LNOTES = LNOTES - 1
      IF (LNOTES .GT. 0) GOTO 407
      LORNAM = 0
      IF (LSLENG .LE. 0) GOTO 43
C     SILENT PART OF NOTE DURATION
      LPLENG = LSLENG
      LSLENG = 0
      LOVOL = 0
      GOTO 405
   43 IF (ISEND .NE. 0) GOTO 8
C     NULL DISPLAY TO PERMIT INTERRUPT
      CALL TWT(K1,KZEROS)
C
C     EVALUATE PLAY PARAMETERS FOR NOTE ITEM
      JNDUR = JNDUR * 100 / LSPD
      IF (ISSMAL .NE. 0) JNDUR = 4
C     SOUND NOTE FOR 90 PER CENT OF FULL DURATION
      LNP = 90
      IF (LGROUP .EQ. 0) GOTO 46
      JNDUR = JNDUR * LGRP / 100
      LGROUP = LGROUP = 1
      LNP = 100
   46 IF (NORNAM .GT. 0 .AND. NORNAM .LE. 9)
     *                        LORNAM = NORNAM
      IF (ISEXTR .EQ. 0) GOTO 48
      JBYTE = NU1
      L1 = NBIT(K4)
      IF (JBYTE .GT. 0 .AND. JBYTE .LE. 4)
```

```
     *                          LNP = LMNP(JBYTE)
      JBYTE = NU2
      L1 = NBIT(K6)
C     PAUSE
      LPAUSE = JBYTE
      GOTO 5
   48 LPAUSE = 0
    5 LPLENG = (LNP+10*LISLUR) * JNDUR / 100
      IF (ISTIE .NE. 0) LPLENG = JNDUR
      LSLENG = JNDUR - LPLENG
      IF (JPITCH .EQ. 0) GOTO 58
C     EVALUATE ACTUAL PITCH
      IF (JAXIDS .EQ. 0) GOTO 55
C     PITCH ADJUSTMENT FOR MARKED ACCIDENTALS
      JBYTE = JAXIDS
      L1 = NBIT(K3)
C     TREAT AS NATURAL IF NO MARKED SHARPS/FLATS
      IF (JBYTE .EQ. 0) JBYTE = 8
C     ENTER ACCIDENTAL IN TEMP KEY SIGNATURE
      CALL STBYTE(JBYTE/K2=1,MTKPCH,JPITCH)
C     NO PITCH CHANGE IF PREVIOUS NOTE WAS TIED
   55 IF (LPVTIE .NE. 0) GOTO 57
C     EVALUATE PITCH INDEX FOR CURRENT NOTE AND
C     NOTES ABOVE AND BELOW FOR ORNAMENTS
      DO 560 L=1,3
      L1 = JPITCH + 2 - L
      IF (L1 .LE. 0) GOTO 552
      IF (L1 .GE. 8) GOTO 554
      LOCTAV = JOCTAV
      GOTO 556
  552 L1 = 7
      LOCTAV = JOCTAV - 1
      GOTO 556
  554 L1 = 1
      LOCTAV = JOCTAV + 1
  556 LMPICH(L) = 12 * LOCTAV + LSPCH(L1)
     *                + NBYTE(MTKPCH,L1) - 3
  560 CONTINUE
   57 LPVTIE = ISTIE
C     TREAT NOTE OUT OF PLAYABLE RANGE AS A REST
      IF (LMPICH(2) .LE. 0 .OR. LMPICH(2) .GT.
     *              LPSIZE) GOTO 58
      LOVOL = LINT
      GOTO 3
C     REST
   58 LOVOL = 0
      GOTO 3
C
    7 CALL TWT(22,LNFMES)
      GOTO 9
   78 INIT = 1
C
C     RESET OUTPUT VOLTAGES TO ZERO
    8 CALL IDAC(K1,K0)
      CALL IDAC(K2,K0)
C     STOP CLOCK
      CALL TIMER
    9 RETURN
      END
```

```
!JOB JH,LOAD CIMBAL FOREGROUND VERSION
!ALL (FILE,MV),(FSIZE,0)
!ALL (FILE,G0),(FORMAT,B),(RSIZE,27),(FSIZE,2000)
!ALL (FILE,X1),(FSIZE,350)
!ALL (FILE,X2),(FSIZE,300)
!ALL (FILE,X3),(FSIZE,35)
!ALL (FILE,X4),(FSIZE,30)
!ALL (FILE,X5),(FSIZE,35)
!ALL (FILE,X6),(FSIZE,0)
!POOL 2
!RADEDIT
:COPY (FILE,D1,JHR00T),(FILE,BT,G0)
:COPY (FILE,D1,JHR00TF),(FILE,BT,G0),ADD
!SY FOR FP,MUSIC
!BLOAD FORE,(UDC,7),(TEM,0),(PUB,COCLIB,ERRLIB);
:(LIB,USE,SYS),(TASKS,2),(FILE,FP,MUSIC),MAP
:ROOT (FILE,BT,G0,END)
:SEG (LINK,2),(FILE,D1,JHR0M2,5)
:SEG (LINK,21,0NT0,2),(FILE,D1,JHR0M21,10)
:SEG (LINK,22,0NT0,2),(FILE,D1,JHR0M22,13)
:SEG (LINK,3),(FILE,D1,JHR0M3,9)
:SEG (LINK,31,0NT0,3),(FILE,D1,JHR0M31,13)
:SEG (LINK,311,0NT0,31),(FILE,D1,JHR0M311,1)
:SEG (LINK,312,0NT0,31),(FILE,D1,JHR0M312,4)
:SEG (LINK,313,0NT0,31),(FILE,D1,JHR0M313,5)
:SEG (LINK,314,0NT0,31),(FILE,D1,JHR0M314,7)
:SEG (LINK,315,0NT0,31),(FILE,D1,JHR0M315,1)
:SEG (LINK,32,0NT0,3),(FILE,D1,JHR0M32,15)
:SEG (LINK,321,0NT0,32),(FILE,D1,JHR0M321,6)
:SEG (LINK,322,0NT0,32),(FILE,D1,JHR0M322,12)
:SEG (LINK,323,0NT0,32),(FILE,D1,JHR0M323,3)
:SEG (LINK,324,0NT0,32),(FILE,D1,JHR0M324,8)
:SEG (LINK,325,0NT0,32),(FILE,D1,JHR0M325,4)
:SEG (LINK,4),(FILE,D1,JHR0M4,6)
:SEG (LINK,41,0NT0,4),(FILE,D1,JHR0M41,1)
:SEG (LINK,42,0NT0,4),(FILE,D1,JHR0M42,1)
:SEG (LINK,421,0NT0,42),(FILE,D1,JHR0M421,9)
:SEG (LINK,422,0NT0,42),(FILE,D1,JHR0M422,6)
:SEG (LINK,423,0NT0,42),(FILE,D1,JHR0M423,10)
:SEG (LINK,43,0NT0,4),(FILE,D1,JHR0M43,10)
:SEG (LINK,44,0NT0,4),(FILE,D1,JHR0M44,4)
:SEG (LINK,5),(FILE,D1,JHR0M5,7)
:SEG (LINK,51,0NT0,5),(FILE,D1,JHR0M51,4)
:SEG (LINK,52,0NT0,5),(FILE,D1,JHR0M52,1)
:SEG (LINK,6),(FILE,D1,JHR0M6,1)
:ASSIGN (F:108,D1,PRINTER)
:ASSIGN (F:119,D2,FGDTEMP)
:ASSIGN (F:120,D2,JHMUSIC)
:ASSIGN (F:121,T2)
:ASSIGN (F:123,T1)
:ASSIGN (M:LL,0C)
:ASSIGN (M:L0,TL)
:ASSIGN (M:D0,TL)
```

```
!JOB JH,PROGRAM TO PACK TABLES FOR MUSIC PROGRAM
!EXTRABGD 6800
!FORTRAN NS,G0,S
C
C
C     ******************************************
C     *                                        *
C     *      PACK TABLES FOR MUSIC PROGRAM     *
C     *                                        *
C     ******************************************
C
C
      COMMON /KK/ JVBAR,KTEXT,JSPACE,JAT
      COMMON /OP/ JOPBUF,MOPBUF(20)
      DIMENSION MTABLE(600),MPTR(256),MDICT(1000),
     *  MTEXT(12),MBRL(2),MSIGN(18),MNAME(2),
     *  MCHAR(2),LMIN(3,5),LMAX(3,5),LCNAM(9),
     *  LTNAM(18),MTYPE(9)
      DIMENSION LBLOCK(19),LDATA(3),LDIMEN(5),
     * LEND(3),LEQUIV(6)
      EQUIVALENCE(LCLASS,LCODE,MSIGN(1))
      EQUIVALENCE(MBRL,MSIGN(2))
      DATA LMIN(1,1)/0/,LMIN(2,1)/0/,LMIN(3,1)/0/,
     * LMIN(1,2)/0/,LMIN(2,2)/0/,LMIN(3,2)/0/,
     * LMIN(1,3)/-63/,LMIN(2,3)/-63/,LMIN(3,3)/0/,
     * LMIN(1,4)/0/,LMIN(2,4)/0/,LMIN(3,4)/0/,
     * LMIN(1,5)/1/,LMIN(2,5)/1/,LMIN(3,5)/0/
      DATA LMAX(1,1)/0/,LMAX(2,1)/0/,LMAX(3,1)/0/,
     * LMAX(1,2)/0/,LMAX(2,2)/0/,LMAX(3,2)/0/,
     * LMAX(1,3)/63/,LMAX(2,3)/63/,LMAX(3,3)/0/,
     * LMAX(1,4)/64/,LMAX(2,4)/0/,LMAX(3,4)/0/,
     * LMAX(1,5)/63/,LMAX(2,5)/64/,LMAX(3,5)/64/
      DATA MTYPE(1)/2/,MTYPE(2)/1/,MTYPE(3)/3/,
     * MTYPE(4)/3/,MTYPE(5)/3/,MTYPE(6)/3/,
     * MTYPE(7)/4/,MTYPE(8)/5/,MTYPE(9)/3/
      DATA LCNAM/2HXA,2HXB,2HXC,2HXD,2HXE,2HXF,
     * 2HXG,2HXH,2HXI/
      DATA LTNAM/6HMPESIJ,6HMCODES,6HMCORD1,
     *  6HMCORD2,6HMCORD3,6HMCORD4,6HMCELLS,
     *  6HMTABLE,6HMPYTAB/
      DATA KBLANK/1H /,KSTAR/1H*/,KAT/1H@/
      DATA KVBAR/1H!/
      DATA KTMAX/256/,KDMAX/4000/,KMBSIZ/2400/
      DATA KNTABS/9/
      DATA KLPS/20/,KWPL/4/
C     TEXT STRINGS FOR BLOCK DATA SUBPROGRAMS
      DATA LBLOCK/28H      BLOCK DATA@C     THIS ,
     * 36HSUBPROGRAM IS PROGRAM-GENERATED.@   ,
     * 11H   COMMON //,LDATA/11H      DATA /,
     * LSLASH/3H/ @/,LOBRAK/1H(/,LCBRAK/2H)@/,
     * LDIMEN/17H      DIMENSION M/,LCOMMA/1H,/,
     * LCBRK2/3H))@/,LEND/10H      END@/,LLETM/
     * 1HM/,LEQUIV/21H)@      EQUIVALENCE(M/
C
      LBSIZE = (KLPS-1) * KWPL
      JSPACE = NBYTE(KBLANK,1)
      JVBAR = NBYTE(KVBAR,1)
      JAT = NBYTE(KAT,1)
C     INITIALISE OUTPUT BUFFER
      DO 110 JOPBUF=1,80
  110 CALL STBYTE(JSPACE,MOPBUF,JOPBUF)
      JOPBUF = 1
C
      DO 70 LTAB=1,KNTABS
      LTYPE = MTYPE(LTAB)
      JENTRY = 0
      NENTRY = 0
      JDICT = 0
      DO 120 L=1,KMBSIZ
  120 CALL STBYTE(0,MTABLE,L)
C     LAST TABLE IS HERSHEY CO-ORDINATE TABLE
      IF (LTAB .EQ. KNTABS) GO TO 2
C     SKIP TABLE HEADER LINE
      READ(1,999)
C     READ NEXT TABLE ENTRY
   15 READ(1,999) LENTRY,MCHAR(1),MSIGN(1),
     *         MCHAR(2),MSIGN(2),MSIGN(3),MTEXT
      IF (LENTRY .EQ. 999) GOTO 55
      JENTRY = JENTRY + 1
C     CHECK DATA FIELDS CONTAIN ALLOWED VALUES
```

```
      IF (LTYPE .NE. 1 .AND. MCHAR(1) .NE.
     * KBLANK .AND. MCHAR(1) .NE. KSTAR) GOTO 75
      IF (MCHAR(2) .NE. KBLANK .AND.
     *             MCHAR(2) .NE. KSTAR) GOTO 75
      DO 170 L=1,3
  170 IF (MSIGN(L) .LT. LMIN(L,LTYPE) .OR.
     *    MSIGN(L) .GT. LMAX(L,LTYPE)) GOTO 75
      IF (MAXO(LENTRY,JENTRY) .GT. KTMAX .AND.
     *             LTYPE .LT. 3) GOTO 76
      GOTO (3,35,4,45,5),LTYPE
C
C     HERSHEY MUSIC SYMBOL CO-ORDINATE TABLE
    2 READ(3,999)
C     READ NEXT LINE OF CO-ORDINATES
   21 READ(3,981) LENTRY,MSIGN
C     -VE SYMBOL NUMBER INDICATES END OF INPUT
      IF (LENTRY) 55,23,22
C     FIRST LINE OF NEW SYMBOL
   22 NENTRY = NENTRY + 1
      MPTR(NENTRY) = JDICT
C     FIRST CO-ORDINATE PAIR ALWAYS POSITION ONLY
      LL = 0
      LX = 0
      LY = 0
      LSTART = 2
   23 DO 250 L=LSTART,9
      LS1 = MSIGN(2*L-1)
      LS2 = MSIGN(2*L)
      IF (LS1 .NE. -64) GO TO 24
      IF (LS2 .EQ. -64) GO TO 26
      LL = 0
      GO TO 25
   24 IF (JDICT+2 .GT. KDMAX) GO TO 72
      CALL STBYTE(128*LL+64+LS1-LX,MDICT,JDICT+1)
      CALL STBYTE(192+LY-LS2,MDICT,JDICT+2)
      LL = 1
      LX = LS1
      LY = LS2
      JDICT = JDICT + 2
   25 CONTINUE
  250 CONTINUE
      LSTART = 1
      GO TO 21
C     REMOVE CONTINUATION BIT FROM LAST CO-ORD
   26 CALL STBYTE(NBYTE(MDICT,JDICT)-128,
     *                          MDICT,JDICT)
      GO TO 21
C
C     INPUT CHARACTER SET TABLE
    3 LCHAR = NECODE(NBYTE(MCHAR(1),1))
      CALL STBYTE(LENTRY,MTABLE,LCHAR+1)
C     INCLUDE LOWER CASE FOR ALPHABETIC CHARACTERS
      IF (LCHAR .GE. 193 .AND. LCHAR .LE. 233)
     *        CALL STBYTE(LENTRY,MTABLE,LCHAR-63)
      GOTO 15
C
C     MESSAGE TABLE
   35 LLENG = NLENG(MTEXT)
      IF (LLENG .GT. KTEXT) GOTO 75
      IF (JDICT + LLENG + 1 .GT. KDMAX) GOTO 72
      MPTR(LENTRY) = JDICT
      LL = LLENG
      IF (MCHAR(1) .EQ. KSTAR) LL = LL + 128
      CALL STBYTE(LL,MDICT,JDICT+1)
      IF (LLENG .EQ. 0) GOTO 37
      DO 360 L=1,LLENG
      LCHAR = NBYTE(MTEXT,L)
      IF (LCHAR .EQ. JAT) GO TO 354
C     CONVERT TO EBCDIC
      LCHAR = NECODE(LCHAR)
      GO TO 356
C     REPLACE -AT- SIGN BY BELL CHARACTER
  354 LCHAR = 7
  356 CALL STBYTE(LCHAR,MDICT,JDICT+L+1)
  360 CONTINUE
   37 JDICT = JDICT + LLENG + 1
      GOTO 15
C
C     DISPLAY CO-ORDINATE TABLE
    4 IF (JDICT + 2 .GT. KDMAX) GOTO 72
```

```
      IF (LENTRY .EQ. 0) GOTO 41
      NENTRY = MAXO(NENTRY,LENTRY)
      MPTR(LENTRY) = JDICT
   41 DO 420 L=1,2
      IF (MCHAR(L) .EQ. KBLANK)
     *                  MSIGN(L) = MSIGN(L) + 128
      CALL STBYTE(MSIGN(L)+64,MDICT,JDICT+L)
  420 CONTINUE
      JDICT = JDICT + 2
      GOTO 15
C
C     MUSIC SIGN TABLE
   45 IF (JENTRY .NE. LENTRY .OR. JENTRY .GT.
     *       KMBSIZ .OR. LCODE .GT. 64) GOTO 75
      IF (MCHAR(1).NE.KBLANK) LCODE = LCODE + 128
      CALL STBYTE(LCODE,MTABLE,JENTRY)
      GOTO 15
C
C     CONTRACTION TABLE
    5 LLENG = NLENG(MTEXT)
      IF (LLENG .LE. 0 .OR. LLENG .GT. 7) GOTO 75
      IF (JDICT + LLENG + 4 .GT. KDMAX) GOTO 72
      MPTR(JENTRY) = JDICT
      CALL STBYTE(LLENG,MDICT,JDICT+1)
      CALL STBYTE(LCLASS,MDICT,JDICT+2)
C     STORE CHARACTER STRING
      DO 510 L=1,LLENG
      LCHAR = NECODE(NBYTE(MTEXT,L))
      CALL STBYTE(LCHAR,MDICT,JDICT+L+2)
  510 CONTINUE
      IF (MBRL(2) .EQ. 0) GOTO 52
      CALL STBYTE(MBRL(1)+128,MDICT,JDICT+LLENG+3)
      CALL STBYTE(MBRL(2),MDICT,JDICT+LLENG+4)
      JDICT = JDICT + LLENG + 4
      GOTO 15
   52 CALL STBYTE(MBRL(1),MDICT,JDICT+LLENG+3)
      JDICT = JDICT + LLENG + 3
      GOTO 15
C
   55 GOTO (551,552,5515,554,552),LTYPE

  551 LLENG = 256
      GOTO 6
 5515 JENTRY = NENTRY
C     CHECK ROOM IN TABLE
  552 LBASE = 2 * (JENTRY + 1)
      IF (LBASE+JDICT .GT. KMBSIZ) GOTO 73
C     STORE NUMBER OF ENTRIES AT START OF TABLE
      CALL STHW(JENTRY,MTABLE,1)
C     TRANSFER POINTERS AND DICTIONARY TO TABLE
      DO 5530 L=1,JENTRY
 5530 CALL STHW(MPTR(L)+LBASE,MTABLE,L+1)
      CALL MBS(MDICT,1,MTABLE,LBASE+1,JDICT)
      LLENG = LBASE + JDICT
      GOTO 6
C
  554 LLENG = JENTRY
C
C     OUTPUT STATISTICS
    6 WRITE(108,998) JENTRY,JDICT
C     OUTPUT PACKED TABLE AS FORTRAN BLOCK DATA
      CALL MBS(LTNAM,8*LTAB-7,MNAME,1,8)
C     *LLENG* = NUMBER OF WORDS ROUNDED UP TO
C     FILL LAST LINE
      LLENG = ((LLENG-1)/4)/KWPL*KWPL+KWPL
      NBLOX = LLENG / LBSIZE
      NLAST = LLENG - NBLOX * LBSIZE
      IF (NLAST .EQ. 0) GO TO 61
      NBLOX = NBLOX + 1
      GO TO 615
   61 NLAST = LBSIZE
  615 LSTART = 1
      LFOUR = (LLENG-1)/4 * 4
      CALL STRING(75,LBLOCK)
      CALL STRING(2,LCNAM(LTAB))
      CALL STRING(2,LSLASH)
      CALL STRING(6,MNAME)
      CALL STRING(1,LOBRAK)
      CALL NUMBER(LLENG)
      CALL STRING(2,LCBRAK)
      IF (NBLOX .EQ. 1) GOTO 621
```

```
C     OUTPUT DIMENSION AND EQUIVALENCE STATEMENTS
      DO 620 L=1,NBLOX
      CALL STRING(17,LDIMEN)
      CALL NUMBER(L)
      CALL STRING(1,LOBRAK)
      LL = LBSIZE
      IF (L .EQ. NBLOX) LL = NLAST
      CALL NUMBER(LL)
      CALL STRING(21,LEQUIV)
      CALL NUMBER(L)
      CALL STRING(1,LCOMMA)
      CALL STRING(6,MNAME)
      CALL STRING(1,LOBRAK)
      CALL NUMBER(LBSIZE*(L-1)+1)
      CALL STRING(3,LCBRK2)
  620 CONTINUE
C     OUTPUT DATA IN HEXADECIMAL FORMAT
  621 DO 640 L=1,NBLOX
      CALL STRING(11,LDATA)
      IF (NBLOX .GT. 1) GOTO 625
      CALL STRING(6,MNAME)
      GOTO 627
  625 CALL STRING(1,LLETM)
      CALL NUMBER(L)
  627 CALL STRING(3,LSLASH)
      LLINES = KLPS - 2
      IF (L .EQ. NBLOX) LLINES = (NLAST/KWPL) - 1
      LSTART = LBSIZE * (L-1) + 1
      LLEND = LSTART + KWPL * LLINES - 1
      IF (LSTART .LE. LLEND)
     * WRITE(2,985) (MTABLE(LL),LL=LSTART,LLEND)
      LSTART = LLEND + 1
      LLEND = LSTART + KWPL - 1
      WRITE(2,986) (MTABLE(LL),LL=LSTART,LLEND)
  630 CONTINUE
  640 CONTINUE
      CALL STRING(10,LEND)
    7 CONTINUE
   70 CONTINUE
      ENDFILE 2
      GOTO 9
C
C     DICTIONARY FULL
   72 OUTPUT JDICT,LLENG,KDMAX
      WRITE(108,994)
      GOTO 78
C     TABLE FULL
   73 WRITE(108,995)
      GOTO 78
C     TABLE FORMAT ERROR
   75 WRITE(108,997)
      GOTO 78
C     ENTRY NUMBER TOO LARGE
   76 WRITE(108,993)
   78 WRITE(108,991) JENTRY,LTAB
    9 STOP
  981 FORMAT(X,I4,2X,18(X,I3))
  985 FORMAT((6H      *,4(2H8Z,Z8,1H,),30X))
  986 FORMAT(5X,3H*8Z,3(Z8,3H,8Z),Z8,1H/,30X)
  991 FORMAT(10H AT ENTRY ,I,9HIN TABLE ,I)
  993 FORMAT(23H ENTRY NUMBER TOO LARGE)
  994 FORMAT(20H DICTIONARY OVERFLOW)
  995 FORMAT(15H TABLE OVERFLOW)
  997 FORMAT(17H BAD ENTRY FORMAT)
  998 FORMAT(X,I,8H ENTRIES,3X,I,10H DICT USED)
  999 FORMAT(I3,X,A1,I3,A1,I3,I2,5X,12A4)
      END
C
C
      SUBROUTINE STRING(NLENG,NTEXT)
C     ADD TEXT STRING *NTEXT* OF LENGTH *NLENG*
C     TO CURRENT OUTPUT LINE.
C     =AT= SIGN INDICATES END OF LINE.
      COMMON /KK/ JVBAR,KTEXT,JSPACE,JAT
      COMMON /OP/ JOPBUF,MOPBUF(20)
      DATA KOPBUF/50/
      DO 50 L=1,NLENG
      LCHAR = NBYTE(NTEXT,L)
      IF (LCHAR .EQ. JAT) GOTO 4
      IF (JOPBUF .GT. KOPBUF) GOTO 7
```

```
      CALL STBYTE(LCHAR,M0PBUF,J0PBUF)
      J0PBUF = J0PBUF + 1
      G0T0 5
C     NEW LINE
    4 IF (J0PBUF .GT. K0PBUF) G0T0 44
C     FILL REST 0F LINE WITH SPACES
      D0 420 LL=J0PBUF,K0PBUF
  420 CALL STBYTE(JSPACE,M0PBUF,LL)
   44 CALL BUFFER0UT(2,1,M0PBUF,20,LSTAT)
      J0PBUF = 1
    5 CONTINUE
   50 CONTINUE
      G0T0 9
    7 WRITE(108,999)
      STOP
    9 RETURN
  999 F0RMAT(19H STATEMENT T00 L0NG)
      END
C
C
      SUBR0UTINE NUMBER(N)
C     ADD N0N-NEGATIVE INTEGER *N* AS TEXT STRING
C     T0 CURRENT 0UTPUT LINE.
      DIMENSION LMTEMP(3),LMDIG(3)
      DATA LMAX/12/,LMDIG/10H0123456789/
      LNDIG = 0
      LNUMB = N
    2 LNDIG = LNDIG + 1
      IF (LNDIG .GT. LMAX) G0T0 7
      LQU0T = LNUMB / 10
      LREM = LNUMB - 10 * LQU0T
      CALL MBS(LMDIG,LREM+1,LMTEMP,LMAX+1-LNDIG,1)
      LNUMB = LQU0T
      IF (LNUMB .GT. 0) G0T0 2
C     SHIFT CHARACTER STRING T0 START 0F *LMTEMP*
      CALL MBS(LMTEMP,LMAX+1-LNDIG,LMTEMP,1,LNDIG)
      CALL STRING(LNDIG,LMTEMP)
      G0T0 9
    7 WRITE(108,999)
      STOP
    9 RETURN
  999 F0RMAT(16H NUMBER T00 L0NG)
      END
C
C
      FUNCTION NLENG(NTEXT)
C     RETURNS *NLENG* = LENGTH 0F CHARACTER STRING
C     IN *NTEXT*, IGN0RING TRAILING BLANKS AND
C     FINAL VERTICAL BAR CHARACTER.
      C0MM0N /KK/ JVBAR,KTEXT,JSPACE,JAT
      DIMENSION NTEXT(1)
      D0 50 L=1,KTEXT
      NLENG = KTEXT + 1 - L
      LCHAR = NBYTE(NTEXT,NLENG)
      IF (LCHAR .NE. JSPACE) G0T0 7
   50 CONTINUE
      NLENG = 0
      G0T0 9
    7 IF (LCHAR .EQ. JVBAR) NLENG = NLENG - 1
    9 RETURN
      END
C
C
      FUNCTION NEC0DE(N)
C     MACHINE-DEPENDENT CHARACTER C0DE C0NVERSION.
C     RETURNS *NEC0DE* = EBCDIC C0DE F0R CHARACTER
C     REPRESENTED BY *N* IN CURRENT MACHINE C0DE.
      NEC0DE = N
      RETURN
      END
C
C
      SUBR0UTINE STBYTE(NNBYTE,NARRAY,NPTR)
C     STORE *NNBYTE* IN *NPTR*TH BYTE 0F *NARRAY*
S        LW,7      *NPTR
S        AI,7      -1
S        LW,9      *NNBYTE
S        STB,9     *NARRAY,7
      RETURN
      END
```

```
C                                                       !POOL 2
C                                                       !REV
      SUBROUTINE STHW(NNHW,NARRAY,NPTR)                 !SY
C     STORE 16-BIT VALUE *NNHW* IN *NPTR*TH             !ASS (F:1,D3,TABOUT)
C     HALFWORD OF *NARRAY*.                             !RUN BP,RJBUILD
S        LW,7       *NPTR                             JH
S        AI,7       -1                                NEWTAB
S        LW,9       *NNHW
S        STH,9      *NARRAY,7
      RETURN
      END
C
C
      FUNCTION NBYTE(NARRAY,NPTR)
C     RETURNS *NBYTE* = *NPTR*TH BYTE OF *NARRAY*
S        LW,7       *NPTR
S        AI,7       -1
S        LB,9       *NARRAY,7
S        STW,9      NBYTE
      RETURN
      END
C
C
      BLOCK DATA
C     CONSTANTS
      COMMON /KK/ JVBAR,KTEXT,JSPACE,JAT
      DATA KTEXT/48/
      END
!OLOAD GO,(UDCB,3)
:ASS (F:1,D3,TABLES)
:ASS (F:2,D3,TABOUT)
:ASS (F:3,D3,HYTAB)
!RADEDIT
:ALL (FIL,D3,TABLES),(FOR,C),(FSI,1500)
:ALL (FIL,D3,TABOUT),(FOR,C),(FSI,1500)
:ALL (FIL,D3,HYTAB),(FOR,C),(FSI,500)
!ASS (F:OUT,D3,TABLES)
!RJFILE JH,MUTABLE
!ASS (F:OUT,D3,HYTAB)
!RJFILE JH,HYTABLE
```

```
*MESSAGE TABLE
001 *
002 *               THE OPTIONS AVAILABLE ARE:
003 *
004 *               AD   ADD TO END OF CURRENT SECTION
005 *               BR   TRANSLATE TO BRAILLE
006 *               BU   BUILD NEW SECTION
007 *               CH   CHANGE PROGRAM PARAMETERS
008 *               CL   CLEAR CURRENT SECTION
009 *               CO   COPY ARCHIVE TAPE TO NEW TAPE
010 *               DE   DELETE SPECIFIED BARS
011 *               DI   DISPLAY SPECIFIED BARS
012 *               ED   EDIT CURRENT SECTION
013 *               EM   EMBOSS BRAILLE
014 *               ER   ERASE MAGNETIC TAPE FILE
015 *               FI   LIST FILES ON ARCHIVE TAPE
016 *               FL   CHANGE BRAILLE FLAG
017 *               FO   FORMAT BRAILLE
018 *               IN   INSERT BAR BEFORE SPECIFIED BAR
019 *               MO   CHANGE MODE
020 *               OP   LIST AVAILABLE COMMAND OPTIONS
021 *               PL   PLAY CURRENT SECTION
022 *               PR   PRINT ON PRINTER OR GRAPH PLOTTER
023 *               RE   RESTORE DISC FILE FROM ARCHIVE TAPE
024 *               SA   SAVE DISC FILE ON ARCHIVE TAPE
025 *               SC   CHANGE HORIZONTAL DISPLAY SCALE
026 *               ST   END OF RUN
027 *               SU   DISPLAY SUMMARY OF DISC FILE
028 *
029                  BEFORE !
030 *               BAR NUMBER ERROR IN FILE
031                  NO MUSIC STORED
032                  FILE ALREADY SAVED
033                  FILE PROTECTED
034                  WRONG MODE
035                  TOO MANY STAVES
036                  COMMAND NOT IMPLEMENTED
037 *               RUN LED2 TO EMBOSS BRAILLE
038                  PROGRAM RELEASED
039                   ... @PRESS TTY BUTTON@
```

```
*MESSAGE TABLE
001 *
002 *               THE OPTIONS AVAILABLE ARE:
003 *
004 *               AD   ADD TO END OF CURRENT SECTION
005 *               BR   TRANSLATE TO BRAILLE
006 *               BU   BUILD NEW SECTION
007 *               CH   CHANGE PROGRAM PARAMETERS
008 *               CL   CLEAR CURRENT SECTION
009 *               CO   COPY ARCHIVE TAPE TO NEW TAPE
010 *               DE   DELETE SPECIFIED BARS
011 *               DI   DISPLAY SPECIFIED BARS
012 *               ED   EDIT CURRENT SECTION
013 *               EM   EMBOSS BRAILLE
014 *               ER   ERASE MAGNETIC TAPE FILE
015 *               FI   LIST FILES ON ARCHIVE TAPE
016 *               FL   CHANGE BRAILLE FLAG
017 *               FO   FORMAT BRAILLE
018 *               IN   INSERT BAR BEFORE SPECIFIED BAR
019 *               MO   CHANGE MODE
020 *               OP   LIST AVAILABLE COMMAND OPTIONS
021 *               PL   PLAY CURRENT SECTION
022 *               PR   PRINT ON PRINTER OR GRAPH PLOTTER
023 *               RE   RESTORE DISC FILE FROM ARCHIVE TAPE
024 *               SA   SAVE DISC FILE ON ARCHIVE TAPE
025 *               SC   CHANGE HORIZONTAL DISPLAY SCALE
026 *               ST   END OF RUN
027 *               SU   DISPLAY SUMMARY OF DISC FILE
028 *
029                 BEFORE 1
030 *               BAR NUMBER ERROR IN FILE
031                 NO MUSIC STORED
032                 FILE ALREADY SAVED
033                 FILE PROTECTED
034                 WRONG MODE
035                 TOO MANY STAVES
036                 COMMAND NOT IMPLEMENTED
037 *               RUN LED2 TO EMBOSS BRAILLE
038                 PROGRAM RELEASED
039                  ... @PRESS TTY BUTTON@
```

```
040                 --INTERRUPT
041 *               ERROR IN THIS OPERATION
042 *               DISC FILE ERROR
043 *               FILE UNCHANGED
044 *               INDEX FULL, FILE NOT SAVED
045 *               WRONG TAPE OR NO TAPE LOADED
046 *               THE TAPE INDEX IS CORRUPTED
047 *               THIS FILE IS NOT ON THE TAPE
048 *               COMMAND CANCELLED
049                 OUTPUT FILE NOT ACCESSIBLE
050 *               TAPE IN USE, PLEASE WAIT
051                 BAR
052                 BAR
053                 LINE
054                 SPEED
055                 NUMBER OF STAVES
056                 STAVE
057                  TO
058                 PRINTED MUSIC
059                 UNFORMATTED BRAILLE
060                 FORMATTED BRAILLE
061 *               NEXT COMMAND? |
062 *               FILE |
063 *               MODE IS |
064 *               MUST BE IN RANGE
065 *               TAPE COPIED CORRECTLY
066 *               PLAYING COMPLETE
067 *               FILE UPDATED
068 *               DISC FILE FULL
069 *               PART
070 *               SUMMARY OF CURRENT DISC FILE
071 *               SCALE |
072 *               TYPE L FOR OPTION LIST |
073 *               THE CHANGEABLE PARAMETERS ARE:
074                 TRANSPOSE
075                 GT40
076                 DEBUGGING
077                 MOD
078                 GRADE 1
079                 SAGEM
```

```
080                 SPACING
081                 SAVED
082                 RESTØRED
083                 DELETED
084                   NAME IS "
085                 "  VERSIØN
086                  RECØRDS
087                  STAVES
088                  DELETED
089                 PLEASE CØNFIRM DELETIØN
090                  CHANGE FRØM
091                 SECTIØN
092                 STATUS
093                 BARS
094                 STAVES
095                 UNPRØTECTED
096                 PRØTECTED
097                 FALSE
098                 TRUE
099
100 *               ØMUSIC PRØGRAM |
101                 BACK
102                 FØRE
103                 GRØUND VERSIØN CØMPILED |
104 *               PRESS ASCII BUTTØN FØR FULL CHARACTER SET
105 *               INITIALISING DISC FILE
106 *               SELECTIØN NUMBER
107                  SELECT STAVES? |
108 *               CHANGE NUMBERS? |
109                  BRAILLE PAGES WRITTEN
110                 SQUEEZING   |
111 *               FØRMATTING
112 *               TRANSLATING
999                 END ØF MESSAGE TABLE
```

```
*INPUT LANGUAGE CHARACTER TABLE
001 !           END-OF-INPUT
002 ;           IN-ACCORD
003 à           SET-RETURN
004 |           BAR-LINE
004 1           BAR-LINE
005 :           DOTS FOR BAR-LINE
006 M           NO TIME CHECK
007 X           CROSSWIRES SWITCH
064 Z           CLEF SIGN
065 T           TIME SIGNATURE
066 K           KEY SIGNATURE
067 N           IRREGULAR GROUPING
068 Q           INKPRINT PAGE
069 L           INKPRINT LINE
070 J           NOTE BEAMING
071 H           HORIZONTAL DISPLAY SCALE
080 (           START OF SLUR
081 )           END OF SLUR
082 4           OCTAVA BELOW
083 -           END OF OCTAVA
084 5           OCTAVA ABOVE
085 <           START OF CRESCENDO
086 >           START OF DECRESCENDO
087             END OF (DE)CRESCENDO
088 P           START OF PEDALLING
089 *           END OF PEDALLING
128 ^           TEXT ABOVE
129 v           TEXT BELOW
130 W           SUNG TEXT
192 R           REST
193 C           NOTE PITCH C
194 D           NOTE PITCH D
195 E           NOTE PITCH E
196 F           NOTE PITCH F
197 G           NOTE PITCH G
198 A           NOTE PITCH A
199 B           NOTE PITCH B
199 C           ZERO
200 1           BREVE
201 2           SEMIBREVE
202 3           MINIM
203 4           CROTCHET
204 5           QUAVER
205 6           SEMIQUAVER
206 7           32ND NOTE
207 8           64TH NOTE
208 9           128TH NOTE
209 Y           OCTAVE UP
210 U           OCTAVE DOWN
211 $           FLAT
212 %           NATURAL
213 #           SHARP
214 I           TIE
215 S           STEM
217 =           NOTE ESCAPE CHARACTER
218 .           DOT FOR NOTE
219 "           REPEAT
220 &           CHORD SIGN
221 ?           CANCEL
222 +           INCREMENT PITCH
223 -           DECREMENT PITCH
224 0           OCTAVE NUMBER
999             END OF CHARACTER TABLE
*DISPLAY COORDINATES, SEGMENT 32
001 *-10 +18    DOUBLE FLAT
      00 -24
     +09 +08
     -09 +04
    *+10 -06
002 * 00 +18    FLAT
      00 -24
     +09 +08
     -09*+04
003 * 00 +12    NATURAL
      00 -16
     +05  00
    * 00 -08
      00 +16
     -05* 00
```

```
004 *+03 +12        SHARP
      00 -24
    *+04  00
      00 +24
    *-07 -09
     +10 +02
    * 00 -08
     -10*-02
005 * 00 -05        DOUBLE SHARP
     +10 +10
    *-10  00
     +10*-10
006 *-10 +10        DELETION CROSS
     +20 -20
    * 00 +20
     -20*-20
999                 END OF 32 COORDINATES
*DISPLAY COORDINATES, SEGMENT 321
004 * 00  00        :::
001 * 00 +17        :::
      00 +02
     +02  00
      00 -02
     -02  00
    * 00 +12
      00 +02
     +02  00
      00 -02
     -02  00
    *+06 -29
002   00 +48        THICK DOUBLE BARLINE
    *+05  00
      00 -48
     +01  00
      00 +48
     +01  00
      00*-48
003   00 +48        :::
     +01  00
      00 -48
      +01  00
    *+05  00
      00 -48
     +01  00
    *+06  00
      00 -48
     +01  00
      00 +48
    *+05  00
      00 -48
     +01  00
    *+06  00
      00*-48
005   00 +48        THIN DOUBLE BARLINE
    *+06  00
      00*-48
006   00 +03        DOTTED BARLINE
    * 00 +05
      00 +04
    * 00 +05
      00 +04
    * 00 +05
      00 +04
    * 00 +05
      00 +04
    * 00 +05
      00*+04
007 *+06 +12        SEGNO
     -06 +03
     -06  00
     -06 -03
      00 -06
     +24 -12
      00 -06
     -06 -03
     -06  00
     -06 +03
    *-06  00
     +24 +24
```

```
      * 00 -11
        00 -02
       +02  00
        00 +02
       -02  00
      *-24  00
       -02  00
        00 -02
       +02  00
        00*+02
008 **+06 +02        ENCIRCLED CROSS SIGN
       -02 +05
       -04 +03
       -04 -03
       -02 -05
       +02 -05
       +04 -03
       +04 +03
       +02 +05
      *+04  00
       -20  00
      *+10 +12
        00*-24
009   +01 -01        PAUSE
       -02  00
        00 +02
       +02  00
        00 -02
      *-11  00
       +03 +07
       +07 +03
       +07 -03
       +03*-07
      *-04  00
       +04 +10
       +04*-10
999                  END OF 321 COORDINATES
*DISPLAY COORDINATES, SEGMENT 322
001 * 00 -04         TREBLE CLEF
        00 +12
       +02 -02
       -04 -04
        C0 -04
       +04  00
        00 +04
       -03* 00
002 * 00 +02         BASS CLEF
       +03  00
        00 -04
       -03*-02
003 *-01 +04         C CLEF
        00 -08
      *+02  00
        00 +08
      * 00 -03
       -02  00
      * 00 -02
       +02* 00
999                  END OF 322 COORDINATES
*DISPLAY COORDINATES, SEGMENT 324
001 * 00 +08         BREVE NOTE HEAD
        00 -16
      *+12  00
        00 +16
      * 00 -05
       -12  00
      * 00 -06
       +12* 00
002 * 00  00         BREVE REST
003   +12  00        SEMIBREVE REST
      * 00 -01
       -12  00
      * 00 -01
004   +12  00        MINIM REST
      * 00 -01
       -12  00
      *+06*+02
005 *+12  00         CROTCHET REST
       -12  00
       +12*-24
```

```
010  +12  00        128TH REST
     -06 -12
    *-12  00
009  +12  00        64TH REST
     -06 -12
    *-12  00
008   12  00        32ND REST
     -06 -12
    *-12  00
007  +12  00        16TH REST
     -06 -12
    *-12  00
006  +12  00        QUAVER REST
     -12*-24
016   00 +16        LOWER MORDENT
      00 -16
018 *-10 +04        UPPER MORDENT
     +04 +08
     +04 -08
     +04 +08
     +04 -08
     +04*+08
017   00 +16        EXTENDED LOWER MORDENT
      00 -16
019 *-14 +04        EXTENDED UPPER MORDENT
     +04 +08
     +04 -08
     +04 +08
     +04 -08
     +04 +08
     +04 -08
     +04*+08
013 *+26  00        TURN AFTER NOTE
011 *-10 +02        TURN ABOVE NOTE
     +04 +08
     +04  00
     +04 -08
     +04  00
     +04*+08
014 *+26  00        INVERTED TURN AFTER NOTE

012 *-10 +10        INVERTED TURN ABOVE NOTE
     +04 -08
     +04  00
     +04 +08
     +04  00
     +04*-08
022 *-06 +08        MEZZO-STACCATO
     +12  00
    *-06 -06
024 *+01 +02        STACCATO
     -01  00
      00 +01
     +01  00
      00 -01
    *-01*-02
023  +02 +04        STACCATISSIMO
     -04  00
     +02*-04
028 *-03  00        SWELL
     -12 +04
     +12 +04
    *+12 -08
025 *-06  00        ATTACK
     +12 +04
    -12*+04
027 *-06 +04        TENUTO
     +12* 00
029  +01 -01        PAUSE
     -02  00
      00 +02
     +02  00
      00 -02
    *-11  00
     +03 +07
     +07 +03
     +07 -03
     +03*-07
026 *-04  00        MARTELLATO
     +04 +10
     +04*-10
```

```
020 * 00  00        DOUBLE ARPEGGIO
021 *-16 -12        SINGLE ARPEGGIO
     +03 -03
     -05 -05
     +05 -05
     -05 -05
     +05 -05
     -05 -05
     +05 -05
     -05 -05
     +05 -05
     -05 -05
     +05*-05
999                 END OF CO-ORDINATE TABLE
*BRAILLE MUSIC SIGN TABLE
001   09            OUTER SLUR
002   09            INNER SLUR
003 * 16            MEZZO-STACCATO
004   38            "
005 * 32            STACCATISSIMO
006   38            "
007   38            STACCATO
008   00            UNUSED
009 * 40            ATTACK
010   38            "
011 * 48            MARTELLATO
012   38            "
013 * 56            TENUTO
014   38            "
015 * 33            SWELL
016   04            "
017   28            WORD SIGN
018   12            SECOND INTERVAL
019   44            THIRD INTERVAL
020   60            FOURTH INTERVAL, NUMBER SIGN
021   20            FIFTH INTERVAL
022   52            6TH INTERVAL, CLOSE QUOTATION
023   18            SEVENTH INTERVAL
024   36            EIGHTH INTERVAL, WORD HYPHEN
025   01            FINGER 1
026   03            FINGER 2
027   07            FINGER 3
028   02            FINGER 4
029   05            FINGER 5, DOTS 1,3 FOR BREVE
030   06            TRIPLET SIGN
031   04            DOT 3
032   00            UNUSED
033   61            BREVE NOTE
034   61            SEMIBREVE NOTE
035   29            MINIM NOTE
036   57            CROTCHET NOTE
037   25            QUAVER NOTE
038   61            16TH NOTE
039   29            32ND NOTE
040   57            64TH NOTE
041   25            128TH NOTE
042   04            DOT FOR NOTE
043   44            SEGNO IN TEXT
044   44            ENCIRCLED CROSS IN TEXT
045   35            PAUSE IN TEXT
046   35            FLAT SIGN
047   33            NATURAL SIGN
048   41            SHARP SIGN
049   08            ACCENT SIGN
050   32            CAPITAL SIGN
051   40            ITALIC SIGN
052   48            LETTER SIGN
053   60            NUMBER SIGN
054   00            UNUSED
055 * 08            OCTAVE 0
056   08            "
057   08            OCTAVE 1
058   00            UNUSED
059   24            OCTAVE 2
060   00            UNUSED
061   56            OCTAVE 3
062   00            UNUSED
063   16            OCTAVE 4, BRAILLE DOT 5
```

064	64	*SPACE	104	C1	64TH TREMOLO
065	40	OCTAVE 5	105	04	128TH TREMOLO
066	00	UNUSED	106	00	*^
067	48	OCTAVE 6	107	02	*,
068	00	UNUSED	108	00	*%
069	32	OCTAVE 7	109	00	*-
070	00	UNUSED	110	00	*>
071 *	32	OCTAVE 8	111	38	*?
072	32	"	112 *	35	DOUBLE FLAT
073	00	UNUSED	113	35	"
074	63	*¢	114	35	FLAT
075	50	*.	115	00	UNUSED
076	00	*<	116	33	NATURAL
077	54	*(& REPEAT SIGN	117	00	UNUSED
078	00	*+	118	41	SHARP
079	00	*|	119	00	UNUSED
080	31	*δ	120 *	41	DOUBLE SHARP
081	01	REPETITION IN CROTCHETS	121	41	"
082	03	REPETITION IN QUAVERS	122	18	*:
083	07	REPETITION IN 8THS	123	60	*#
084	02	REPETITION IN 16THS	124	00	*@
085	05	REPETITION IN 32NDS	125	04	*'
086	04	REPETITION IN 64THS	126	54	*=
087	18	END OF CRESCENDO	127	38	*"
088	50	END OF DECRESCENDO	128	00	UNUSED
089	00	UNUSED	129	01	*A
090	22	*!	130	03	*B
091	00	*$	131	09	*C
092	00	**	132	25	*D
093	54	*)	133	17	*E
094	06	*;	134	11	*F
095	00	*¬	135	27	*G
096	36	*-	136	19	*H
097	00	*/	137	10	*I
098	04	SEMIBREVE STEM	138	00	A ACUTE
099	05	CROTCHET STEM	139	63	E ACUTE
100	01	QUAVER STEM	140	00	I ACUTE
101	03	8TH STEM OR TREMOLO	141	00	O ACUTE
102	07	16TH STEM OR TREMOLO	142	00	U ACUTE
103	02	32ND STEM OR TREMOLO	143 *	16	APPOGGIATURA

144	34	ACCIACCATURA	184 *	28	SINGLE STAVE ARPEGGIØ
145	30	*J	185	05	"
146	05	*K	186	00	A UMLAUT
147	07	*L	187	43	E UMLAUT
148	13	*M	188	59	I UMLAUT
149	29	*N	189	42	Ø UMLAUT
150	21	*Ø	190	51	U UMLAUT
151	15	*P	191	00	UNUSED
152	31	*Q	192	00	UNUSED
153	23	*R	193	01	*A
154	55	A GRAVE	194	03	*B
155	46	E GRAVE	195	09	*C
156	00	I GRAVE	196	25	*D
157	63	Ø GRAVE	197	17	*E
158	62	U GRAVE	198	11	*F
159 *	04	DØUBLE DØT FØR NØTE	199	27	*G
160	04	SINGLE DØT FØR NØTE	200	19	*H
161	00	NØ DØT FØR NØTE	201	10	*I
162	14	*S	202	00	*A ACUTE
163	30	*T	203	63	*E ACUTE
164	37	*U	204	00	*I ACUTE
165	39	*V	205	00	*Ø ACUTE
166	58	*W	206	00	*U ACUTE
167	45	*X	207	00	UNUSED
168	61	*Y	208	00	UNUSED
169	53	*Z	209	26	*J
170	33	A CIRCUMFLEX	210	05	*K
171	35	E CIRCUMFLEX	211	07	*L
172	41	I CIRCUMFLEX	212	13	*M
173	57	Ø CIRCUMFLEX	213	29	*N
174	49	U CIRCUMFLEX	214	21	*Ø
175 *	00	UNUSED	215	15	*P
176 *	00	UNUSED	216	31	*Q
177	C0	*1	217	23	*R
178	00	*2	218	55	*A GRAVE
179	00	*3	219	46	*E GRAVE
180	00	*4	220	00	*I GRAVE
181	00	*5	221	00	*Ø GRAVE
182	00	NØ ARPEGGIØ	222	62	*U GRAVE
183 *	16	DØUBLE STAVE ARPEGGIØ	223	00	UNUSED

224	00	UNUSED	264	39	64TH REST
225	00	UNUSED	265	45	128TH REST
226	14	*S	266	61	BREVE C
227	30	*T	267	61	SEMIBREVE C
228	37	*L	268	29	MINIM C
229	39	*V	269	57	CROTCHET C
230	58	*W	270	25	QUAVER C
231	45	*X	271	61	16TH C
232	61	*Y	272	29	32ND C
233	53	*Z	273	57	64TH C
234	33	*A CIRCUMFLEX	274	25	128TH C
235	35	*E CIRCUMFLEX	275	53	BREVE D
236	41	*I CIRCUMFLEX	276	53	SEMIBREVE D
237	00	*O CIRCUMFLEX	277	21	MINIM D
238	00	*U CIRCUMFLEX	278	49	CROTCHET D
239	00	UNUSED	279	17	QUAVER D
240	26	*C	280	53	16TH D
241	01	*1	281	21	32ND D
242	03	*2	282	49	64TH D
243	09	*3	283	17	128TH D
244	25	*4	284	47	BREVE E
245	17	*5	285	47	SEMIBREVE E
246	11	*6	286	15	MINIM E
247	27	*7	287	43	CROTCHET E
248	19	*8	288	11	QUAVER E
249	10	*9	289	47	16TH E
250	00	*A UMLAUT	290	15	32ND E
251	43	*E UMLAUT	291	43	64TH E
252	59	*I UMLAUT	292	11	128TH E
253	42	*O UMLAUT	293	63	BREVE F
254	51	*U UMLAUT	294	63	SEMIBREVE F
255 *	13	THREE MEASURES REST	295	31	MINIM F
256 *	13	TWO MEASURES REST	296	59	CROTCHET F
257	13	BREVE REST, MEASURE REST	297	27	QUAVER F
258	13	SEMIBREVE REST	298	63	16TH F
259	37	MINIM REST	299	31	32ND F
260	39	CROTCHET REST	300	59	64TH F
261	45	QUAVER REST	301	27	128TH F
262	13	16TH REST	302	55	BREVE G
263	37	32ND REST	303	55	SEMIBREVE G

304		23	MINIM G	344	*	35	DOUBLE BAR FOLLOWED BY DOTS
305		51	CROTCHET G	345		54	"
306		19	QUAVER G	346		00	UNUSED
307		55	16TH G	347	*	35	DOUBLE BAR PRECEDED AND
308		23	32ND G	348		54	...FOLLOWED BY DOTS
309		51	64TH G	349		00	UNUSED
310		19	128TH G	350	*	35	THIN DOUBLE BAR LINE
311		46	BREVE A	351	*	05	"
312		46	SEMIBREVE A	352		04	"
313		14	MINIM A	353		05	DOTTED BAR LINE
314		42	CROTCHET A	354		00	UNUSED
315		10	QUAVER A	355		00	UNUSED
316		46	16TH A	356		00	UNUSED
317		14	32ND A	357		00	UNUSED
318		42	64TH A	358		00	UNUSED
319		10	128TH A	359	*	44	SEGNO
320		62	BREVE B	360		64	"
321		62	SEMIBREVE B	361		00	UNUSED
322		30	MINIM B	362	*	12	ENCIRCLED CROSS
323		58	CROTCHET B	363		07	"
324		26	QUAVER B	364		00	NO PAUSE
325		62	16TH B	365	*	56	PAUSE ABOVE BAR LINE
326		30	32ND B	366	*	35	" & PAUSE ON NOTE OR REST
327		58	64TH B	367		07	"
328		26	128TH B	368	*	40	RIGHT HAND SIGN
329	*	35	WHOLE MEASURE IN-ACCORD	369		28	"
330		28	"	370		00	UNUSED
331		00	UNUSED	371	*	56	LEFT HAND SIGN
332	*	40	MEASURE DIVISION	372		28	"
333		05	"	373		00	UNUSED
334		00	UNUSED	374	*	60	FIRST ENDING SIGN
335	*	16	PART-MEASURE IN-ACCORD	375		02	"
336		02	"	376		00	UNUSED
337		00	UNUSED	377	*	60	SECOND ENDING SIGN
338	*	35	DOUBLE BAR PRECEDED BY DOTS	378		06	"
339		06	"	379	*	32	TURN ABOVE/BELOW NOTE
340		00	UNUSED	380		50	"
341	*	35	THICK DOUBLE BAR LINE	381		00	UNUSED
342		05	"	382	*	32	INVERTED TURN ABOVE/BELOW NOTE
343		00	UNUSED	383	*	50	"

```
384    07          "
385    50          TURN FOLLOWING NOTE
386  * 48          OPEN BRACKET SLUR
387    03          "
388  * 50          INVERTED TURN FOLLOWING NOTE
389    07          "
390    00          UNUSED
391    22          TRILL
392  * 24          CLOSE BRACKET SLUR
393    06          "
394  * 16          LOWER MORDENT
395  * 22          "
396    07          "
397  * 48          EXTENDED LOWER MORDENT
398  * 22          "
399    07          "
400  * 16          UPPER MORDENT
401    22          "
402    00          UNUSED
403  * 48          EXTENDED UPPER MORDENT
404    22          "
405  * 35          PEDALLING
406    09          "
407  * 33          END OF PEDALLING
408    09          "
409  * 16          HALF PEDALLING
410  * 35          "
411    09          "
412  * 40          COMMON TIME
413    09          "
414  * 56          CUT TIME
415    09          "
416    00          UNUSED
417    00          UNUSED
418    00          UNUSED
419  * 08          TIE
420    09          "
421  * 40          CHORD TIE
422    09          "
423  * 16          BRAILLE MUSIC HYPHEN
424    64          "
425  * 32          BREATH MARK
426    12          UNUSED
427    00          UNUSED
428    00          UNUSED
429  * 56          START PARALLEL MOVEMENT
430  * 28          "
431  * 04          "
432  * 36          "
433    36          "
434  * 16          INKPRINT PAGE NUMBER SIGN
435  * 18          "
436    60          "
437  * 40          RIGHT HAND SIGN
438    28          "
439  * 56          LEFT HAND SIGN
440    28          "
441  * 24          ORGAN PEDALS
442    28          "
999                 END OF SIGN TABLE
*BRAILLE CONTRACTION TABLE
       63  43      ED
       63  59      ER
       15  4817    ENCE
       63  34      EN
       09  02      EA
       63  28      AR
       63  47      AND
       54  53      AS
       54  12      STILL
       63  12      ST
       54  41      SHALL
       63  4125    SHOULD
       49  41      SH
       15  4029    SION
       63  1425    SAID
       49  1614    SOME
       54  14      SO
       63  1433    SUCH
       49  1646    THERE
```

```
63  5646    THEIR
54  2446    THESE
63  46      THE
54  30      THAT
54  57      THIS
63  1657    THROUGH
63  57      TH
15  4829    TION
63  1630    TIME
15  44      ING
54  20      IN
09  20      IN
54  4514    ITS
54  45      IT
63  55      OF
15  4025    OUND
15  4030    OUNT
63  1651    OUGHT
63  51      OU
63  42      OW
15  4827    ONG
63  1621    ONE
54  07      LIKE
63  1607    LORD
63  1623    RIGHT
54  29      NOT
63  1629    NAME
15  4814    NESS
54  33      CHILD
63  33      CH
49  18      CON
63  0925    COULD
54  09      CAN
09  18      CC
54  25      DE
49  50      DIS
09  50      DC
63  1615    PART
15  4830    MENT
63  1333    MUCH
14  1312    MUST
54  37      US
54  03      BUT
18  06      BE
09  06      BB
15  60      BLE
63  63      FOR
63  1123    FRIEND
54  11      FROM
54  1112    FIRST
15  4807    FUL
09  22      FF
63  35      GH
54  27      GO
09  54      GG
63  62      WITH
54  58      WILL
54  49      WHICH
63  1649    WHERE
63  2449    WHOSE
63  49      WH
18  54      WERE
18  52      WAS
63  5825    WOULD
63  1658    WORK
63  2458    WORD
63  5658    WORLD
63  1661    YOUNG
63  6123    YOUR
54  61      YOU
54  5619    HAD
54  19      HAVE
54  1913    HIM
18  38      HIS
63  1619    HERE
63  1605    KNOW
54  26      JUST
63  3105    QUICK
54  31      QUITE
999         END OF CONTRACTION TABLE
```

```
*HERSHEY CO-ORDINATES FOR MUSIC SYMBOLS
2367  : -4   4: -1  -1: -1   1:  1   1:  1  -1: -1  -1:-64   0: -1  -1:  1   1:
       -64   0:  1  -1: -1   1:-64 -64:
2368  : -8   8: -5  -2: -1   0:  2   2:  4   4:  5   7:  5   9:  4  11:  3  12:
       -64   0: -5  -1:  1   2:-64   0: -5   0: -2   1:  2   3:  4   5:  5   7:
       -64 -64:
2369  : -8   8:  5  -7:  4  -5:  2  -3: -2  -1: -5   0:-64   0:  1  -2: -5   1:
       -64   0:  3 -12:  4 -11:  5  -9:  5  -7:  4  -4:  2  -2: -1   0: -5   2:
       -64 -64:
2370  :-10  10: -2  -5: -5  -4: -6  -3: -7  -1: -7   1: -6   3: -5   4: -2   5:
         2   5:  5   4:  6   3:  7   1:  7  -1:  6  -3:  5  -4:  2  -5: -2  -5:
       -64   0: -5  -4: -6  -2: -6   1: -5   3: -4   4: -2   5:-64   0:  5   4:
         6   2:  6  -1:  5  -3:  4  -4:  2  -5:-64 -64:
2371  :-10  10:  1  -5: -3  -4: -6  -2: -7   0: -7   2: -6   4: -4   5: -1   5:
         3   4:  6   2:  7   0:  7  -2:  6  -4:  4  -5:  1  -5:-64   0:  6  -4:
         1  -5:-64   0:  4  -5: -1  -4: -6  -2:-64   0: -3  -4: -7   0:-64   0:
        -6   4: -1   5:-64   0: -4   5:  1   4:  6   2:-64   0:  3   4:  7   0:
       -64 -64:
2372  : -8   9:  1  -5: -2  -4: -4  -2: -5   0: -5   2: -4   4: -2   5:  0   5:
         3   4:  5   2:  6   0:  6  -2:  5  -4:  3  -5:  1  -5:-64   0: -3  -2:
         3  -5:-64   0: -4   0:  4  -4:-64   0: -5   2:  5  -3:-64   0: -4   3:
         6  -2:-64   0: -3   4:  5   0:-64   0: -2   5:  4   2:-64 -64:
2373  : -8   8: -3 -11: -3  12:-64   0:  3 -12:  3  11:-64   0: -5  -4:  5  -6:
       -64   0: -5  -3:  5  -5:-64   0: -5   5:  5   3:-64   0: -5   6:  5   4:
       -64 -64:
2374  : -8   8: -4 -12: -4   6:-64   0:  4  -6:  4  12:-64   0: -4  -4:  4  -6:
       -64   0: -4  -3:  4  -5:-64   0: -4   5:  4   3:-64   0: -4   6:  4   4:
       -64 -64:
2375  : -8   8: -4 -16: -4   5:-64   0: -4  -4: -1  -6:  2  -6:  4  -5:  5  -3:
         5  -1:  4   1:  1   3: -1   4: -4   5:-64   0: -4  -4: -1  -5:  2  -5:
         4  -4:-64   0:  3  -5:  4  -3:  4  -1:  3   1:  1   3:-64 -64:
2376  :-13  13:-10  -9:-10  -6:-64   0: 10  -9: 10  -6:-64   0:-10  -9: 10  -9:
       -64   0:-10  -8: 10  -8:-64   0:-10  -7: 10  -7:-64   0:-10  -6: 10  -6:
       -64 -64:
2377  : -8   8: -5  -4: -5  -1:-64   0:  5  -4:  5  -1:-64   0: -5  -4:  5  -4:
       -64   0: -5  -3:  5  -3:-64   0: -5  -2:  5  -2:-64   0: -5  -1:  5  -1:
       -64 -64:
2378  : -8   8: -1 -15:  4  -5:  0   2:  0   3:-64   0:  3  -6: -1   1:-64   0:
         2  -9:  2  -7: -2   0:  0   3:  3   7:-64   0:  5  10:  3   7:  1   6:
        -1   6: -3   7: -4   9: -4  11: -3  13:  0  15:-64   0:  5  10:  3   8:
```

```
         1   7: -3   7: -3  11: -2  13:  0  15:-64   0:  1   6: -2   8: -4  11:
      -64 -64:
2379  : -8   8: -2  -3: -3  -5: -3  -7: -5  -7: -5  -5: -4  -4: -2  -3:  1  -3:
         3  -4:  5  -6:-64   0: -4  -7: -4  -4:-64   0: -5  -6: -3  -6:-64   0:
        -5  -5:  1  -3:-64   0: -2  -3:  3  -4:-64   0:  5  -6:  1   7:-64 -64:
2380  :-17  12:-11  20:-10  20: -9  19: -9  18:-10  17:-11  17:-12  18:-12  20:
       -11  22: -9  23: -7  23: -4  22: -2  20: -1  18:  0  14:  0   3: -1 -23:
        -1 -30:  0 -35:  1 -37:  3 -38:  4 -38:  6 -37:  7 -35:  7 -31:  6 -28:
         5 -26:  3 -23: -2 -19: -8 -15:-10 -13:-12 -10:-13  -8:-14  -4:-14   0:
       -13   4:-11   7: -8   9: -4  10:  0  10:  4   9:  6   8:  8   5:  9   2:
         9  -2:  8  -5:  7  -7:  5  -9:  2 -10: -2 -10: -5  -9: -7  -7: -8  -4:
        -8   0: -7   3: -5   5:-64   0:-11  18:-11  19:-10  19:-10  18:-11  18:
       -64   0:  3 -23: -1 -19: -6 -15: -9 -12:-11  -9:-12  -7:-13  -4:-13   0:
       -12   4:-11   6: -8   9:-64   0:  0  10:  3   9:  5   8:  7   5:  8   2:
         8  -2:  7  -5:  6  -7:  4  -9:  2 -10:-64 -64:
2381  : -9  24: -4  -1: -3  -3: -1  -4:  1  -4:  3  -3:  4  -1:  4   1:  3   3:
         1   4: -1   4: -3   3: -4   2: -5  -1: -5  -4: -4  -7: -2  -9:  1 -10:
         5 -10:  9  -9: 12  -7: 14  -4: 15   0: 15   5: 14   9: 13  11: 11  14:
         8  17:  4  20: -1  23: -5  25:-64   0:  5 -10:  8  -9: 11  -7: 13  -4:
        14   0: 14   5: 13   9: 12  11: 10  14:  7  17:  2  21: -1  23:-64   0:
        -2  -3:  2  -3:-64   0: -3  -2:  3  -2:-64   0: -4  -1:  4  -1:-64   0:
        -4   0:  4   0:-64   0: -4   1:  4   1:-64   0: -3   2:  3   2:-64   0:
        -2   3:  2   3:-64   0: 19  -6: 19  -4: 21  -4: 21  -6: 19  -6:-64   0:
        20  -6: 20  -4:-64   0: 19  -5: 21  -5:-64   0: 19   4: 19   6: 21   6:
        21   4: 19   4:-64   0: 20   4: 20   6:-64   0: 19   5: 21   5:-64 -64:
2382  :-14  14:-10 -20:-10  20:-64   0: -9 -20: -9  20:-64   0: -5 -20: -5  20:
       -64   0: -1 -16:  1 -16:  1 -14: -1 -14: -1 -17:  0 -19:  2 -20:  5 -20:
         7 -19:  9 -17: 10 -14: 10  -9:  9  -6:  7  -4:  5  -3:  3  -3:  1  -4:
         0  -6: -1  -4: -3  -1: -4   0: -3   1: -1   4:  0   6:  1   4:  3   3:
         5   3:  7   4:  9   6: 10   9: 10  14:  9  17:  7  19:  5  20:  2  20:
         0  19: -1  17: -1  14:  1  14:  1  16: -1  16:-64   0:  0 -16:  0 -14:
       -64   0: -1 -15:  1 -15:-64   0:  7 -19:  8 -17:  9 -14:  9  -9:  8  -6:
         7  -4:-64   0:  0  -6:  0  -4: -2  -1: -4   0: -2   1:  0   4:  0   6:
       -64   0:  7   4:  8   6:  9   9:  9  14:  8  17:  7  19:-64   0:  0  14:
         0  16:-64   0: -1  15:  1  15:-64 -64:
-1
```

Appendix 2. CIMBAL flow diagrams

List of diagrams

Appendix 2. CIMBAL flow diagrams

List of diagrams

Notes on flow diagrams

The flow diagrams should be read in conjunction with the program listing. Where a diagram roughly corresponds to a subprogram or group of subprograms in the program, the routine names are shown following the diagram title. The diagrams show the logical flow of the program conceptually; they do not specify all operations in complete detail and do not generally refer to the program variables in which information is stored and transmitted. Operations on individual items are not included.

In the diagrams, circular boxes represent entry and exit points; each contains the diagram number. Rectangular boxes represent operations. Operation boxes containing only text within quotation marks represent messages displayed on the screen of the terminal. Hexagonal boxes represent decisions. Each decision box has two exits marked "Y" and "N" corresponding to answers "yes" and "no" to the question in the box.

Boxes having an additional vertical bar at each side represent operations or decisions which are elaborated in a separate diagram, the number of which is shown in or adjacent to the box.

The word "next" should be read as "first" the first time an operation is executed.

Abbreviations used in the flow diagrams

FB	Formatted braille
HFC	Head of free chain
LT	Location table
MB	Measure buffer
O/P	Output
PTR	Pointer
TB	Temporary buffer
UB	Unformatted braille
#	Number (of)

Figure A2:1 CIMBAL

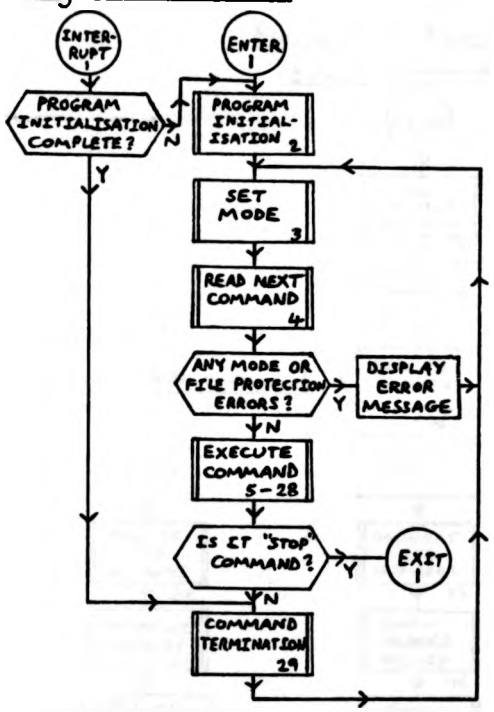


Figure A2:2 Program initialisation

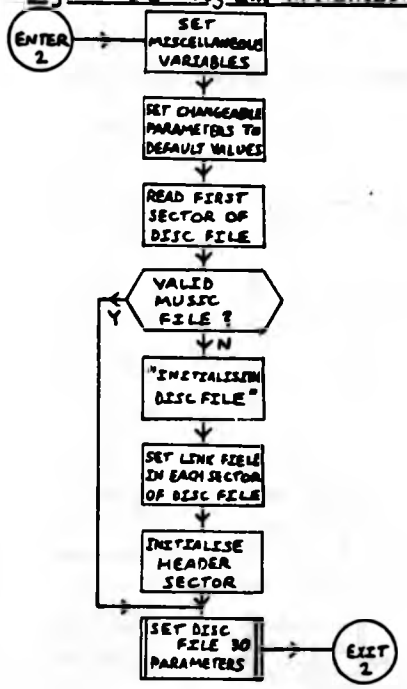


Figure A2:3 Set mode

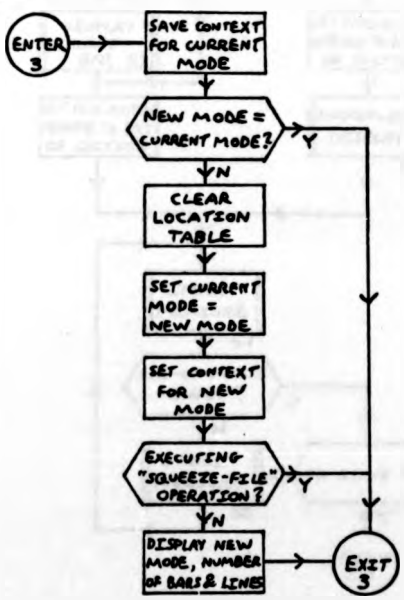


Figure A2:4 Read next command

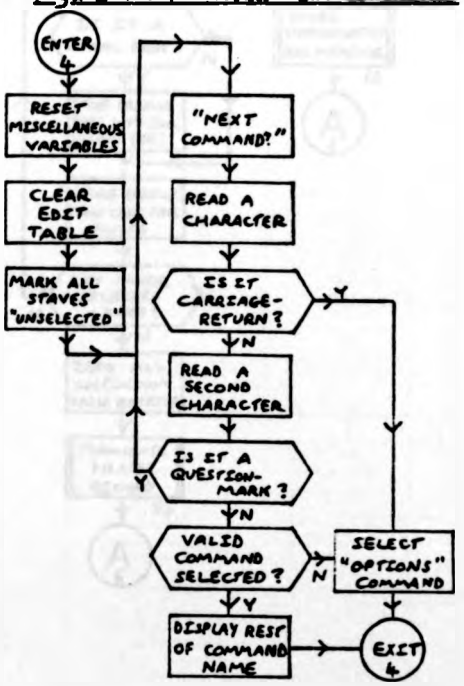


Figure A2:5
Execute ADD command

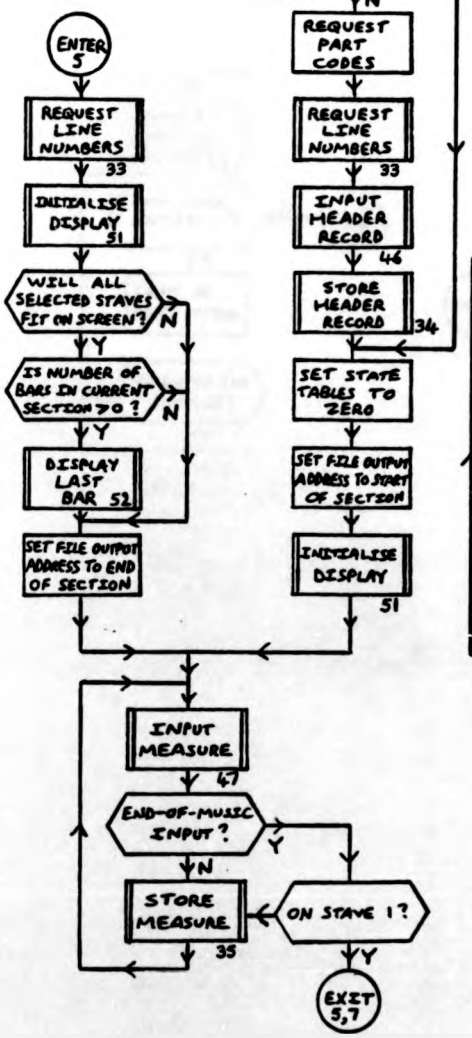


Figure A2:7 Execute
BUILD command

Figure A2:6
Execute BRAILLE command

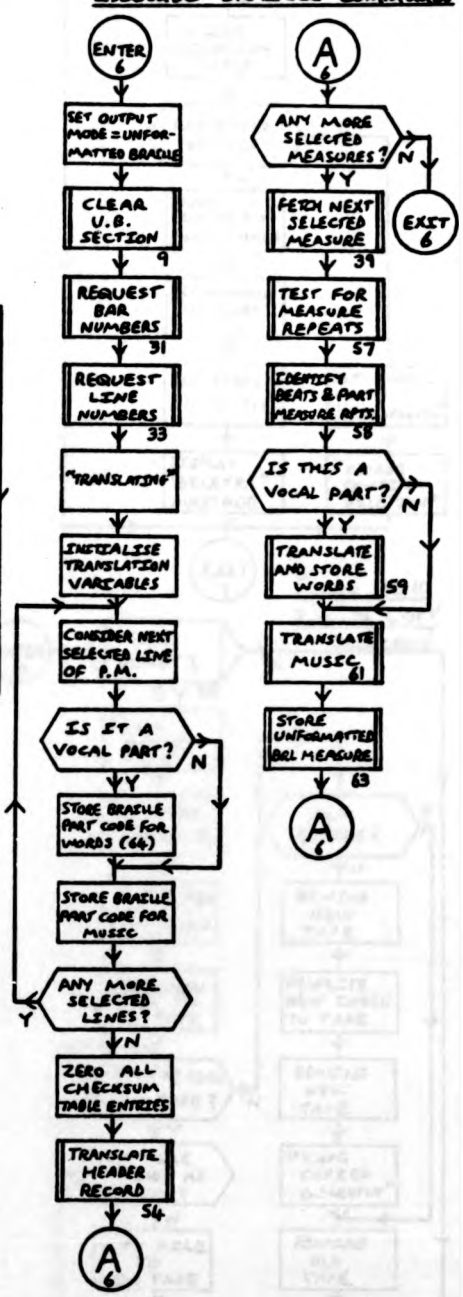


Figure A2:8 Execute CHANGE command

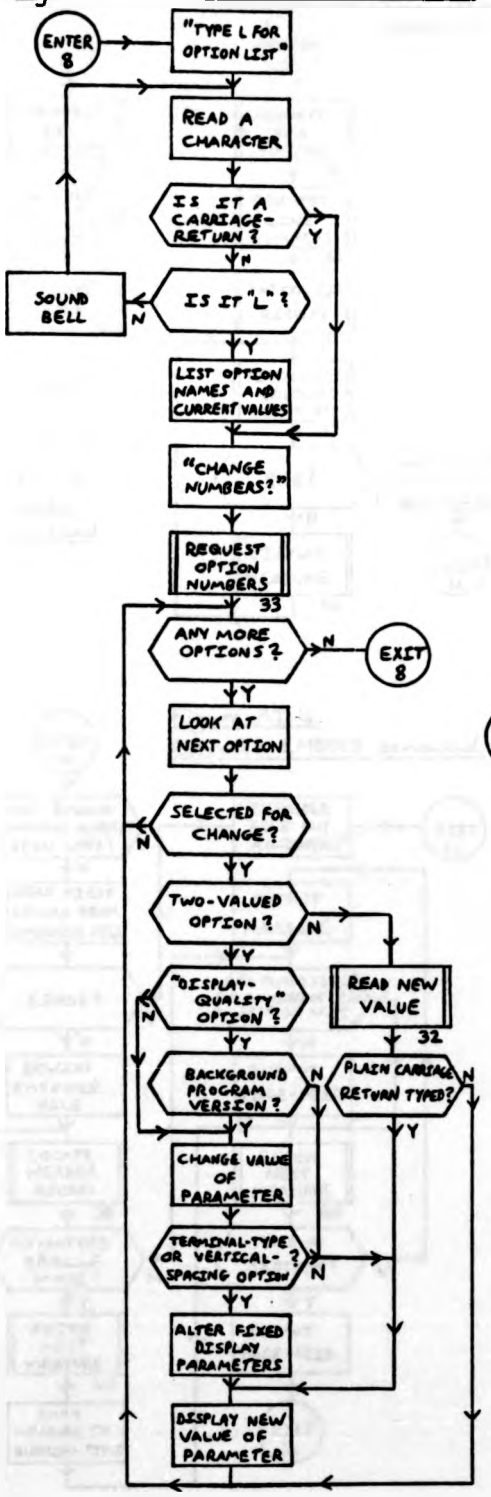


Figure A2:9 Execute CLEAR command

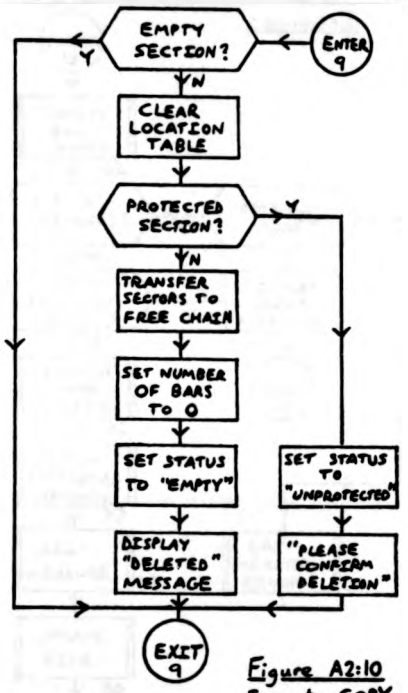
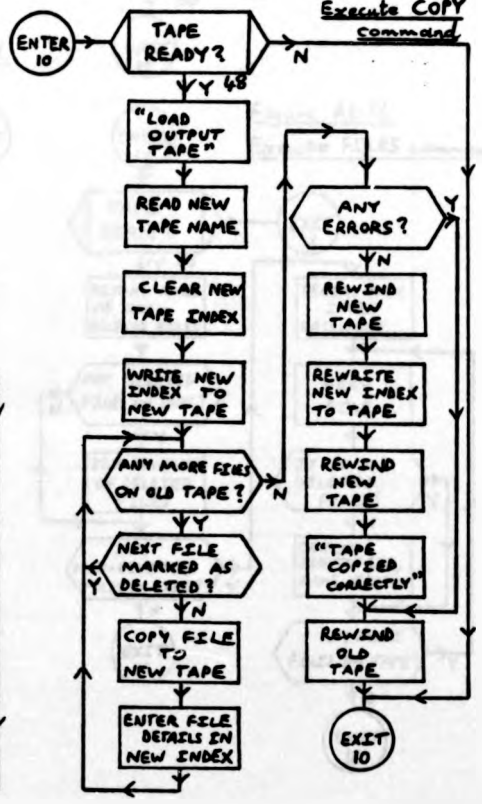


Figure A2:10 Execute COPY command



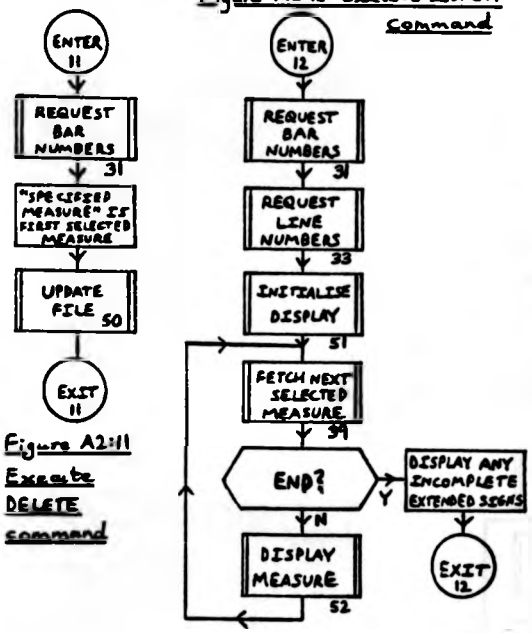


Figure A2:11 Execute DELETE command

Figure A2:12 Execute DISPLAY command

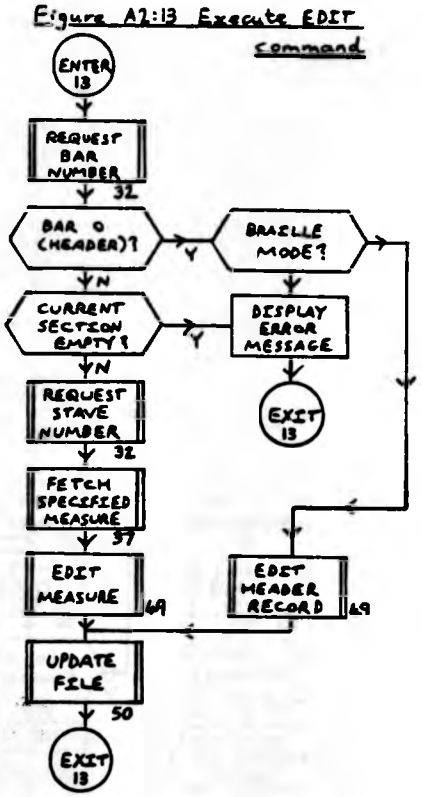


Figure A2:13 Execute EDIT command

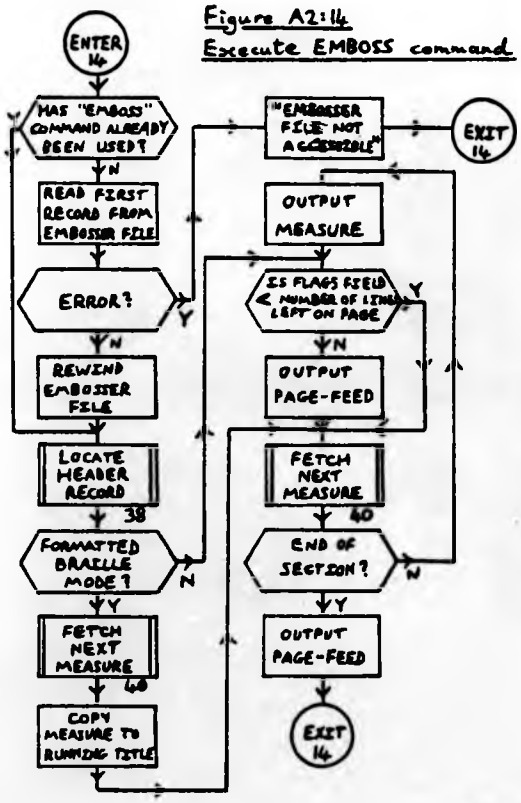


Figure A2:14 Execute EMBOSS command

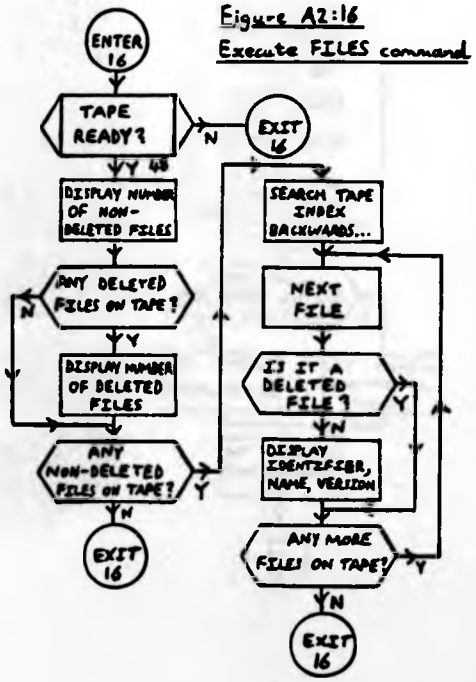


Figure A2:16 Execute FILES command

Figure A2:15
Execute ERASE command

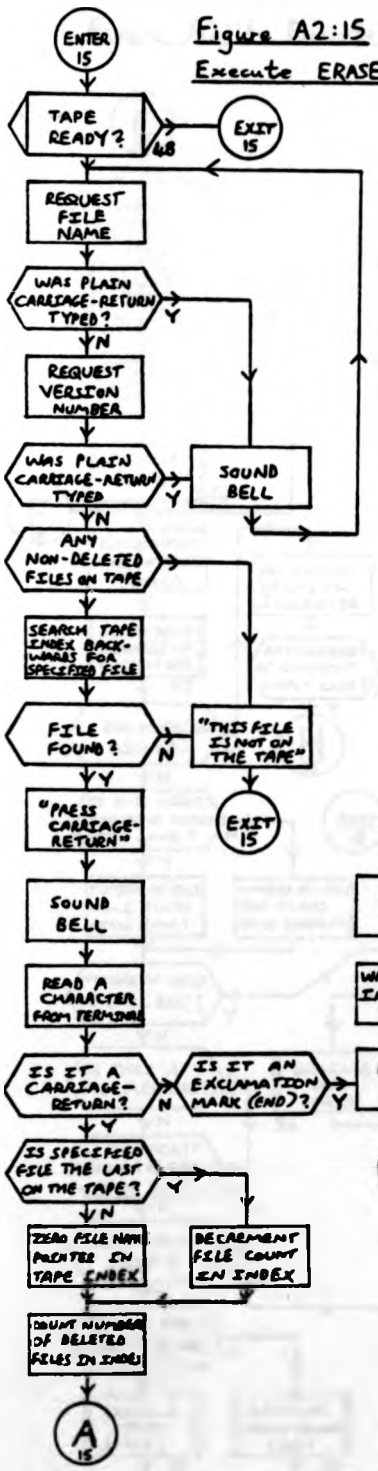


Figure A2:18 Execute FORMAT command

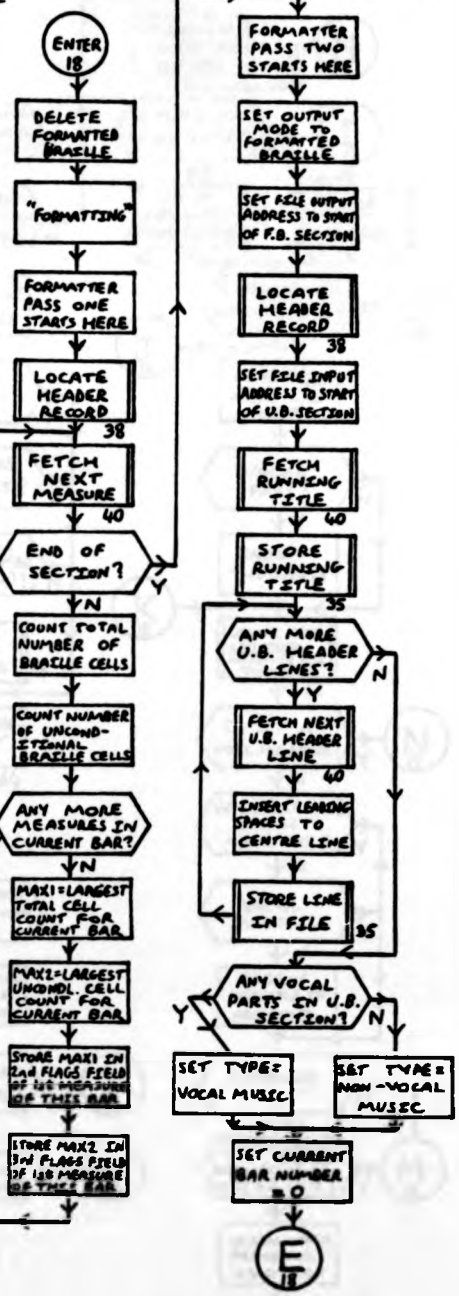


Figure A2:18 Execute FORMAT command (continued)

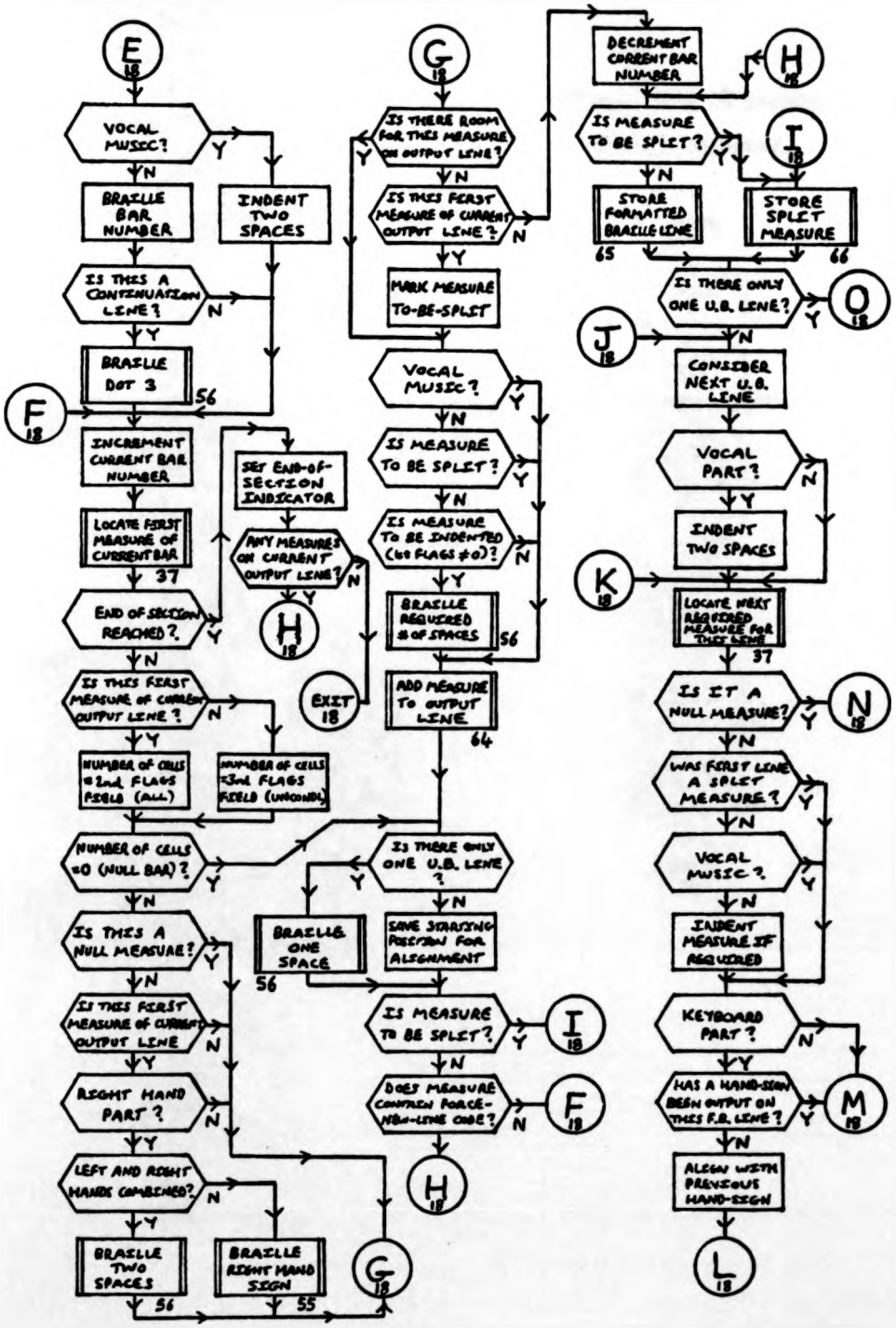


Figure A2:18 Execute FORMAT command (continued)

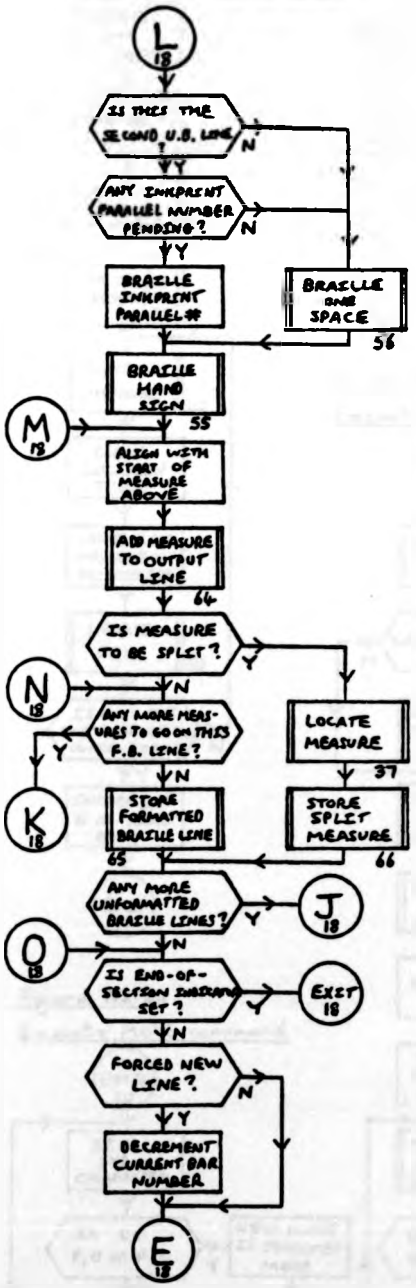


Figure A2:17 Execute FLAG command

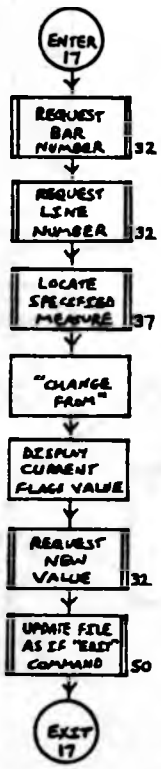


Figure A2:19
Execute INSERT command

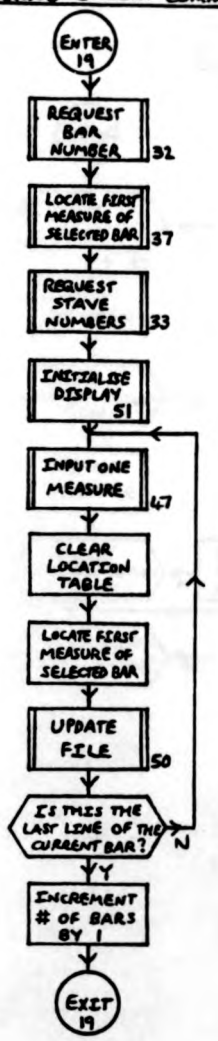


Figure A2:20
Execute MODE command

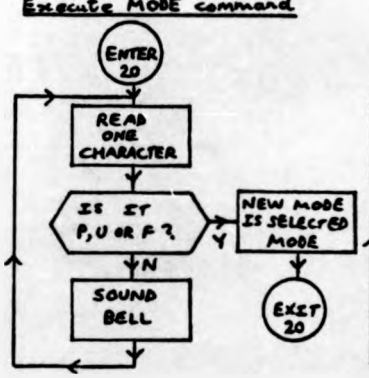


Figure A2:21
Execute OPTIONS command

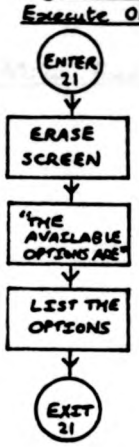


Figure A2:22
Execute PLAY command

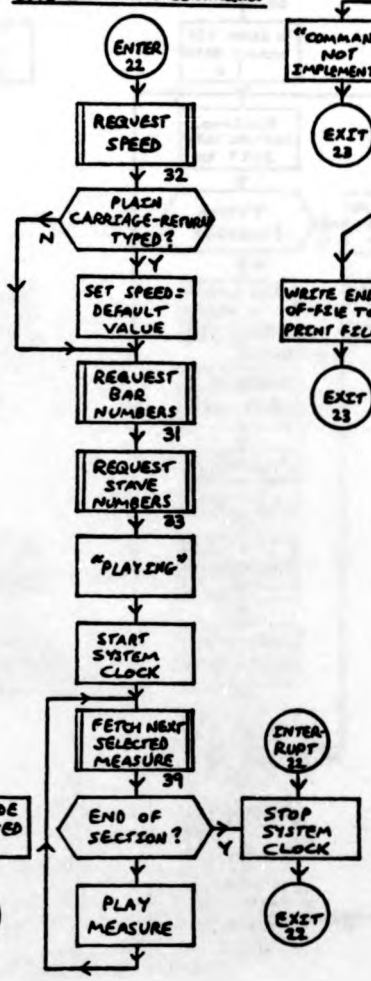


Figure A2:23
Execute PRINT command

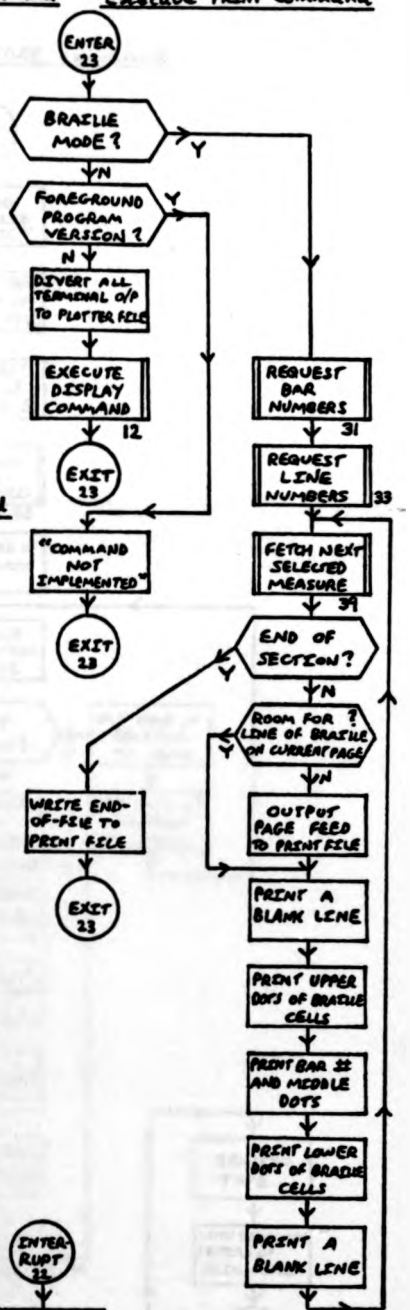


Figure A2:24 Execute RESTORE command

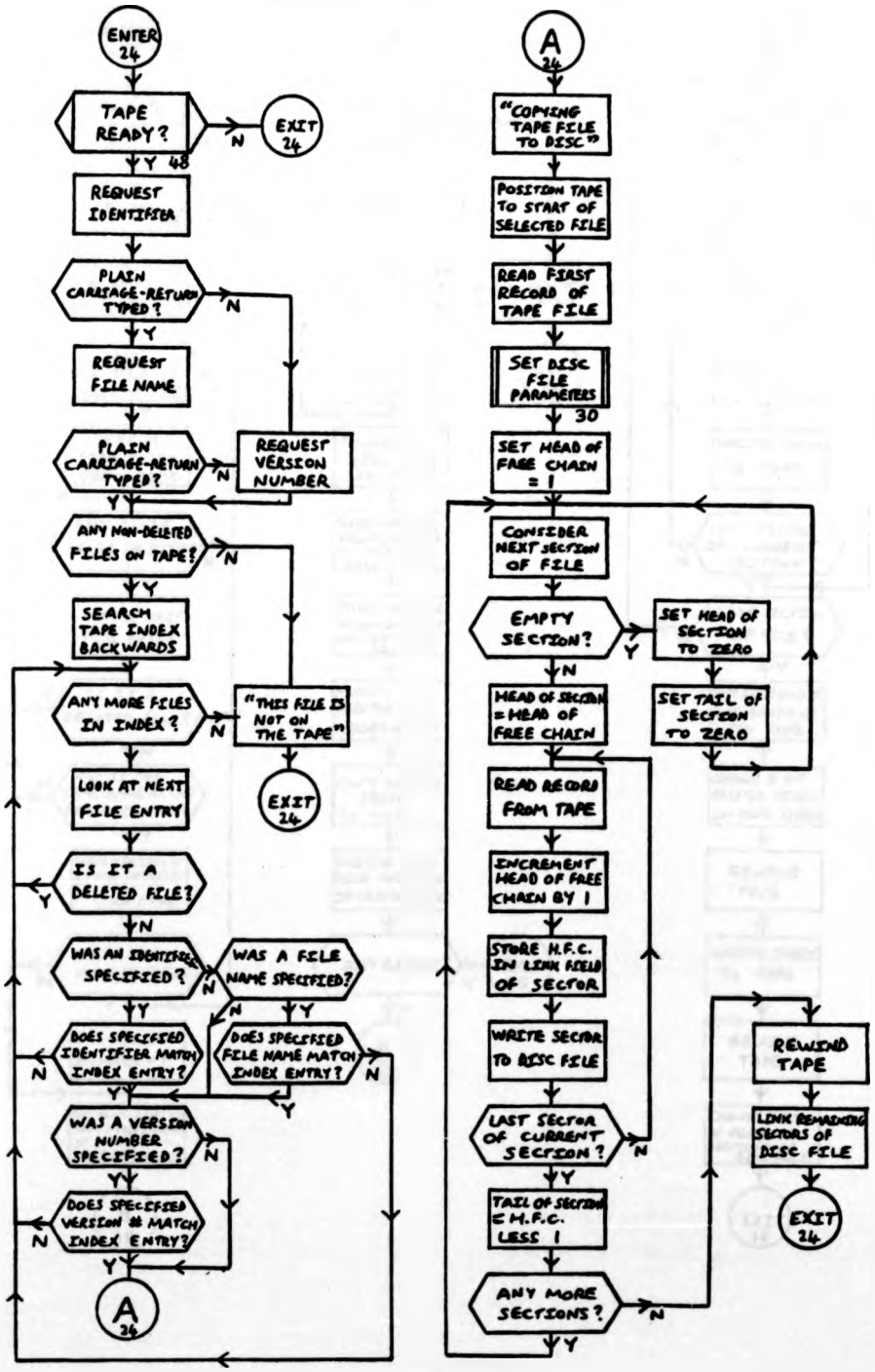


Figure A2:24 Execute RESTORE command

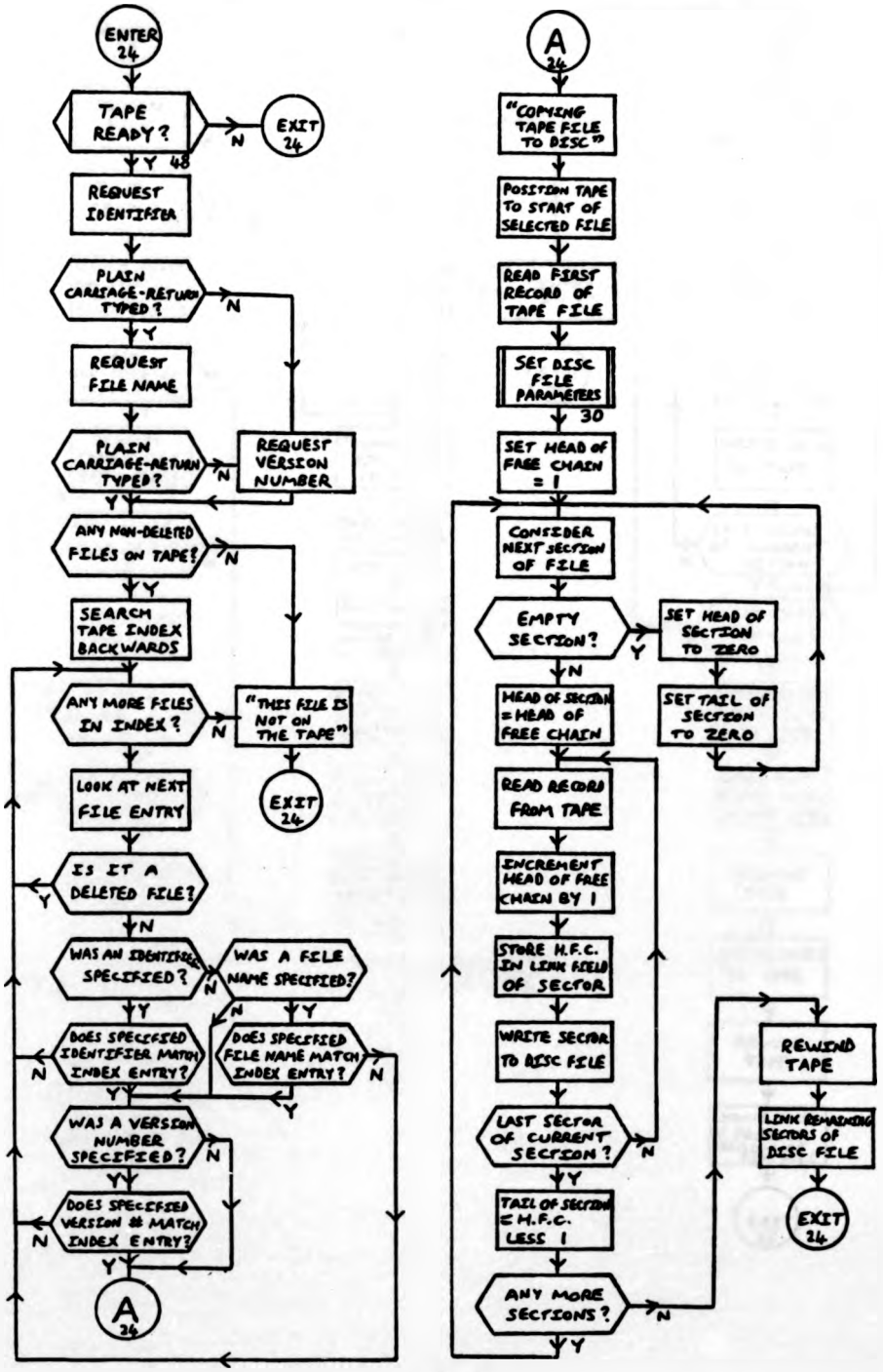


Figure A2:25 Execute SAVE command

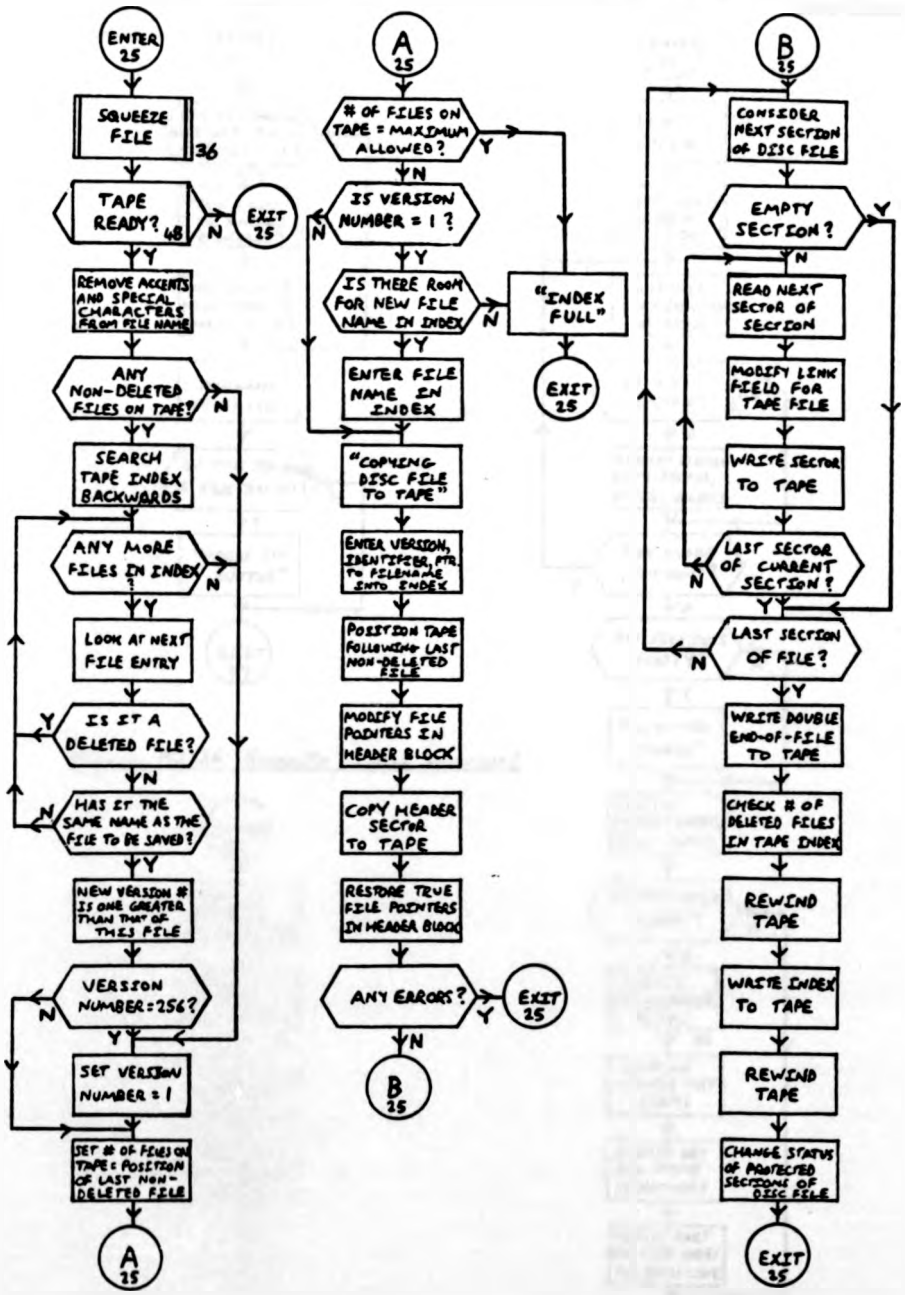


Figure A2:27 Execute STOP command

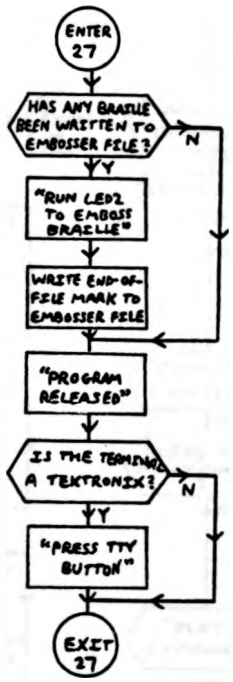


Figure A2:28 Execute SUMMARY command

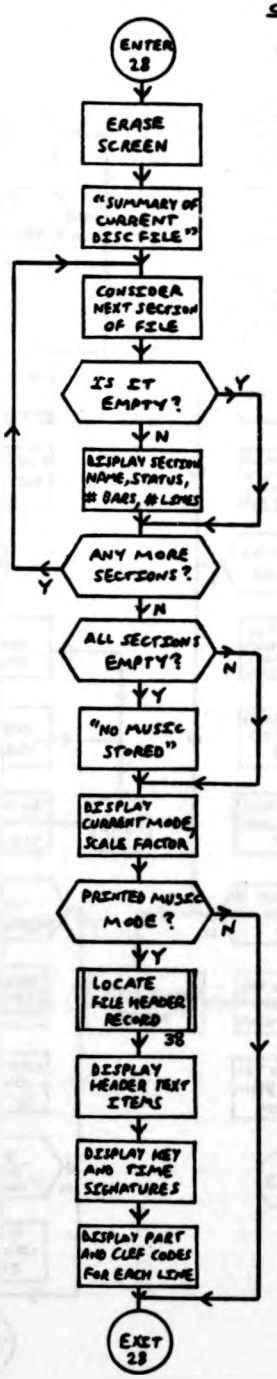


Figure A2:26 Execute SCALE command

Figure A2:29 Command termination

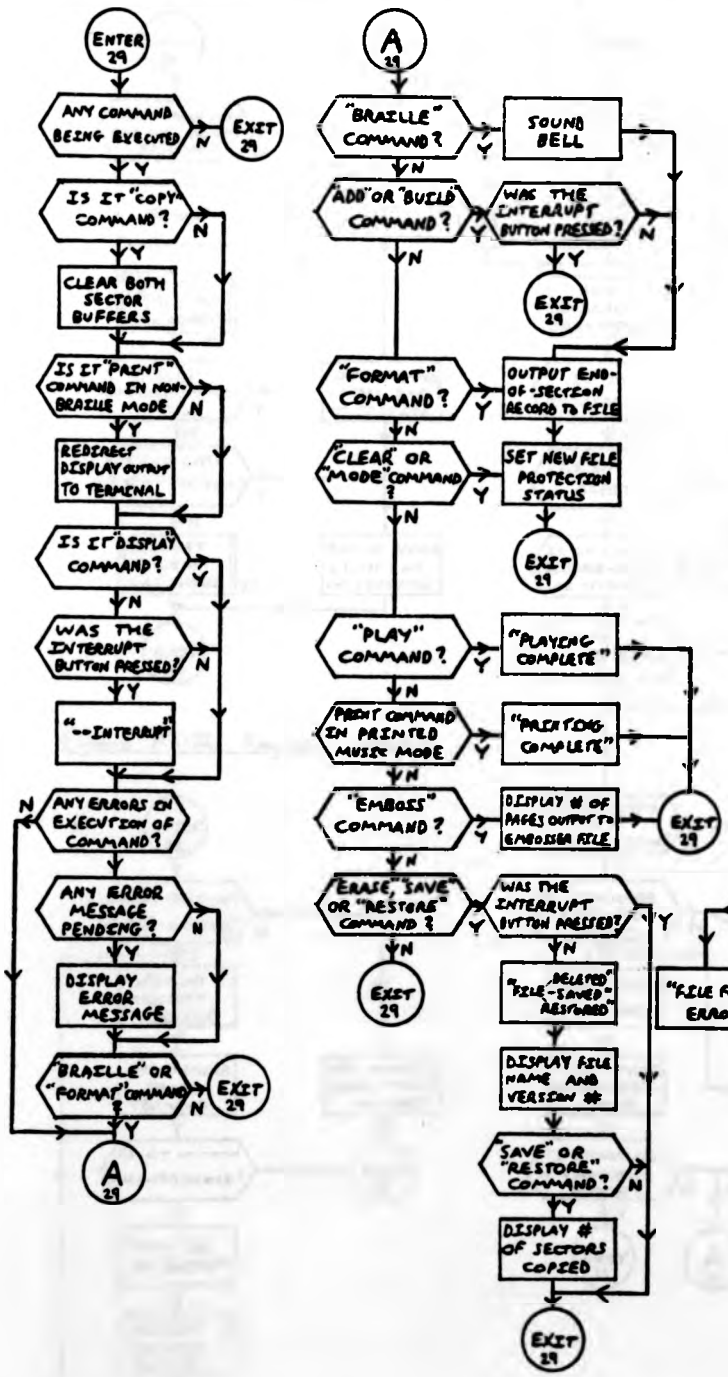


Figure A2:30 Set disc file parameters

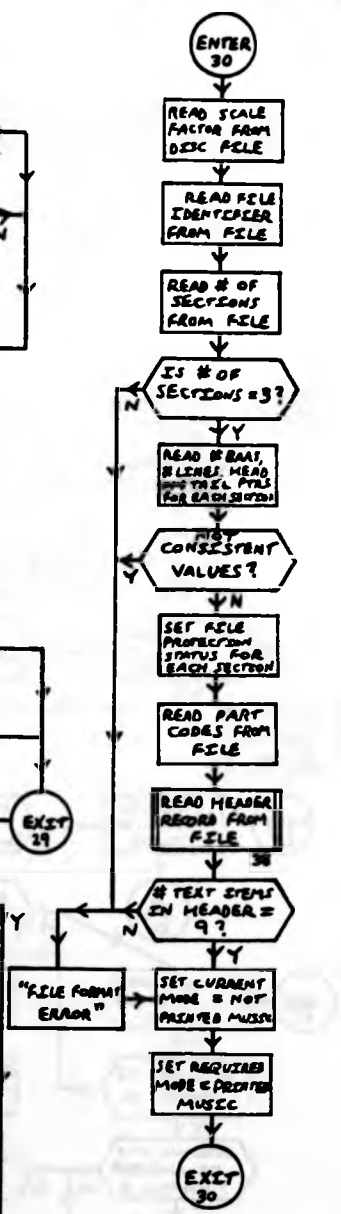


Figure A2:31 Request bar numbers

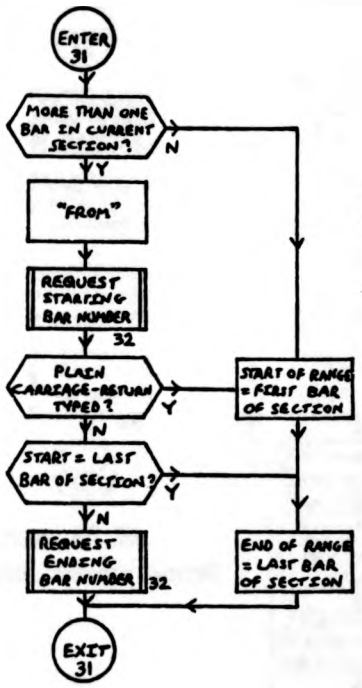


Figure A2:32 Request number

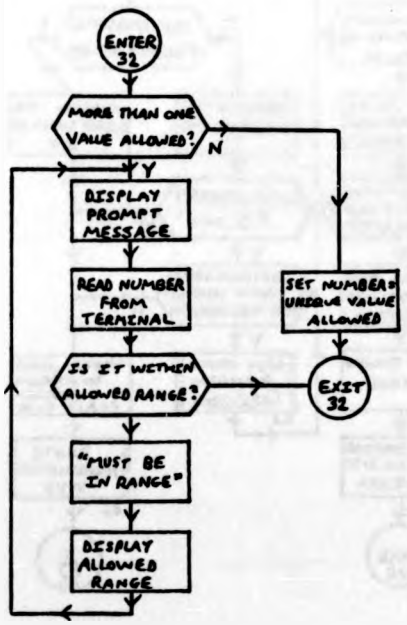


Figure A2:33 Request line numbers

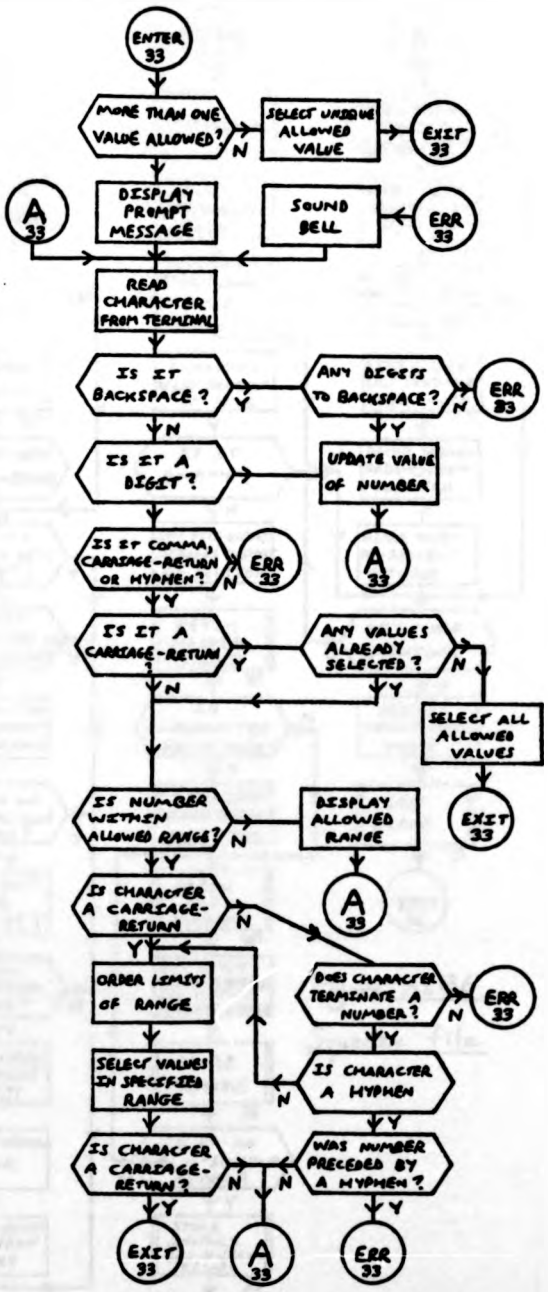


Figure A2:35 Store measure

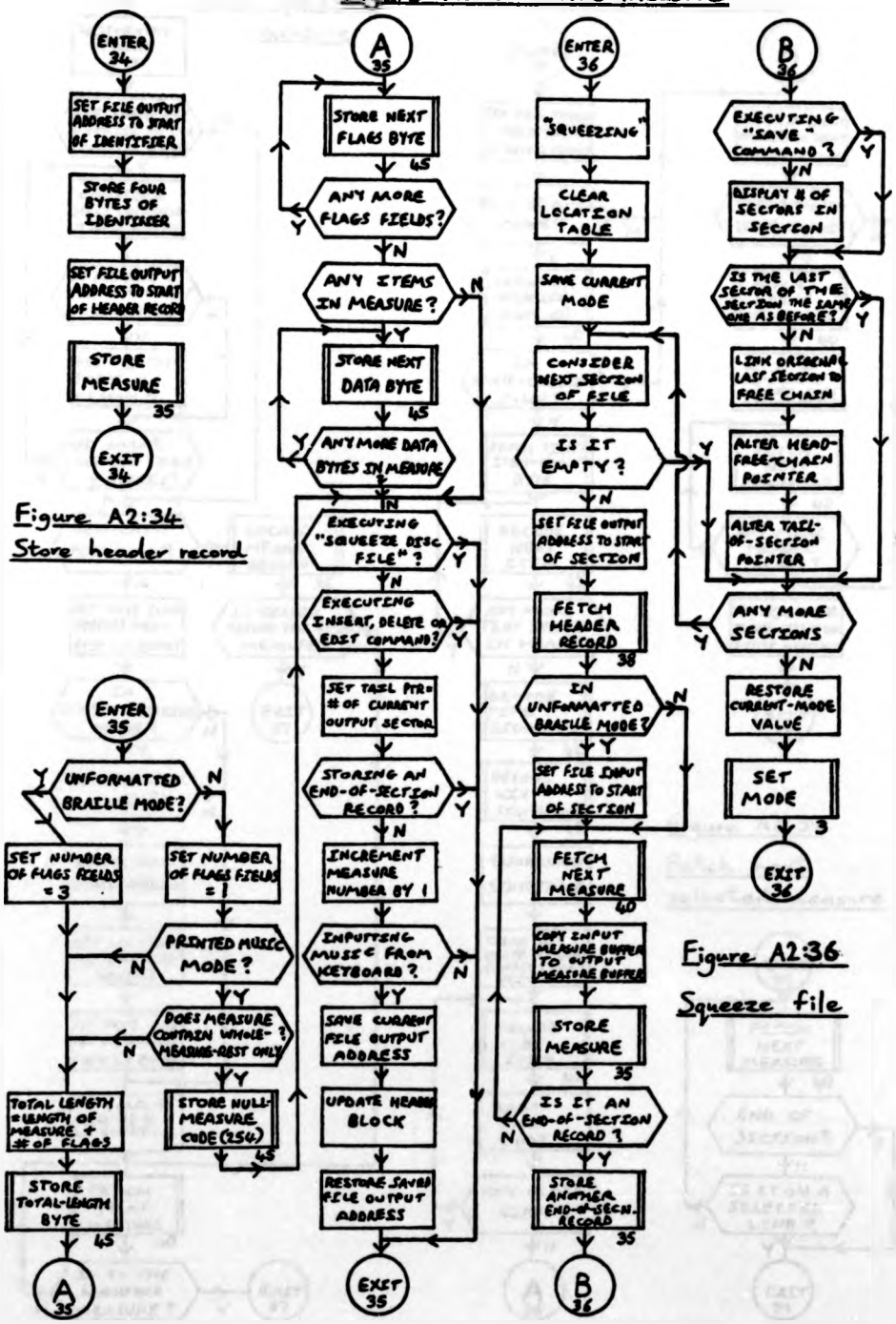


Figure A2:34
Store header record

Figure A2:36
Squeeze file

Figure A2:37
Locate specified measure

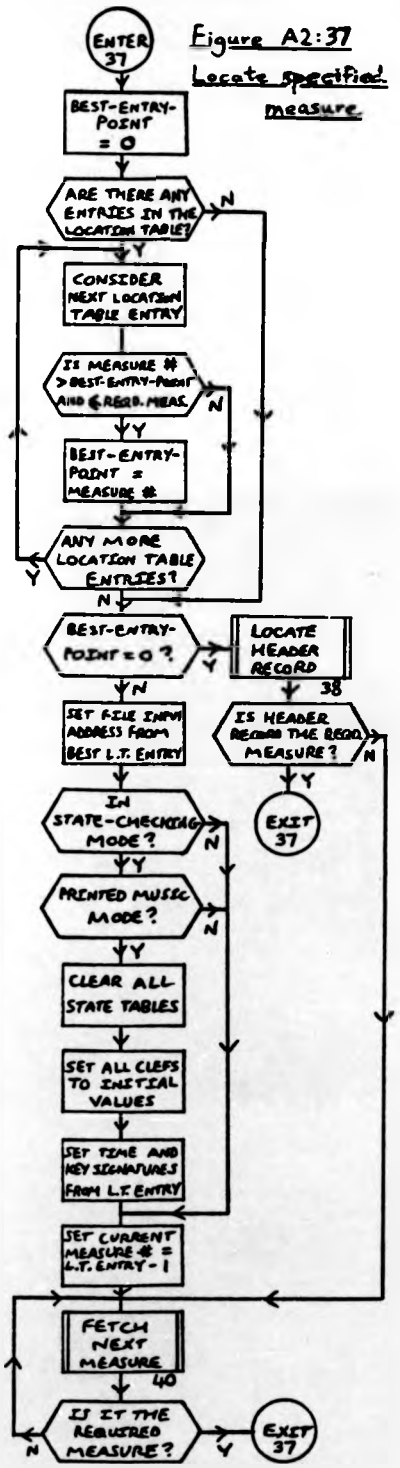


Figure A2:38
Locate header record

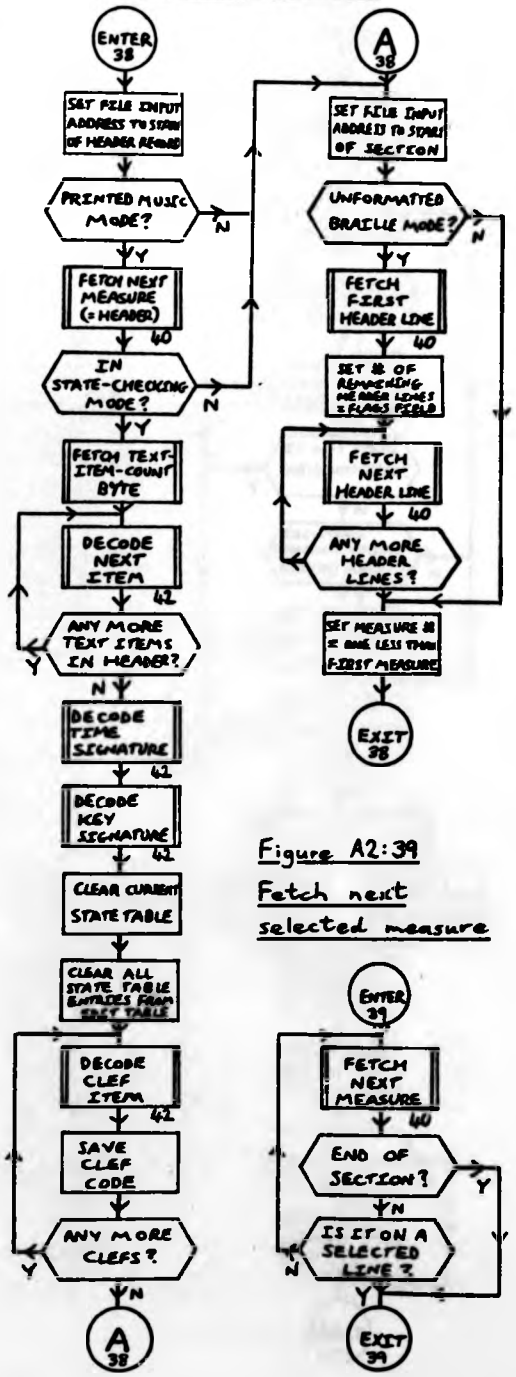


Figure A2:39
Fetch next selected measure

Figure A2:40 Fetch next measure

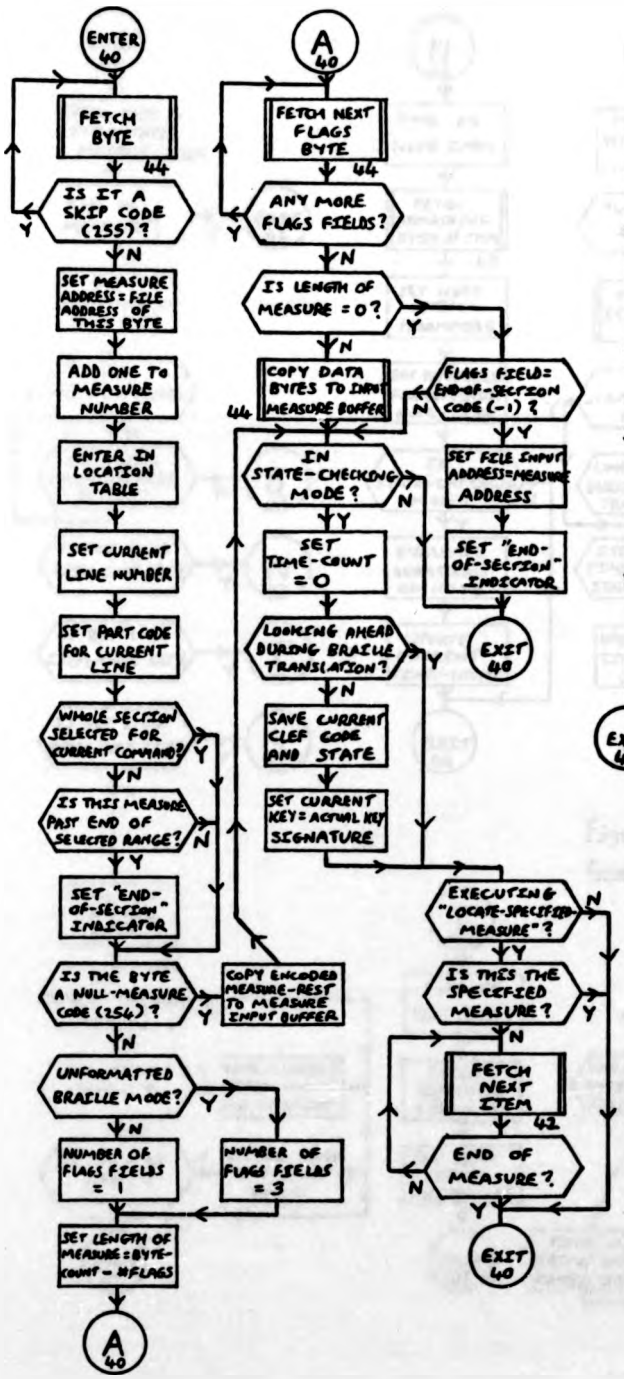


Figure A2:41
Enter measure in location table

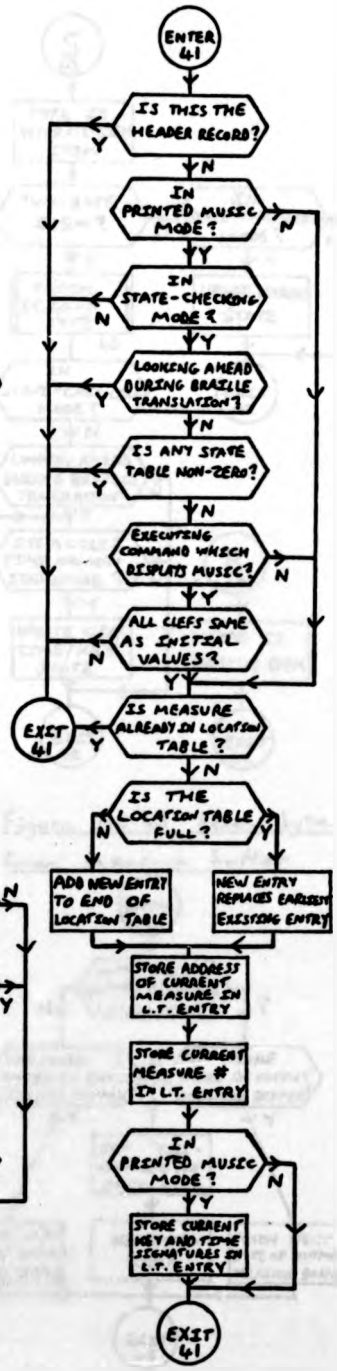


Figure A2:42 Decode item

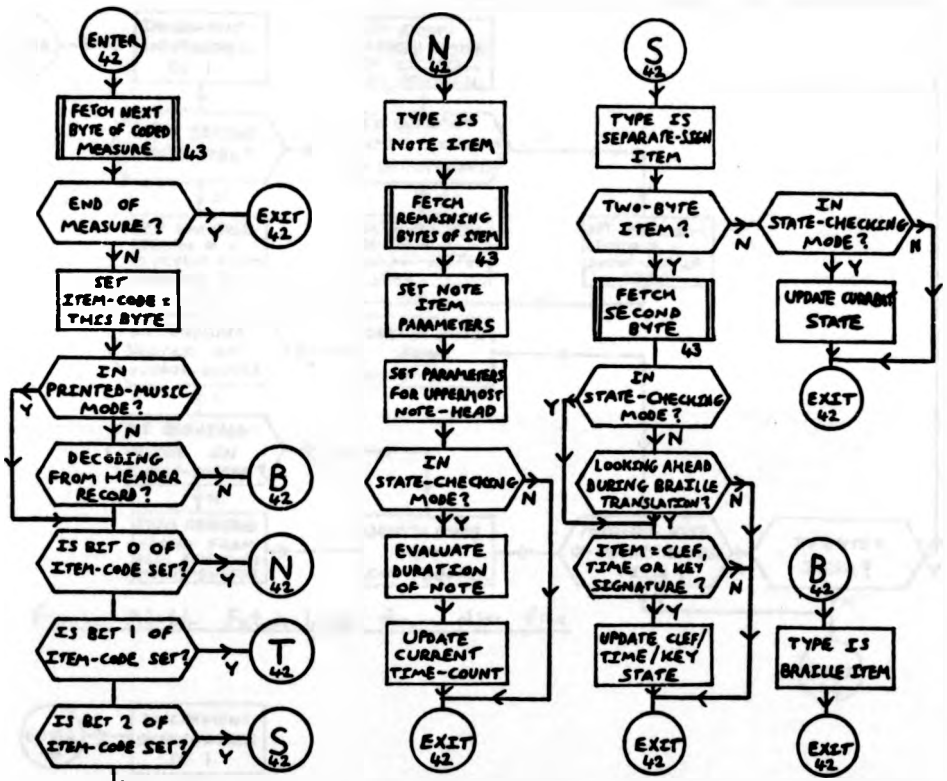
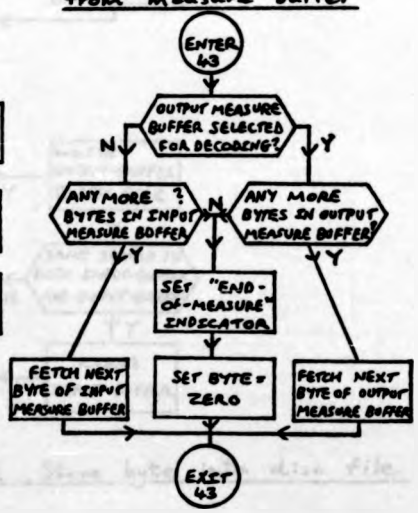


Figure A2:43 Fetch byte from measure buffer



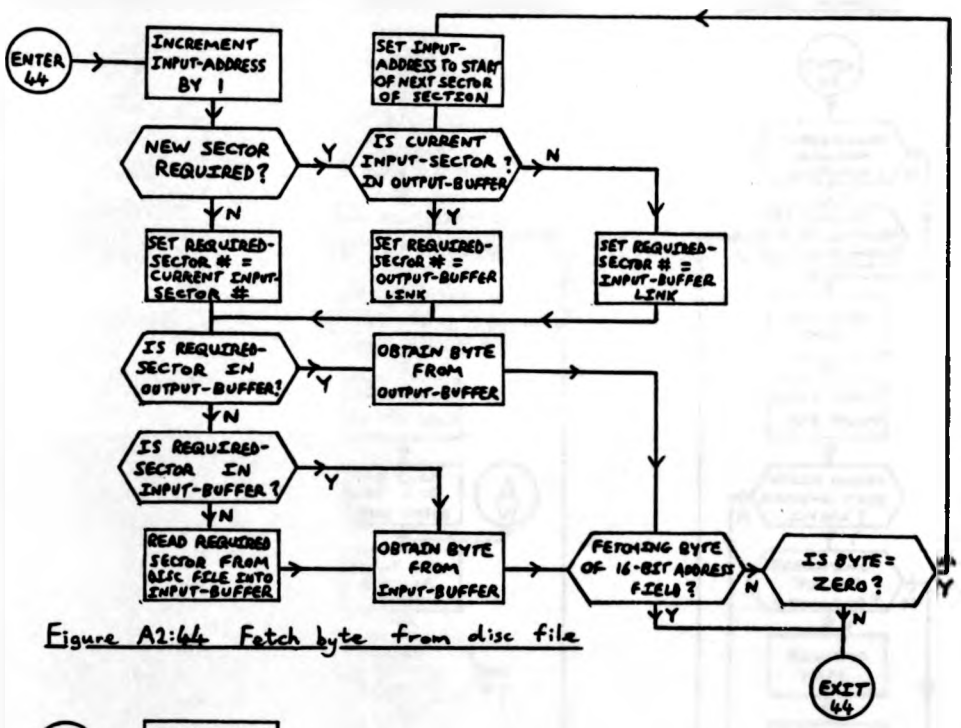


Figure A2:64 Fetch byte from disc file

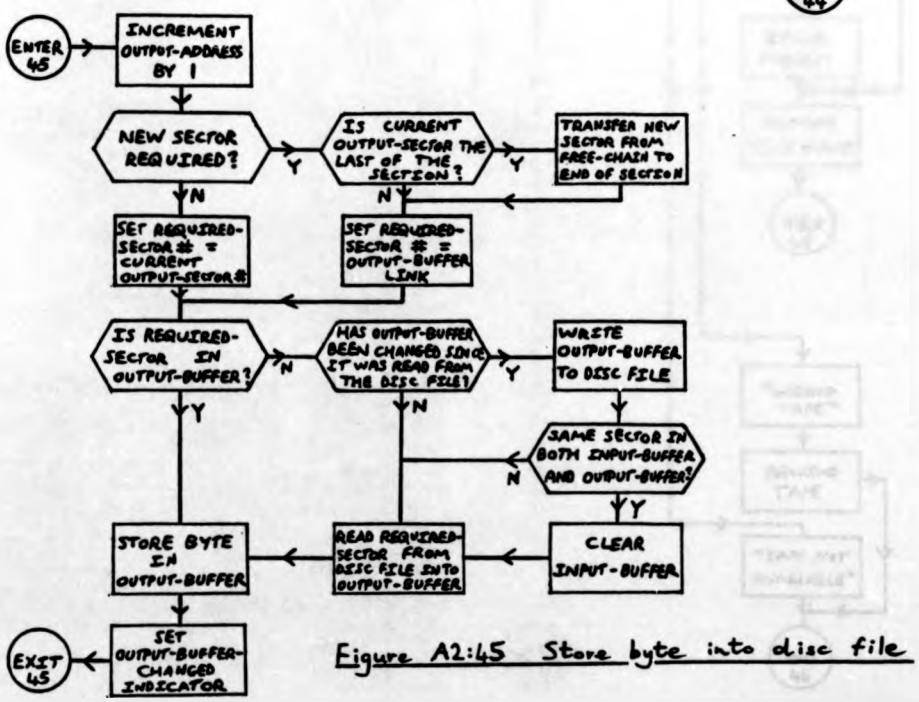


Figure A2:45 Store byte into disc file

Figure A2:46
Input header record

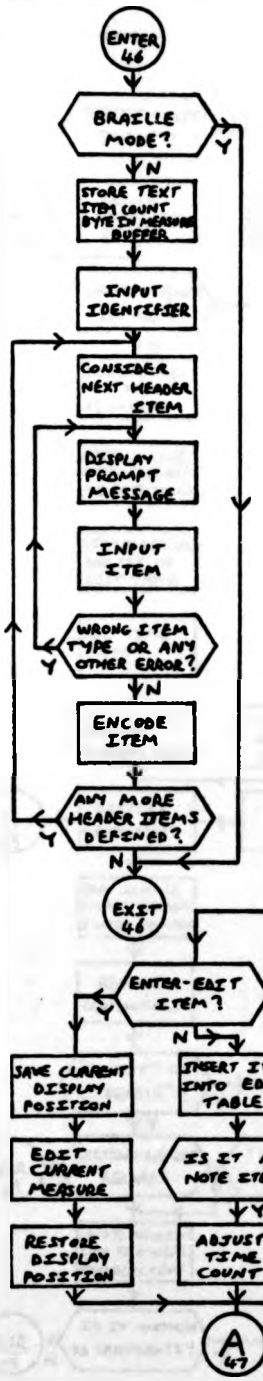


Figure A2:47
Input measure

Figure A2:48
Tape ready?

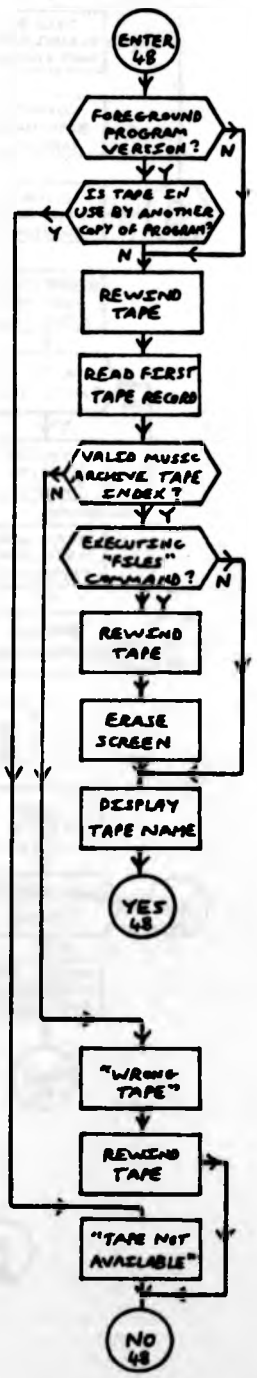


Figure A2:49 Edit measure

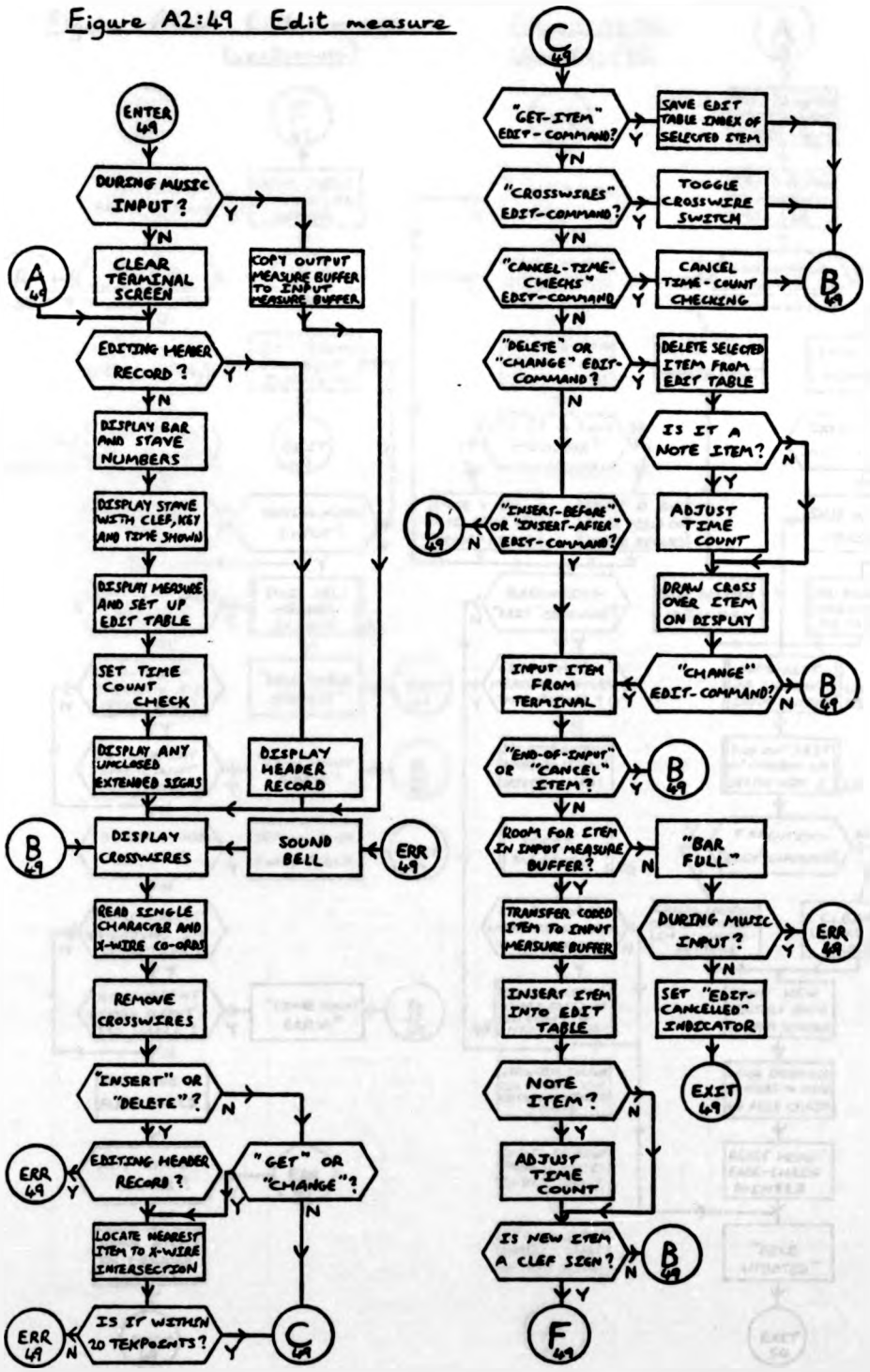


Figure A2:49 Edit measure (continued)

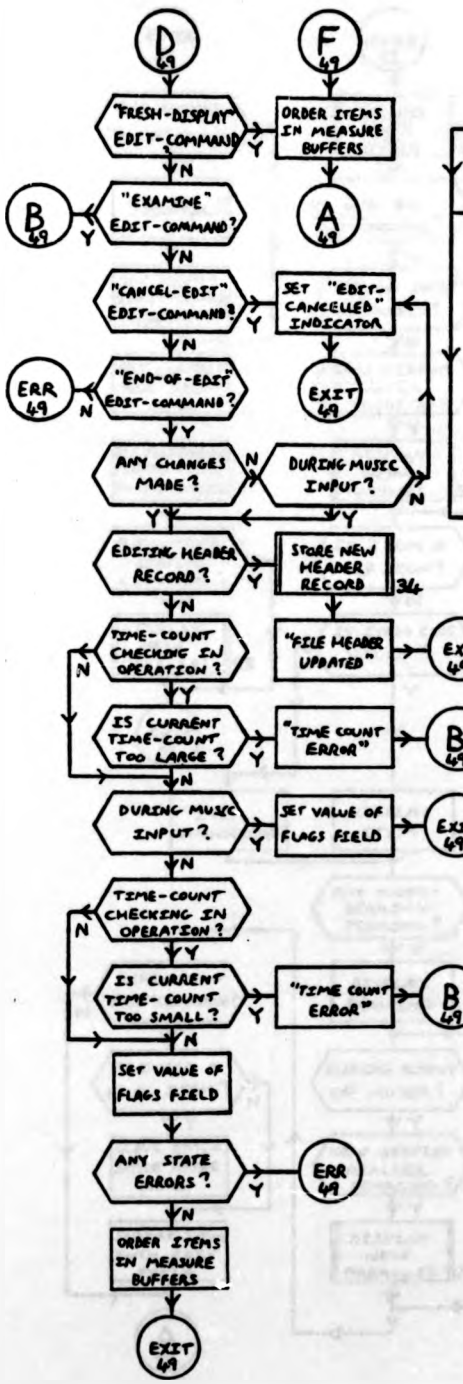


Figure A2:50 Update File

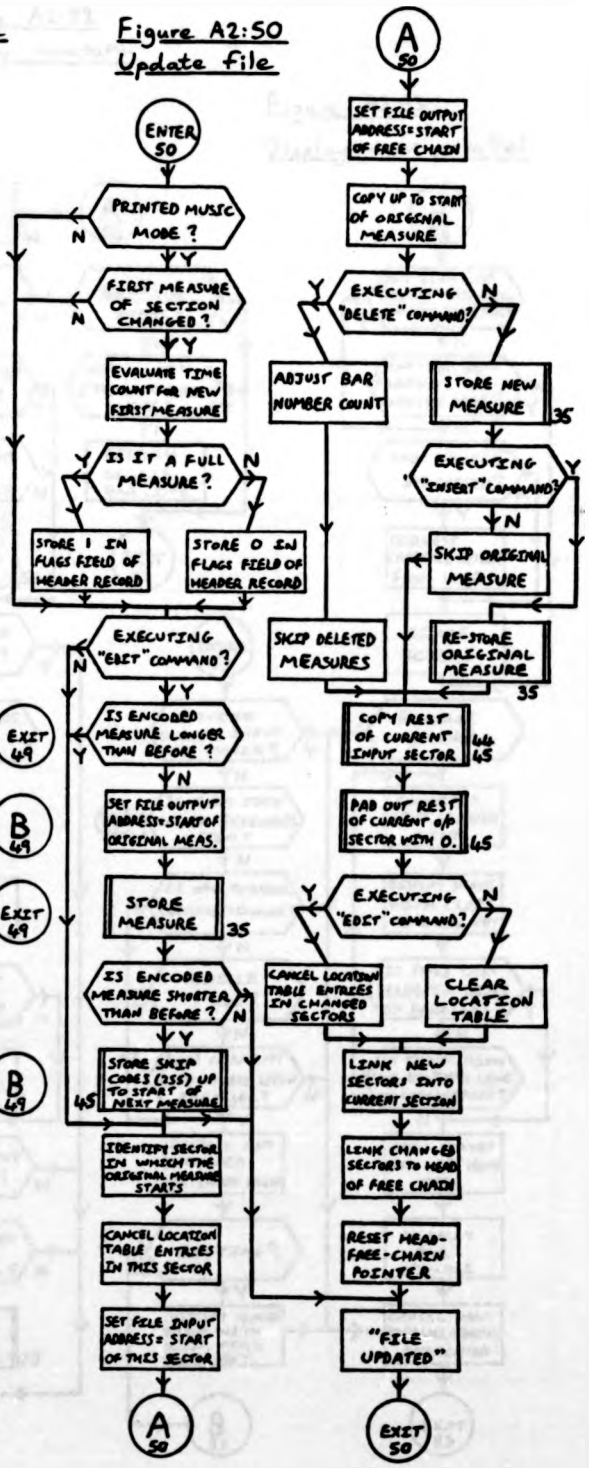
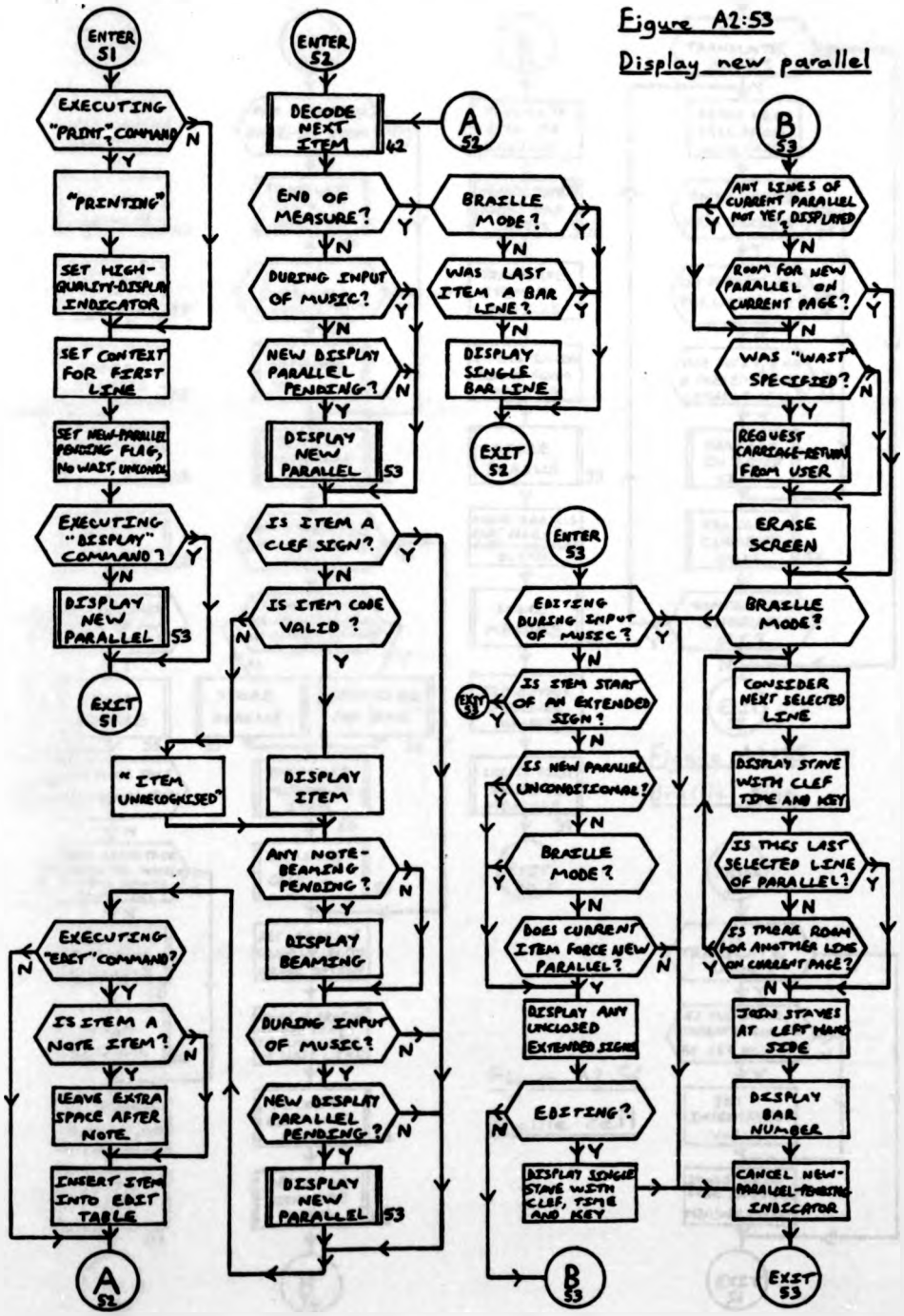


Figure A2:51
Initialise display

Figure A2:52
Display measure

Figure A2:53
Display new parallel



BRAILLE TRANSLATION

Figure A2:54 Translate header record

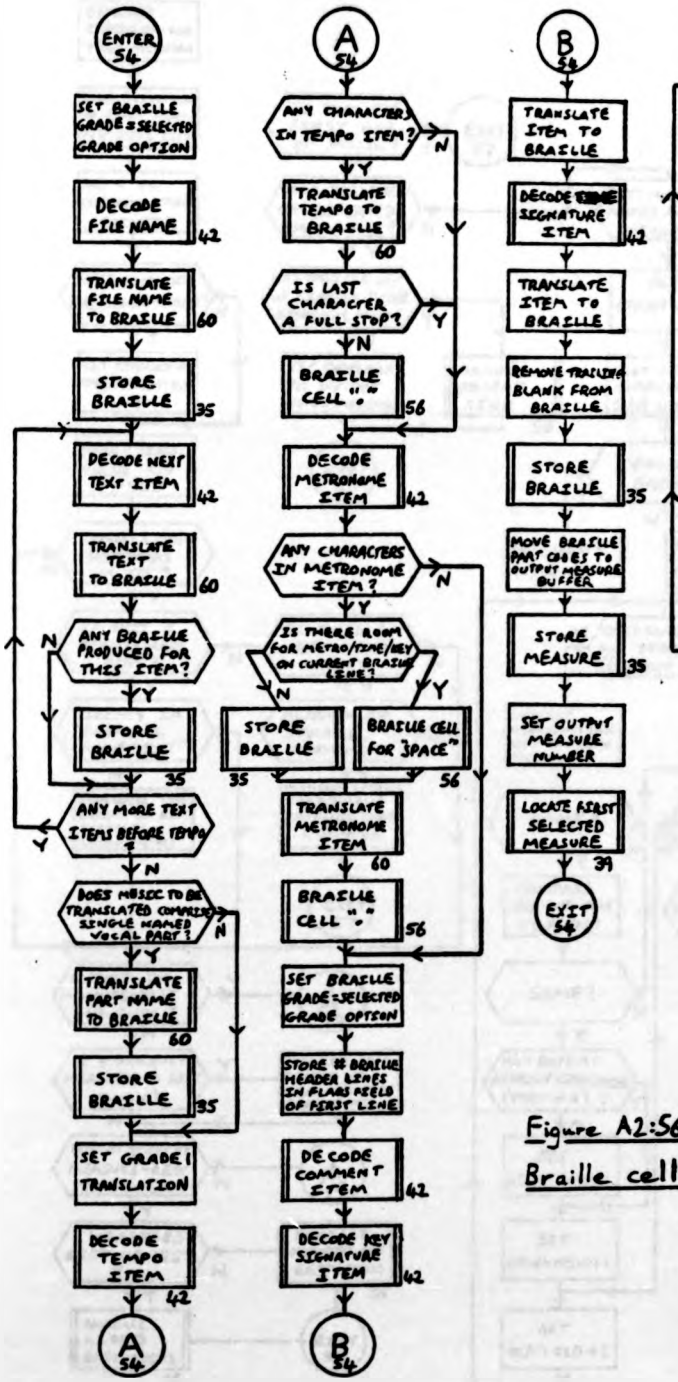


Figure A2:55 Braille sign

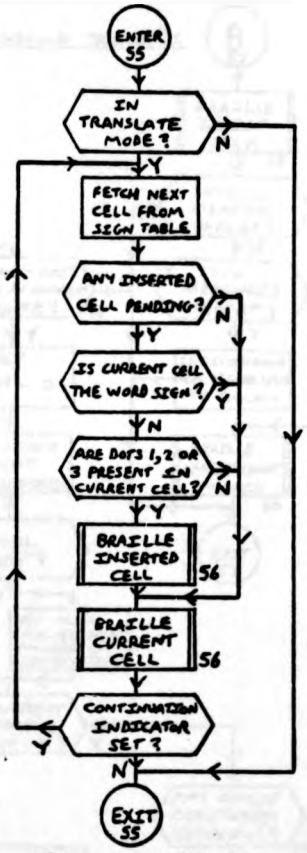


Figure A2:56 Braille cell

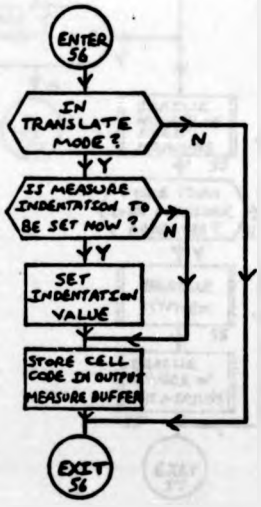


Figure A2:57 Test for measure repeats

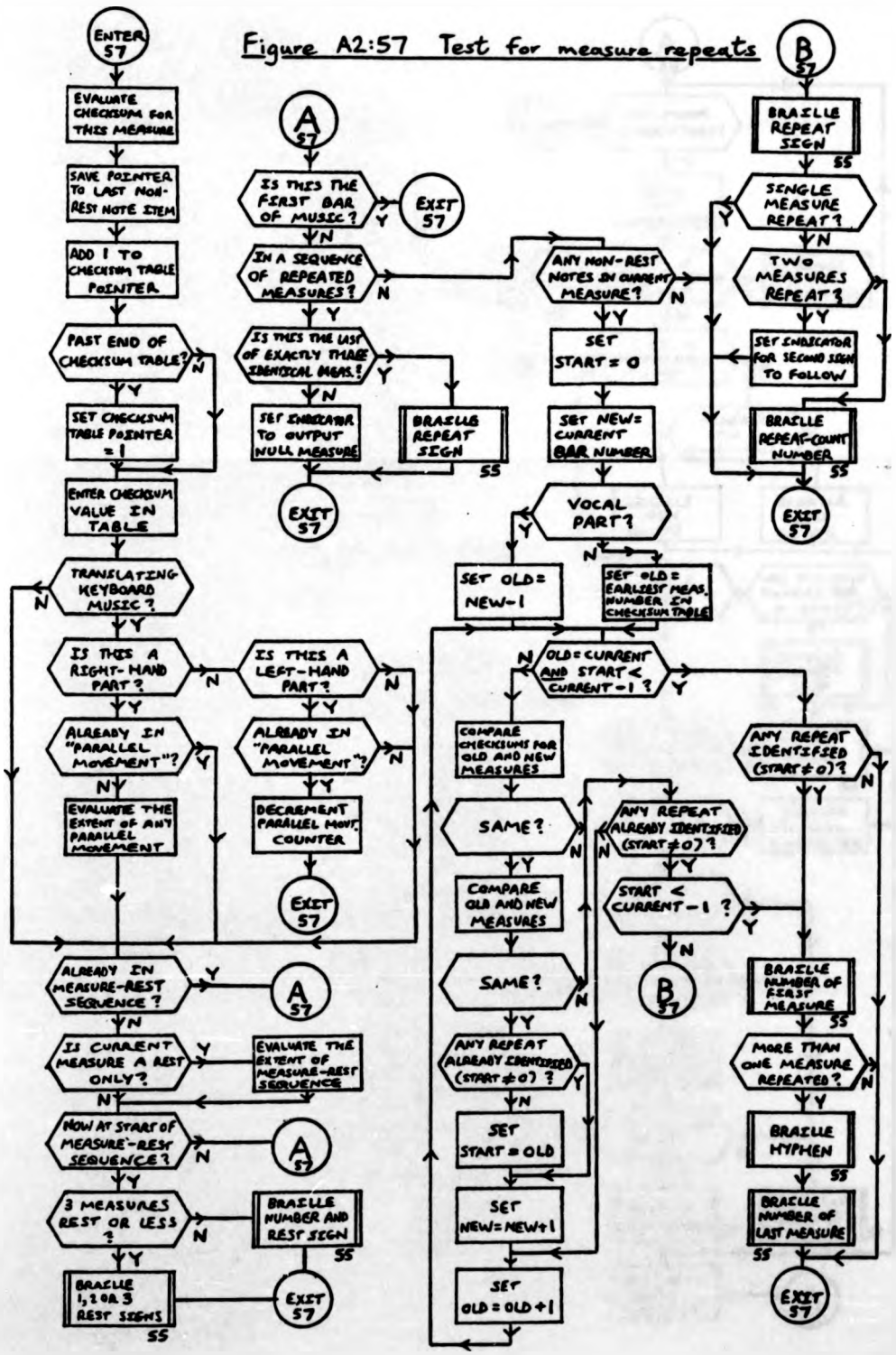


Figure A2:57 Test for measure repeats

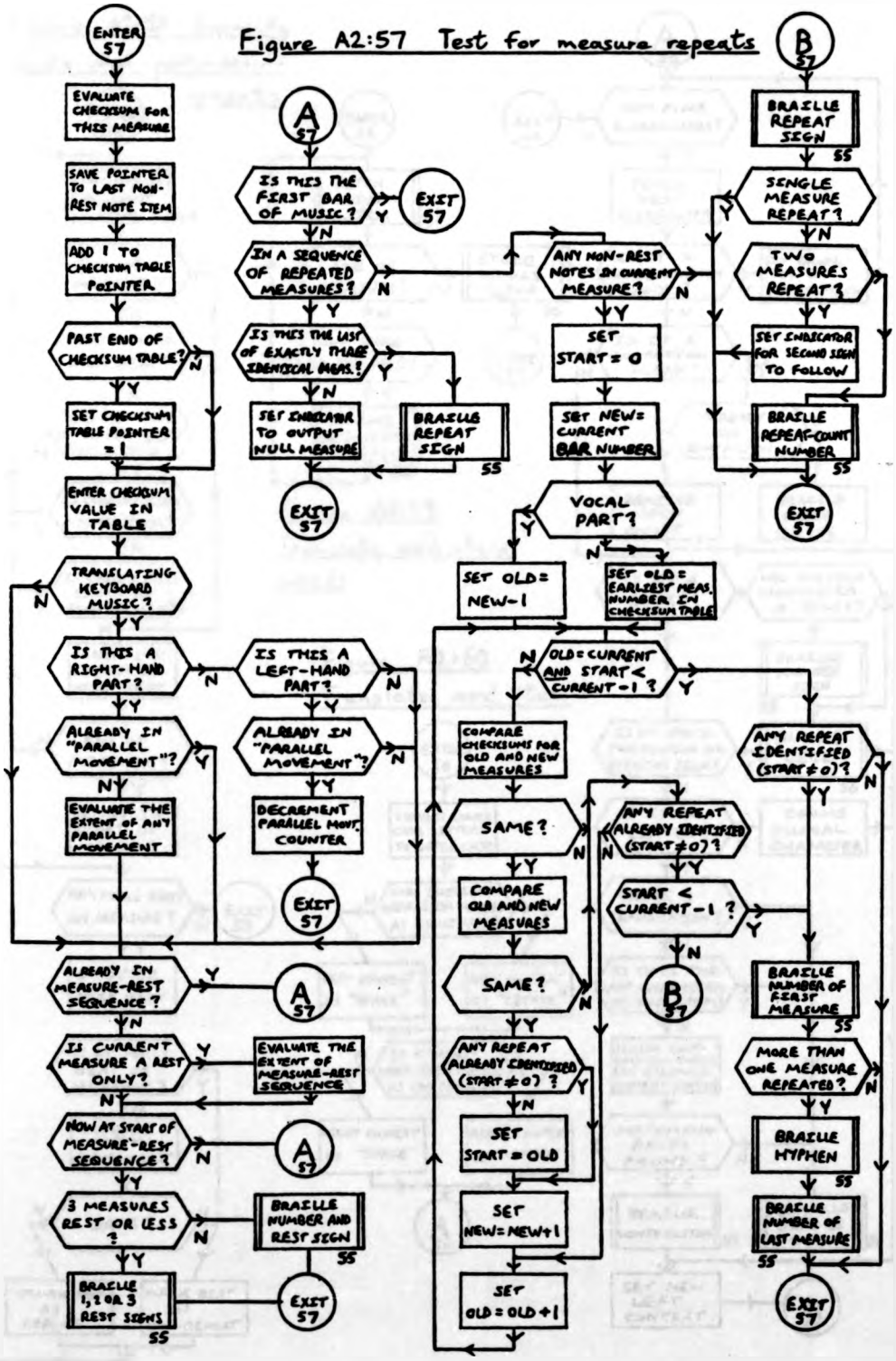
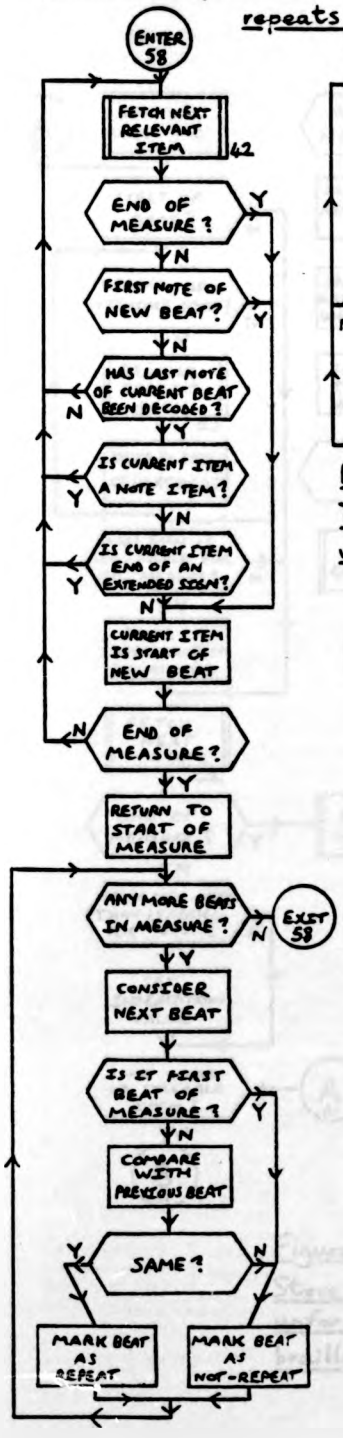


Figure A2:58 Identify beats and part-measure repeats



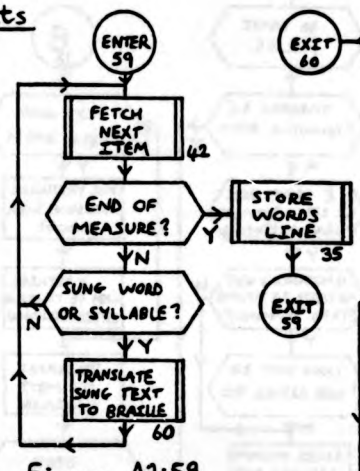
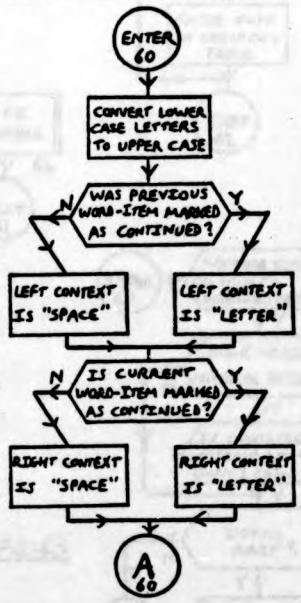


Figure A2:59 Translate and store words

Figure A2:60 Translate word item



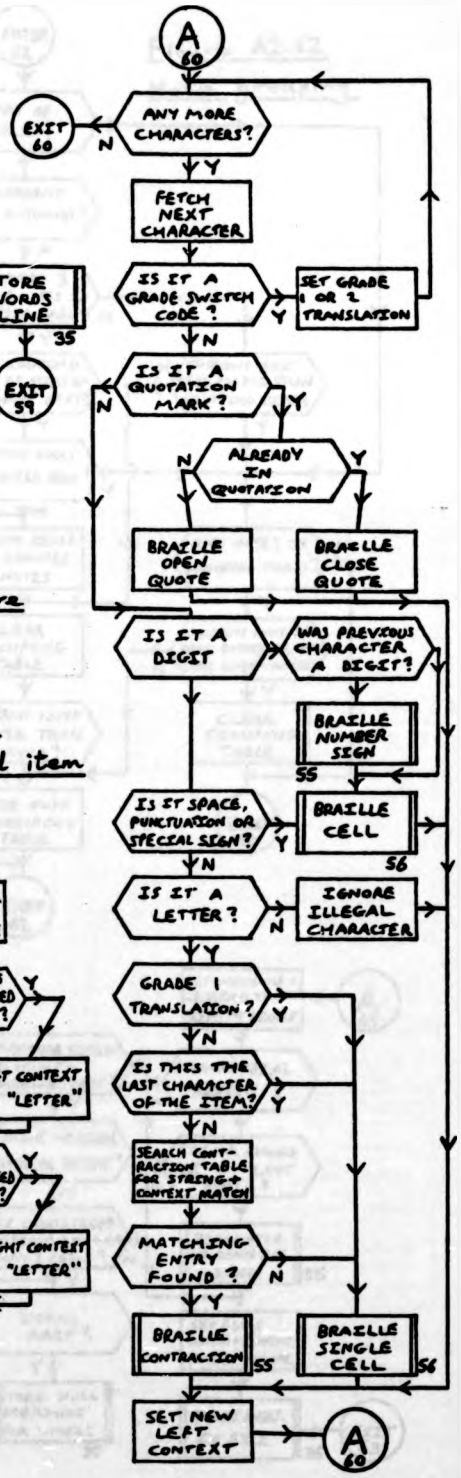


Figure A2:61 Translate music

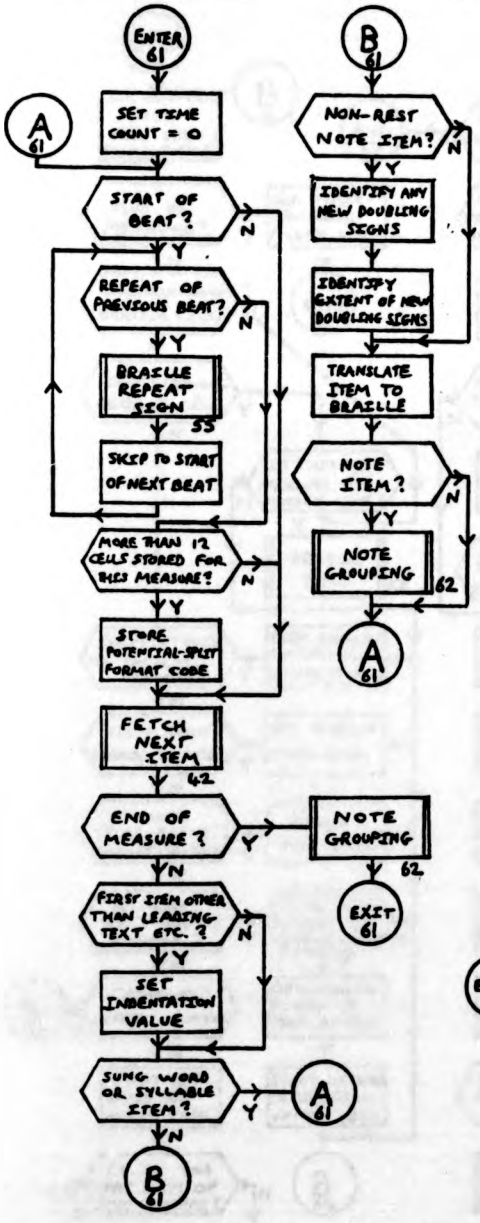


Figure A2:62 Note grouping

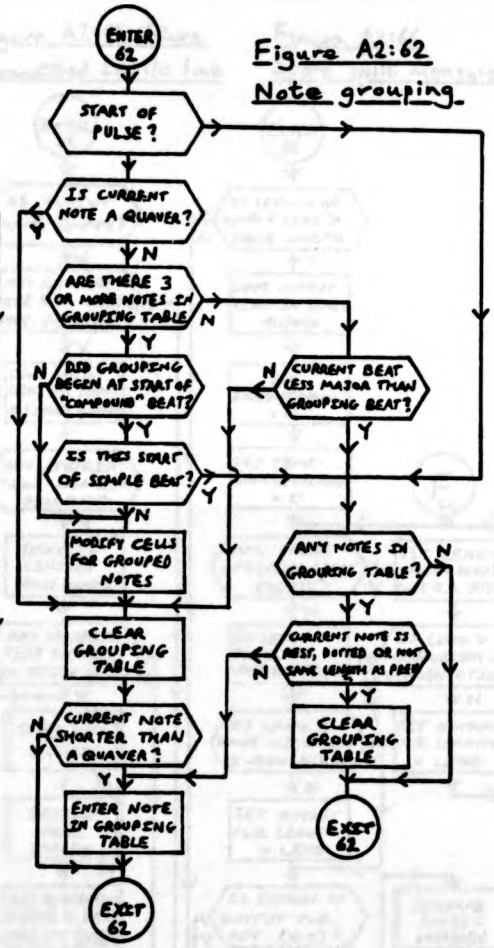


Figure A2:63 Store unformatted braille measure

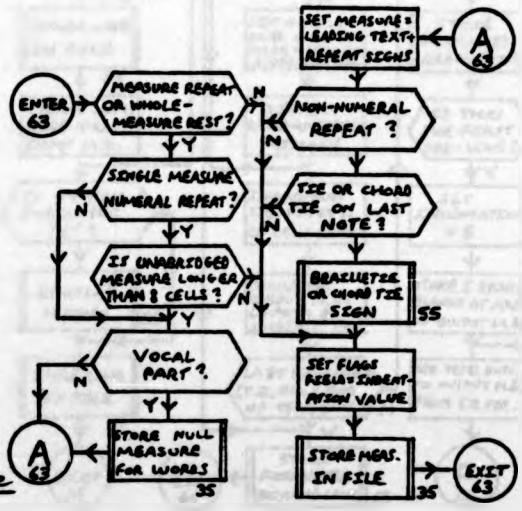


Figure A2:64 Add measure to output line

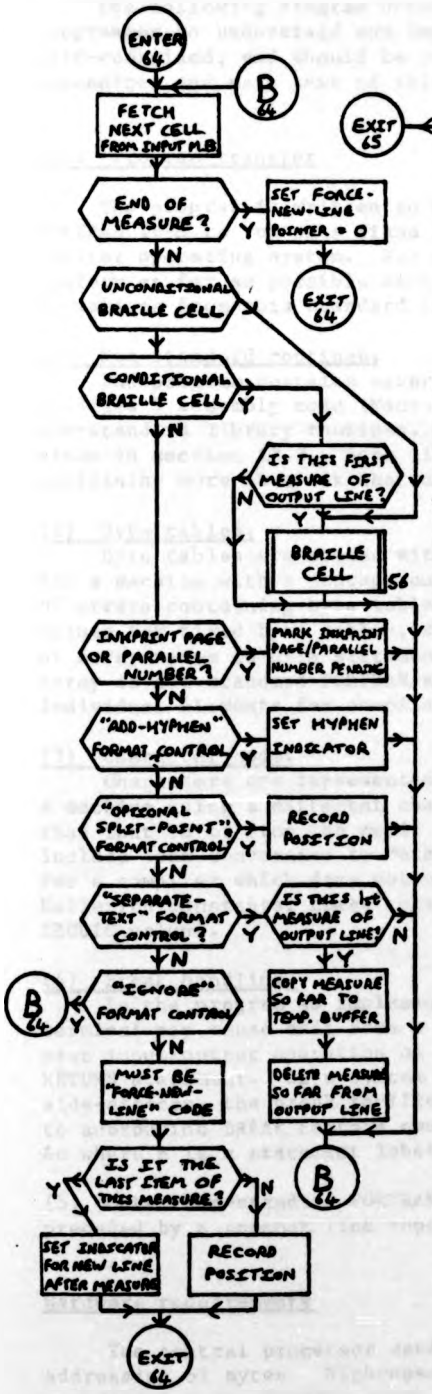


Figure A2:65 Store formatted braille line

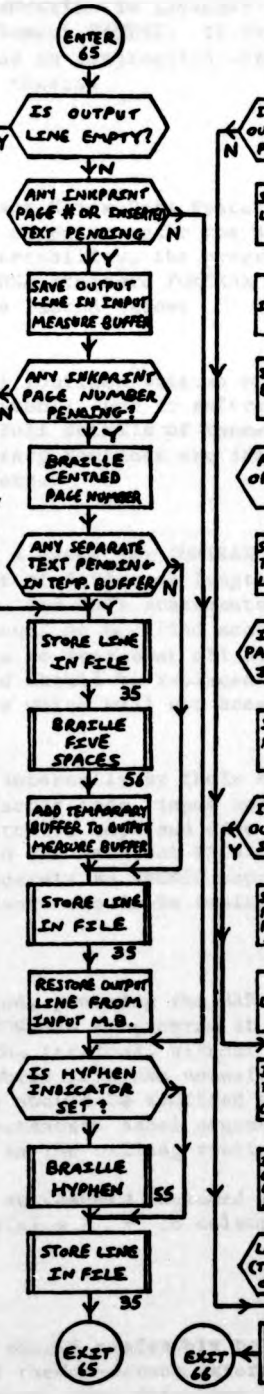
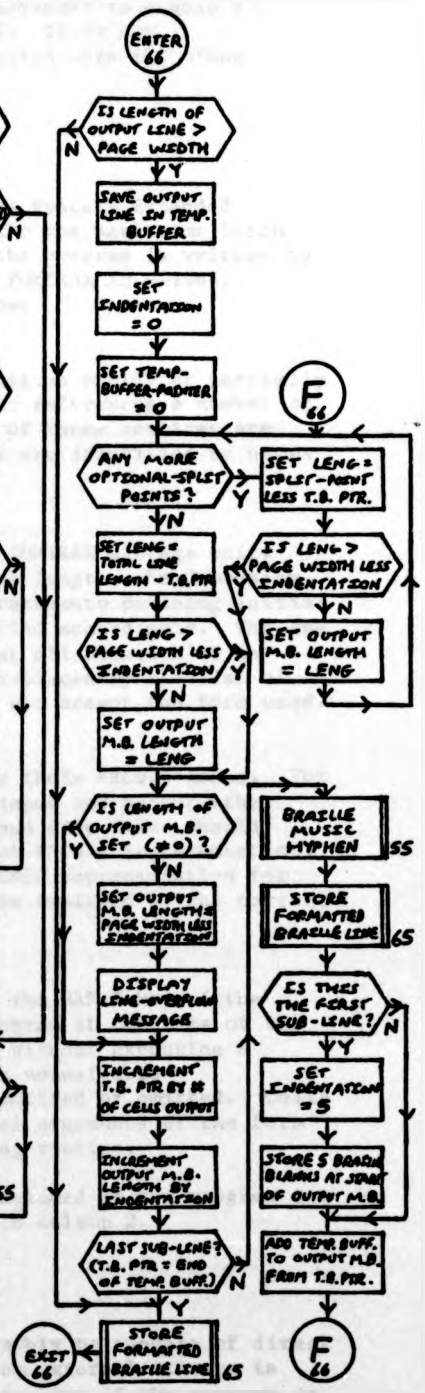


Figure A2:66 Store split measure



Appendix 3. Program documentation

The following program documentation is intended to enable a programmer to understand and implement CIMBAL. It is not self-contained, and should be read in conjunction with the other appendices and main text of this thesis.

A3.1 Program transfer

The program is written in Rank Xerox Data Systems Extended FORTRAN IV-H to run on a Sigma 5 computer under the Real-time Batch Monitor operating system. For portability, the program is written to conform as far as possible with USA Standard FORTRAN X3.9-1966. Deviations from this standard are listed below:

(1) Non-standard routines.

The program contains several routines written wholly or partially in Sigma 5 assembly code (Macrosymbol), and it references a number of non-standard library routines. Full details of these routines are given in section A3.3. Some library routines are identified by names containing more than six characters.

(2) Byte tables.

Byte tables are stored with 4 bytes per FORTRAN storage unit. For a machine with a storage unit of different length, the dimension of arrays containing byte tables, and DATA statements defining initial values for fixed byte tables, should be modified accordingly. The use of array names in DATA statements to represent all elements of an array is not standard FORTRAN and should be replaced by the list of individual elements for compilers which will not accept the form used.

(3) Character code.

Characters are represented internally by their EBCDIC codes. For a machine using a different character code, input and output other than that to or from the music archive tape and disc files should include code conversion to retain the internal EBCDIC representation. For a compiler which does not generate an EBCDIC representation for Hollerith constants, these constants should be replaced by the correct EBCDIC values.

(4) Break handling.

In the program as implemented, pressing the BREAK key of the terminal may cause exit from a FORTRAN subprogram at the time of the next input/output operation on the terminal, without executing a RETURN statement. On a system where this has undesirable side-effects, the break facility should be modified or omitted. Calls to subroutine BREAK contain non-standard label arguments of the form &n where n is a statement label in the calling routine.

(5) Other non-standard FORTRAN statements included in the program are preceded by a comment line containing a "+" in column 2.

Hardware requirements

The central processor used should preferably be capable of direct addressing of bytes. High-speed random-access external storage is required for the random-access music data file, and if the program is

overlaid, for the program file. In the prototype system the program file requires about 115K bytes; 60K bytes has been found sufficient for the music data file to hold a reasonably large piece of music. Some form of large-capacity sequential-access storage, such as magnetic tape, is required for archiving. A graphical display terminal is necessary for operator communication. The prototype system uses a Tektronix model T4002 display with a six and a half inch by eight and a quarter inch screen and 1024 x 760 addressable points. The program has also been successfully run from a DEC GT40 graphical display using a Tektronix simulation program. A braille embossing terminal is required for braille output. A lineprinter is required if a printed proof-listing of the braille is desired. A digital plotter is required if a printed proof display of printed music is desired.

Program size and overlay structure

The overall core space required by the prototype CIMBAL program is 115K bytes. The core space available to run the program in a time-sharing environment on the Sigma 5 computer is 34K bytes. The program is therefore overlaid according to the structure shown in figure A3:1. Program overlay segments are loaded as required at execution time from a read-only program file by the subroutine <LSEG>. On a machine with sufficient available core space, the overlay structure may be dispensed with or segments may be combined, starting at the lowest level. Alternatively, the input/display, braille translation, and archiving functions may be implemented as three separate programs.

A3.2 Use of identifiers

The use of identifiers to represent variables is described below. All variables in COMMON storage are listed, together with selected variables local to individual subprograms. All variables used in the program except <PSCALE> are of type INTEGER. Except for those marked L or X in the list below, they contain either integer values in the range -32767 to 32767 or byte strings. The "encoded" form of an item is the same as its external representation (see appendix 5).

In the list of identifiers, the information given for each variable is:

1 Identifier
2 Name of common block or subprogram containing the variable. All labelled common blocks have two-character names. In the following list, blank common is denoted by "**". All subprogram names except <SY> have three or more characters.
3 One or more single-letter codes meaning:

A	Array	All arrays are one-dimensional.
B	Byte	The variable contains a byte string (character string or packed table of 8-bit values). All byte strings are stored in arrays.
F	Flag	The variable takes one of the two values 0 (denoting "reset" or "false") or 1 (denoting "set" or "true").
H	Halfword	The variable contains a packed table of 16-bit values. All such tables are stored in arrays.
K	Constant	The variable is assigned a value at program compilation time and is not modified during

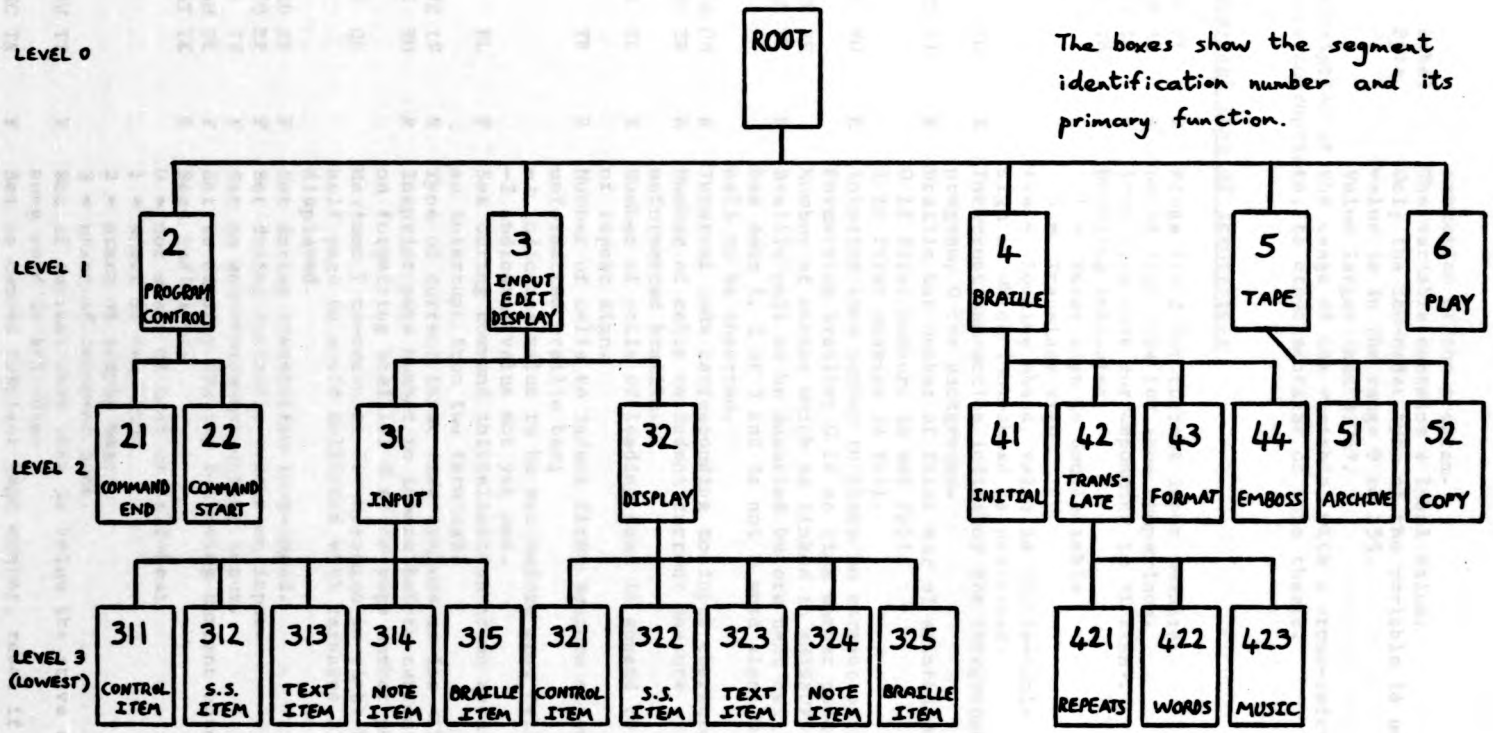


Figure A3:1 Program overlay structure

execution of the program.

L	Label	The variable contains a label value.
R	Byte	Only the low-order byte of the variable is used; the value is in the range 0 to 255.
X		Value larger than 32767.

4 Description of the usage of the variable, with a cross-reference, where appropriate, to other sections of this thesis.

Alphabetical list of identifiers

ICFLAG	FF	R	Flags field for current input measure.
ICITEM	IS	R	Saved item code for item comparison.
ICTYPE	IS	R	Item type code corresponding to <ICITEM>.
IDOUB	TR		Doubling indicator: -1 = Enter sign in double table 0 = Translate sign +ve = Looking ahead, value is double-table index of sign on which look-ahead is performed.
IEA	FG	X	Interrupt end-action indicator for foreground program, 0 for background.
IFIRST	B1	R	Braille bar number of first bar of printed music: 0 if first measure is not full 1 if first measure is full.
ILINE	FO	R	Inkprint line number to place on current parallel on formatting braille; 0 if no line number pending.
INBACK	FH		Number of sector which is linked to <MINBUF>.
INCELL	TR	R	Braille cell to be inserted before next cell if it has dots 1, 2 or 3 and is not a word sign; 0 if no cell to be inserted.
INCODE	CH	R	Internal code corresponding to input character.
INDENT	TR	R	Number of cells to indent current measure of unformatted braille.
INDTX	TR	R	Number of cells of leading text to append to start of repeat sign.
IND1	TR	R	Number of cells to indent first measure of current unformatted braille bar; -1 indicates value to be set before next cell -2 indicates value not yet set.
INIT	FL	F	Set during command initialisation or on receipt of an interrupt from the terminal.
INTYPE	IV	R	Type of current input item (values as for <ITTYPE>).
IPAGE	FO	R	Inkprint page number to insert before current line on formatting braille; 0 if no page number pending.
IQMAX	QL		Maximum Y co-ordinate for messages in right-hand half page to avoid collision with messages already displayed.
ISAHED	FF	F	Set during translation look-ahead.
ISAUTO	RP	F	Set during automatic repeat on input.
ISBAR	IV	F	Set on end-of-measure during input.
ISBEAM	DI	F	Set to display beaming following current note item.
ISBEAT	TK	R	Beat indicator: 0 = not start of beat or half-beat 1 = start of half-beat 2 = start of simple beat 3 = start of compound beat.
ISBLOW	TY	F	Set if current text item is below the stave or is a sung word or syllable.
ISCANC	TA	F	Set to cancel inkprint page number, reset if a null

			bar is output.
ISCHEK	FL	F	Set if checking and updating of current state is required on music retrieval.
ISCHOR	NO	F	Set if current note item is a chord or rest.
ISCMPD	TK	F	Set within first simple beat of compound beat.
ISCONT	TY	F	Set if current text item is a sung syllable which is not the last of a word.
ISCTIE	NO	F	Set if last decoded note item is a chord with ties on all heads.
ISCTK	FF	F	Set if current measure contains a clef sign, time signature, key signature or scale change.
ISDBUG	PP	R	Program debugging level: 0 indicates no debugging messages required. Positive value indicates that calls of subroutine <DBUG> are to produce a message if the first argument is not less than <ISDBUG>.
ISDISP	RP	F	Set to display item during input.
ISDOT	FO	F	Set if dot 3 is required after hand sign.
ISDUB	DB	R	0 = current doubling sign is not doubled, 1 = sign is doubled, 2 = sign is doubled and a second conditional single sign at the start of a measure is pending.
ISEDIT	FL	F	Set during editing.
ISEMB	FF	F	Set if any braille has been output to the embosser file.
ISEND	FL	R	General-purpose "end" indicator: 0 = no "end" condition, 1 = unspecified "end" condition, 2 = simulate interrupt on music input.
ISERR	FL	R	Error indicator: 0 = no error condition 1 = unspecified error condition 2 or more = specified error (used to index message table at end of execution of current command).
ISEXTR	NO	F	Set if current note item has "extras".
ISFEND	TR	F	Set on translating first-ending sign to prevent double barline forcing rest of measure onto new braille line.
ISGRD1	PP	F	Set to force grade 1 translation of braille text throughout.
ISGR1	TK	F	Set if current translation of literary text is to grade 1 braille.
ISHDR	FL	F	Set during processing of file header record.
ISHQD	PP	F	Set during high-quality display.
ISHYPH	FO	F	Set if hyphen is to be added to end of braille line on formatting.
ISINPT	FL	F	Set during input of music.
ISITEM	EE	R	Edit table index of item selected by previous edit-command if it was "get-item"; 0 otherwise.
ISJOY	CH	F	Set if joystick is in use for pitch specification during printed music input.
ISKEYB	TR	F	Set during translation of keyboard music.
ISLINR	TA	F	Set if braille left and right hand parts are currently combined using doubled octave interval.
ISNXST	FL	R	Indicator to display new parallel: 0 = no new parallel pending 1 = new parallel with wait, but not for certain items 2 = new parallel without wait, but not for certain items

			3 = force new parallel with wait 4 = force new parallel without wait.
ISOCV	TR	F	Set if compulsory octave sign required before next note.
ISOPT	TR	F	Set if following braille sign is to be included only if it is in first measure of a braille line.
ISPAUS	TK	F	Set if current note item has a pause sign.
ISPCON	TA	F	Set if previous syllable of sung text on current line was not the last of a word.
ISPLOT	PL	F	Set during output to digital plotter file.
ISPMT	IS	F	Set during part-measure repeat evaluation.
ISPOSE	PP	F	Set if transposition from treble clef is in effect on input.
ISPVBT	TK	F	Value of <ISBEAT> for previous note item of current measure; 0 if no previous note item.
ISQUOT	TR	F	Set if sung text is currently within quotation marks.
ISREST	FF	F	Set if current measure contains only a whole-measure rest.
ISRPT	FL		Number of measures remaining to be repeated during measure repeat; 0 if no measure-repeat is in effect.
ISSAG	PP	F	Set for SAGEM embosser, reset for LED embosser.
ISSCAN	FI	F	Set if scanning of measure required during music retrieval.
ISSIG	DB	F	Set if current item has specified doubling sign.
ISSMAL	NO	R	Small-note indicator for current note-head: 0 = ordinary note 1 = acciaccatura 2 = appoggiatura.
ISSPLT	FO	F	Set if measure for first braille part is split onto two or more braille lines.
ISSQZ	FF	F	Set during file squeezing operation.
ISSTEM	NO	F	Set if current note item is a stem sign.
ISTERM	TR	R	Braille termination sign pending flag: 0 = no sign pending 1 = end of crescendo pending 2 = end of decrescendo pending.
ISTIE	NO	F	Set if current note item is an implied rest or current note-head is tied to following note.
ISVOC	FF	F	Set if current line is a vocal part.
ISWIDE	PP	F	Set for extra spacing between printed music staves on display.
ISWORD	TX	F	Set if the current text item is a sung word or syllable.
ISWRIT	FI	F	Set if the sector of the disc file currently in the output buffer differs from the corresponding sector of the physical disc file.
ISXPOS	FF	F	Set if note pitches are to be transposed from the treble clef on input.
ITEM	IT	R	Current item code (first byte of encoded form).
ITERR	IT	F	Set if an item error message is currently displayed on the screen.
ITLENG	IT	R	Length in bytes of encoded form of current item.
ITPTR	IT	R	Pointer to start of encoded form of current item in input buffer or output buffer.
ITTYPE	IT	R	Item type code for current item: 1 = control item 2 = separate sign item 3 = text item

```
                    4 = note item
                    5 = braille item.
IXB     TD          X co-ordinate of start of current measure.
IXC     TD          X co-ordinate for reflection of input character.
IXINC   SC      R   Horizontal display unit.
IXE     TE          X co-ordinate of current item.
IXM     TE          X co-ordinate at which current note item should
                    appear according to time signature and scaling.
IXN     TE          X co-ordinate of current part of item.
IXP     PV          X co-ordinate of previous item.
IXQ     QL          X co-ordinate for display of messages.
IXQMIN  QL          Minimum X co-ordinate for reflected input characters
                    to avoid collision with displayed messages.
IXR     TD          X co-ordinate to return to for in-accord.
IXSHIF  DP          X co-ordinate increment for alignment of characters
                    and graphics on display.
IXSM    DN          X co-ordinate of note stem.
IXST    MM          X co-ordinate of start of extended sign.
IXT     TE          X co-ordinate of current cursor position.
IYC     TD          Y co-ordinate for reflection of input character.
IYE     TE          Y co-ordinate of current item.
IYINC   SC      R   Vertical display unit.
IYN     TE          Y co-ordinate of current part of item.
IYQ     QL          Y co-ordinate for display of messages.
IYS     TE          Y co-ordinate of bottom line of current stave.
IYSHIF  DP          Y co-ordinate increment for alignment of characters
                    with graphics on display.
IYSM    DN          Y co-ordinate of tail of note stem.
IYSMAX  DN          Y co-ordinate of top of note stem.
IYSMIN  DN          Y co-ordinate of bottom of note stem.
IYSTV   SC          Vertical display interval between staves.
IYT     TE          Y co-ordinate of current cursor position.
JADDR1  TR          File address of first flags field of first measure
                    of current bar of unformatted braille.
JALEV   DN          Average display level of uppermost and lowermost
                    heads of current note item.
JAXIDS  NO      R   Accidentals specification for current note head:
                    Bits 0-1 are zero
                    Bits 2-7 value same as encoded form.
JBAR    NB          Current bar number for display.
JBBUF   **      F   Indicates internal buffer for item decoding:
                    0  means decode item from input buffer
                    1  means decode item from output buffer.
JBDUR   TI          Total duration of current measure.
JBE     MD          Number of last bar of range specified for current
                    command.
JBEAM   DI      R   Position of current note item within beamed group; 0
                    if not in beamed group.
JBNO    TA          Bar number of measure being translated.
JBS     MD          Number of first bar of range for current command.
JBXL    ND          Display length of current measure in screen units.
JBYTE   WS          Temporary byte storage for separating individual
                    bits.
JCKMAX  TJ          Maximum number of bars which can be represented in
                    the checksum table.
JCLEF   IU      R   Current clef code (values same as second byte of
                    encoded form of clef item).
JCMD    SE          Number of command currently being executed:
                    -1  Initialisation 1     0  No current command
```

		1 Delete	2 Edit
		3 Insert	4 Build
		5 Add	6 Display
		7 Braille	8 Format
		9 Emboss	10 Save
		11 Restore	12 Erase
		13 Files	14 Clear
		15 Mode	16 Stop
		17 Summary	18 Scale
		19 Play	20 Change
		21 Print	22 Squeeze
		23 Copy	24 Flags
		25 Options	26 Initialisation 2
JCSTAT	FS	R	Protection status for current section: 1 = empty, 2 = unprotected, 3 = protected.
JDHIGH	JD	R	Line number of first line of current display parallel.
JDIST	JO		Square of distance of crosswire position of edit-command from nearest displayed item.
JDLINE	JD		Selection number of lowest line of current display parallel.
JDLOW	JD	R	Number of lowest line of current display parallel.
JDOTS	NO	R	Number of dots for current note item (in range 0-2).
JEBACK	EE		Number of sector linked to sector containing start of measure currently being edited.
JECLEF	ST	R	Clef code as at start of current measure.
JEDCMD	ED	R	Number of current edit-command: 1 = select item, 2 = delete item, 3 = change item, 4 = insert before, 5 = insert after, 6 = fresh display, 7 = examine, 8 = end of edit, 9 = cancel edit, 10 = override time check, 11 = crosswire switch
JEDCOD	RP	R	Edit table index of next note item to be decoded to give default values for note repetition.
JEDIT	EE	R	Index to current edit table entry.
JEFREE	EE	R	Edit table index of first entry of free chain (A3.5).
JESEL	RP	R	Edit table index of item selected for editing.
JFDIR	DN	R	Force-stem-direction indicator for display: 1 Note stems point upwards; 2 Note stems point downwards; 3 Note stems point upwards above middle line of stave, downwards otherwise; 4 Note stems point downwards above middle line of stave, upwards otherwise.
JFREE	FH		Number of first sector of free chain in disc file.
JGROUP	GP		Multiplying factor for duration of first note of irregular grouping to give correct overall time count when the remaining notes of the group are given their normal durations.
JHDR	HD		File address of header record.
JIADDR	AD		File address of last retrieved measure, or address following last stored measure during input of music.
JIBAR	**	R	Pointer to current position in input buffer.
JIMODE	MO	R	Current mode: 1 = printed music, 2 = unformatted braille, 3 = formatted braille.

		1 Delete	2 Edit
		3 Insert	4 Build
		5 Add	6 Display
		7 Braille	8 Format
		9 Emboss	10 Save
		11 Restore	12 Erase
		13 Files	14 Clear
		15 Mode	16 Stop
		17 Summary	18 Scale
		19 Play	20 Change
		21 Print	22 Squeeze
		23 Copy	24 Flags
		25 Options	26 Initialisation 2

JCSTAT FS R Protection status for current section:
1 = empty, 2 = unprotected, 3 = protected.

JDHIGH JD R Line number of first line of current display parallel.

JDIST JO Square of distance of crosswire position of edit-command from nearest displayed item.

JDLINE JD Selection number of lowest line of current display parallel.

JDLOW JD R Number of lowest line of current display parallel.

JDOTS NO R Number of dots for current note item (in range 0-2).

JEBACK EE Number of sector linked to sector containing start of measure currently being edited.

JECLEF ST R Clef code as at start of current measure.

JEDCMD ED R Number of current edit-command:

1 = select item	7 = examine
2 = delete item	8 = end of edit
3 = change item	9 = cancel edit
4 = insert before	10 = override time check
5 = insert after	11 = crosswire switch
6 = fresh display	

JEDCOD RP R Edit table index of next note item to be decoded to give default values for note repetition.

JEDIT EE R Index to current edit table entry.

JEFREE EE R Edit table index of first entry of free chain (A3.5).

JESEL RP R Edit table index of item selected for editing.

JFDIR DN R Force-stem-direction indicator for display:
1 Note stems point upwards
2 Note stems point downwards
3 Note stems point upwards above middle line of stave, downwards otherwise
4 Note stems point downwards above middle line of stave, upwards otherwise.

JFREE FH Number of first sector of free chain in disc file.

JGROUP GP Multiplying factor for duration of first note of irregular grouping to give correct overall time count when the remaining notes of the group are given their normal durations.

JHDR HD File address of header record.

JIADDR AD File address of last retrieved measure, or address following last stored measure during input of music.

JIBAR ** R Pointer to current position in input buffer.

JIMODE MO R Current mode:
1 = printed music
2 = unformatted braille
3 = formatted braille.

JINBUF	**		Pointer to input address within sector.
JINLK	**		Sector to which sector <JINSEC> is linked.
JINSEC	**		Sector containing current input address.
JISNO	TA	R	Line number of measure being translated.
JKEY	IU	R	Current key signature (values same as second byte of encoded form of key signature item).
JKEYNO	KY	R	Number of signs in current key signature (0 to 7).
JKEYSG	KY	R	Type of signs in current key signature: 1 for flats, 2 for naturals or no sign, 3 for sharps.
JLABEL	CT	L	Address of program location to which control is to be transferred on receipt of a terminal interrupt.
JLASTN	JS	R	Pointer to last non-rest note in input measure.
JLENTH	NO	R	Duration code for current note item: 1 = Breve 2 = Semibreve (whole note) 3 = Minim (half note) 4 = Crotchet (quarter note) 5 = Quaver (1/8 note) 6 = Semiquaver (1/16 note 7 = Demisemiquaver (1/32 note) 8 = Hemidemisemiquaver (1/64 note) 9 = Semihemidemisemiquaver (1/128 note).
JLEV	DN		Display level for current note-head.
JLINE	NS		Current line number.
JLINR	TA		Number of consecutive measures (including current measure) for which left hand is an octave below right hand.
JLOCN	LO	R	Pointer to current position for insertion into location table.
JMATCH	JS	R	Result of measure comparison; <CMPARE> returns one of following values: 0 no match 1 match 2 measures are an octave apart; <MTEST> returns one of the following values: 0 no repeats or measure rests 1 single or multiple measure rest 2 single or multiple repeat of previous measure 3 single measure numeral repeat 4 multiple measure numeral repeat 5 null measure.
JMEM	MM	R	Index to position in current state table.
JNDUR	TI		Duration of current note item.
JNGRP	NG	R	Number of notes in current braille note-grouping.
JNHEDS	NH	R	Number of note-heads input so far for note item currently being input.
JNLAST	JS	R	Temporary value of <JLASTN>.
JNSTAT	FS	R	New protection status for current section on successful completion of current command (values as for <JCSTAT>).
JOADDR	AD		File address of flags field of most recently stored measure.
JOBAR	**	R	Pointer to current position in output buffer.
JOCTAV	NO	R	Octave number for current note-head, in range 0 to 8.
JOLINE	EM	R	Current line number on braille output page.
JOMODE	MO	R	Mode of section being created (values as for <JIMODE>.
JOPAGE	B1		Current page number in embosser output file.

JOPBUF	**		Pointer to output address within sector.
JOPLK	**		Sector to which sector <JOPSEC> is linked.
JOPSEC	**		Sector containing current output address.
JOSNO	TA	R	Unformatted braille line number of measure being translated.
JOUT	EM		Pointer to current position in embosser file output buffer.
JPART	NS	R	Current part code (A5.1).
JPITCH	NO	R	Pitch code for current note-head: 0 indicates item is a rest 1 = C, 2 = D, 3 = E, 4 = F, 5 = G, 6 = A, 7 = B.
JPLEN	PV		Duration code for previous note item.
JPN	PV	R	Edit table index of previous non-rest note item in current measure if any during input of printed music; 0 otherwise.
JPPCH	PV	R	Octave/pitch value for previous note-head (set to zero during braille translation when a forced octave sign is required).
JPVAX	DX	F	Set if an accidental has been displayed for the current note item and the most recent such accidental was not shifted left.
JPVLEV	DN		Display level of previous note-head.
JREPT	PV	R	Number of note items remaining to be automatically repeated on input; 0 if no automatic repeat in effect.
JSCALE	HD	R	Horizontal scale factor for printed music display.
JSDIR	DN		Stem direction for current note item: -1 = stem pointing upwards 1 = stem pointing downwards.
JSLUR	DB	AR	Slur status indicators for outer and inner slurs respectively during braille translation: 0 no slurs in sight 1 to output close bracket slur at end-of-slur 2 in slur but no sign yet output 3 in bracket slur 4 to output no slur sign after note 5 close bracket slur to follow next note 6 to output double sign after current note 7 to output final single sign after note 8 to output other single slur sign
JSSEL	MD	R	Number of first selected line.
JSTAFF	JD	R	Number of current parallel on display.
JSTART	FO	R	Position in braille output line of next measure of formatted braille.
JSUM	JS	R	Checksum for current measure for measure repeat testing during braille translation.
JTERM	PP	F	Terminal type: 0 = Tektronix, 1 = GT40.
JTIME	TI		Time count within current measure.
JTIME1	TI	R	Upper figure of current time signature.
JTIME2	TI	R	Lower figure of current time signature.
JTMAX	TI		Maximum allowed time duration for measure.
JTMIN	TI		Minimum allowed time duration for measure.
JTTEMP	TT		Temporary storage for time count.
JTYME	IU	R	Second byte of encoded form of current time signature.
JVOIC	NS	R	Bits 0-1 in-accord part within current line (first part is part 0); Bits 2-7 current line number (= <JLINE>).
JX	CH		X co-ordinate of crosswire position.

JOPBUF	**		Pointer to output address within sector.
JOPLK	**		Sector to which sector <JOPSEC> is linked.
JOPSEC	**		Sector containing current output address.
JOSNO	TA	R	Unformatted braille line number of measure being translated.
JOUT	EM		Pointer to current position in embosser file output buffer.
JPART	NS	R	Current part code (A5.1).
JPITCH	NO	R	Pitch code for current note-head: 0 indicates item is a rest 1 = C, 2 = D, 3 = E, 4 = F, 5 = G, 6 = A, 7 = B.
JPLEN	PV		Duration code for previous note item.
JPN	PV	R	Edit table index of previous non-rest note item in current measure if any during input of printed music; 0 otherwise.
JPPCH	PV	R	Octave/pitch value for previous note-head (set to zero during braille translation when a forced octave sign is required).
JPVAX	DX	F	Set if an accidental has been displayed for the current note item and the most recent such accidental was not shifted left.
JPVLEV	DN		Display level of previous note-head.
JREPT	PV	R	Number of note items remaining to be automatically repeated on input; 0 if no automatic repeat in effect.
JSCALE	HD	R	Horizontal scale factor for printed music display.
JSDIR	DN		Stem direction for current note item: -1 = stem pointing upwards 1 = stem pointing downwards.
JSLUR	DB	AR	Slur status indicators for outer and inner slurs respectively during braille translation: 0 no slurs in sight 1 to output close bracket slur at end-of-slur 2 in slur but no sign yet output 3 in bracket slur 4 to output no slur sign after note 5 close bracket slur to follow next note 6 to output double sign after current note 7 to output final single sign after note 8 to output other single slur sign
JSSEL	MD	R	Number of first selected line.
JSTAFF	JD	R	Number of current parallel on display.
JSTART	FO	R	Position in braille output line of next measure of formatted braille.
JSUM	JS	R	Checksum for current measure for measure repeat testing during braille translation.
JTERM	PP	F	Terminal type: 0 = Tektronix, 1 = GT40.
JTIME	TI		Time count within current measure.
JTIME1	TI	R	Upper figure of current time signature.
JTIME2	TI	R	Lower figure of current time signature.
JTMAX	TI		Maximum allowed time duration for measure.
JTMIN	TI		Minimum allowed time duration for measure.
JTTEMP	TT		Temporary storage for time count.
JTYME	IU	R	Second byte of encoded form of current time signature.
JVOIC	NS	R	Bits 0-1 in-accord part within current line (first part is part 0); Bits 2-7 current line number (= <JLINE>).
JX	CH		X co-ordinate of crosswire position.

JXBEAM	DX		Shift in X co-ordinate for note-beaming display.
JXMAX	QL		X co-ordinate of right hand side of page.
JXPOSE	TK		Number of octaves by which braille is to be transposed down from print.
JXTEMP	TT		X co-ordinate to return to after in-accord.
JY	JO		Y co-ordinate of most recently read crosswire position.
JYBEAM	DX		Y co-ordinate of tails of beamed notes.
JYMAX	QL		Y co-ordinate of top of page.
KATDEV	AR	KR	Unit number of music archive tape.
KBAR	KK	KR	Internal code for barline character.
KBDOT	KB	KR	Braille cell dot 3.
KBDOTS	KK	KR	Internal code for bar-line dots character.
KBFSIZ	BF	K	Size of disc file sector in bytes.
KBFWDS	BF	K	Size of disc file sector in machine words.
KBHSIG	BE	KAR	Second cell of braille hand signs, indexed by part code.
KBHYPH	KB	KR	Braille cell for hyphen.
KBIG	KO	K	Largest integer used in the program (excluding those marked X in this list).
KBLENG	BE	KR	Maximum number of cells per braille line.
KBNUMB	KB	KR	Braille cell for number sign.
KBRDEV	EK	KR	Unit number for embosser file.
KBREAK	KC	KR	Braille format code for force new line.
KBRPT	KB	KR	Braille cell for repeat sign.
KBSIZE	MC	KR	Maximum number of bytes in encoded measure.
KBSPAC	KB	KR	Braille cell for space.
KCLEF	KK	KR	Internal code for clef character.
KDATA	BF	K	Number of bytes in data field of disc file sector.
KDOT	KT	KB	EBCDIC characters ". ".
KEOD	CH	KR	Internal code for end-of-data character.
KFLAT	KK	KR	Internal code for flat character.
KFSIZE	MC	K	Number of last sector of disc file. (= number of sectors in file less 1).
KHS	NO	KR	Maximum number of bytes in encoded form of note-head.
KIBASE	AR	K	Pointer to start in tape index of table of identifiers.
KISIZE	AR	K	Length in bytes of tape index.
KKEY	KK	KR	Internal code for key-signature character.
KLEVEL	FG	KR	Interrupt level for foreground program, 0 for background program.
KMLPP	EK	KAR	Number of braille lines per page, indexed by embosser type code <ISSAG> + 1.
KMRSIZ	KM	KR	Number of bytes in encoded form of measure rest.
KMSIZE	ST	KR	Length in bytes of state table.
KMTMES	AR	K	EBCDIC characters "TAPE NAME = ".
KNATUR	KK	KR	Internal code for natural character.
KNBASE	AR	K	Pointer to start in tape index of filenames field.
KNDOT	KK	KR	Internal code for note-dot character.
KNHMAX	NH	KR	Maximum number of note heads per note item.
KNLIST	FH	KR	Number of sections in disc file.
KNREPT	KK	KR	Internal code for repeat character.
KNSDEL	KK	KR	Internal code for cancel character.
KQERY	KT	KB	EBCDIC characters "? ".
KSBASE	AR	K	Pointer to start in tape index of table of pointers to filenames in tape index.
KSEGS	SE	KB	Table of overlay segment numbers indexed by command number <JCMD>.

KSHARP	KK	KR	Internal code for sharp character.
KSPACS	KT	KB	8 EBCDIC spaces.
KSPLIT	KC	KR	Braille format code for potential line division.
KSWICH	KK	KR	Internal code for joystick-switch character.
KTEXDN	KK	KR	Internal code for text-below character.
KTEXUP	KK	KR	Internal code for text-above character.
KTEXWD	KK	KR	Internal code for sung-word character.
KTHDR	HD	KB	File and tape type identifier "MUZK".
KTIE	KK	KR	Internal code for tie character.
KTIME	KK	KR	Internal code for time-signature character.
KTOUT	KC	KR	Braille format code indicating that preceding cells of current measure are to be transferred to a separate line.
KVBASE	AR	K	Pointer to start in tape index of table of version numbers.
KWHYPH	KC	KR	Braille format code indicating that a hyphen is to be added if at the end of a braille text line.
KWSIZE	MC	KR	Length of machine word in bytes.
KXMAX	MC	K	X co-ordinate of right hand side of screen.
KYMAX	MC	K	Y co-ordinate of top of screen.
KZEROS	KM	KB	Zero state table.
K0-K8	KO	KR	Constants 0 to 8.
LBELL	REJECT	KB	Bell character.
LBKPTR	FEDIT		Number of sector linked to sector <LEIN1>.
LEBYTE	FEDIT		Byte position within sector of start of measure to be edited.
LEIN1	FEDIT		Number of sector containing start of measure to be edited.
LEIN2	FEDIT		Number of sector containing end of measure to be edited.
LELINK	FEDIT		Number of sector to which sector <LEIN2> is linked.
LELIST	CMDSET	KR	Number of lists in edit table (A3.5).
LEOUT1	FEDIT		Number of sector containg start of edited measure.
LEOUT2	FEDIT		Number of sector containg end of edited measure.
LESIZE	CMDSET	KR	Number of available entries in edit table (A3.5).
LISREW	ARCHIV	F	Set if tape is to be rewound.
LKNATS	IDCODE	KB	Key table for no sharps or flats.
LKSMAX	ININIT	KAR	Maximum number of lines allowed in each section.
LLCA	NXCMD	KR	EBCDIC code for lower case A.
LLCZ	NXCMD	KR	EBCDIC code for lower case Z.
LMKP	IDCODE	KB	Table of pitches within octave of sharps in key signature, indexed by accidental position (4152637 = FCGDAEB).
LMTYPE	WTRANS	KAB	Type codes for translation of text characters (byte table indexed by EBCDIC character code): 0 = illegal character 1 = space 2 = punctuation character 3 = quotation mark 4 = set grade 1 translation 5 = set grade 2 translation 6 = digit 7 = upper case letter 8 = lower case letter
LNCMDS	NXCMD	KR	Number of commands.
LNMOD	CHANGE	KR	Number of changeable parameters.
LNPINS	ININIT	KR	Number of instrumental part codes.
LNPVOC	ININIT	KR	Number of vocal part codes.
LPLDEV	PREND	KR	Unit number for graph plotter file.

LREQM	BFIND		Measure number of required measure.
LTCMD	NXCMD	KB	Command names (8 characters each).
LTPINS	ININIT	KB	Input abbreviations for instrumental part codes (4 characters each).
LTPVOC	ININIT	KB	Input abbreviations for vocal part codes (4 characters each).
LTRANS	NXCMD	KB	Disc file status transition table, indexed by command number and current mode; 0 = command not allowed in this mode +ve = new status (values as for <JCSTAT>).
L1-L3	WS		Pseudo-local variables. These are used in such a way that if they were not declared as COMMON variables, program execution would not be affected.
MBARS	FH	A	Number of bars in each section of disc file, indexed by section number.
MBEAML	DI	B	Length codes for notes of beamed group.
MBEAMX	DI	H	X co-ordinates for beamed group.
MBEAT	TR	B	Pointers to start of beats in current measure.
MBHDR	EM	B	Braille cell codes for running page title.
MBLINK	LK	B	Backward link pointer field of edit table (A3.5).
MBMYH	DI	H	Y co-ordinates of upper heads of beamed group.
MBMYL	DI	H	Y co-ordinates of lower heads of beamed group.
MBPART	BE	B	Braille part codes, indexed by unformatted braille line number.
MBRPT	TR	B	Flag for each beat of current measure, set if beat is a repeat of the previous beat of the same measure.
MBYTES	BY	A	Number of bytes in each section of disc file, indexed by section number.
MCELLS	XG	KB	Table of braille music signs (A3.5).
MCHAR	CH	B	Character read from terminal.
MCHECK	TJ	B	Table of checksum values for measure comparison (4.2).
MCLEF	HE	B	Current clef codes, indexed by line number.
MCMEM	ST	B	Current state table (A3.5).
MCODES	XB	KB	Table of internal codes corresponding to music input characters, indexed by EBCDIC code of character typed on terminal (A3.5).
MCORD1	XC	KB	Display co-ordinate table for segment 32 (A3.5).
MCORD2	XD	KB	Display co-ordinate table for segment 321 (A3.5).
MCORD3	XE	KB	Display co-ordinate table for segment 322 (A3.5).
MCORD4	XF	KB	Display co-ordinate table for segment 324 (A3.5).
MCSAVE	IS	B	Parameters of item saved for comparison.
MCTK	LTB	B	Time and key signature fields of location table (A3.5).
MCXMEM	TA	B	Saved values of <NSLURS> and <JSLUR> entries for each unformatted braille line.
MCXTRA	IS	AR	Extras for note item saved for comparison.
MEMPTR	ME	B	Table of edit-table indices of first of pair of entries containing saved state table, indexed by line number; 0 indicates no state table saved for given line (A3.5).
MEND	FO	A	+ve = position in measure at which to force new line in formatted braille, 0 = no forced new line, -1 = force new line at end of measure; indexed by unformatted braille line number.
MEXTRA	NO	AR	Extras for current note item: first entry is unused (0), remaining entries contain extras codes of types

			1, 2, 3 respectively (zero if absent), low-order 5 bits only.
MFIND	LT	A	Measure number field of location table (A3.5).
MFLAGS	**	AR	Flags fields for current measure (second and third flags used only for unformatted braille).
MFLINK	LK	B	Forward link pointer field of edit table (A3.5).
MFNAME	AR	B	File name.
MGROUP	GP	KA	Table of values for <JGROUP>, indexed by number of notes in irregular grouping.
MHCLEF	SU	B	Table of clef codes in header record, indexed by line number.
MHD	NO	B	Encoded note-heads for current note item.
MHEAD	FH	A	First sector of each disc file section, indexed by section number; 0 if section is empty.
MHEXT	NO	AR	Extras for current note head: (code values same as external format) 1st = fingering code 2nd = ornament code (in range 0-10) 3rd = first accidental code for ornament 4th = second accidental code for ornament.
MHYTAB	XJ	KB	Hershey co-ordinate table (A3.5).
MIBAR	**	B	Input buffer, containing one encoded measure.
MIDENT	HD	AB	Current disc file identifier: four EBCDIC characters (including trailing blanks if necessary).
MINBUF	**	B	Buffer containing sector of disc file currently being accessed for input.
MINDEX	AR	B	Copy of tape index during tape operations (A5.2).
MJNOTE	NG	B	Pointers to positions in output buffer of braille notes to be grouped.
MKPCH	KY	B	Key table for current key signature (A3.5).
MLINES	FH	A	Number of lines in each section of disc file, indexed by section number.
MLIST1	LK	B	First data field of edit table (A3.5).
MLIST2	LK	B	Second data field of edit table (A3.5).
MLIST3	LK	B	Third data field of edit table (A3.5).
MLIST4	LK	B	Fourth data field of edit table (A3.5).
MLOCN	LT	B	Disc file address field of location table (A3.5).
MMESIJ	XA	KB	Message table (A3.5).
MNAME	TL	B	Name of current disc file.
MOBAR	**	B	Output buffer, containing one encoded measure.
MOPBUF	**	B	Buffer containing sector of disc file currently being accessed for output.
MPART	HE	B	Current part codes, indexed by line number.
MPCONT	TA	AF	Values of <ISPCON> for each unformatted braille line.
MPITCH	TA	A	Values of <JPPCH> for each unformatted braille line.
MRBAR	TK	B	Cells of current braille repeat sign if any.
MREPT	TA	A	+ve = number of measures in repeat sequence; 0 = no repeat in effect; -1 = second of two measure repeats pending; indexed by selected line number.
MREST	KM	KB	Coded form of whole-measure rest.
MRESTS	TA	A	Number of measures in sequence of consecutive measure rests, for each unformatted braille line.
MSBAR	FO	B	Braille cell codes for text line to be inserted on formatting.
MSMEM	ST	B	State table as at start of current measure (A3.5).
MSPLIT	FO	A	Up to three points at which to split a long measure between two formatted braille lines; unused entries

			1, 2, 3 respectively (zero if absent), low-order 5 bits only.
MFIND	LT	A	Measure number field of location table (A3.5).
MFLAGS	**	AR	Flags fields for current measure (second and third flags used only for unformatted braille).
MFLINK	LK	B	Forward link pointer field of edit table (A3.5).
MFNAME	AR	B	File name.
MGROUP	GP	KA	Table of values for <JGROUP>, indexed by number of notes in irregular grouping.
MHCLEF	SU	B	Table of clef codes in header record, indexed by line number.
MHD	NO	B	Encoded note-heads for current note item.
MHEAD	FH	A	First sector of each disc file section, indexed by section number; 0 if section is empty.
MHEXT	NO	AR	Extras for current note head: (code values same as external format) 1st = fingering code 2nd = ornament code (in range 0-10) 3rd = first accidental code for ornament 4th = second accidental code for ornament.
MHYTAB	XJ	KB	Hershey co-ordinate table (A3.5).
MIBAR	**	B	Input buffer, containing one encoded measure.
MIDENT	HD	AB	Current disc file identifier: four EBCDIC characters (including trailing blanks if necessary).
MINBUF	**	B	Buffer containing sector of disc file currently being accessed for input.
MINDEX	AR	B	Copy of tape index during tape operations (A5.2).
MJNOTE	NG	B	Pointers to positions in output buffer of braille notes to be grouped.
MKPCH	KY	B	Key table for current key signature (A3.5).
MLINES	FH	A	Number of lines in each section of disc file, indexed by section number.
MLIST1	LK	B	First data field of edit table (A3.5).
MLIST2	LK	B	Second data field of edit table (A3.5).
MLIST3	LK	B	Third data field of edit table (A3.5).
MLIST4	LK	B	Fourth data field of edit table (A3.5).
MLOCN	LT	B	Disc file address field of location table (A3.5).
MMESIJ	XA	KB	Message table (A3.5).
MNAME	TL	B	Name of current disc file.
MOBAR	**	B	Output buffer, containing one encoded measure.
MOPBUF	**	B	Buffer containing sector of disc file currently being accessed for output.
MPART	HE	B	Current part codes, indexed by line number.
MPCONT	TA	AF	Values of <ISPCON> for each unformatted braille line.
MPITCH	TA	A	Values of <JPPCH> for each unformatted braille line.
MRBAR	TK	B	Cells of current braille repeat sign if any.
MREPT	TA	A	+ve = number of measures in repeat sequence; 0 = no repeat in effect; -1 = second of two measure repeats pending; indexed by selected line number.
MREST	KM	KB	Coded form of whole-measure rest.
MRESTS	TA	A	Number of measures in sequence of consecutive measure rests, for each unformatted braille line.
MSBAR	FO	B	Braille cell codes for text line to be inserted on formatting.
MSMEM	ST	B	State table as at start of current measure (A3.5).
MSPLIT	FO	A	Up to three points at which to split a long measure between two formatted braille lines; unused entries

			take value <KBIG>.
MSSEL	MD	B	Table of selected lines, indexed by line number: 0 = unselected line, +ve = sequence number of line within selected group.
MSTAT	FS	AR	Status codes for sections of disc file, indexed by section number (values as for <JCSTAT>).
MTABLE	XH	KB	Literary braille contraction table (A3.5).
MTAIL	FH	A	Number of last sector of each section of disc file, indexed by section number; 0 for empty section.
MTBAR	MT	B	Temporary storage for one measure.
MTEXT	TX	B	Characters of current text item.
MTKEY	TK	B	Two temporary key tables used during beat comparison and look-ahead (A3.5).
MTKPCH	KY	B	Key table for current key signature as modified by marked accidentals in current measure (A3.5).
MXCOM	CB	A	Communication variables between control segment and other program overlay segments.
MXTEXT	DN	A	X co-ordinates of end of text items of types text-above, text-below and word respectively.
MYAXID	SD	AX	Vertical display offsets from centre line of treble stave for display of key signature flats, naturals and sharps respectively; packed as decimal digits.
MYSTV	SC	AR	Values of <IYSTV> for each section, indexed by section number.
NBARS	FH		Number of bars in current section.
NBEAM	IU		Number of notes in current beamed group (in range 0 to 16).
NBEATS	TR	R	Number of beats in current measure.
NBHDR	EM	R	Number of cells in braille running title.
NBYTES	BY		Number of bytes in current section.
NCHARS	TX	R	Number of characters in current text item (in range 0 to 31).
NDBL	DB	R	Number of remaining consecutive notes to which current doubling sign applies.
NDL	NS	R	Number of lines of printed music which will fit onto one page of display.
NDP	NS	R	Number of parallels of music which will fit onto one page of display.
NEDBAR	ED		Bar number of measure being edited.
NEDSTV	ED	R	Line number of measure being edited.
NELENG	ED	R	Number of bytes in source of measure being edited, as stored in file.
NEXTX	ND		X co-ordinate for start of next measure.
NFLAGS	**	R	Number of flags bytes in each measure of current section (3 for unformatted braille, 1 otherwise).
NFNAME	AR	R	Number of characters in <MFNAME>.
NGROUP	IU	R	Number of notes in current or most recent irregular note grouping.
NHEDS	NO	R	Number of note-heads in current note item.
NHTEXT	HD	R	Number of text items in header record, excluding file name item.
NIB	NB		Bar number of input measure.
NIBS	NB		Measure number of input measure.
NILENG	**	R	Length in bytes of input measure.
NINBUF	**		Number of disc file sector currently in <MINBUF> or -1 indicating no sector in <MINBUF>.
NIPL	TR	R	Number of braille format control codes output so far for current unformatted braille line.
NIS	NB		Line number of input measure.

NLINE	IU	R	Current inkprint parallel number.
NLINES	NS		Number of lines in current section.
NLOCS	LO	R	Number of entries of location table used (A3.5).
NNAME	TL	R	Number of characters in file name <MNAME>.
NNDOWN	MR	R	Pitch adjustment for measure repeat.
NNINCB	PV	R	Number of note items in current measure.
NNOTES	TI	R	Number of note items decoded for current measure.
NOB	NB		Bar number of output measure.
NOLENG	**	R	Length in bytes of output measure.
NOPBUF	**		Number of disc file sector currently in <MOPBUF>; -1 means no sector in <MOPBUF>.
NOS	NB	R	Line number of output measure.
NPAGE	IU	R	Current inkprint page number.
NRLENG	TK	R	Number of cells in <MRBAR>.
NSCALE	IU	R	Horizontal display scaling factor.
NSECTS	BY		Number of sectors in current section of disc file.
NSHIFT	ND		Vertical shift in display of notes due to 8vas.
NSLENG	FO	R	Number of cells in separate text line to be inserted on formatting.
NSLURS	DB	R	Number of slurs open during braille translation (maximum 2).
NSSEL	MD	R	Number of lines currently selected.
NTLENG	MT	R	Number of bytes in temporary measure <MTBAR>.
NVERS	BY	R	Version number of saved or restored file.
NXLENG	ND		Horizontal display spacing for current note item according to scale factor and time signature.
PSCALE	P1		Graph plotter display scaling factor (inches/unit).

A3.3 Use of non-standard routines

This section contains a description of all non-standard library routines called by the program, and utility routines included in the program but written entirely or partly in assembly code. The routines are listed in alphabetical order in the following format:

ROUTINE NAME(PARAMETERS)

Function: Summary of the purpose of the routine.

Parameters: Indicates type of all parameters which are not scalar integers.

Entry: Indicates values of parameters and any other relevant variables on entry to the routine. The value on entry of any parameter not listed here is irrelevant.

Exit: Indicates values of parameters and any other relevant variables on exit from the routine. The value of any parameter not listed here is unchanged by the routine.

The non-standard routines may be classified as follows:

(1) Manipulation of non-FORTRAN information units: CBS, ISBIT, MBS, NBYTE, NHW, STBIT, STBYTE, STHW.
(2) Disc and magnetic tape input/output: BUFFERIN, BUFFEROUT, BUFFIN, BUFFOUT, DISKIO, ICHECK, SKIPFILE, WAIT.
(3) Terminal input/output: BRKLABEL, ERASE, JOY, TPLOT, URD, UWT.
(4) Graph plotter output: NUPAGE, PLOT, SYMBOL.
(5) Audio output: IDAC, TIMER.
(6) RBM operating system interface: INTLEVEL, OVMOVE, SY.
(7) Internal formatting: ICCONV.
(8) Overlay segment loader: LSEG.

(9) Space saving: NPOWER.

For program transfer, these routines should be replaced by equivalent routines written in FORTRAN or otherwise. For efficiency, FORTRAN should not be used for byte-handling routines. RBM operating system interface routines may be omitted.

```
            SUBROUTINE BRKLABEL(NLABEL)
Function:   Causes program control to be transferred to a specified
            location on receipt of an interrupt from the terminal.
            This transfer occurs immediately if the program is
            currently reading from or writing to the terminal,
            otherwise it is delayed until the next read or write
            operation on the terminal.
Parameters: <NLABEL> is a label value.
Entry:      <NLABEL> is of the form &n, where n is the number of a
            label occurring in the calling routine, to which control
            is to be transferred on receipt of an interrupt.

            SUBROUTINE BUFFERIN(NUNIT,NTYPE,NARRAY,NLENG,NSTAT)
Function:   Direct input from an external storage device.
Parameters: <NARRAY> is an integer array.
Entry:      <NUNIT> is the FORTRAN unit number of the device.
            <NTYPE> is 1.
            <NLENG> is the maximum number of machine words to be read.
Exit:       <NSTAT> is an operation completion status indicator:
            2 = operation successful, 3 = end-of-file, 4 = input
            error.
            <NARRAY> contains the data read.

            SUBROUTINE BUFFEROUT(NUNIT,NTYPE,NARRAY,NLENG,NSTAT)
Function:   Direct output to an external storage device.
Parameters: <NARRAY> is an integer array.
Entry:      <NUNIT> is the FORTRAN unit number of the device.
            <NTYPE> is 1.
            <NLENG> is the number of machine words of data to be
            written.
            <NARRAY> contains the data to be written.
Exit:       <NSTAT> is an operation completion status indicator:
            2 = operation successful, 3 or 4 = output error.

            SUBROUTINE BUFFIN(NUNIT,NTYPE,NARRAY,NLENG)
Function:   To initiate direct input from an external storage device.
Parameters: <NARRAY> is an integer array.
Entry:      As for subroutine <BUFFERIN>.
Exit:       An input operation is initiated to read data into
            <NARRAY>, and control is returned to the calling routine.

            SUBROUTINE BUFFOUT(NUNIT,NTYPE,NARRAY,NLENG)
Function:   To initiate direct output to an external storage device.
Parameters: <NARRAY> is an integer array.
Entry:      As for subroutine <BUFFEROUT>.
Exit:       The output operation is initiated and control is returned
            to the calling program.

            LOGICAL FUNCTION CBS(NTEXT1,N1,NTEXT2,N2,NLENG)
Function:   To compare two byte strings.
Parameters: <NTEXT1> and <NTEXT2> are integer arrays.
```

Entry: <NTEXT1> contains the first byte string.
<N1> is the starting byte position within the first byte string. The most significant byte of the first word is byte 1.
<NTEXT2> contains the second byte string.
<N2> is the starting byte position within the second byte string. The most significant byte of the first word is byte 1.
<NLENG> is the number of bytes to be compared.

Exit: <CBS> is TRUE if the compared byte strings are identical, FALSE otherwise.

SUBROUTINE DISKIO(NIO,NSECT,NBUFF)

Function: Random-access disc file input and output. For the read function, a specified sector of the disc file is read into one of the two file-image buffers <MINBUF> or <MOPBUF>. For the write function, the contents of one of the two buffers are written to a specified sector of the disc file.

Entry: <NIO> is read/write indicator: 0 = read, 1 = write.
<NSECT> is the sector number within the disc file of the block to be read from or written to, The first sector of the file being considered as sector 0.
<NBUFF> indicates which internal buffer is to be used: 0 for <MINBUF>, 1 for <MOPBUF>.

Exit: If any input/output error occurs, <ISERR> is set to 2 and a terminal interrupt is simulated (because continuation is usually meaningless). Otherwise a single sector of data is transferred between the file and the specified internal buffer.

SUBROUTINE ERASE

Function: Erases the screen of the terminal.

SUBROUTINE ICCONV(N,NLENG,NTEXT)

Function: Converts a 16-bit non-negative integer from binary to EBCDIC character form.

Parameters: <NTEXT> is an integer array of length 8 bytes.

Entry: <N> is the integer to be converted.

Exit: <NLENG> is one greater than the number of decimal digits in the number.
The first <NLENG> bytes of <NTEXT> contain the character representation of the integer in EBCDIC code, including a leading blank. The remaining bytes of <NTEXT> are undefined.

SUBROUTINE ICHECK(NUNIT,NSTAT,NWORDS)

Function: Checks the current status of the last input/output operation initiated on a specified device using <BUFFIN> or <BUFFOUT>.

Entry: <NUNIT> is the FORTRAN unit number of the device.

Exit: <NSTAT> is a status indicator having one of the values: 1 = operation not yet complete, 2 = operation successfully complete, 3 = end-of-file, 4 = input/output error.
If <NSTAT> = 2 then <NWORDS> is the actual amount of data transferred in machine words.

SUBROUTINE IDAC(NCHAN,NVALUE)

Note: Used only for audio output.

Function: Integer digital to analogue conversion. Causes a specified analogue output channel to be set to a given voltage.
Entry: <NCHAN> is the channel number.
<NVALUE> is proportional to the output voltage.

SUBROUTINE INTLEVEL(NLEVEL)
Note: Relevant only to operation on the Sigma 5.
Function: The program is connected to the specified interrupt level.
Entry: <NLEVEL> is the number of the interrupt level.

FUNCTION ISBIT(NARRAY,NPTR)
Function: To obtain a specified bit from an array.
Parameters: <NARRAY> is an integer array.
Entry: <NARRAY> is the array from which the bit is to be obtained.
<NPTR> is the position of the bit within the array, the high-order bit of the first word being considered as bit 0.
Exit: <ISBIT> contains the value of the bit, 0 or 1.

SUBROUTINE JOY(NCHAR,NX,NY)
Function: Displays crosswires on the screen of the terminal, waits for a single character to be typed, and returns the co-ordinates of the position at which the crosswires intersect when the character is typed.
Parameters: <NCHAR> is an integer array.
Exit: If a carriage-return is input, <NCHAR>, <NX> and <NY> remain unchanged, otherwise:
<NCHAR> contains the EBCDIC code for the character typed in the high order byte and blanks in the remaining bytes.
<NX> is the X co-ordinate of the vertical crosswire.
<NY> is the Y co-ordinate of the horizontal crosswire.

SUBROUTINE LSEG(N)
Function: To load a program overlay segment into core.
Entry: <N> is the identification number of the segment to be loaded, or zero.
Exit: If either <N> is zero or the required segment is the one most recently loaded into core, no action is taken. Otherwise the specified segment is loaded.

SUBROUTINE MBS(NTEXT1,N1,NTEXT2,N2,NLENG)
Function: To copy a byte string to a new position.
Parameters: <NTEXT1> and <NTEXT2> are integer arrays.
Entry: <NTEXT1> contains the byte string to be copied.
<N1> is the starting byte position within <NTEXT1> of the byte string to be copied, the most significant byte of the first word being considered as byte 1.
<NLENG> is the length in bytes of the byte string to be copied.
Exit: <NTEXT2> is equal to <NTEXT1> in the first <NLENG> bytes. The remainder of <NTEXT2> is unchanged.

FUNCTION NBYTE(NARRAY,NPTR)
Function: To obtain a specified byte from an array.
Parameters: <NARRAY> is an integer array.
Entry: <NARRAY> is the array from which the byte is to be obtained.

<NPTR> is the position of the byte within the array. The high order byte of the first word is byte 1.

Exit: <NBYTE> contains the value of the byte, in the range 0 to 255.

FUNCTION NHW(NARRAY,NPTR)

Function: To obtain a specified double-byte (Sigma 5 halfword) from an array.

Parameters: <NARRAY> is an integer array.

Entry: <NARRAY> is the array from which the double-byte is to be obtained.
<NPTR> is the position of the double-byte within the array. The high-order two bytes of the first word are double-byte 1.

Exit: <NHW> contains the value of the double-byte, in the range 0 to 32767.

FUNCTION NPOWER(N)

FUNCTION: To evaluate an integral power of 2.

Entry: <N> is a non-negative integer.

Exit: <NPOWER> is 2 to the power <N>.

Note: NPOWER(N) is equivalent to 2**N.

SUBROUTINE NUPAGE(SPACE)

Function: Start new page on digital plotter.

Parameters: <SPACE> is real.

Entry: <SPACE> is the number of inches between pages.

SUBROUTINE OVMOVE

Note: This routine is specific to operation on the Sigma 5 computer and is not required for implementation on any other machine. Its sole purpose is to reduce the amount of core storage occupied by the program by re-allocating system tables associated with the program more efficiently.

Function: The segment table for the overlay loader is repacked in a form suitable for use by subroutine <LSEG>. The program's temporary stack is set up in the area occupied by the original segment table. The routine is executed at the start of a run and is not called anywhere within the program.

SUBROUTINE PLOT(X,Y,IC)

Function: To move the pen of the digital plotter from its current position to a new position.

Parameters: <X> and <Y> are the page co-ordinates in inches.
<IC> = 1 for no change in vertical position of pen, 2 for pen down (draw line), 3 for pen up (no line).

SUBROUTINE SKIPFILE(NUNIT,NFILES)

Function: To skip files on a magnetic tape. The tape is wound forward until the specified number of tape marks (end-of-file marks) have been read, then positioned at the start of the following file.

Entry: <NUNIT> is the FORTRAN unit number assigned to the tape.
<NFILES> is the number of files to be skipped.

SUBROUTINE STBIT(NNBIT,NNBYTE,NPTR)

Function: To store a bit into a word.

Entry: <NNBIT> contains the value of the bit to be stored (0 or 1).
<NPTR> is the bit position within the word in which the bit is to be stored, the low order bit of the word being considered as bit 8.
Exit: <NNBYTE> is modified only in the specified bit position.

SUBROUTINE STBYTE(NNBYTE,NARRAY,NPTR)
Function: To store a byte into an array.
Parameters: <NARRAY> is an integer array.
Entry: <NNBYTE> contains the value in the range 0 to 255 to be stored.
<NPTR> is the byte position in the array at which the value is to be stored, the high order byte of the first word being considered as byte 1.
Exit: <NARRAY> is modified only in the specified byte position.

SUBROUTINE STHW(NNHW,NARRAY,NPTR)
Function: To store a double-byte (Sigma 5 halfword) into an array.
Parameters: <NARRAY> is an integer array.
Entry: <NNHW> contains a 15-bit integer value in the range 0 to 32767 to be stored.
<NPTR> is the double-byte position in the array at which the value is to be stored, the high order two bytes of the first word being considered as double-byte 0.
Exit: <NARRAY> is modified only in the specified double-byte position.

SUBROUTINE SY
Note: Relevant only to operation on the Sigma 5.
Function: To change the system write-protect indicator for the printer output file.

SUBROUTINE SYMBOL(X,Y,HEIGHT,IADD,THETA,N)
Function: To draw a character string on the digital plotter.
Parameters: <X>, <Y>, <HEIGHT> and <THETA> are real.
<IADD> is an integer array.
Entry: <X> and <Y> are the page co-ordinates of the lower left corner of the first character of the string.
<HEIGHT> is the height of the characters in inches. Spacing is 6/7 height.
<IADD> contains the character string in EBCDIC code.
<THETA> is the angle between the base of the characters and the horizontal in degrees.
<N> is the number of characters to be drawn.

SUBROUTINE TIMER(N,NLEVEL)
Note: Used only for audio output.
Function: To trigger the program's system interrupt level at regular intervals. If the routine is called with no arguments, triggering of the interrupt is stopped.
Entry: <N> is the period in clock pulses. The clock frequency is 2000 pulses per second.
<NLEVEL> is the interrupt level number.

SUBROUTINE TPLOT(NTYPE,NX,NY)
Function: To display graphics on the terminal.
Entry: <NTYPE> is 1 if a line is to be drawn, 0 otherwise.
<NX> is the X co-ordinate of the new cursor position.

<NY> is the Y co-ordinate of the new cursor position.
Exit: The cursor of the terminal is moved from its current position to the new position specified, with or without drawing a straight line between the two points.

SUBROUTINE URD(NLENG,NTEXT)
Function: To read a character from the terminal.
Entry: <NLENG> is 1, indicating number of characters to be read.
<NTEXT> is an integer array of dimension 1.
Exit: <NTEXT> contains the EBCDIC code for a character read from the terminal in the high-order byte.

SUBROUTINE UWT(NLENG,NTEXT)
Function: To write a character string to the terminal.
Parameters: <NTEXT> is an integer array.
Entry: <NLENG> is a non-negative integer indicating the number of characters in the text string.
<NTEXT> contains the text string in the first <NLENG> bytes.

SUBROUTINE WAIT
Function: To wait for completion of an input/output operation.
Note: On the Sigma 5 this permits other programs to use the processor time during an input or output operation initiated using <BUFFIN> or <BUFFOUT>.

A3.4 Fixed byte tables

A number of tables used in the program contain information, arranged in bytes, which is not modified during execution of the program. The contents of these tables have been defined in the program source rather than being set up dynamically at execution time because:
(1) These tables are likely to be modified rarely, if at all, when the program is in production use;
(2) Extra program code and execution time required to read the tables dynamically is thereby avoided;
(3) The tables can be located in individual program segments within the overlay structure, reducing overall program execution size.

The following tables are automatically encoded in the form of BLOCK DATA subprograms from corresponding source tables (listed in appendix A1.3) by a separate program (listed in appendix A1.2):

Message table
Input character conversion table
Display co-ordinate tables
Braille music sign table
Literary braille contraction table
Hershey co-ordinate table

Automatic encoding of these tables is used in preference to manual encoding to retain the advantages of reading and setting up of tables at execution time, viz.:

(1) The source tables can be specified in a natural form;
(2) The tables can be modified easily;
(3) The format for direct coding of binary information in FORTRAN

programs is machine dependent. With automatic encoding, minor modifications in the auxiliary program may be used to generate tables in a form acceptable to a particular FORTRAN compiler.

The source for the fixed byte tables is in card-image form (80-character records). The end of each table is indicated by 999 in columns 1 to 3, except for the Hershey co-ordinate table for which it is indicated by -1 in columns 4 to 5. The format of the lines of each table is as follows:

Message table

Cols 1-3	Message identification number.
Col 5	Space if the message is to be displayed at the current cursor position; asterisk if the message is to be displayed starting on a new line.
Cols 20-67	Text of message. A message which includes trailing blanks is followed by a vertical bar character, which is not treated as part of the message itself. The "@" sign is interpreted as a bell character.

Input character conversion table

Cols 1-3	Internal code
Col 5	Input character
Cols 20+	Comment

Display co-ordinate tables

Cols 1-3	Symbol identification number (on first line for each symbol only; otherwise blank).
Col 5	Space means draw line from previous cursor position; asterisk means position cursor only.
Cols 6-8	Signed x co-ordinate increment from current position (in range -63 to +63).
Col 9	Asterisk for the last co-ordinate pair of a symbol; space otherwise.
Cols 10-12	Signed y co-ordinate increment from current position (in range -63 to +63).
Cols 20+	Comment

Braille music sign table

Cols 1-3	Sign identification number
Col 5	Space means last cell of a sign; asterisk otherwise.
Cols 7-8	Braille cell code (in range 0 to 63).
Col 20+	Comment

Braille contraction table

Cols 7-8	Context class, sum of following values: 1 may be followed by a letter 2 may be followed by a space 4 may be followed by another character 8 may be preceded by a letter 16 may be preceded by a space 32 may be preceded by another character.
Cols 11-16	Up to three braille cell codes
Cols 20-29	Text of contraction

Hershey co-ordinate table

This table is an extract from Hershey's co-ordinate

tables for graphic symbols (Wolcott & Hilsenrath 1976)

Cols 1-5	Symbol identification number (first line of symbol only, otherwise blank).
Cols 9-80	Pairs of signed integers (see section A1.3 for precise format). For each symbol, the first pair (not used by CIMBAL) give the minimum and maximum x co-ordinates of the symbol; for the remaining pairs: (-64,-64) means end of symbol; (-64,0) means the next co-ordinate pair is position only (no line to be drawn); otherwise numbers are the x and y co-ordinates respectively with respect to a fixed origin.

A3.5 Internal tables

Fixed byte tables

The input character conversion table <MCODES> is a byte table of internal codes corresponding to characters typed on the terminal, indexed by the EBCDIC character code. For the remaining fixed byte tables the first double-byte contains the number (n) of entries in the table. The following n double-bytes contain pointers to the start of the corresponding entries within the table. The rest of the table contains the entries, not necessarily in order. The format of each entry depends on the type of table as follows:

Message table

Byte 1 bit 0	0 display at current position; 1 start new line.
bits 1-7	number of characters in message.
Following bytes:	text of message in EBCDIC code.

Display co-ordinate tables (including Hershey table)

Odd bytes: bit 0	0 position cursor only; 1 draw line.
bits 1-7	X co-ordinate increment plus 64.
Even bytes:bit 0	0 not last byte of entry; 1 last byte of entry.
bits 1-7	Y co-ordinate increment plus 64.

Braille music sign table

A sequence of bytes of the form:

bit 0	0 = last byte of entry; 1 otherwise.
bit 1	0
bits 2-7	braille cell code.

Literary braille contraction table

Byte 0	length of text of contraction
Byte 1	context code (value as in external table, see A3.4)
Following bytes	text of contraction in EBCDIC code
Following bytes	corresponding braille sign in same form as braille music sign table entries.

Edit table

The edit table is used where non-sequential insertion and deletion of entries is necessary. It contains three circular bi-directionally linked lists of entries, used for editing of measures, doubling of braille signs, and storage of state tables respectively. <LELIST> contains the number of lists in the edit table. <LESIZE> contains the number of available positions in the table. A fourth list links unused positions of the table. <JEFREE> contains a pointer to the first entry of this list. The forward link field of the last entry is zero. Each entry comprises four information bytes (stored in <MLIST1>, <MLIST2>, <MLIST3> and <MLIST4>), a forward link pointer (stored in <MFLINK>), and a backward link pointer (stored in <MBLINK>). The first three positions of the edit table contain dummy entries representing the head of the respective lists. Each is forward linked to the first entry of the corresponding list and backward linked to the last entry of the list. For an empty list it is forward and backward linked to itself. The information fields associated with these three positions are not used.

The information fields of the remaining entries contain:

List 1 (items for editing):
Byte 0 pointer to item in encoded measure
Byte 1 length in bytes of encoded item
Byte 2 X co-ordinate of item, divided by 4
Byte 3 Y co-ordinate of item, divided by 4

List 2 (currently doubled braille signs):
Byte 0 braille sign number
Byte 1 line number and in-accord part (value of <JVOIC>)
Byte 2 doubling indicator (value of <ISDUB>)
Byte 3 number of notes remaining in doubled sequence

List 3 (state tables):
One state table is stored for each line other than the current one for which any state table entry is non-zero. Each state table occupies two linked edit table entries. <MEMPTR> contains pointers to the first entry of each such pair.

State tables

The state tables <MCMEM> and <MSMEM> represent the "state" (in the sense described in section 2.6) of the printed music extended signs at a given point on the current line. Each state table comprises eight bytes:

Bytes 0-3 extended sign indicators:
0 = sign not open
1 = sign open but display co-ordinates not evaluated
2 = sign open, starts off left hand side of screen
3+ = X co-ordinate of start of sign, divided by 8
Byte 0 (outer) slur
Byte 1 inner slur
Byte 2 crescendo (bit 0 = 0) or decrescendo (bit 0 = 1)
Byte 3 ottava above (bit 0 = 0) or below (bit 0 = 1)
Byte 4 display shift for 8vas (value of <NSHIFT>)

Bytes 5-7 zero (unused)

Location table

The location table contains up to 10 entries representing potential entry points for accessing the current section of the disc file. A measure number is entered in the location table only if the state table for each line contains only zeros at the start of the measure and, for printed music, the clefs are the same as those at the start of the section. Each entry comprises a measure number (stored in <MFIND>), a time and key signature value (stored in <MCTK>), and the file address of the start of the measure (stored in <MLOCN>). <NLOCS> contains the number of entries of the table currently occupied.

Key tables

The key tables <MKPCH>, <MTKPCH> and <MTKEY> represent the modification of note pitch by accidentals. Each key table comprises seven bytes corresponding to the notes C, D, E, F, G, A, and B respectively. Each byte has one of the values:

1	double flat	4	sharp
2	flat	5	double sharp
3	natural		

Appendix 4. CIMBAL Users' reference manual

Contents

(Sections not concerned with input of printed music notation are marked with an asterisk).

A4.1 Introduction

This reference manual contains information on the use of CIMBAL, an interactive computer program for the input, editing, storage and translation to braille of printed musical notation.

The manual contains a full description of the use of CIMBAL, although some of the information given is not required by the user concerned only with input of printed music notation, who may disregard sections marked with an asterisk.

The information given in this manual is believed to describe the behaviour of CIMBAL correctly. Users are requested to make a note of any apparent discrepancy between the manual and the actual behaviour of the program.

Non-standard terminology

The following words have the special meanings given below when used in this manual:

Bar — A measure of music, in the conventional sense, including the music for all parts of the piece.

Measure — A measure of music, in the conventional sense, for one part only.

Part — The line of music associated with one stave.

Staff — A set of parallel staves, one corresponding to each part of the piece.

Stave — A single set of five parallel lines in the printed version and the associated music representing a single part.

A4.2 Operating details on Sigma 5 computer

Sections A4.2.1, A4.2.2 and A4.2.4 are specific to the implementation on the Sigma 5 computer in the Engineering Department at the University of Warwick, using the RBM operating system.

*A4.2.1 Program requirements

Two versions of CIMBAL are available on the Sigma 5 computer: foreground and background. These are identical except that the background version has no audio output (see "PLAY command") and cannot be time-shared with other computer users, while the foreground version does not include the high-quality display or graph-plotter output options, which require more core space than is available. The foreground version is the one normally used except when graph plotter output is required. The resources required by CIMBAL are:

Core locations X'4E00' to X'6FFF' (foreground version).
Interrupt level X'69' (foreground version only).
A 1200-baud COC line and graphical display unit.
Magnetic tape unit 2 (foreground version) or unit 0 (background version) during execution of any command requiring an archive tape.
Magnetic tape unit 1 for output during execution of the "COPY" command, or for graph-plotter output.
The program file is disc file FP,MUSIC (foreground version) or BP,JHMUSIC (background version).
The music data file on disc is D2,JHMUSIC (foreground version) or D1,JHMUSIC (background version).
The disc file for braille output is D2,FGDTEMP.
The disc file for lineprinter output is D1,PRINTER.

The graphical display unit normally used is a Tektronix 4002. It is also possible to use a DEC GT40 (see "Running CIMBAL" and the "CHANGE" command), but the Tektronix is preferable unless no printed music display is required.

A4.2.2 Loading CIMBAL

CIMBAL is run interactively from a graphical display unit. If the Tektronix is used, the illuminated buttons should be set thus:
"ON-LINE/LOCAL" button: set to "ON-LINE".
"INPUT" button: both the "KEYBOARD" and "AUX" sections should be lit. If not, press the button until they are. This allows use of the joystick for editing.
"ASCII/TTY" button: initially set to "TTY" for communication with the operating system. This automatically converts all letters typed to upper case.

Type three or more "ESCAPE"s in quick succession to invoke the Sigma 5 operating system ("ESCAPE" is SHIFT+ESC on the Tektronix, or "ALT" on the GT40). The operating system should respond with the message:

SIGMA 5 FGD EXECUTIVE ON LINE <n>

where <n> is a number in the range 2 to 5, followed by a $$ prompt. Type "RUN MUSIC" followed by a carriage-return to load CIMBAL. If CIMBAL is successfully loaded it will erase the screen and display a

message of the form:

Music program foreground version compiled on <date>.

From here on, the user communicates directly with CIMBAL as described below.

A4.2.3 Running CIMBAL

If the music disc file is not correctly set up, CIMBAL will first display the message "INITIALISING DISC FILE", and there will be a few seconds delay while the file is linked. CIMBAL will then display the question "SELECTION NUMBER?". The selection number allows the user to choose certain options. The selection number, which is typed followed by a carriage-return, is the sum of the values listed below for the options which the user initially wishes to select. A fuller description of the meaning and use of these options is given in section A4.4.6, which also explains how any of these options may subsequently be changed. If a plain carriage-return is typed for the selection number, none of the options will be selected. The options are:

1 Input all note pitches as if in treble clef
2 The terminal used is a GT40
4 Debugging mode (used only for program testing)
8 High quality display (not normally recommended)
16 Translate text to grade 1 braille only
32 SAGEM model of embosser to be used
64 Extra spacing between staves on display

Typing an illegal selection number will cause CIMBAL to repeat the question. When the user has typed a correct selection number or plain carriage-return, CIMBAL will enter command status, indicated by the message "NEXT COMMAND?". If a Tektronix terminal is used CIMBAL will also display the message "PRESS ASCII BUTTON FOR FULL CHARACTER SET". In this case, press the button so that the "ASCII" section is lit. Continue as described in section A4.4.

A4.2.4 Associated programs

The braille file created for embossing may be embossed using the braille embossing program LED2. The printer file containing braille for printing on the lineprinter may be printed using the PRINT program. The graph-plotter file containing printed music notation for plotting on the digital plotter may be printed by loading the tape on tape unit 2 and using the PLOT program. Details of these programs are given in separate documentation not included in this manual.

A4.3 Music storage

This section describes from the user's point of view the way in which the printed and braille music notations are stored. It defines some terms needed to understand section A4.4 which describes the use of "commands".

Each piece of music is stored in a "file", either on disc or on magnetic tape. Only one disc file is used by CIMBAL for storage of music. This contains the piece of music currently being accessed by the user. There may be any number of files of music stored on magnetic tapes. These are retained indefinitely unless deleted by the user using the "ERASE" command. The only commands which operate on a magnetic tape file are "SAVE", which creates the file as a copy of the current disc file, "RESTORE", which copies a magnetic tape file to the disc file, "ERASE", which deletes a file from the tape, and "COPY", which copies the whole archive tape to a new tape.

A4.3.1 Disc file

The piece of music currently accessible to the user is stored in a disc file. Only one piece of music may be stored in this file at any given time. A summary of the contents of the disc file may be displayed using the "SUMMARY" command. The disc file is in three sections containing different forms of the current piece of music as follows:

1 Printed music. This section contains a representation of the music as printed. It is created by typing on the terminal.

2 Unformatted braille. This section contains a representation of the braille music, arranged bar by bar without any formatting.

3 Formatted braille. This section contains a representation of the braille music formatted, apart from pagination, as it will finally appear in the braille output.

Each section of the disc file has a "protection status" associated with it which is one of the following:

1 "Empty". This means that nothing is currently stored in the section.

2 "Unprotected". This means that there is music stored in the section and either a copy of the file has been saved on magnetic tape, or the section can be re-created without any further input or editing.

3 "Protected". This means that there is music stored in the section and no copy of the file is saved on magnetic tape, and further input or editing would be required to re-create the section if it were destroyed.

The purpose of the protection status is to safeguard the user against accidentally deleting a file which has not been saved on magnetic tape.

At any given time, commands which access the disc file operate on one particular section of the file, referred to as the "current section", depending on the current "mode" of CIMBAL, which is one of: "printed music", "unformatted braille", or "formatted braille". The exceptions to this rule are the "SAVE" and "RESTORE" commands which copy the whole disc file to or from magnetic tape, regardless of the current mode.

At the start of each run of CIMBAL, the mode is set to "printed music". For input, editing and display of printed music this is the only mode required. However, the mode can be changed if necessary by the user using the "MODE" command. Certain commands cause CIMBAL to change its mode automatically (for example the "BRAILLE" and "FORMAT" commands). Any change of mode is accompanied by a message on the screen informing the user of the new mode, protection status, and number of bars and staves or lines of music stored in the corresponding section of the file.

The "printed music" and "unformatted braille" sections of the disc file are divided into "bars", corresponding to the bars of the music and identified by bar number. Each bar is divided into one or more "measures", corresponding to the measures of the music, and identified by stave number if the section has more than one stave. The first bar of a piece is considered as bar 1. Thus if this bar is incomplete, bar numbers shown in the print may be one less than corresponding numbers used by CIMBAL.

The "formatted braille" section of the disc file is divided into "lines", corresponding to the lines of braille, and identified by line number. Braille pagination is not included, as this is determined by the type of embosser used at the time of final output. Further details of the braille sections are given in section A4.9 ("Braille editing"). The terms "bar" and "measure" will be taken in formatted braille context to refer to a line of braille.

Each measure contains a number of "items". An item is the basic unit of musical or braille information for input, editing, and storage. The "printed music" section contains four types of item:

1 Control item, defining the structure of the music
2 Separate sign item, defining markings not associated with a specific note and not covered by the other item types
3 Text item, defining text in the music
4 Note item, defining notes and rests and their associated markings.

The "unformatted braille" and "formatted braille" sections contain only one type of item:

Braille item, defining a braille cell or (in the unformatted braille section only) a format control code.

The various items of different types and their use are described in section A4.5.

A4.3.2 Magnetic tape files

A music archive tape is used for storage of files of music, either permanently or between runs of CIMBAL. Five of the commands ("COPY", "ERASE", "FILES", "RESTORE", and "SAVE") operate on the archive tape, and a music archive tape should be loaded during execution of these commands. The tape is rewound on completion of each of these commands, and may then be removed if the tape unit is required by other computer users. Different archive tapes may used at different times during the same run. All commands using an archive tape initially display the name of the currently loaded tape on the screen.

At the start of each run of CIMBAL, the mode is set to "printed music". For input, editing and display of printed music this is the only mode required. However, the mode can be changed if necessary by the user using the "MODE" command. Certain commands cause CIMBAL to change its mode automatically (for example the "BRAILLE" and "FORMAT" commands). Any change of mode is accompanied by a message on the screen informing the user of the new mode, protection status, and number of bars and staves or lines of music stored in the corresponding section of the file.

The "printed music" and "unformatted braille" sections of the disc file are divided into "bars", corresponding to the bars of the music and identified by bar number. Each bar is divided into one or more "measures", corresponding to the measures of the music, and identified by stave number if the section has more than one stave. The first bar of a piece is considered as bar 1. Thus if this bar is incomplete, bar numbers shown in the print may be one less than corresponding numbers used by CIMBAL.

The "formatted braille" section of the disc file is divided into "lines", corresponding to the lines of braille, and identified by line number. Braille pagination is not included, as this is determined by the type of embosser used at the time of final output. Further details of the braille sections are given in section A4.9 ("Braille editing"). The terms "bar" and "measure" will be taken in formatted braille context to refer to a line of braille.

Each measure contains a number of "items". An item is the basic unit of musical or braille information for input, editing, and storage. The "printed music" section contains four types of item:

1 Control item, defining the structure of the music
2 Separate sign item, defining markings not associated with a specific note and not covered by the other item types
3 Text item, defining text in the music
4 Note item, defining notes and rests and their associated markings.

The "unformatted braille" and "formatted braille" sections contain only one type of item:

Braille item, defining a braille cell or (in the unformatted braille section only) a format control code.

The various items of different types and their use are described in section A4.5.

A4.3.2 Magnetic tape files

A music archive tape is used for storage of files of music, either permanently or between runs of CIMBAL. Five of the commands ("COPY", "ERASE", "FILES", "RESTORE", and "SAVE") operate on the archive tape, and a music archive tape should be loaded during execution of these commands. The tape is rewound on completion of each of these commands, and may then be removed if the tape unit is required by other computer users. Different archive tapes may used at different times during the same run. All commands using an archive tape initially display the name of the currently loaded tape on the screen.

The archive tape contains an index at the beginning, followed by the files of music. The index contains the tape name, number of files on the tape, and identifier, name, and version number for each file. The tape index is updated on successful completion of each "ERASE" or "SAVE" command.

A magnetic tape file is created using the "SAVE" command, and may be deleted only by use of the "ERASE" command. The "RESTORE" command, which makes the disc file a copy of a specified tape file, does not delete or affect the file stored on the tape. The identifier and name of a file on the tape are as input in response to the corresponding questions during input of the piece, except that accented characters are converted to the corresponding unaccented characters, and the special codes input using the "@" sign (see section A4.5.5) are removed.

When the disc file is copied to a magnetic tape using the "SAVE" command, it is assigned a version number on the tape, to distinguish it from any other file already on the tape which has the same name. Thus a piece of music may be saved at intermediate stages of input or editing, and any of the saved versions may subsequently be recovered. Version numbers apply only to magnetic tape files and not to the disc file, which may or may not currently be a copy of a tape file. The version number is assigned at the time the file is saved, according to the following rule: if there are no files already on the archive tape with the same name, the new file is given version number 1. If there are already files on the archive tape with the same name, the new file is given a version number one greater than that of the most recent such file. The highest version number allowed is 255, after which the version number reverts to 1.

CIMBAL will not attempt to write to any tape which does not begin with an index in the correct format, and is thus safeguarded against corrupting non-music tapes. A new music archive tape must first be initialised using a separate program or the "COPY" command.

A4.4 Commands

The user communicates with CIMBAL by giving a series of "commands", the last of which must be the "STOP" command. A command is given by typing the first two letters of the command name in either upper or lower case. CIMBAL indicates that it is in "command status" (ready and waiting to receive a command) by displaying the message "NEXT COMMAND?" at the left hand side of the screen. It enters command status following the display of initial messages at the start of the run, and subsequently on completion of each command. Execution of most commands may be cancelled and the program returned to command status by pressing the interrupt or break button, except during input of music, in which case the interrupt has a special use (see section A4.5). If the first character of the command name is wrongly typed it may be cancelled by typing a question mark, and the "NEXT COMMAND?" message will be re-displayed.

If the two characters typed by the user are the first two letters of a recognised command name, the remainder of the command name is displayed, then either the command is executed or an error message is displayed explaining why the chosen command is not allowed in the

The archive tape contains an index at the beginning, followed by the files of music. The index contains the tape name, number of files on the tape, and identifier, name, and version number for each file. The tape index is updated on successful completion of each "ERASE" or "SAVE" command.

A magnetic tape file is created using the "SAVE" command, and may be deleted only by use of the "ERASE" command. The "RESTORE" command, which makes the disc file a copy of a specified tape file, does not delete or affect the file stored on the tape. The identifier and name of a file on the tape are as input in response to the corresponding questions during input of the piece, except that accented characters are converted to the corresponding unaccented characters, and the special codes input using the "@" sign (see section A4.5.5) are removed.

When the disc file is copied to a magnetic tape using the "SAVE" command, it is assigned a version number on the tape, to distinguish it from any other file already on the tape which has the same name. Thus a piece of music may be saved at intermediate stages of input or editing, and any of the saved versions may subsequently be recovered. Version numbers apply only to magnetic tape files and not to the disc file, which may or may not currently be a copy of a tape file. The version number is assigned at the time the file is saved, according to the following rule: if there are no files already on the archive tape with the same name, the new file is given version number 1. If there are already files on the archive tape with the same name, the new file is given a version number one greater than that of the most recent such file. The highest version number allowed is 255, after which the version number reverts to 1.

CIMBAL will not attempt to write to any tape which does not begin with an index in the correct format, and is thus safeguarded against corrupting non-music tapes. A new music archive tape must first be initialised using a separate program or the "COPY" command.

A4.4 Commands

The user communicates with CIMBAL by giving a series of "commands", the last of which must be the "STOP" command. A command is given by typing the first two letters of the command name in either upper or lower case. CIMBAL indicates that it is in "command status" (ready and waiting to receive a command) by displaying the message "NEXT COMMAND?" at the left hand side of the screen. It enters command status following the display of initial messages at the start of the run, and subsequently on completion of each command. Execution of most commands may be cancelled and the program returned to command status by pressing the interrupt or break button, except during input of music, in which case the interrupt has a special use (see section A4.5). If the first character of the command name is wrongly typed it may be cancelled by typing a question mark, and the "NEXT COMMAND?" message will be re-displayed.

If the two characters typed by the user are the first two letters of a recognised command name, the remainder of the command name is displayed, then either the command is executed or an error message is displayed explaining why the chosen command is not allowed in the

circumstances. If the two characters typed are not the first two characters of a recognised command, the "OPTIONS" command is executed, listing the available commands. This listing may be stopped by pressing the interrupt button. Error messages are listed in section A4.7, with details of their meaning and what to do about them.

The use of the individual commands is described in detail in sections A4.4.3 to A4.4.26, in alphabetical order of command name. All commands are available in any mode unless otherwise indicated. The available commands are:

ADD	extend the current section
BRAILLE	convert printed music to unformatted braille
BUILD	input a new piece of music
CHANGE	change specified program parameters
CLEAR	delete the current section
COPY	copy an archive tape to a new tape
DELETE	delete a specified range of bars
DISPLAY	display a selected part of the current section
EDIT	edit a selected measure of the current section
EMBOSS	create an embosser file from the current section
ERASE	erase a file from an archive tape
FILES	list the files on an archive tape
FLAG	change a braille "flag" value
FORMAT	convert unformatted braille to formatted braille
INSERT	insert a new bar before a specified bar
MODE	change the program mode
OPTIONS	list the available command options
PLAY	play the current section using audio output
PRINT	print music using the graph plotter or lineprinter
RESTORE	restore the disc file from an archive tape
SAVE	copy the disc file to an archive tape
SCALE	change the horizontal display scale
STOP	release CIMBAL
SUMMARY	display summary information for the disc file

A4.4.1 Summary of use of commands

To input music:
(1) To start a new piece of music, use the "BUILD" command.
(2) To add new bars to the end of a piece of music already stored in the disc file, use the "ADD" command.
(3) To add new bars anywhere else in a piece of music already stored in the disc file, use the "INSERT" command.
(4) To add or change items in an existing measure, use the "EDIT" command.

To delete music:
(1) To delete a magnetic tape file, use the "ERASE" command.
(2) To delete the current section of the disc file, use the "CLEAR" command.
(3) To delete specified bars from the disc file, use the "DELETE" command.
(4) To delete or change items within a measure, use the "EDIT" command.

To copy music:
(1) To copy the disc file to magnetic tape, use the "SAVE" command.
(2) To copy a magnetic tape file to the disc file, use the "RESTORE" command.
(3) To copy an archive tape to a new tape, use the "COPY" command.

To produce braille:
Use the "BRAILLE", "FORMAT", and "EMBOSS" commands in succession.

To display:
(1) To display a listing of the files on a magnetic tape, use the "FILES" command.
(2) To display a summary of the music in the disc file, use the "SUMMARY" command.
(3) To display the contents of the disc file, use the "DISPLAY" command.
(4) To display a listing of the available commands, use the "OPTIONS" command or any unrecognised command name.
(5) To hear the contents of the disc file, use the "PLAY" command.
(6) To print the contents of the disc file on the graph plotter (printed music) or lineprinter (braille), use the "PRINT" command.

To finish a session and release CIMBAL, use the "STOP" command.

A4.4.2 Specifying bar and stave numbers

Certain commands request the user to type bar numbers and/or stave numbers. Each number is input as a series of digits followed by a carriage-return. A question mark acts as a backspace character in typing a number. If the user types any other non-digit character in the middle of a number, CIMBAL will ignore the character and sound the bell. If a number is not within the required range, CIMBAL will display the message:

MUST BE IN RANGE <m> TO <n>

where <m> and <n> are the lowest and highest values the number may take. If this happens, type the correct number, or press the interrupt button to cancel the command and return to command status.

A4.4.2.1 Specifying bar numbers

The following commands require the user to specify a bar number or range of bars, unless the current section contains only one bar: "BRAILLE", "DELETE", "DISPLAY", "EDIT", "FLAG", "FORMAT", "INSERT", "PLAY", "PRINT".

If a single bar number is required ("EDIT", "FLAG" and "INSERT" commands) CIMBAL will display the question "BAR?". Type the number of the required bar.

If a range of bars is required (the remaining commands) CIMBAL will display the question "FROM BAR?". Type the number of the first bar of the required range. If this is not the last bar of the section, CIMBAL will then display the question "TO BAR?". Type the number of the last bar of the required range.

If a zero or a plain carriage-return is input for the first bar number, no second number is requested and the effect is:

"BRAILLE", "DISPLAY", "FORMAT", "PLAY", and "PRINT" commands: the command operates on the entire current section.
"EDIT" command: The file header is displayed for editing.
"FLAG", "INSERT" and "DELETE" commands: not allowed.

A4.4.2.2 Specifying stave numbers

The following commands require the user to specify stave numbers if the current section has more than one stave: "ADD", "BRAILLE", "BUILD", "DISPLAY", "EDIT", "FORMAT", "INSERT", "PLAY", "PRINT".

For the "EDIT" and "FLAG" commands a single stave number is required. CIMBAL will display the question "STAVE?". Type the stave number.

For the remaining commands in the above list, CIMBAL will display the question "SELECT STAVES?". Any combination of staves may be selected by typing a sequence of stave numbers and/or pairs of stave numbers separated by a hyphen, each separated by a comma. The numbers may be typed in any order and CIMBAL will sort them into the correct order. A pair of numbers separated by a hyphen indicates that all staves from the lower of the two numbers to the higher are to be selected. A zero or plain carriage-return typed in response to this question indicates that all staves are to be selected.

Example: 3-5,1,7 selects staves 1, 3, 4, 5, and 7

A4.4.3 ADD command

The "ADD" command is used to add to the end of an existing section. It may not be used if the current section is "empty" - in this case the "BUILD" command should be used instead. The command is used in the same way as the "BUILD" command except that the initial specifications are omitted. If there is room for all the selected staves on a single display, the last bar of music already stored in the current section is first displayed.

*A4.4.4 BRAILLE command

The "BRAILLE" command is used to create a section of unformatted braille from the current section. It may be used only in printed music mode. It is not allowed if the unformatted braille section of the disc file is "protected". If the command is accepted, any music already stored in the unformatted braille section is first deleted. On completion of the translation, the mode is changed to unformatted braille, and CIMBAL returns to command status. The range of bars and staves to be translated are given as described in section A4.4.2. The new unformatted braille section is given "unprotected" status.

Each instrumental part produces one line in the unformatted braille section; each vocal part produces two lines (one for words and one for music). A maximum of four lines of unformatted braille music (not counting text lines) is allowed in the current implementation.

A4.4.5 BUILD command

The "BUILD" command is used to input a new piece of music to the current section. It may not be used if the current section is "protected" - in this case its status must first be changed to "unprotected", either by copying the file to magnetic tape using the "SAVE" command or by deleting the current section using the "CLEAR" command. If the command is accepted, any music already stored in the current section is first deleted. Staves are selected as described in section A4.4.2.

The screen is cleared and some questions are asked, the answers to which comprise the initial specifications for the piece of music. The music itself is then typed using the special music input language. Full details of the input language and the questions and answers for the initial specifications are given in section A4.5. The music is displayed on the screen as it is input. Whenever the screen is full the message "PRESS CARRIAGE-RETURN" is displayed; press the carriage-return key to clear the screen and continue. It is not necessary to input a whole piece at one time: a piece may be partially input and later extended using the "ADD" command. Any mistakes made during input may be corrected immediately if noticed during input of the same measure, otherwise they may be corrected later using the "EDIT" command. The new current section is given "protected" status.

A4.4.6 CHANGE command

The "CHANGE" command is used to change the values of certain parameters affecting the operation of CIMBAL. Each of these parameters except "debugging" has one of the two values "TRUE" or "FALSE". One of these two values is assigned to the parameter at the start of a run, according to the selection number specified by the user. Whenever the parameter is changed using the "CHANGE" command, it takes the opposite value from its current value.

CIMBAL displays the message "TYPE L FOR OPTION LIST". Type either the letter L to obtain a listing of the changeable parameters, or a carriage-return to omit this listing. The listing of the changeable parameters includes their identification numbers and current values. Any combination of these parameters may be changed with one execution of the "CHANGE" command. The numbers of the parameters to be changed are specified in the same way as stave numbers (see section A4.4.2). The new values for the changed parameters are displayed. The changeable parameters are:

1. Transpose: If this is FALSE, notes are input using their actual pitch values. If it is TRUE, notes are input as if they were written in the treble clef, and CIMBAL transposes them to their correct values before storing and displaying them.

2. GT40: This parameter should be FALSE if a Tektronix terminal is being used, TRUE if a GT40 terminal is being used. If a GT40 terminal is being used it is advisable to change this parameter before inputting or displaying any music.

3. Debugging: should normally be left FALSE (zero). Set to a

positive integer for program testing and development.

4. High-quality display: If this is set to TRUE, the quality of certain symbols on the printed music display is improved. These symbols take longer to draw and are therefore not recommended for normal use.

5. Grade 1 braille: If this is set to TRUE all literary braille text will be produced in grade 1 (that is, without contractions).

6. SAGEM embosser: If this is set to TRUE, the braille output file produced by the "EMBOSS" command will be suitable for embossing on a SAGEM model TEM 8BR braille embosser. If it is set to FALSE the braille output file will be suitable for embossing on a Triformation model LED120 braille embosser. It is not sensible to change this parameter after the first use of the "EMBOSS" command during a session.

7. Wide spacing: If this is set to TRUE, the vertical separation between staves on the printed music display will be increased. This may occasionally be necessary for pieces of music where the notation for adjacent staves is liable to collide using normal spacing.

A4.4.7 CLEAR command

The "CLEAR" command is used to delete the current section of the disc file. It does not affect the other sections of the file. It is not accepted if the current section is already empty. If the current section has "unprotected" status, it is deleted immediately. However, for safety, if the current section is "protected", it is not actually deleted, but instead its status is changed to "unprotected" and the message "PLEASE CONFIRM DELETION" is displayed. Deletion may then be confirmed by repeating the "CLEAR" command or by giving another command which overwrites the section ("BRAILLE", "BUILD", "FORMAT", or "RESTORE").

*A4.4.8 COPY command

The "COPY" command is used to copy an archive tape to a new archive tape, omitting deleted files and recovering index space used by these files. A music archive tape should be loaded before the command is used, and a new tape should be loaded on a second tape unit. CIMBAL will display the message "LOAD OUTPUT TAPE" and "PRESS CARRIAGE-RETURN". Press the carriage-return key to confirm that the output tape is correctly loaded. The program will then request a name for the new tape. This should be typed as a sequence of up to eight characters, followed by a carriage-return. On completion of the copying CIMBAL will display the message "TAPE COPIED CORRECTLY".

A4.4.9 DELETE command

The "DELETE" command is used to delete a specified sequence of bars from the current section. The range of bars to be deleted is specified as described in section A4.4.2. The specified bars are deleted from the current section and all following bars are

immediately re-numbered accordingly. The command operates on entire bars: it is not possible to delete only selected measures within a bar. The status of the current section is set to "protected", even if all bars of the current section are deleted. To delete the entire section including the header, the "CLEAR" command should be used.

A4.4.10 DISPLAY command

The "DISPLAY" command is used to display a selected part of the current section on the screen. The range of bars and staves to be displayed are specified as described in section A4.4.2. Whenever the screen is full during the display the bell is sounded: to clear the screen and display further music, press the carriage-return key. Execution of the display command may be stopped and CIMBAL returned to command status at any time during the display by pressing the interrupt button. For details of the display format, see section A4.6.

A4.4.11 EDIT command

The "EDIT" command is used to alter music stored in the current section. To insert complete bars into the current section or delete complete bars from the current section, see the "INSERT" and "DELETE" commands respectively.

The command operates either on the file header or on a single measure or line of music. The bar and stave numbers are specified as described in section A4.4.2. The screen is then erased and the specified measure is displayed in the middle of the screen, with extra spacing between items to allow for the display of insertions in the correct place.

Editing is performed by giving a series of "edit-commands". The program indicates that it is in edit-command status (ready and waiting to receive an edit-command) by displaying a pair of perpendicular cross-wires on the screen. If the character typed is a letter it may be in either upper or lower case. If it is not a recognised edit-command, or that particular edit-command is not allowed in the circumstances, the bell will sound and CIMBAL will return to edit-command status. If the command is accepted, the crosswires will disappear and the command will be executed. Edit-commands should not be confused with characters of the music input language. Neither the edit-command name nor the single character typed is displayed. CIMBAL enters edit-command status after first displaying the selected measure, and subsequently on completion of each edit-command, except for "end-of-edit" and "cancel-edit" which end the editing and return CIMBAL to command status.

The disc file is not changed in any way by the "EDIT" command until the "end-of-edit" edit-command is given and accepted. The entire edit may be cancelled using the "cancel-edit" edit-command. Pressing the interrupt button at any time during the edit except during input of a data item has the same effect as the "cancel-edit" edit-command.

The "insert-after", "insert-before", "change-item", "delete-item" and "get-item" edit-commands require an item to be selected before the

appropriate character is typed. The item is selected by using the joystick or light pen to position the crosswires on the screen so that they intersect at or close to the required item. If the intersection is close to more than one item, the nearest item is selected. If the intersection is not sufficiently close to any displayed items, these five edit-commands will be rejected. The position of the crosswires is irrelevant for the remaining edit-commands. To determine the exact screen position associated with each item, see section A4.6 ("Display format").

Control items, not all shown in the normal display, are displayed for the "EDIT" command so that the user may edit them if required. The single bar line at the end of a measure does not count as an item in the measure. However, any other type of bar line in or at the end of a measure is an item which may be selected for an edit-command.

The following edit-commands are available in edit-command status (the character to be typed is shown first):

A insert-after. This is used to insert a new item into the current measure immediately following the selected item. The new item is specified using the music input language described in section A4.5. If the new item is a note item, the length and pitch specifications are taken from the selected item if it is a note item, or from the last note item of the measure otherwise. If this edit-command is used by mistake, it may be cancelled by inserting the "cancel" item (see section A4.5.3).

B insert-before. This is used to insert a new item into the current measure immediately before the selected item. It is used in exactly the same way as the "insert-after" edit-command.

C change-item. This is used to change or replace the selected item. If the selected item is a note item, all the default attributes are taken from it. The displayed item is crossed out when the command is first typed, and re-drawn on completion of the replacement item. Replacing an item by the "cancel" item has the effect of converting the "change-item" edit-command into a "delete-item". To change the attributes of a note item, use the "change-item" edit-command and add or cancel only the specifications for the new values of the attributes to be changed.

D delete-item. The selected item is deleted and a cross is drawn through it on the display.

E examine. This has no direct effect. It may be used in case of confusion to distinguish editing crosswires from pitch-specification crosswires (see section A4.5.6.3): if the character "E" is typed, then the bell will sound if the pitch-specification crosswires are displayed, indicating that a character of the input language is required, but not if the editing crosswires are displayed, indicating that an edit-command is required. This edit-command may also be used to cancel the "get-item" edit-command (see below).

F fresh-display. This is used to re-display the current measure during an edit. The screen is erased, and the measure being

edited is re-displayed with deleted items removed and the remaining items suitably spaced. The "fresh-display" edit-command affects only the display and not the contents of the current measure or the disc file.

G get-item. This is used to copy the selected item, which must be a note item, to a different position within the same measure, or to duplicate it. If the following command is "insert-after", "insert-before" or "change-item", the item selected for the "get-item" edit-command will be used as the new or replacement item. If any other edit-command is used immediately after "get-item", the effect of the "get-item" is cancelled. The "get-item" edit-command does not affect the selected item, which should be deleted separately if required.

M ignore-check. This is used to allow the total time-count of notes in the measure being edited to be changed. After it has been used, the "end-of-edit" edit-command will not check the total duration of notes in the measure. The "fresh-display" edit-command reinstates the time-count check.

! end-of-edit. If this is accepted, the disc file is updated to incorporate the changes made during the execution of the current "EDIT" command, and CIMBAL returns to command status. The current section is given "protected" status, and the message "FILE UPDATED" is displayed. "End-of-edit" will not be accepted if the total time count of the notes in the measure as edited is different from that at the start of the edit, regardless of whether this conforms to the time signature, unless the "ignore-check" edit-command has been used previously during the current edit.

? cancel-edit. This is used to cancel the entire "EDIT" command and return CIMBAL to command status without changing the disc file. The message "FILE UNCHANGED" is displayed.

X crosswires. This has the same effect as the "crosswires" control item (see section A4.5.3), that is, it switches on or off the crosswires for note pitch specification. It should be used only with great care during editing because the inexperienced user is liable to confuse the crosswires for pitch-specification with the editing crosswires. The pitch-specification crosswires are initially switched off at the start of each "EDIT" command.

It is advisable to use the "fresh-display" edit-command immediately after removing or inserting a clef sign, other than at the end of a measure, because otherwise the following notes will be be displayed at the wrong vertical position on the stave.

The file header, which contains the information given in the initial specifications (see section A4.5.2) is edited by typing a plain carriage-return or zero in response to the request for a bar number. The file header information is displayed as for the "SUMMARY" command, and may be edited in the same way as an ordinary measure of music, except that the "insert-before", "insert-after" and "delete-item" edit-commands are not available. The "change-item" edit-command should be used to change items in the file header. These

items should be changed only to items of the same kind (that is, text item, clef, key signature or time signature). All textual entries in the file header have a corresponding text item, which may however contain no characters.

*A4.4.12 EMBOSS command

The "EMBOSS" command is used to create an embosser file from the current section. It is available only in "unformatted braille" or "formatted braille" mode. If the embosser output file is not available or is too small or has the wrong format, the message "BRAILLE OUTPUT FILE NOT AVAILABLE" will be displayed and the command will not be accepted. On successful completion of the command, a message will be displayed indicating the total number of braille pages produced during the current session. The embosser file may subsequently be embossed using an appropriate embossing program.

The initial use of the "EMBOSS" command during a session will overwrite any braille previously stored in the embosser file. Each subsequent use of the "EMBOSS" command will add the new braille to the end of the existing file, enabling more than one piece of music to be included in a single braille document. After each use of the "EMBOSS" command the total number of braille pages output to the embosser file during the current session will be displayed.

The default output device for the braille is the LED120 embosser. The file may be made suitable for embossing on a SAGEM embosser by setting the SAGEM parameter to TRUE before first using the "EMBOSS" command (see section A4.4.6).

A4.4.13 ERASE command

The "ERASE" command is used to delete unwanted files from an archive tape. The archive tape should be loaded at the time the command is used. CIMBAL requests a name and version number for the file to be deleted. Both must be typed in full exactly as they appear in the listing produced by the "FILES" command. If there is no file on the tape corresponding to the specified name and version number, the message "THIS FILE IS NOT ON THE TAPE" is displayed and the tape remains unchanged, otherwise the file specified is marked as deleted in the tape index. Although the file is not physically removed from the tape until subsequent use of the "COPY" command, it is no longer accessible to the user. No other files on the tape are affected, including files with the same name but different version numbers.

A4.4.14 FILES command

The "FILES" command is used to obtain a listing of the files on an archive tape. The tape should be loaded at the time the command is used. The tape name and number of files stored on the tape and deleted from it are displayed, followed by a listing of identifier, name and version number for each file on the tape in increasing order of age, that is, the most recently added file first. "Deleted" files refers to files which have been deleted from the index, but not yet physically deleted from the tape. The number of deleted files on the tape is reset to zero when these files are physically removed by use

of the "COPY" command. The listing may be interrupted at any time by pressing the interrupt button.

*A4.4.15 FLAG command

The "FLAG" command is used to change the "flag" value associated with a measure of unformatted or formatted braille. It is not available in printed music mode. The bar and stave numbers, or line number, are specified as described in section A4.4.2. A message is displayed giving the current value of the flag and requesting a new value. Type a number in the range 0 to 127, followed by a carriage-return, or a plain carriage-return which is equivalent to 0. The significance of the "flag" value is described in section A4.9.

*A4.4.16 FORMAT command

The "FORMAT" command is used to create a "formatted braille" section from the current section. It is available only in "unformatted braille" mode. The command will not be accepted if the "formatted braille" section of the disc file is currently "protected". If the command is accepted, any braille already stored in the "formatted braille" section is first deleted. On successful completion of the command, the mode is changed to "formatted braille", and the new "formatted braille" section is given "unprotected" status.

A4.4.17 INSERT command

The "INSERT" command is used to insert a new bar into the current section, other than at the end. To add further music to the end of the current section, use the "ADD" command instead. To insert more than one bar, use the "INSERT" command repeatedly.

A bar number and stave numbers are requested. These are specified as described in section A4.4.2. The bar number is that of the bar before which the new bar is to be inserted. Music is input as for the "BUILD" command. On completion of input of one bar, CIMBAL returns to command status, and the current section is given "protected" status. All bars following the inserted bar are re-numbered accordingly immediately on completion of the command.

*A4.4.18 MODE command

The "MODE" command is used to change the current mode of CIMBAL. The program mode determines which section of the disc file commands operate on, as described in section A4.3. Type a single character specifying the new mode required:

P	Printed music mode
U	Unformatted braille mode
F	Formatted braille mode

A4.4.19 OPTIONS command

The "OPTIONS" command clears the screen and displays a list of

available commands with a brief description of their use. This command is also executed if an unrecognised command name is typed when CIMBAL is in command status. The listing may be cancelled and CIMBAL returned to command status by pressing the interrupt button.

*A4.4.20 PLAY command

The "PLAY" command is used to produce an audible output of a selected part of the current section, using the digital to analogue converter and analogue outputs of the computer. It is available only in "printed music" mode. Bar and stave numbers are specified as described in section A4.4.2. A playing speed is requested. At a medium tempo this corresponds to the number of beats per minute. If a plain carriage-return is typed in response to the request for a speed, a value of 200 is used. Pressing the interrupt button during playing will cancel the command at the end of the current measure.

In the current implementation, only monophonic output is available. It is therefore not suitable for music with chords or in-accord parts and not sensible to select more than one stave.

A4.4.21 PRINT command

The "PRINT" command is used to create a print file from a selected part of the current section. A printed copy of the music may later be produced on a graph plotter or lineprinter from the print file (see section A4.2.3). Bar and stave numbers are specified as described in section A4.4.2. Printed music notation is produced graphically on a digital graph plotter in a form similar to that displayed on the screen. Braille is printed on a lineprinter using a pattern of dots and spaces to represent the braille cells, with bar or line numbers shown at the left hand side of the page.

A4.4.22 RESTORE command

The "RESTORE" command is used to copy a file from an archive tape to the disc file. The archive tape should be loaded before the command is used. The command will not be accepted if any section of the existing disc file is "protected". A file identifier is requested. If a plain carriage-return is typed for the identifier, a name is requested instead. If a plain carriage-return is typed for the name, the most recent file on the tape is selected. If either an identifier or a name is specified, a version number is requested. If a plain carriage-return is typed for the version number, the most recent version of the file with the given identifier or name is selected, otherwise the specific version requested is selected.

If the selected file is not on the tape, the message "THIS FILE IS NOT ON THE TAPE" is displayed and the disc file remains unchanged. If the selected file is on the tape, the disc file is made a copy of the selected tape file, the status of each non-empty section of the new disc file is set to "unprotected", and the current mode is set to "printed music". All sections of the file are copied regardless of the current mode. The tape itself is not changed by the "RESTORE" command.

A4.4.23 SAVE command

The "SAVE" command is used to copy the disc file to an archive tape. The archive tape should be loaded before the command is used. The command operates on the whole file, regardless of the current mode, and will not be accepted if all sections of the disc file are "empty". On successful completion of the command, a message is displayed giving the name and version number of the new file on the tape, and the number of tape records occupied. The disc file is unchanged except that the status of any "protected" section is changed to "unprotected". The current mode is automatically set to "printed music" on successful completion of the command. Version numbers are explained in section A4.3.2.

A4.4.24 SCALE command

The "SCALE" command is used to change the horizontal scale factor at the start of the file for display of the printed music, in order to improve alignment between notes on different staves or to avoid large gaps between notes. The scale factor is given as a number in the range 8 to 128. The scale factor is proportional to the distance between notes on the display, subject to a minimum spacing. The default scale factor on creation of a new printed music section is 64. The scale factor may also be changed at any point in the printed music section using the scale-factor item (see section A4.5.4). The display scale does not affect the braille translation in any way.

A4.4.25 STOP command

The "STOP" command is used to finish a session and release CIMBAL. For safety, it will not be accepted if any section of the disc file is "protected". In this case, the status of any "protected" sections should be changed to "unprotected" by using either the "CLEAR" command or the "SAVE" command. The disc file should normally be copied to a magnetic tape using the "SAVE" command before the "STOP" command is used. If the command is accepted, the message "PROGRAM RELEASED" is displayed and the terminal is then available to other users. If the "PRINT" or "EMBOSS" commands have been used during the current session, additional messages will indicate that there are files awaiting printing or embossing (see section A4.2 for means of doing this).

A4.4.26 SUMMARY command

The "SUMMARY" command is used to display summary information for the current disc file. This includes the number of bars and staves stored in, and protection status of, each non-empty section of the file. If the current mode is printed music, the contents of the header record are displayed, showing the information input as initial specifications (see section A4.5.2).

A4.4.23 SAVE command

The "SAVE" command is used to copy the disc file to an archive tape. The archive tape should be loaded before the command is used. The command operates on the whole file, regardless of the current mode, and will not be accepted if all sections of the disc file are "empty". On successful completion of the command, a message is displayed giving the name and version number of the new file on the tape, and the number of tape records occupied. The disc file is unchanged except that the status of any "protected" section is changed to "unprotected". The current mode is automatically set to "printed music" on successful completion of the command. Version numbers are explained in section A4.3.2.

A4.4.24 SCALE command

The "SCALE" command is used to change the horizontal scale factor at the start of the file for display of the printed music, in order to improve alignment between notes on different staves or to avoid large gaps between notes. The scale factor is given as a number in the range 8 to 128. The scale factor is proportional to the distance between notes on the display, subject to a minimum spacing. The default scale factor on creation of a new printed music section is 64. The scale factor may also be changed at any point in the printed music section using the scale-factor item (see section A4.5.4). The display scale does not affect the braille translation in any way.

A4.4.25 STOP command

The "STOP" command is used to finish a session and release CIMBAL. For safety, it will not be accepted if any section of the disc file is "protected". In this case, the status of any "protected" sections should be changed to "unprotected" by using either the "CLEAR" command or the "SAVE" command. The disc file should normally be copied to a magnetic tape using the "SAVE" command before the "STOP" command is used. If the command is accepted, the message "PROGRAM RELEASED" is displayed and the terminal is then available to other users. If the "PRINT" or "EMBOSS" commands have been used during the current session, additional messages will indicate that there are files awaiting printing or embossing (see section A4.2 for means of doing this).

A4.4.26 SUMMARY command

The "SUMMARY" command is used to display summary information for the current disc file. This includes the number of bars and staves stored in, and protection status of, each non-empty section of the file. If the current mode is printed music, the contents of the header record are displayed, showing the information input as initial specifications (see section A4.5.2).

A4.5 Input language for music notation

A4.5.1 General notes on music input

The "ADD", "BUILD" and "INSERT" commands allow music notation to be input. When a new file is created using the "BUILD" command, one stave is allocated for each stave of the printed score. The total number of staves must be specified correctly, because this affects the file structure, and cannot subsequently be changed. Any combination of staves may be selected for music input as described in section A4.4.2. Staves not selected for input have whole-measure rests stored in them. Thus for orchestral scores it is only necessary to select staves for instruments actually playing at a given time.

The "BUILD" command requires the user to type some initial specifications as described in section A4.5.2 before the music itself is input. The music is input using a special character code. For an example of the use of this code, see section A4.8 ("Example"). Upper case and lower case letters are treated as the same throughout, except where (in text items) they correspond to letters in the printed music. The main additional meanings of the different keyboard characters are listed in table A4:2.

Each character typed by the user is inspected by CIMBAL to check whether it is allowed in the given context. If the character is not acceptable, it will not be displayed on the screen, but the bell will sound instead to indicate an error. The user may then investigate the error and continue as if the erroneous character had not been typed. Characters which are accepted are displayed along the lower part of the screen for checking purposes. A carriage-return is represented as a slash (/) in this display. In certain circumstances (notably the first character typed for a note item where there is no previous note item in the same measure) there may be a slight delay before a valid character is displayed. The user should wait for each character to appear on the screen before typing another character, unless the bell is sounded.

Music is input bar by bar. Within each bar the music is input measure by measure from the uppermost line to the lowermost line. Each measure is terminated with an "end-of-measure" item (see section A4.5.3). Subsequent items will then automatically refer to the following measure. Within each measure the music is input item by item. The order of items within a measure is described in section A4.5.8.

An item is input by typing an appropriate sequence of characters. Some items require only one character to be typed. Others need several characters. The item itself is not displayed until all the characters necessary for its input have been typed. Some single-character items (for example, start-of-slur) are not displayed immediately, because the position of the symbol is not yet known, but their acceptance is indicated by the reflection of the input character on the screen. The input of an item may be cancelled at any time before it is complete by pressing the interrupt button. Any characters displayed at the bottom of the screen which are associated with the item will be crossed out. The user may then proceed as if that item had not been started.

For the "INSERT" command, music input is terminated after the input of one complete bar. For the "ADD" and "BUILD" commands, input is terminated by inputting the "end-of-input" item (see section A4.5.3). If this item is input on stave 1, the current measure will be discarded, otherwise the current measure will be stored, regardless of whether or not it is complete, and any remaining measures of the current bar will be stored as whole-measure rests. Thus the "end-of-input" item should normally be input following the "end-of-measure" item on the lowest stave currently being input (or instead of it if stave 1 is not currently selected).

Sections A4.5.3 to A4.5.7 contain a description of the various items, grouped by item type. The layout of the description for each item is as follows:

Name:	The name by which the item is referred to throughout this manual.
Format:	The sequence of characters typed to input the item. Characters actually typed are described in words or shown enclosed in angle brackets in this description. These angle brackets are not themselves typed.
Meaning:	The meaning of the item in terms of conventional music notation.
Restrictions:	Limitations on the context in which the particular item is allowed.
Comments:	Any further description of the use of the item not covered by the above.

A4.5.2 Initial specifications

The initial specifications define the file header record, and refer to the file as a whole. If the interrupt button is pressed during input of specifications 1 to 3, CIMBAL will return immediately to command status. If the interrupt button is pressed during input of specifications 4 to 12, the current question will be repeated. If the interrupt button is pressed during input of specifications 13 to 15 it will have the normal effect of cancelling any characters already input for the current item.

For each of the initial specifications, CIMBAL will display a question asking for the specification to be typed. The dialogue for the initial specifications is shown below. Those indicated as "optional" may be omitted by replying with a plain carriage-return. The answers to questions 4 to 12 may be any text-below item (see section A4.5.5) with the initial <V> omitted.

1a.	Question:	NUMBER OF STAVES?
	Answer:	a number between 1 and 40 inclusive, indicating the number of parts, followed by a carriage-return.
	Note:	The number of staves once specified may not be changed, because it determines the structure of the current section.
1b.	Question:	SELECT STAVES?
	Answer:	staves are selected as described in section A4.4.2. Normally all staves are selected by typing a plain carriage-return.

2. Questions: PART 1? , ... , PART n? (One such question will be asked for each part)
   Answer: for each part, one of the abbreviations listed in table A4:1 should be used, followed by a carriage-return.

3. Question: IDENTIFIER?
   Answer: any sequence of up to four characters followed by a carriage-return. This is a label to permit easy identification of the file when saved on an archive tape. It is advisable to choose a unique identifier for each piece of music.

4. Question: NAME?
   Answer: the name of the piece. This is the full name by which the file is known when stored on magnetic tape, and is also used as the running page title in the braille version. It should normally contain the surname of the composer and the title of the piece, or an abridged version of the title if this is too long.

5. Question: TITLE? (optional)
   Answer: the actual title of the piece.

6. Question: SUBTITLE? (optional)
   Answer: any subsiduary title of the piece, or additional information which follows the title.

7. Question: COMPOSER? (optional)
   Answer: the name of the composer.

8. Question: PUBLISHER? (optional)
   Answer: the name of the publisher (if required in the braille).

9. Question: NUMBER? (optional)
   Answer: the number of the piece if any (e.g. K.381 or op. 68). Full stops and spacing should be typed as printed.

10. Question: TEMPO? (optional)
    Answer: the word or phrase normally printed in heavy type above the first bar indicating speed etc. (e.g. Andante).

11. Question: METRONOME? (optional)
    Answer: the metronome indication as printed, if any. See section A4.5.5 regarding input of musical symbols within text. No space should be left on either side of the "=" sign. The expression should be enclosed in parentheses if these or square brackets are shown around the metronome marking in the print.

12. Question: COMMENTS? (optional)
    Answer: any additional comment which the user wishes to include. This will not appear in the braille.

13. Question: KEY SIGNATURE?
    Answer: the key signature is input as described in section A4.5.4.

14. Question: TIME SIGNATURE?

Answer: the time signature is input as described in section A4.5.4.

15. Question: CLEF FOR STAVE 1?, ... , CLEF FOR STAVE n? (one such question is asked for each part)
Answer: each clef is input as described in section A4.5.4.

When the initial specification questions have been answered satisfatorily the screen will be cleared, and a staff will be displayed showing a stave of five lines for each part, with clefs, time signature and key signature shown. CIMBAL is then ready for input of the music itself. The individual items available for this purpose are described in sections A4.5.3 to A4.5.7 and the order in which they are used is described in section A4.5.8.

A4.5.3 Control items

Control items either define the structure of the music, control the operation of CIMBAL, or represent miscellaneous music symbols.

The following items control the operation of CIMBAL and do not cause anything to be stored in the current section of the disc file:

Name: cancel
Format: <?>
Meaning: during editing, this item returns CIMBAL immediately to edit-command status. Otherwise it causes CIMBAL to enter edit-command status, enabling the user to edit the current measure (see section A4.5.10).
Restrictions: the cancel item will not be accepted if there are no items already stored in the current measure to edit.

Name: end-of-input
Format: <!>
Meaning: music input is complete.
Comments: during editing, this item has the same effect as the "cancel" item (see above). During the "INSERT" command, it has the same effect as the "end-of-measure" item (see below). Otherwise it terminates input of music and returns CIMBAL to command status. Any incomplete measure on the first stave will be discarded.

Name: end-of-measure
Format: <|> followed by a carriage-return.
Meaning: ordinary single bar line indicating the end of a measure.
Restrictions: the total time count for the notes in the current measure must be correct according to the current time signature, unless the "ignore-check" item has already been input or the "ignore-check" edit-command has been used during input of the current measure.
Comments: where a measure ends with a bar line which is not an ordinary single bar line, the appropriate form of the "bar-line" item should be used, followed by the "end-of-measure" item.

Name: ignore-check
Format: <M>

Meaning: CIMBAL is not to perform the usual check on the time count for the current measure (see section A4.5.9).
Comments: the item is effective only for the current measure.

Name: crosswires
Format: <X>
Meaning: if crosswires are currently in use for pitch specification (see section A4.5.6.3) they are switched off. Otherwise they are switched on.

The following items are stored in the file:

Name: bar-line
Format: (1) <||>
or (2) <:||>
or (3) <||:>
or (4) <:||:>
or (5) <||T>
or (6) Vertical bar followed by <.>,
each followed by a carriage-return.
Meaning: (1) Ordinary thick double bar line
(2) Double bar line preceded by dots indicating end of repeat section
(3) Double bar line followed by dots indicating start of repeat section
(4) Double bar line preceded and followed by dots indicating end of one repeated section and start of another
(5) Thin double bar line
(6) Dotted bar line.
Comments: a bar line may appear anywhere within a measure. Where a bar line extends across more than one stave, the same bar-line item should be input in the correct position on each stave. An ordinary single bar line is not input as such, but is implied by the "end-of-measure" item (see above). A double bar line preceded and followed by dots which coincides with the end of a measure should be input as a double bar line preceded by dots at the end of the first measure and a double bar line followed by dots at the start of the second measure. A bar-line item does not automatically imply the end of a measure and the "end-of-measure" item must also be used if these occur at the end of a measure.

Name: in-accord
Format: <;> followed by a carriage-return
Meaning: go back to the position in the current measure at which the last "set-return" item (see below) was input, or to the start of the bar if no set-return item has been input during the current measure.
Comments: this item is used where the music contains notes written "in-accord". See section A4.5.8 ("Order of input") for details.

Name: set-return
Format: <@> followed by a carriage-return
Meaning: indicates the position to return to after an in-accord, or the end of part of a measure written in-accord.
Comments: See section A4.5.8 ("Order of input") for details.

Meaning: CIMBAL is not to perform the usual check on the time count for the current measure (see section A4.5.9).
Comments: the item is effective only for the current measure.

Name: crosswires
Format: <X>
Meaning: if crosswires are currently in use for pitch specification (see section A4.5.6.3) they are switched off. Otherwise they are switched on.

The following items are stored in the file:

Name: bar-line
Format: (1) <||>
or (2) <:||>
or (3) <||:>
or (4) <:||:>
or (5) <||T>
or (6) Vertical bar followed by <.>, each followed by a carriage-return.
Meaning: (1) Ordinary thick double bar line
(2) Double bar line preceded by dots indicating end of repeat section
(3) Double bar line followed by dots indicating start of repeat section
(4) Double bar line preceded and followed by dots indicating end of one repeated section and start of another
(5) Thin double bar line
(6) Dotted bar line.
Comments: a bar line may appear anywhere within a measure. Where a bar line extends across more than one stave, the same bar-line item should be input in the correct position on each stave. An ordinary single bar line is not input as such, but is implied by the "end-of-measure" item (see above). A double bar line preceded and followed by dots which coincides with the end of a measure should be input as a double bar line preceded by dots at the end of the first measure and a double bar line followed by dots at the start of the second measure. A bar-line item does not automatically imply the end of a measure and the "end-of-measure" item must also be used if these occur at the end of a measure.

Name: in-accord
Format: <;> followed by a carriage-return
Meaning: go back to the position in the current measure at which the last "set-return" item (see below) was input, or to the start of the bar if no set-return item has been input during the current measure.
Comments: this item is used where the music contains notes written "in-accord". See section A4.5.8 ("Order of input") for details.

Name: set-return
Format: <@> followed by a carriage-return
Meaning: indicates the position to return to after an in-accord, or the end of part of a measure written in-accord.
Comments: See section A4.5.8 ("Order of input") for details.

Name: measure-repeat

Format: <|"> optionally followed by a measure-number specification, optionally followed by a length-of-repeat specification, optionally followed by a pitch-change specification, followed by a carriage-return, where a measure-number specification is either:
(1) a single integer
or (2) two integers separated by a dot
or (3) a dot followed by an integer;
a length-of-repeat specification is an asterisk followed by a positive integer;
and a pitch-change specification is any sequence of "increment-pitch", "decrement-pitch", "increment-octave" or "decrement-octave" note-specifications as described in section A4.5.6.3.

Meaning: this item is used to make the current measure a repeat of an already existing measure, with optional change of pitch. The measure-number specification identifies the first measure to be copied:
(1) The integer specifies the bar number, the current stave is assumed.
(2) The first integer specifies a bar number and the second a stave number.
(3) The integer specifies a stave number, the current bar is assumed.
If no measure-number specification is given, the immediately preceding measure on the same stave is repeated.
The length-of-repeat specification defines the number of measures to be repeated, the measures being counted in the order in which they would otherwise be typed by the user (that is, including all selected staves). If the length-of-repeat specification is absent, a single measure is repeated.
The pitch-change specification defines the change in pitch required if any. If this is absent there is no pitch change. If present, all note-heads are incremented or decremented in pitch by the specified amount.

Restrictions: The "measure-repeat" item must be the first and only item in the current measure. The measure-number specification must correspond to an already existing measure of the current section, which in the case of the "INSERT" command need not necessarily come before the new measure. This item is not available during execution of the "EDIT" command.

Comments: If the item is accepted, the entire new measure is displayed. No "end-of-measure" item is required; CIMBAL is then ready to continue with input of the next measure. The pitch change feature cannot be used to add or remove marked accidentals.

Name: footnote

Format: <|F>

Meaning: this item introduces a musical footnote.

Restrictions: time count checking is performed in the same way as for the "end-of-measure" item.

Comments: the "footnote" item should be input at the end of the first measure in which the musical footnote is referred

to, followed by the contents of the footnote, followed by the "end-of-measure" item.

Name: segno
Format: <@S>
Meaning: the segno symbol

Name: encircled cross
Format: <@O>
Meaning: the encircled cross sign

Name: pause
Format: <@P>
Meaning: the pause sign, when not directly associated with a note or rest.
Comments: a pause sign which is associated with a note or rest is input as part of the corresponding note item (see section A4.5.6).

Name: right-hand sign
Format: <@+>
Meaning: this item precedes the first note input in the right hand part following a note which has been input in the right hand part because of hand alternation, but actually appears in the left hand part in the print. It is also used to represent R.H. (right hand) or M.D. (main droit) where these occur in the print.
Restrictions: only available in the right hand part of keyboard music.

Name: left-hand sign
Format: <@->
Meaning: this item precedes the first note input in the right hand part which in the print is shown in the left hand part, following a note which is correctly shown in the right hand part. It is also used to represent L.H. (left hand) or M.G. (main gauche) where these occur in the print.
Restrictions: only available in the right hand part of keyboard music.
Comments: where, in keyboard music, the right and left hand parts alternate, all notes should be input in the right hand part with appropriate right-hand and left-hand signs, and implied rests should be inserted instead in the corresponding left-hand part.

Name: first or second ending
Format: <@1> or <@2>
Meaning: this item corresponds to the start of a first or second alternative ending shown in the print.
Comments: where the alternative endings occupy one or more complete measures, they should be input as separate measures in the order printed.

Name: null
Format: <@X>
Comments: this may be used if necessary to prevent the braille translator from using a braille repeat sign but otherwise has no effect.

Name: note-stem direction

Format: <@Y> or <@U> or <@=>
Meaning: the first form indicates that subsequent notes on the current stave are to be displayed with stems pointing upwards.
The second form indicates that subsequent notes on the current stave are to be displayed with stems pointing downwards.
The third form indicates that CIMBAL is to determine the direction for the stems of subsequently displayed notes using the usual convention.
Comments: normally CIMBAL will determine the correct stem direction according to the usual convention. This item is only necessary if it is required to override this choice. It does not affect the braille translation. The item itself is not displayed except for editing.

A4.5.4 Separate sign items

Separate sign items define markings (other than text) which are not in general associated with a specific note but apply to a sequence of notes. An "extended sign", that is, one which in the print may extend above or below a sequence of consecutive notes (viz. slur, crescendo, decrescendo, ottava), is represented by two separate-sign items. The first of these corresponds to the start of the sign and precedes the first note to which the sign applies. The second corresponds to the end of the sign and follows the last note to which the sign applies. The separate-sign items are:

Name: clef
Format: <Z> followed by a clef-type character followed by a carriage-return, where the clef-type character is one of:
<T> or <G> for the treble (G) clef,
or <B> or <F> for the bass (F) clef,
or <1> for C clef on bottom line (soprano),
or <3> for C clef on middle line (alto),
or <4> for C clef on 4th line (tenor),
or <8> for treble clef with a small 8 below.
Meaning: clef sign, either in the initial specifications or where a clef change is marked in the printed score.
Comments: the clef signs at the beginning of each printed line are not input, unless they indicate a change of clef. Those at the beginning of the piece are included in the initial specifications.

Name: time-signature
Format: <T> followed by a time-specification followed by a carriage-return, where the time-specification is one of:
(1) Two integers separated by a carriage-return,
or (2) <C>,
or (3) <C|>,
or (4) <U>.
Meaning: a time signature, either in the initial specifications or where the time signature changes during the piece.
(1) is used for a time signature shown as two numbers, one above the other: the first number is the upper one and the second the lower;

(2) is used for a time signature shown as C;
(3) is used for a time signature shown as C with a vertical bar through it;
(4) is used to indicate unmeasured music or music with no time signature.

Restrictions: in case (1) the first number must be in the range 1 to 32 and the second number must be 1, 2, 4, 8, 16 or 32.

Comments: (2) and (3) are equivalent to 4/4 and 2/2 respectively for time-count checking purposes. (4) causes all time-count checks to be ignored. A change of time signature is input for each stave of the music as shown in the print.

Name: key-signature

Format: <K> followed by a number in the range 1 to 7 followed by <#> for sharp, <$> for flat, or <%> for natural;
or: <K> followed by a carriage-return.

Meaning: key signature, either in the initial specifications or where the key signature changes during the piece. The number gives the number of signs in the key signature and the following character indicates whether they are sharps, flats, or naturals. The second form indicates a key signature having no sharps, flats or naturals, and is used for the initial specifications for a C major key signature (no accidentals shown).

Restrictions: naturals are not allowed in the initial specifications.

Comments: the key signature at the beginning of each printed line is not input, unless it represents a change of key. A change of key signature is input for each stave of the music as shown in the print. Where a key signature comprises sharps or flats preceded by naturals which cancel an earlier specification, this should be typed as two separate items, one for the naturals and one for the sharps or flats.

Name: irregular-note-grouping

Format: <N> followed by an integer in the range 2 to 11 followed by a carriage-return.

Meaning: indicates that the following notes form an irregular grouping. The number gives the number of notes grouped.

Comments: the item is input preceding the first note of the group.

Name: inkprint-page-number

Format: <Q> followed by an integer in the range 1 to 255 followed by a carriage-return.

Meaning: page number of the printed score.

Comments: for each printed page, the page number should be input as the first item of the first measure on that page. This item is required only if print page numbers are wanted in the braille.

Name: inkprint-line-number

Format: <L> followed by an integer in the range 1 to 9 followed by a carriage-return.

Meaning: the number of the parallel on the current page of the printed score. The first parallel of each page is numbered 1.

Comments: the line number is input as the first item (other than an inkprint-page-number item) of the first measure of

each parallel of the printed score. This item is required only if print line numbers are wanted in the braille.

Name: note-beaming
Format: <J> followed by an integer n in the range 2 to 16 followed by a carriage-return.
Meaning: the following n notes are to be beamed together on the display.
Restrictions: this should not be used within the scope of a previous note-beaming, or for beaming which extends beyond the end of the measure.
Comments: this item does not affect braille translation. The item is not required if the beamed group of notes coincides with an irregular grouping. The number does not include rests occurring between notes of the beamed group. It does not include grace notes unless the first note-item following the "note-beaming" item is a grace note.

Name: scale factor
Format: <H> followed by an integer in the range 8 to 255 followed by a carriage-return.
Meaning: changes the horizontal scale factor for display of subsequent music (see section A4.4.24).

Name: pedal
Format: <P>
Meaning: pedalling sign (P or ped in the print)
Comments: The item should precede the first played note to which it corresponds.

Name: end-pedal
Format: <*>
Meaning: end of pedalling (asterisk, or end of a horizontal line extending from the P in the print).
Comments: The item should follow the last played note or rest after which it is written.

Name: start-of-slur
Format: <(>
Restrictions: a maximum of two concurrent slurs is allowed on each stave.
Comments: the start-of-slur item is input preceding the note on which the slur starts. The slur is not displayed because its length is not yet known, but the appearance of the <(> on the screen is evidence of its acceptance. Where in-accord parts are used, a slur must start and end in the same part.

Name: end-of-slur
Format: <)>
Restrictions: each end-of-slur must correspond to an earlier start-of-slur on the same stave.
Comments: the end-of-slur corresponds to the most recently input start-of-slur item on the same stave which has not already been matched by an end-of-slur item. The end-of-slur is input following the note on which the slur ends. The slur is displayed when this item is input.

Name: ottava-above
Format: <]>
Meaning: corresponds to 8va written above the printed stave
Comments: if the 8va appears above a note, the ottava-above item is input preceding the note item.

Name: ottava-below
Format: <[>
Meaning: corresponds to 8va written below the printed stave
Comments: as for ottava-above.

Name: end-of-ottava
Format: <_>
Meaning: corresponds to "loco" or the end of a horizontal line extending from the 8va in the print.
Restrictions: The item must correspond to a previous ottava-above or ottava-below item on the same stave.

Name: start-of-crescendo
Format: <<> (less-than sign)
Meaning: beginning of diverging lines appearing above or below the stave.
Restrictions: only one crescendo or decrescendo is allowed on one stave at a given time.
Comments: the start-of-crescendo sign precedes the first note item to which the crescendo applies. The crescendo is not displayed when this item is input because its length is not yet known.

Name: start-of-decrescendo
Format: <>> (greater-than sign)
Meaning: start of converging lines written above or below the stave.
Restrictions: as for start-of-crescendo
Comments: as for start-of-crescendo.

Name: end-of-crescendo-or-decrescendo
Format: < > (space)
Meaning: corresponds to the end of converging or diverging lines
Restrictions: must correspond to a preceding start-of-crescendo or start-of-decrescendo on the same stave.
Comments: this item is input following the corresponding note item. The crescendo or decrescendo sign is then displayed.

A4.5.5 Text items

Text items define text or letters written above or below the stave, excluding:

tr (for trill) which is part of a note item
P (that is, upper case P representing pedalling, not to be confused with lower case p standing for piano, which is a text item)

Name: text
Format: <W> or <^> or <V>, followed by up to 31 characters of text followed by a carriage-return.

Meaning: initial character <W>: sung word or syllable. If the syllable is not the last of a word, a hyphen should be added as the last character before the carriage-return. If there is a genuine hyphen at the end of a syllable, the final hyphen is in addition to this;
initial character <^>: text written above the stave representing dynamics, expression marks etc.;
initial character <V>: similarly text written below the stave representing dynamics, expression marks etc.
The text is typed as it appears in the print. Upper and lower case characters are distinguished by CIMBAL and displayed as such.

Restrictions: all printable characters are allowed in the text. The question mark acts as a backspace character. Any number of characters may be backspaced within a text item. Backspacing past the first character of text causes the item to be cancelled.

The "@" sign is used within text items as a special character which together with the immediately following character defines a special code. This is to permit inclusion of accents, musical symbols and the question mark within text. The "@" sign itself is not counted in the permitted 31 characters. The character following the "@" must be one of those listed below, and has the significance shown. A single question mark has the effect of backspacing both the "@" and the character following the "@".

Accents:
<@'> acute accent
<@`> grave accent
<@^> circumflex accent
<@_> umlaut (two dots above a letter)
The accent codes are used for accented vowels. The accent code should immediately follow the affected vowel. The accent is attached to the vowel and is not treated as a separate character for backspacing or storage purposes.

Music symbols:
<@1> breve
<@2> semibreve
<@3> minim (1/2 note)
<@4> crotchet (1/4 note)
<@5> quaver (1/8 note)
<@6> semiquaver (1/16 note)
<@7> demisemiquaver (1/32 note)
<@8> hemidemisemiquaver (1/64 note)
<@9> semihemidemisemiquaver (1/128 note)
<@$> flat sign
<@%> natural sign
<@#> sharp sign
<@.> dot (for dotted note)
<@S> segno
<@O> encircled cross
<@P> pause sign
The music symbol codes are used to represent musical symbols where these occur within text items as in metronome markings and D.S. indications.

Foreign text:
<@F> start of foreign text
<@E> end of foreign text (start of English text).

The foreign text code is input preceding foreign text. The English text code is used to cancel the foreign text code, and is input at the end of foreign text if English text follows. The foreign text code causes subsequent text to be converted to uncontracted braille; the English text code causes subsequent text to be converted to contracted braille. Heading items (other than tempo), sung-word items, and other text items are treated as three independent categories for this purpose and the effect of the foreign text or English text codes continues from one text item to the next within each of these categories until contradicted. By default, text-above and text-below items (expression marks, dynamics etc.) and the tempo indication in the heading are assumed to be foreign text, and sung words and the remaining heading items are assumed to be English text.

Special types: <@R> rehearsal letter or number
A rehearsal number or letter should be input as a text item above the stave, starting with the <@R> code.

Braille codes: <@A> accent sign
<@C> capital sign
<@I> italic sign
<@L> letter sign
<@N> number sign
<@/> null sign
These codes correspond to the braille cells of the same name and should be included if specially required in the braille. The only one normally required is the letter sign, which should be inserted before a single letter standing alone in a heading item, for example, representing a key (but not before dynamics etc. within the music). The number sign is normally inserted automatically in the braille by CIMBAL wherever it is required. The null sign has no effect except to prevent contraction in the braille of groups of letters containing it.

Others: <@@> is used to produce a genuine "@" sign.
<@?> is used to produce a genuine question mark.
(note therefore that the single "@" sign cannot be backspaced - this effect may be achieved by following it with two question marks.)

The breath mark or break mark shown as a comma written above the stave is input as a text-above item comprising a single comma. The break mark shown as two short parallel oblique lines at the top of the stave is input as a text-above item comprising the characters <//>.

A4.5.6 Note items

A note item represents a note or a rest or a complete chord, together with associated dots, accidentals, inflexions, nuances, ornaments, etc.

For the purpose of this description, unless otherwise indicated,

"note" refers to either a single individual note, or a rest, or a complete chord. The term "note-head", or "head", is used to refer to one of the pitches associated with a note. Thus a chord has two or more note-heads and a single note in the conventional sense has one note-head.

A note item will not be accepted if it would cause the total time-count of the current measure to exceed the limit required by the time signature unless the user overrides the time-count check (see section A4.5.9).

A4.5.6.1 Note specifications

A note item is input as a sequence of "note-specifications" followed by a carriage-return. Each note-specification defines or modifies an attribute of the note. In the printed score these attributes are indicated by the form and vertical position of the note symbol, or by various other markings adjacent to the note symbol. Some note specifications refer to the note as a whole, while others refer to individual note-heads. This distinction is not significant for the user, except for the separation of the individual note-heads of a chord by the chord sign (see section A4.5.6.6).

It is not necessary to type specifications for all (or even any) of the attributes of each note. Each attribute of the note or note-head has a default value which is used if the corresponding specification is not input. The "current value" of each attribute of the note is set to its default value at the start of input of the item. The user may then replace or modify the attributes using the appropriate note specifications.

The default values for the attributes of a note-item are as follows:

For notes input using the "change-item" edit-command (see section A4.4.11) the default values for all attributes are the same as the values of the attributes of the selected item if it is a note item, or of the last note item in the measure otherwise.

For notes input using the "insert-after" or "insert-before" edit-commands, default attributes are taken from the selected item in the same way as they are taken from the previous note during ordinary input.

If a note-repeat is in effect, the default values are the same as those of a previous note as described in section A4.5.6.4.

Otherwise: length (but not dots) is the same as that for the previous non-rest note item on the same stave (semiquaver for the first note of the stave); pitch and octave for the uppermost note-head are the same as those for the uppermost note-head of the previous non-rest note item on the same stave (octave 4, pitch C for the first note of the stave); pitch and octave for subsequent note-heads (if any) of the note item are the same as those for the previous note-head of the current note item; all other attributes are "absent".

A particular attribute of the current note may be specified more than once. In this case the last specification for that attribute is

used. This provides an easy way of correcting certain mistakes. For example, if <#A.5> is typed by mistake instead of <#B.5>, this may be corrected by simply adding the correct pitch specification <B> giving <#A.5B>. (If the two values of the same attribute are given consecutively and there is any ambiguity, as many characters as possible are associated with the first of the two specifications. For example: <...> is two "dots" specifications <..> and <.>, not <.> and <..>; <...> is therefore equivalent to <.> rather than <..>.)

A question mark typed immediately following specification of a note attribute which cannot be individually cancelled (for example, pitch) has the same effect as pressing the interrupt button, that is, all specifications typed for the item are cancelled and the displayed characters are crossed out.

A4.5.6.2 Specifications applying to whole note

Apart from note-length and rest, each of these attributes may be reset to absent by typing the appropriate specification followed by a question mark. "Rest" is cancelled by typing any pitch-letter specification.

Name: length
Format: one of the nine note-length characters, which are:
<1> for breve
<2> for semibreve (open head) or whole bar rest
<3> for minim (open head with stem)
<4> for crotchet (solid head with stem)
<5> for quaver (stem with one flag)
<6> for semiquaver (two flags)
<7> for demisemiquaver (three flags)
<8> for hemidemisemiquaver (four flags)
<9> for semihemidemisemiquaver (five flags)
Meaning: defines the length of the note.
Restrictions: for a chord the length must be specified before the first "chord-sign" if the default value would cause a time-count error.

Name: dots
Format: <.> or <..>
Meaning: one or two dots following the note, indicating extended duration.

Name: rest
Format: <R>
Meaning: the current note-item is a rest.
Comments: where a whole-measure rest is shown as a semibreve rest in the print, this should be specified as a semibreve rest regardless of the time signature.

Name: expression marks
These are specified by typing the <=> sign followed by a single character identifying the type of expression mark. Apart from the three types of staccato, which are mutually exclusive and cancel each other, any combination of expression marks may be specified for one note-item. Only the pause is meaningful for a rest.

printed symbol	character typed	name
·	<.>	staccato
▾	<:>	staccatissimo
⊣	<->	mezzo-staccato
>	<>>	attack
<>	<S>	swell
–	<_>	agogic accent (tenuto)
𝄐	<P>	pause (fermata)
^	<^>	martellato
≋	<A>	arpeggio on one stave only
≋	<D>	arpeggio across two staves (should be input on both staves)

Name: note-fractioning
Format: <=/0> <=/1> <=/2> <=/3> <=/4> or <=/5>
Meaning: fractioning of a note, indicated in print by one or more short bars through the stem of the note. The digit corresponds to the number of these bars. The zero digit corresponds to repetition in crotchets.

Name: tremolo
Format: <=J1> <=J2> <=J3> <=J4> or <=J5>
Meaning: tremolo between the current note and the following note, indicated in print by short bars between the two notes, or beaming of minims. The digit corresponds to the number of these bars.
Comments: no tremolo indication is required for the second of the two notes.

Name: stem-sign
Format: <S>
Meaning: indicates that the note forms an additional stem attached to the previous note (see section A4.5.8 for details of its usage).

A4.5.6.3 Note-pitch specifications

Pitch is an attribute of each head of a note-item. It does not include accidentals, which are specified separately. The pitch of a note-head comprises an octave-number and a pitch-letter (C, D, E, F, G, A or B) within the octave. The octaves are numbered consecutively in ascending order, each including notes from C to the next B above. Octave 4 corresponds to the octave of the piano starting at middle C; octaves 1 and 7 correspond to the lowest and highest complete octaves of the piano respectively. The lowest and highest pitches accepted by CIMBAL are octave 0 C and octave 8 B respectively.

Pitch may be specified in one of two modes: non-transposing or transposing. In non-transposing mode, the pitch-specification corresponds to the true pitch of the written note-head. In transposing mode, pitch is specified as if all notes were written in the treble clef, regardless of the actual clef shown, and CIMBAL transposes it to the correct value before displaying and storing it. The note-head will not be accepted if this would cause the true pitch to be outside the permitted range. Example: in transposing mode, a note on the middle line of the stave would always be input as octave 4 B, regardless of the clef sign. For the convenience of musicians familiar with music written in the different clefs, CIMBAL operates by default in non-transposing mode. The user who wishes to change this option may select transposing mode using the initial "selection number" or the "CHANGE" command (see sections A4.2.3 and A4.4.6).

Three methods are available for specifying the pitch of note-heads. The most appropriate of these for each note-head depends on the context and user's musical knowledge.

(1) Using crosswires. The crosswires are switched on and off using the "crosswires" item (see section A4.5.3) or "crosswires" edit-command (see section A4.4.11). When switched on, the crosswires will appear whenever a character of the input language is expected, regardless of the type of the current item. With the crosswires switched on, the pitch of the note-head is determined by the vertical position of the horizontal crosswire at the time the last character for the note-head other than a carriage-return is typed. For a chord, for all but the lowest head, this is the <&> following specification of the note-head. Use of the crosswires for pitch specification will work correctly in both transposing and non-transposing modes. The other note-pitch specifications described below will not be accepted while the crosswires are switched on.

(2) Absolute pitch specification. The absolute pitch of the note-head may be specified using the following two specifications:

Name: pitch-letter
Format: one of the note-pitch characters which are: <A>, <B>, <C>, <D>, <E>, <F> and <G>
Meaning: pitch of the note-head.

Name: octave
Format: <O> followed by an integer in the range 0 to 8.
Meaning: the number of the octave, as defined above.

(3) Relative pitch specification. The pitch may be specified relative to its current value using one or more of the following specifications. To determine the current value of the pitch, see section A4.5.6.1. One pitch step corresponds to the interval between a line of the stave and an adjacent space, regardless of the actual musical interval. Relative pitch specification cannot be used for a rest.

Name: increment pitch
Format: <+>
Meaning: raises the current pitch value by one step (incrementing the octave number if appropriate).

Name: decrement pitch

Format: <->
Meaning: lowers the current pitch value by one step (decrementing the octave number if appropriate).

Name: increment octave
Format: <Y>
Meaning: raises the current octave number by 1.
Restrictions: this cannot be used if the current octave is 8.

Name: decrement octave
Format: <U>
Meaning: lowers the current octave number by 1.
Restrictions: this cannot be used if the current octave is 0.

A4.5.6.4 Other note-head specifications

Each of the following attributes may be set to absent by typing the appropriate specification followed by a question mark. For a chord, these specifications apply independently to each individual head.

Name: accidentals
Format: one of the following: flat (<$>), sharp (<#>), natural (<%>), double flat (<$$>), double sharp (<##>), or any of the above preceded by natural (<%$>, <%#>, <%$$>, or <%##>); this is followed by <^> for editorial accidentals written above or below the note, or <V> for editorial accidentals written in parentheses.
Meaning: accidental signs printed to the left of the note-head.

Name: fingering
Format: <=F> followed by either:
(1) An integer in the range 1 to 5
or (2) two integers in the range 1 to 5, separated by a comma <,> or colon <:>
Meaning: (1) Fingering for the note-head, printed as a small number above, below or alongside the note.
(2) The comma indicates a change of fingering on one note, printed as two small numbers side by side with a short line or slur between them; the colon indicates two alternative fingerings for the same note, usually printed as one small number above the other (notice, however, that for chords, fingering numbers stacked in this way refer to the fingering for each separate head).

Name: tie or implied rest
Format: <I>
Meaning: tie from current note head to the following note.
Comments: for a rest this specification indicates that the rest is implied but not actually shown in the print. No special indication is needed for the end of a tie.

Name: small note
Format: <=1> or <=2>
Meaning: note head written smaller than normal (grace note).
<=1> represents an acciaccatura (one of several consecutive small notes, or a single small note printed with a stroke through the stem, preceding a normal

note).
<=2> represents an appoggiatura (a single small note without a stroke through the stem, preceding a normal note).

Comments: this attribute is cancelled by ornament specifications.

Name: ornaments
Format: <=>, optionally followed by one or two accidental characters (<#>, <$>, or <%>), followed by a single character representing the type of ornament as follows:

printed symbol	character typed	name
	<M>	lower mordent (or mordent)
	<E>	extended lower mordent
	<U>	upper mordent (or inverted mordent)
	<X>	extended upper mordent
	<~>	turn above or below the note
	<I>	inverted turn above or below the note
	<N>	turn following the note
	<Z>	inverted turn following the note
	<T>	trill

A4.5.6.5 Special note specifications

Unlike the above note specifications, the position of the following note specifications in the sequence of specifications for a particular note is significant.

Name: chord-sign
Format: <&>
Meaning: the chord sign indicates that note specifications referring to a particular head of the current note item are complete, and subsequent specifications referring to an individual head are associated with the next head (see section A4.5.6.6).

Name: note-repeat
Format: (1) <">
or (2) <"*> followed by a single-digit integer n
or (3) <"*> followed by a single-digit integer n followed by <">
Meaning: the current values of the attributes of the current note item are set equal to those of a selected previous note, which is determined as follows:
(1) Single note repeat. The selected note is the immediately preceding note on the same stave.
(2) Multiple note repeat. Used to repeat a group of notes, with modifications, or with other items between

them. The selected note is the <n>th note preceding the current note, or the first note of the current measure if there are less than <n> notes in the current measure. Also the default attributes of the following <n-1> notes are taken from the <n-1> notes following the selected note respectively, instead of each from its own preceding note. The notes may then be modified individually by giving appropriate specifications for each.

(3) Automatic multiple note-repeat. Used to repeat a group of notes exactly. This is equivalent to using form (2) then typing <n> carriage-returns. <n> notes are automatically repeated and displayed.

Restrictions: the effect of a note-repeat specification is cancelled if an illegal note would be produced, for example, one exceeding the allowed time count for the current measure, but any notes already displayed will have been stored. This specification is also cancelled by editing the current measure while it is in effect. It may not be used during editing. Only notes within the current measure may be repeated using this specification.

Comments: the note-repeat specification applies only to note items. It does not affect and is not affected by non-note items. For a multiple note repeat the notes to be repeated need not necessarily exist at the time the note-repeat specification is typed, provided that each exists by the time it is required. Example: if the current measure contains a single note and <"*5"> is typed, this will produce a further five notes identical to the first, assuming that the time count limit for the measure is not exceeded. The second note is a repeat of the first, the third of the second and so on.

A4.5.6.6 Chords

A chord is input as a single note item. The individual heads of the chord are specified in order starting with the uppermost one. The specifications applying to individual note-heads are separated by the chord-sign <&>. The specifications for each note-head which apply to the chord as a whole (that is, those listed in section A4.5.6.2) may be interspersed with these in any position. If the pitch value for the second or subsequent note-head of a chord is equal to or higher than that of the previous note-head it will automatically be changed to the highest pitch with the same pitch letter which is lower than that of the previous note-head. Example: <05A&/> gives octave 5 A and octave 4 A. A maximum of eight heads per chord is allowed in the current implementation.

*A4.5.7 Braille items

Braille items are used only in unformatted or formatted braille modes, for inputting braille directly from the keyboard, or editing the braille. A braille item is one of the following:

Name: braille cell
Format: a sequence of digits in the range 1 to 7 and/or question marks followed by a carriage-return, or a plain

carriage-return.

Meaning: each digit 1-6 represents a dot of the braille cell, using conventional numbering 1,2,3 down the left-hand side of the cell and 4,5,6 down the right-hand side. Presence of a digit indicates presence of the corresponding dot in the cell, unless the digit is followed by a question mark, in which case the dot is cancelled. The carriage-return is used alone to represent the blank cell. Presence of a 7 indicates that on conversion to formatted braille, the cell is to be included if and only if it is in the first measure of a braille line.

Restrictions: the "delete" character (question mark) may only be typed immediately after a digit. The digit 7 may be used only in unformatted braille mode.

Name: short-form braille cell
Format: any letter
Meaning: this produces the braille cell corresponding to the letter according to the Standard English Braille code.

Name: format control
Format: same as braille cell, except that digit 8 is also used.
meaning: defines a special code controlling the operation of the braille formatter. Particular values and their meanings are defined in section A4.9 ("Braille editing").
Restrictions: format codes may not be used in formatted braille mode.

Name: end-of-measure
Format: <|>
Meaning: end of the current measure. Subsequent braille cells belong to the following measure.
Restrictions: not available during editing.

Name: end-of-input
Format: <!>
Meaning: as described in section A4.5.3.

A4.5.8 Order of input

The order of input of measures is the same as the order in which they are stored (described in section A4.3.1). Signs shown between two staves and equally applicable to both are input in the stave containing the first played note (not rest) to which they apply.

A4.5.8.1 Order of notes within a measure

If within a measure no two notes are played at the same time, considering a chord as a single note, the notes are input in order from left to right. If the whole measure or a portion of the measure consists of two or more sets of notes played together, that is, written in parallel, or "in-accord", this portion is input in the following order: the notes for one subsidiary part are input in order from left to right, followed by those for the next subsidiary part, and so on. The first group of notes is preceded by the set-return item (see section A4.5.3) and each subsequent group is preceded by the in-accord item. The final group is followed by the set-return item

unless there are no further notes in the measure. The order of subsidiary parts is from top to bottom for the right hand part of keyboard music, or non-keyboard music written in the treble clef, and from bottom to top for the left hand part of keyboard music, or non-keyboard music not in the treble clef.

Where individual notes have a second stem pointing in the opposite direction from the main stem and the additional notes cannot be written using an in-accord part, the additional stem should be represented as a separate note-item of the correct length immediately following the note-item to which the stem is attached, using the "stem-sign" specification to indicate that this note-item coincides in attack-time and pitch with the previous one.

A4.5.8.2 Order of other items within a measure

The start of an extended sign (see section A4.5.4) is input before the first note to which it applies and the end of the sign is input following the last note to which it applies. Other signs not associated with one particular note are input before the first note to which they apply. Sung words and syllables are input immediately before the note to which they apply.

The following is a general guide to the order of items preceding and following a note-item:

Start of pedalling
Text above or below stave
Start of crescendo or decrescendo
Start of slur
Note-beaming
Irregular note-grouping
Sung word or syllable
Note-item
End-of-slur
End of crescendo or decrescendo
Breath mark or break mark
End of pedalling.

A4.5.9 Time count check on input

There is a time-count check in effect during input unless the user overrides the check using the "ignore-checks" item or edit-command or the current time signature is "unmeasured". If the user cancels the time-count check, it will be restored at the start of each new measure, or by use of the "fresh-display" edit-command. For the first bar of a piece, the time count may be less than the normal bar length, but must be the same for each stave. Notes which cause the time count to exceed the required limit will not be accepted if time-count checking is in effect. Similarly, the "end-of-measure" item will be accepted only if the total duration of notes within the measure is correct according to the time signature. Allowance is made in evaluating the time-count for grace notes and stem-sign notes (considered for this purpose to have zero duration), in-accord parts, and irregular note-groupings. If grace notes occur at the end of a measure, it is necessary to use the "ignore-checks" item, because otherwise the time-count check will cause CIMBAL to reject any

note-item before it can be identified as a grace note.

A4.5.10 Editing during input

The measure currently being input may be edited at any time before the end-of-measure item is input. Previous measures may be edited only by using the "EDIT" command later, even if they are still displayed on the screen.

The editing mode is entered using the "cancel" item (question mark). The crosswires are then displayed on the screen and CIMBAL enters edit-command status. Editing is performed as described in section A4.4.11 with the following exceptions:
(1) The "cancel-edit" and "fresh-display" edit-commands are not available.
(2) The "end-of-edit" edit-command does not return CIMBAL to command status, but allows input to continue from the point where the editing mode was entered.
(3) Only items displayed for the current measure may be edited.
(4) The time count at the end of the edit need not correspond to that at the start of the edit but the overall time count for the current measure at the end of the edit may not exceed that allowed for the measure as a whole if time-count checking is in effect.

A4.6 Display format

Communication from CIMBAL to the user is via the display screen and bell of the terminal. Erasure of the screen and positioning of the cursor is performed by CIMBAL and it is therefore not necessary or desirable for the user to try to perform these operations with local controls on the terminal.

CIMBAL is designed for use with a storage tube rather than a continually refreshing display, so items may not be moved or removed from the display except by complete erasure of the screen. Deletion of an item displayed on the screen is temporarily indicated by a cross drawn through it. When CIMBAL wishes to erase the screen it will normally display the message "PRESS CARRIAGE-RETURN" and sound the bell. If CIMBAL is executing the "DISPLAY" command, no message is displayed, and only the bell is sounded. The user should then press the carriage-return key to allow CIMBAL to clear the screen and continue. Any other input in response to this message will be rejected. However, if the interrupt button is pressed, this will take effect immediately after the screen has been erased.

The following types of display are used:

1 Printed music
2 Braille music
3 Textual messages

A4.6.1 Printed music display

The music is displayed in a form roughly resembling the printed score. All symbols apart from note-heads, text and dots are composed

of straight line segments. Note-heads are displayed using a letter O (upper case for ordinary notes, lower case for small notes) overtyped in the case of solid heads with an asterisk (for ordinary notes) or a dot (for small notes).

The size of the Tektronix screen allows up to six parallel staves to be displayed at one time. During music input, up to four staves are displayed at one time and the lower part of the screen is used for messages and reflection of input characters. Thus, for example, for two-stave music, two parallels may be displayed during input and three during playback. The number of the first bar of each displayed parallel is shown at the beginning of that parallel. If the number of staves selected is greater than the number which can be shown on the screen at one time, or less than the total number of staves in the current section of the file, the stave number is also displayed at the start of each stave.

Some control items are not normally displayed, but all are displayed during execution of the "EDIT" command to enable them to be accessed if necessary. In this case set-return is shown as "@", in-accord as ";", right-hand and left-hand signs as "RH" and "LH" respectively, and other control items as the characters used to input them.

Each data item displayed has an associated screen position, which identifies it for editing purposes, and which is at the centre of the cross drawn through the item when it is deleted. The screen position of the various items is as follows:

item	screen position
start of extended sign	left hand side of displayed sign
end of extended sign	right hand side of displayed sign
text	centre of first character
note item	centre of note-head furthest from tail of stem (or occasionally, for chords, centre of the note-head nearest to tail of stem)
other items	near centre of item as displayed

A4.6.2 Braille music display

The braille music is displayed in a form resembling the embossed braille using the dot character to represent the dots. A blank cell is represented by a horizontal line beneath the cell.

In unformatted braille mode a non-blank cell displayed with a horizontal line beneath it represents either a cell which is to be omitted on formatting unless the cell is in the first measure of a braille line, or a special format control item.

The size of the Tektronix screen allows up to 14 lines of up to 40 braille cells each to be displayed at one time. Bar or line numbers are shown at the right hand side of the screen, using a

separate line if the braille line contains 40 cells.

A4.6.3 Messages

Messages are of three types: questions, informatory messages, and error messages. They are displayed either at the left hand side of the screen, or halfway across the screen, starting below the displayed music if any, or at the top of the screen otherwise.

All questions displayed by CIMBAL end with a question mark and require the user to type something. Informatory messages are self-explanatory. Error messages are described in full in section A4.7.

A4.7 Error messages

This section lists possible error messages produced by CIMBAL in response to erroneous input by the user, indicating their meaning and what action is required.

Message: <bell>.
Reason: a character has been typed which is not allowed in the given context by the music input language.
Note: the bell may also accompany other messages or sound as a margin indicator when items are displayed near the right hand side of the screen. The bell is also used during display to indicate that the screen is full and the user is to press carriage-return to clear the screen and continue.
Remedy: consult the "restrictions" entry of the appropriate part of section A4.5 to find out why the character is not acceptable. For a note-item or "end-of-measure" item, check that the time-count is correct. If CIMBAL appears to reject valid characters, press the interrupt button and start the current item again.

Message: NO MUSIC STORED.
Reason: the specified command requires music to be stored in the current section of the disc file.
Remedy: change mode using the "MODE" command or produce some music using the "RESTORE" or "BUILD" commands.

Message: FILE PROTECTED.
Reason: a command has been given which would cause a protected section of the disc file to be deleted.
Remedy: remove the protection from the appropriate section or sections of the file using the "CLEAR" or "SAVE" commands. If necessary, use the "SUMMARY" command to find which sections are "protected".

Message: WRONG MODE.
Reason: a command has been given which is not appropriate to the current mode.
Remedy: change the mode using the "MODE" command.

Message: MUST BE IN RANGE <m> TO <n>. (where <m> and <n> are

integers)

Reason: a number has been typed which is not in the required range.

Remedy: type the correct number in the range <m> to <n> inclusive, or press the interrupt button to cancel the command.

Message: TIME COUNT ERROR.

Reason: an attempt has been made to end an edit when the total time count value has been changed.

Remedy: either continue editing to correct the time count or use the "ignore-checks" edit-command to ignore the check, then repeat the "end-of-edit" edit-command.

Message: *ERROR <x> <y> <z>. (where <x>, <y> and <z> are integers).

Reason. there is a mismatch between items which should be paired: The number <x> is a code for the item as follows:

49 Start of slur
50 End of slur
51 Ottava below
52 End of ottava
53 Ottava above
54 Start of crescendo
55 Start of decrescendo
56 End of crescendo or decrescendo

The numbers <y> and <z> are the bar and stave numbers respectively of the measure in which the erroneous item was found. Example: *ERROR 50 10 1 means an "end-of-slur" item has been found in bar 10, stave 1, which has no corresponding earlier "start-of-slur" item.

Remedy: insert or delete items using "EDIT" command so that the items match correctly.

Message: THE TAPE INDEX IS FULL.

Reason: an attempt has been made to save a file when there is no room for it in the tape index. If there are deleted files on the tape, index space used by these files may be recovered using the "COPY" command, otherwise a new tape must be started.

Message: WRONG TAPE OR NO TAPE LOADED.

Reason: the loaded tape is not a music archive tape, or no tape is loaded, or the tape is loaded on the wrong unit.

Remedy: load the correct tape (see section A4.2) and try again.

Message: THIS FILE IS NOT ON THE TAPE.

Reason: a file identifier or name and/or version number have been specified for a "RESTORE" or "ERASE" command which do not correspond to any of the files on the archive tape.

Remedy: try again with the correct identifier or name and version number. If necessary use the "FILES" command to find out the identifiers, names and version numbers of files on the tape.

Message: BRAILLE OUTPUT FILE NOT AVAILABLE.

Reason: The "EMBOSS" command has been used when the braille

output file is not ready.
Remedy: See section A4.2.

The following error messages may be produced by CIMBAL but should not appear in normal use. Remedy for all the following: note the circumstances and call an expert.

Message: BAR FULL.
Message: EDIT TABLE FULL.
Message: BAR TOO LONG.
Reason: too many items have been input for the current measure.

Message: ITEM UNRECOGNISED.
Reason: the file contains an item code which is not recognised by CIMBAL.

Message: TAPE INDEX ERROR.
Reason: the archive tape index is not in the correct format.

Message: TAPE ERROR.
Reason: an error has been encountered on trying to read from or write to the archive tape.

Message: FILE FORMAT ERROR.
Message: DISC FILE ERROR.
Reason: the disc file has overflowed or is corrupted.

A4.8 Example

The following example indicates how part of the printed piece of music shown in figure A7:1 might be input. In this description a slash (/) indicates that a carriage-return is to be typed.

The actual sequence of characters typed is shown at the left hand side with an explanatory commentary to the right. The division of characters typed into separate lines in the description is merely for correspondence with the commentary. In practice the characters are typed consecutively. The commentary becomes progressively less detailed. For the first four bars of the piece, each item is shown on a separate line.

For illustration, alternative methods of indicating pitch are given, using either pitch and octave specifications or + and - to give the pitch relative to the preceding note. Either may be easier to use, depending on the familiarity of the user with the pitch names. Lines with commentary preceded by an asterisk contain deliberate mistakes included to demonstrate editing during and after input. It is assumed that CIMBAL is running in non-transposing mode. Characters are typed as follows:

characters typed	commentary
BU	initiates the "BUILD" command
2/	first initial specification: number of staves
RH/	part 1: right hand

```
LH/                     part 2: left hand
/                       select both staves for input
EX.1/                   identifier: anything up to four characters
MOZART. SONATA          file name: @L represents braille letter sign
 IN @LD/                rest of file name
Sonata in @LD/          title
Second movement         subtitle
 (primo part)/          rest of subtitle
Mozart/                 composer
/                       no publisher given
K.381/                  number
Andante/                tempo
/                       no metronome mark is given
This is an example/     comment
k1#                     key signature: one sharp sign
t3/4/                   time signature: 3/4
zt/                     treble clef for right hand
zb/                     bass clef for left hand
                        here the screen is cleared and a staff displayed
vsotto voce/            bar 1, right hand: text below stave
o5d4./                  1st note: octave 5, pitch d, length code 4,
                        dotted
(                       start of slur (precedes first of slurred notes)
j3/                     join next three notes
--5/                    2nd note: two pitch steps down from 1st, length
                        code 5
++/                     3rd note: up two pitch steps, same length
----/                   4th note: down four pitch steps (could use ug/
                        instead)
)                       end of slur (follows last of slurred notes)
|/                      bar line
j4/                     join next four notes
o3G6/                   bar 1, left hand, 1st note: octave 3 G, length
                        code 6
++++/                   2nd note: up four pitch steps, same length
--/                     3rd note: back down 2
++/                     4th note: up again
j4/                     join four notes
"*4"                    5th to 8th notes: automatically repeat 4 notes
j4/                     join four notes
"*4"                    9th to 12th notes: repeat last four notes
|/                      bar line
j3/                     bar 2, right hand: join three notes
(                       start of slur
.=T/                    1st note: same length and pitch, dotted, trill
-7/                     2nd note: one pitch step down, length code 7
+/                      3rd note: up again, same length
-4                      *4th note: mistake, end of slur should come first
                        press interrupt button to cancel -4
)                       end of slur
-4/                     4th note: down one pitch step, length code 4
R/                      rest: same length as previous note
|/                      bar line
j4/                     left hand: join four notes
---/                    1st note: down 3, same length as previous note
+++/                    2nd note: back up 3
-/                      3rd note: down one
+/                      4th note: up again
j4/                     join four notes
```

```
"*4"                5th to 8th notes: automatically repeat 4 notes
j4/                 join four notes
"*4"                9th to 12th notes: repeat 4 notes again
|/                  bar line
                    here a second staff is displayed
o5C/                *Bar 3, right hand, 1st note: re-specify pitch
                    after rest (mistake:  dot missing from note)
(                   start of slur
j3/                 join three notes
5--/                2nd note
++/                 3rd note
uf/                 4th note: octave down, pitch F (could use ----/
                    instead)
)                   end of slur
?/                  cancel attempted bar line
?                   attempted edit: not accepted because in middle of
                    bar line
?                   initiate editing of current measure
c                   position crosswires to first note, type C for
                    change
./                  same pitch and length as before, add dot
!                   end of editing
|/                  bar line now accepted
j4/                 bar 3, left hand
u/                  1st note: one octave down, same length
y-/                 2nd note: up an octave, down one pitch step
--/                 3rd note
++/                 4th note
j4/                 join four notes
"*4"                repeat four notes
j4/                 join four notes
"*4"                repeat four notes
|/                  bar line
(                   bar 4, right hand: start of slur
4++/                1st note: up two pitch steps, length 4
-/                  2nd note
)                   end of slur
r/                  rest
|/                  bar line
j4/                 bar 4, left hand, join four notes
---/                1st note
+++/                2nd note
--/                 3rd note
++/                 4th note
j4/                 join four notes
"*4/                5th note: repeat 4 notes but...
+/                  6th note: raise the second by one pitch step
+/                  7th note: likewise the third
+/                  8th note: and fourth
j4/                 join four notes
"*4"                9th to 12th notes: repeat previous 4 notes
                    exactly
|/                  bar line
                    here press carriage-return to clear screen (in
                    response to a message from CIMBAL)
o4E./j3/5/+/+/|/    *bar 5, right hand (missing trill)
j4/u-/g/--/++/      bar 5, left hand
j4/"*4"j4/"*4"|/
4/(j4/D5/G/b/yd/)|/ bar 6, right hand
```

```
j4/ub/yg/d/g/          bar 6, left hand
j4/"*4"j4/"*4"|/
4&--/                  bar 7, right hand: chord, top pitch same, bottom
                       2 below
-&-/                   *2nd chord, top pitch down one, lower should be
                       two below, mistakenly typed one below
-&g/|/                 3rd chord (could use -- instead of g) and bar
                       line
j4/-/yd/---/+++/       bar 7, left hand
j4/"*4"j4/"*4"|/
@/                     bar 8, right hand: set return for in-accord
j3/5.=T/(-7/+/)-5/     upper part of in-accord
;/                     in-accord sign
-4/                    pitch one less than last note input
-5/                    quaver
@/                     set-return sign indicates end of in-accord
r/                     rest (crescendo goes with left hand)
r4/|/                  repeat R otherwise last non-rest note is used
j4/u/y/#-/+/u/         bar 8, left hand, first 5 notes
zt/                    treble clef sign
<                      start of crescendo
j3/y/                  6th note: up an octave (note clef change)
+/+/j4/+/+/+/+%/       remaining notes
 |/                    space for end of crescendo, and bar line
vpiu@` f/              bar 9, right hand: text precedes lst note (@  is
                       accent)
4o6D./(j3/--5/++/      rest of right hand
ug/)|/
o5D4./(j3/--5/++/      bar 9, left hand
ug/)|/
(j3/.=T/-7/+/)-4/      bar 10, right hand
r/|/
|".lu/                 repeat right hand measure an octave below
o6C./(j3/--5/++/       bar 11, right hand
uf/)|/
|".lu/                 left hand: repeat right hand an octave below
(4++/-/)r/|/           bar 12, right hand
|".lu/                 left hand: repeat right hand an octave below
E./(j3/=t5/+/+/)|/     bar 13, right hand
|".lu/                 left hand octave below right hand
(4/j4/---5/)+++=./     bar 14, right hand: first three notes
"++/"++/|/             use of note-repeat saves re-typing staccato
|".lu/                 left hand octave below right hand
"++/"++/|/
=26/j3/-5/             bar 15, right hand: appoggiatura, same pitch
-6=./"-/-4/-=T/|/
|".lu/                 left hand octave below right hand
!                      end of input

ED                     initiate EDIT command
5/                     bar number
1/                     stave number
C                      position crosswires to head of 2nd note,type C
                       for change
=t/                    add trill
!                      end of edit

ED                     initiate EDIT command
7/                     bar number
```

1/	stave number
C	position crosswires to 2nd chord, C for change
&-/	1st head same, 2nd head one pitch step lower than before
!	end of edit
DI	DISPLAY command
/	select all bars stored
/	select both staves
	the entire piece of music just input will be displayed. press carriage-return each time the page is full and the bell sounds.
ST	attempted "STOP" command. CIMBAL will say "FILE PROTECTED". (Acceptance of the "STOP" command could result in losing the music just input)
SA	"SAVE" command. First check that the archive tape is loaded. The file is copied to magnetic tape.
ST	"STOP" command to release CIMBAL.

*A4.9 Braille editing

Two commands are available for editing of the braille music if this is considered necessary: the "EDIT" command may be used as for printed music to edit braille cells and format control codes, and the "FLAG" command may be used to change a numeric value associated with each measure of unformatted or formatted braille music which influences the layout of the braille.

For a measure of unformatted braille, the "flag" value is the number of cells which the measure should be indented relative to the start of the bar, to give correct alignment of first notes in bar-over-bar format. For a line of formatted braille, the "flag" value is the number of lines following the current line which must appear on the same braille page. This should be set greater than or equal to the number of lines per braille page if the current line is required to start a new page regardless of the position on the current page.

The following format control codes are defined:

Code	Characters typed	meaning
252	345678	text separator: preceding cells of the current measure are put on a separate braille line on formatting.
253	1345678	potential split: a measure exceeding one braille line in length may be split at this point.
254	2345678	force new line: the following cell starts a new braille line on formatting.
255	12345678	filler: this code is ignored.

Table A4:1 Instrument and voice codes

Accordion	ACC	Horn	HORN
Alto flute	AFL	Kettledrum	KD
Alto voice	ALTO	Marimba	MAR
Anvil	ANV	Marimba gong	MG
Bass clarinet	BCL	Oboe	OBOE
Bass drum	BD	Oboe d'amore	OBA
Bass trombone	BTRB	Organ pedals	PED
Bass trumpet	BTRP	Piano left hand	LH
Bass tuba	BTUB	Piano right hand	RH
Bass voice	BASS	Piccolo	PIC
Bassoon	BN	Rattle	RAT
Bells	BELL	Saxophone	SAX
Castanets	CAST	Side drum	SD
Celesta	CEL	Soprano voice	SOP
Cello	VC	Tabor	TAB
Clarinet	CL	Tambourine	TAMB
Cornet	COR	Tenor drum	TD
Cymbals	CYM	Tenor voice	TEN
Double bass	DB	Timpani	KD
Double bassoon	DBN	Triangle	TRI
Dulcitone	DUL	Trombone	TRB
English horn	EH	Trumpet	TRP
Euphonium	EUPH	Tuba	TUBA
Flugel horn	FLH	Viola	VA
Flute	FL	Violin (first)	VN1
Glockenspiel	GL	Violin (second)	VN2
Gong	GONG	Vocal	VOC
Guitar	GUIT	Wire brushes	WB
Harp	HARP	Xylophone	XY

Table A4:2 Meanings of keyboard characters

The following table gives the usage of the keyboard characters in the music input language. "Primary meaning" is the meaning of the character when it is the first character typed for an item or note specification. "Secondary meanings" gives other meanings for the character which are dependent on previous characters typed for the same item. In addition to these special meanings, each character may also represent itself (in numbers and text items).

Character	Primary meaning	Secondary meanings
A	Note pitch	Single stave arpeggio
B	Note pitch	Bass clef
C	Note pitch	Common time
D	Note pitch	Double stave arpeggio
E	Note pitch	Extended lower mordent
F	Note pitch	Fingering, Bass clef, Footnote
G	Note pitch	Treble clef
H	Horizontal display scale factor	
I	Tie, implied rest	Inverted turn above/below note
J	Note beaming	Tremolo
K	Key signature	
L	Inkprint line number	

M	Ignore-checks	Lower mordent
N	Irregular grouping	Turn following note
O	Octave number	Encircled cross
P	Pedalling	Pause
Q	Inkprint page number	
R	Rest	
S	Stem sign	Swell, Segno
T	Time signature	Trill, Treble clef
U	Decrement octave	Upper mordent, Stem direction
V	Text below stave	
W	Word or syllable	
X	Crosswires	Extended upper mordent, Null
Y	Increment octave	Stem direction
Z	Clef sign	Inverted turn after note
1	Breve	Soprano clef, First ending
2	Semibreve	Second ending
3	Minim	Alto clef
4	Crotchet	Tenor clef
5	Quaver	
6	Semiquaver	
7	32nd note	
8	64th note	Treble clef with 8 below
9	128th note	
0	Unused	
!	End of input	
"	Note repeat	Bar repeat
#	Sharp	
$	Flat	
%	Natural	
&	Chord sign	
'	Unused	Acute accent
(	Start of slur	
)	End of slur	
=	Escape character	Stem direction
{	Unused	
}	Unused	
[	Ottava below	
]	Ottava above	
-	Decrement pitch	Mezzo-staccato, Left hand
+	Increment pitch	Extended lower mordent, Right hand
`	Unused	Grave accent
@	Set-return, Control-item escape character	
|	Bar line	
\	Bar line	
_	End of ottava	Agogic accent, Umlaut
;	In-accord with	
:	Dots before barline	Dots after barline, Staccatissimo
*	End of pedalling	
,	Unused	
.	Dotted note	Staccato
/	Unused	Fractioning
?	Enter edit mode	Delete current item or specification
<	Start of decrescendo	
>	Start of crescendo	Attack
Space	End of crescendo or decrescendo	
^	Text above stave	Martellato, Circumflex accent
~	Unused	Turn above/below note
C/R	End of item or number	

Index to appendix 4

Appendix 5. External data format definition

This appendix gives a detailed definition of the format of the disc and magnetic tape files used for storage of the digital representation of print and braille music notation.

Throughout this description, numbering starts from zero unless otherwise indicated. Thus the first sector of a file is sector 0, the bytes of a sector are numbered 0 to 359, the bits of a byte are numbered 0 to 7 from left to right (high-order to low-order). All numbers are decimal unless otherwise stated. Where a value is specified for a field occupying more than one bit or byte, this is the value associated with the field as a whole rather than each bit or byte of the field.

A5.1 Disc file format

The disc file is a randomly-accessed file containing a number of sectors of fixed length. This length is the actual sector size of the Sigma 5 disc, 360 bytes, in the prototype implementation. The first sector of the file is a header sector containing the music header record and information describing the structure of the file. The remaining sectors are linked together in up to four linked lists. One linked list is always present and contains sectors of unused storage space. The remaining lists, referred to as "sections" of the file, contain representations of printed music, unformatted braille music, and formatted braille music respectively. Consecutive sectors of a linked list are not necessarily contiguous or in order in physical storage.

Header sector format

Bytes 0 - 1	Undefined.
Byte 2	0
Byte 3	Initial display scale factor (16 to 255).
Bytes 4 - 7	File type identifier: 4 EBCDIC characters "MUZK".
Bytes 8 - 11	File identifier: four EBCDIC characters as input by the user, filled with trailing blanks.
Bytes 12 - 13	Number of sections in file (=3).
Bytes 14 - 15	Number of bars of printed music.
Bytes 16 - 17	Number of bars of unformatted braille.
Bytes 18 - 19	Number of lines of formatted braille.
Bytes 20 - 21	Number of staves of printed music.
Bytes 22 - 23	Number of staves of unformatted braille.
Bytes 24 - 25	1
Bytes 26 - 27	Pointer to first sector of printed music.
Bytes 28 - 29	Pointer to first sector of unformatted braille.
Bytes 30 - 31	Pointer to first sector of formatted braille.
Bytes 32 - 33	Pointer to last sector of printed music.
Bytes 34 - 35	Pointer to last sector of unformatted braille.
Bytes 36 - 37	Pointer to last sector of formatted braille.
Bytes 38 - 39	Pointer to first sector of free storage.
Bytes 40 +	Printed music part codes (one byte per stave). These are provisionally assigned as follows:

1 Right hand	16 Trumpet	31 Marimba	46 Violin 2
2 Left hand	17 Bass trumpet	32 Marimba gongs	47 Viola
3 Pedals	18 Trombone	33 Side Drum	48 Cello
4 Piccolo	19 Bass trombone	34 Tenor drum	49 Double bass
5 Flute	20 Tuba	35 Bass drum	50 Harp
6 Alto flute	21 Bass tuba	36 Tabor	51 Guitar
7 Oboe	22 Cornet	37 Tambourine	52 Accordion
8 Oboe d'amore	23 Euphonium	38 Triangle	
9 English horn	24 Saxophone	39 Cymbals	
10 Clarinet	25 Kettledrum	40 Gong	65 Vocal
11 Bass clarinet	26 Bells	41 Castanets	66 Soprano
12 Bassoon	27 Glockenspiel	42 Rattle	67 Alto
13 Dble bassoon	28 Celesta	43 Anvil	68 Tenor
14 Horn	29 Dulcitone	44 Wire brushes	69 Bass
15 Flugel horn	30 Xylophone	45 Violin 1	

These are followed by the header record (see below). The unused remainder of the header sector is filled with bytes containing zero. For empty sections, zero is stored in the fields used for the pointers to the first and last sector of the section.

Non-header sector format

Bytes 0 - 1	Forward link: pointer to next sector of linked list. Undefined for the last sector of each section.
Bytes 2 - end	Data field.

A zero byte in the data field indicates that the remainder of the sector is to be ignored, and data continues at the start of the data field of the next sector in the chain. The form of records and data items is such that no zero byte can occur in the data itself.

Record format

Within each section of the file, data is stored as a sequence of variable-length records, with minimum length 1 byte and maximum length 254 bytes. The header sector contains one special record which contains the header information input by the user as initial specifications (see appendix 4).

The records are stored consecutively within each section, except that adjacent records may be separated by one or more padding bytes containing the value 255. A single record may continue from one sector to the next. The final data record in each section is followed by a special end-of-section record.

In the printed music and unformatted braille sections, each record contains the data for one measure of music. In the formatted braille section, each record contains the data for one line of braille.

Music record format

A music record is either of length one byte, containing:

254	measure rest (printed music section only)

or of variable length from 2 to 254 bytes, of the form:

Byte 0	the total number of bytes in the record, excluding this byte itself (in the range 1 to 253).
Byte 1	
Bit 0	1
Bits 1-7	flags field
Following two bytes (unformatted braille section only): (reserved for use by the formatter):	
Bit 0	1
Bits 1-7	number of cells in longest measure of current bar of unformatted braille (first stave only, after formatting only). For the first of the two bytes this count includes all cells; for the second it excludes conditional cells.
Following bytes:	
A (possibly empty) sequence of data items (defined below).	

Braille items may appear only in the braille sections and other items may appear only in the printed music section.

The value of the flags field is dependent on the section:

Printed music section:	
Bits 0-5	0
Bit 6	1 if the measure contains any clef, key signature, time signature or scale-factor item, or the "state" at the end of the measure is different from that at its start; 0 otherwise.
Unformatted braille section:	
Bit 0	1 if a dot is required following hand signs (stave 1 only, after formatting only); 0 otherwise.
Bits 1-6	number of spaces to indent first cell of measure.
Formatted braille section:	
Bits 0-6	number of blank lines which must remain on the current braille page following the current line; a number greater than the maximum number of lines per page indicates an unconditional new page.

Header record format

The header record has the same format as a music record except that the first byte of the data field contains 8, the number of text items in the header record other than the name. Each of these items is always present, but may have a character count of zero. The items in the header record are:

```
 1     Name  (text item)
 2     Title  (text item)
 3     Subtitle  (text item)
 4     Composer  (text item)
 5     Publisher  (text item)
 6     Number  (text item)
 7     Tempo  (text item)
 8     Metronome mark  (text item)
 9     Comments  (text item)
10     Time signature  (separate-sign item)
11     Key signature  (separate-sign item)
12+    Clefs for each stave of printed music (separate-sign items).
```

The flags field of the header record contains 1 if the first measure of printed music is not less than a full measure according to the time signature in the header, otherwise 0.

End-of-section record format

An end-of-section record is the last record of each section. It is of fixed length 2 bytes:

```
Byte 0        1
Byte 1        127
```

Data item

A data item is one of the following:

```
1     Control item
2     Separate-sign item
3     Text item
4     Note item
5     Braille item
```

Control item

A control item is of fixed length one byte and has the form:

```
Byte 0  Bits 0-2      0
        Bits 3-7      item-code
```

where item-code has one of the following values:

```
 2   In-accord sign
 3   Return point for in-accord
 4   Single barline (no longer used)
 5   Double barline preceded by dots
 6   Double barline without dots
 7   Double barline followed by dots
 8   Double barline preceded and followed by dots
 9   Thin double barline
10   Dotted barline
```

11	Start of footnote
12	Segno
13	Encircled cross sign
14	Pause sign
15	Subsequent notes are written on the correct stave
16	Subsequent notes are written on the stave below
17	First ending (prima volta)
18	Second ending (seconda volta)
19	Cancel force-stem-direction
20	Subsequent note stems point upwards
21	Subsequent note stems point downwards
22	Null item

Separate-sign item

There are two groups of separate-sign items. Those in the first group have fixed length 2 bytes and those in the second group have fixed length 1 byte. The first byte of separate-sign items has the form:

Byte 0	Bits 0-2	1
	Bits 3-7	Item-code

where item-code has one of the following values:

1	Clef sign
2	Time signature
3	Key signature
4	Irregular note-grouping
5	Inkprint page number
6	Inkprint line number
7	Note beaming
8	Display scale change
17	Start of slur
18	End of slur
19	Start of ottava below
20	End of ottava above or below
21	Start of ottava above
22	Start of crescendo
23	Start of decrescendo
24	End of crescendo or decrescendo
25	Start of pedalling
26	End of pedalling
27	Half pedalling

If item-code = 1 (clef sign) then:

Byte 1 128 plus the position of middle C with respect to the centre line of the stave. Only the following values are currently used:

122	Treble clef
124	Soprano clef (C clef on bottom line)
128	Alto clef (C clef on middle line)
129	Treble clef with "8" below
130	Tenor clef (C clef on fourth line)
134	Bass clef

```
If item-code = 2 (time signature) then either:
Byte 1  Bits 0-2      6 for C;
                      7 for C bar.
        Bits 3-7      0
Or:
Byte 1  Bits 0-2      logarithm (base 2) of lower figure of time
                      signature (in the range 1 to 5)
        Bits 3-7      upper figure of time signature less 1 (in the
                      range 0 to 31).

If item-code = 3 (key signature) then:
Byte 1  Bits 0-2      0
        Bits 3-4      1 for flats;
                      2 for naturals or no sign;
                      3 for sharps.
        Bits 5-7      number of accidentals (in range 0 to 7).

If item-code = 4 (irregular note-grouping) then:
Byte 1  number of following notes in the group.

If item-code = 5 (inkprint page number) then:
Byte 1  number of printed page (in range 1 to 255).

If item-code = 6 (inkprint line number) then:
Byte 1  number of parallel on current printed page (in range 1 to 9).

If item-code = 7 (note-beaming) then:
Byte 1  number of following notes beamed (in range 1 to 16); includes
        rests contained within beamed group; the nature of the
        beaming is determined by the length of the notes themselves.

If item-code = 8 (display scale) then:
Byte 1  display scale factor.
```

Text item

A text item is of variable length from 1 to 33 bytes and has the form:

```
Byte 0  Bits 0-1      1
        Bit  2        0  for musical expression mark or dynamic;
                      1  for sung word or syllable.
        Bit  3        if bit 2 = 0 then:
                      0  for text appearing above the stave;
                      1  for text appearing below the stave.
                      if bit 2 = 1 then:
                      0  if not last syllable of word;
                      1  otherwise (i.e. to be continued).
        Bits 4-7      0 to 14   number of bytes of text following;
                      15 indicates byte count is in following byte.

If byte 0 bits 4-7 contain 15 then:

Byte 1  number of bytes of text following  (in range 15 to 31)
```

The following bytes contain the text in EBCDIC code. Certain unassigned EBCDIC codes are used for inflected letters and special

symbols. The meanings associated with the following values are shown in the braille music sign table (see appendix A1.3): 33-53, 64, 74-80, 90-97, 106-111, 122-127, 129-142, 145-158, 162-174, 177-181, 186-190, 193-206, 209-222, 226-238, 240-254. Other values are not used.

Note item

A note item is of variable length from 2 to (5+4n) bytes, where n is the maximum number of note-heads permitted in one note item (8 in the current implementation). A note item has the form:

Byte 0 Bit 0 — 1

Bit 1 — 1 if the note has "extras" (see below); 0 otherwise.

Bit 2 — 1 if the note item is a chord, rest or stem sign; 0 otherwise.

Bits 3-7 — Duration of note in the form 3m + n, where the length code m is:

1	for a breve	(1024 units)
2	for a semibreve	(512 units)
3	for a minim	(256 units)
4	for a crotchet	(128 units)
5	for a quaver	(64 units)
6	for a semiquaver	(32 units)
7	for a 32nd note	(16 units)
8	for a 64th note	(8 units)
9	for a 128th note	(4 units)

and the dots code n is the number of dots following the note (in range 0 to 2).

Following bytes: "Extras" specifications (defined below; present if and only if byte 0 bit 1 is 1).

Following byte (present if and only if byte 0 bit 2 is 1):

Bit 0 — 1 for implied rest or tied stem sign; 0 otherwise.

Bits 1-6 — 1 for a rest; 65 for a stem sign; number of heads for a chord (in range 2 to 8).

Following bytes: Note-head specifications for each note-head in descending order of pitch (present only if the note item is not a rest or stem sign).

The extras specifications form a sequence of up to 3 bytes, each of the form:

Bit 0 — 0 for the last byte of the sequence; 1 otherwise.

Bits 1-2 — type code (in range 1 to 3).

If type code is 1 then:

Bit 3 — 0

Bit 4 — 0 for note fractioning; 1 for tremolo with following note.

Bits 5-7 code for unit of repetition:
0 = crotchet (tremolo only)
1 = quaver
2 = semiquaver
3 = 1/32nd note
4 = 1/64th note
5 = 1/128th note.

If type code is 2 then:

Bit 3	1 for double stave arpeggio; 0 otherwise.
Bit 4	1 for single stave arpeggio; 0 otherwise.
Bit 5	1 for mezzo-staccato; 0 otherwise.
Bit 6	1 for staccatissimo; 0 otherwise.
Bit 7	1 for staccato; 0 otherwise.

If type code = 3 then:

Bit 3	1 for attack; 0 otherwise.
Bit 4	1 for martellato; 0 otherwise.
Bit 5	1 for agogic accent; 0 otherwise.
Bit 6	1 for swell; 0 otherwise.
Bit 7	1 for pause; 0 otherwise.

Each note-head specification has up to four bytes as follows:

Byte 0	Bit 0	1 if the note head has marked accidentals or is tied to the following note; 0 otherwise.
	Bit 1	1 if the note has fingering or ornaments; 0 otherwise.
	Bits 2-7	pitch specification in the form 7m + n, where m is the octave number and the pitch code n is one of: 0 for rest, 1 for C, 2 for D, 3 for E, 4 for F, 5 for G, 6 for A, 7 for B.

Following byte (present if and only if byte 0 bit 0 is 1):

Bit 0	1 if the note head is tied to the following note; 0 otherwise.
Bit 1	0
Bit 2	1 if the accidentals are in parentheses; 0 otherwise.
Bit 3	1 if the accidentals are written above or below; 0 otherwise.
Bit 4	1 if the note head is preceded by a natural sign; 0 otherwise.
Bit 5	1 if the note-head is preceded by a flat sign; 0 otherwise.
Bits 6-7	Number of sharps or flats preceding the note-head (in the range 0 to 2).

Following byte: fingering (present only if byte 0 bit 1 is 1):

Bit 0	0
Bit 1	1 if ornaments specification is present; 0 otherwise.
Bit 2	1 if a change of fingering is indicated on the same note; 0 otherwise.
Bits 3-7	5m + n, where m is the second fingering (in range 1 to 5) or zero if no second fingering present, and n is the first fingering (in range

1 to 5).

Following byte: ornaments (present if and only if byte 0 bit 1 is 1 and fingering byte is absent or has bit 1 = 1).

Bit 0	1
Bit 1	1 if the ornament is a turn; 0 otherwise.
Bits 2-3	inflexion code for ornament (in range 0 to 3).
	If bit 1 is 1 (turn) then:
Bits 4-5	inflexion code for accidental below turn (in range 0 to 3).
Bits 6-7	ornament type code (in range 0 to 3).
	If bit 1 is 0 (mordent, trill or grace note) then:
Bits 4-7	ornament type code (in range 4 to 10).

where inflexion code = 0 for no inflexion
1 for flat
2 for natural
3 for sharp

and ornament type = 0 for turn above or below note
1 for inverted turn above or below note
2 for turn following note
3 for inverted turn following note
4 for trill
5 for lower mordent
6 for extended lower mordent
7 for upper mordent
8 for extended upper mordent
9 for acciaccatura
10 for appoggiatura.

Braille item

A braille item is of fixed length one byte and has the form:

either (braille cell):

Byte 0	Bit 0	0
	Bit 1	1 if the cell is conditional (that is, to be omitted on formatting unless it occurs in the first measure on a braille line; always 0 for formatted braille section).
	Bit 2	1 if braille dot 6 is present; 0 otherwise.
	Bit 3	1 if braille dot 5 is present; 0 otherwise.
	Bit 4	1 if braille dot 4 is present; 0 otherwise.
	Bit 5	1 if braille dot 3 is present; 0 otherwise.
	Bit 6	1 if braille dot 2 is present; 0 otherwise.
	Bit 7	1 if braille dot 1 is present; 0 otherwise.

or (format-control code; unformatted braille section only):

Byte 0	Bit 0	1
	Bit 1	0 for inkprint page number;

1 otherwise.

If bit 1 = 0 then:

Bits 2-7 inkprint page number (in range 1 to 63).

If bit 1 = 1 then:

Bits 2-7
1-9 inkprint line number
60 to add hyphen to word at end of line
61 potential bar split
62 force new braille line
63 ignore.

A5.2. Archive tape format

A music archive tape contains an index file followed by a number of music data files. The final music data file is terminated with a double tape mark. The allocated space in the index permits a maximum of 120 music data files on one tape. This includes files which are marked in the index as deleted but have not yet been removed from the tape by copying to a new tape.

The index file comprises a single 2160-byte record in the format:

Bytes	0 - 7	Tape name: up to 8 EBCDIC characters, filled with trailing blanks.
Bytes	8 - 11	Tape type identifier: EBCDIC "MUZK".
Bytes	12 - 15	Total number of data files on tape.
Bytes	16 - 19	Number of files marked as deleted.
Bytes	20 - 39	Unused.
Bytes	40 - 519	File identifiers. (4-byte table indexed by position of file on tape)
Bytes	520 - 759	Byte address within tape index of start of file title. The first byte of the index is byte 1. A zero entry indicates that the file is marked to be deleted. (double-byte table indexed by file position on tape)
Bytes	760 - 839	Version number of file. (byte table indexed by position of file on tape)
Bytes	880 - 2159	File names. Each name consists of a single byte containing the number of characters in the name, followed by the name itself in EBCDIC code.

Each music data file comprises a number of 360-byte sectors which are of the same form as those in the disc file. The first sector is the header sector. This is followed by the printed music, unformatted braille and formatted braille sections respectively, consecutive sectors of each section being stored in the correct order within the tape file.

Appendix 6. Syntax of music input language

The syntax of the input language for music notation is described formally in this appendix using an extension of the BNF metalanguage whose symbols and their meanings are:

```
<   >   angle brackets enclosing symbols used in the definition
::=     "is defined as"
|       "or"
```

The extensions to the BNF metalanguage used for convenience are parentheses enclosing symbols which may or may not be present, and double angle brackets enclosing symbols which may be repeated an arbitrary number of times. These extensions are defined in terms of the standard notation by:

```
(<X>)  ::=  <X> |
<<X>>  ::=  <X> <<X>> | <X>
```

Any restriction on the number of repetitions allowed is indicated by a comment in parentheses following the definition.

The syntax definition given below does not define all aspects of the input language. It does not include, for example, the use of the interrupt and question mark for backspacing, editing during input, or time count checks. Lower case letters of the input language are considered as being syntactically equivalent to the corresponding upper case letters.

```
<printed-music>  ::=  <initial-specs> (<<print-measure>>)
        <end-of-input>

<initial-specs>  ::=  <<part-code>> <identifier> <name> <title>
        <subtitle> <composer> <publisher> <number> <tempo>
        <metronome> <comments> <key-signature> <time-signature>
        <<clef>>
        (the number of <part-code>s and the number of <clef>s must
        each equal the number of lines in the section)
<part-code>  ::=  <part-abbrev> <eod>
<part-abbrev>  ::=  ACC | AFL | ALTO | ANV | BASS | BCL | BD | BELL |
        BN | BTRB | BTRP | BTUB | CAST | CEL | CL | COR | CYM | DB |
        DBN | DUL | EH | EUPH | FL | FLH | GL | GONG | GUIT | HARP |
        HORN | KD | LH | MAR | MG | OBA | OBOE | PED | PIC | RAT | RH
        | SAX | SD | SOP | TAB | TAMB | TD | TEN | TRB | TRI | TRP |
        TUBA | VA | VL | VN1 | VN2 | VOC | WIRE | XY
<eod>  ::=  {CARRIAGE-RETURN}
<identifier>  ::=  <non-empty-text>
<name>  ::=  <non-empty-text>
<title>  ::=  <text>
<subtitle>  ::=  <text>
<composer>  ::=  <text>
<publisher>  ::=  <text>
<number>  ::=  <text>
<tempo>  ::=  <text>
<metronome>  ::=  <text>
<comments>  ::=  <text>
```

```
<printed-measure>  ::=  <<printed-item>> <end-of-measure>
<printed-item>  ::=  <control-item> | <separate-sign-item> |
        <text-item> | <note-item>
<end-of-measure>  ::=  <bar> <eod>
<bar>  ::=  \ | {VERTICAL-BAR}
<end-of-input>  ::=  <end-char> <eod>
<end-char>  ::=  !


<control-item>  ::=  <in-accord> | <set-return> | <barline> |
        <footnote> | <control-symbol> | <ignore-checks> |
        <crosswire-switch>
<in-accord>  ::=  <in-accord-char> <eod>
<in-accord-char>  ::=  ;
<set-return>  ::=  <control-escape> <eod>
<control-escape>  ::=  @
<barline>  ::=  <bar-dots> <bar> <bar> <bar-dots> | <bar-dots> <bar>
        <bar> | <bar> <bar> <bar-dots> | <bar> <bar> <thin-char> |
        <bar> <bar> | <bar> <dotted>
<bar-dots>  ::=  :
<thin-char> ::=  <time-char>
<dotted>  ::=  <dot>
<footnote>  ::=  <bar> <footnote-char>
<footnote-char>  ::=  F
<control-symbol>  ::=  <control-escape> <ctrl-symbol-type>
<ctrl-symbol-type>  ::=  0 | P | S | U | X | Y | 1 | 2 | + | - | =
<ignore-checks>  ::=  M
<crosswire-switch>  ::=  X


<separate-sign-item>  ::=  <clef> | <time-signature> |
        <key-signature> | <note-grouping> | <line-number> |
        <page-number> | <note-beaming> | <scale-factor> |
        <single-char-sign>

<clef>  ::=  <clef-char> <clef-type> <eod>
<clef-char>  ::=  Z
<clef-type>  ::=  B | T | G | F | 1 | 3 | 4 | 8

<time-signature>  ::=  <time-char> <time> <eod>
<time-char>  ::=  T
<time>  ::=  <c-char> <bar> | <c-char> | <unmeasured-char> |
        <numerator> <eod> <denominator>
<c-char>  ::=  C
<unmeasured-char>  ::=  U
<numerator>  ::=  1 | 2 | 3 | 4 | 5 | ... | 32
<denominator>  ::=  1 | 2 | 4 | 8 | 16 | 32

<key-signature>  ::=  <key-char> <key-number> <accidental> |
        <key-char> <eod>
<key-char>  ::=  K
<key-number>  ::=  1 | 2 | 3 | 4 | 5 | 6 | 7
<accidental>  ::=  <flat> | <natural> | <sharp>
<flat>  ::=  $
<natural>  ::=  %
<sharp>  ::=  #

<note-grouping>  ::=  <group-char> <group-number> <eod>
<group-char>  ::=  N
```

```
<group-number>  ::=  2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11

<page-number>  ::=  <page-char> <number> <eod>
<page-char>  ::=  Q
<number>  ::=  1 | 2 | 3 | 4 | 5 | ... | 255
<line-number>  ::=  <line-char> <number> <eod>
<line-char>  ::=  L

<note-beaming>  ::=  <join-char> <join-number> <eod>
<join-char>  ::=  J
<join-number>  ::=  2 | 3 | 4 | 5 | ... | 16
<scale-factor>  ::=  <scale-char> <number> <eod>
<scale-char>  ::= H

<single-char-sign>  ::=  P | * | ( | ) | [ | ] | _ | < | > | {SPACE}


<text-item>  ::=  <text-type> <text>
<non-empty-text>  ::=  <character> <text>
<text-type>  ::=  V | W | ^
<text>  ::=  (<<character>>) <eod>
        (maximum 31 <character>s in text item)
<character>  ::=  <text-escape> <special-char> | <vowel> (<accent>) |
        <consonant> | <other-character>
<text-escape>  ::=  <control-escape>
<special-char>  ::=  A | C | E | F | I | L | N | O | P | R | S | 1 |
        2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | . | $ | % | # | / | @ | ?
<vowel>  ::=  A | E | I | O | U
<accent>  ::=  <text-escape> <accent-sign>
<accent-sign>  ::=  ' | ` | ^ | _
<consonant>  ::=  B | C | D | F | G | H | J | K | L | M | N | P | Q |
        R | S | T | V | W | X | Y | Z
<other-character>  ::=  ! | " | # | $ | % | & | ' | ( | ) | - | = | ~
        | ^ | \ | | | ` | { | } | [ | ] | + | * | _ | ; | : | < | > |
        , | . | / | {SPACE}


<note-item>  ::=  (<<note-specification>>) <eod>
<note-specification>  ::=  <non-resettable-spec> | <resettable-spec>
        | <note-repeat-spec>

<non-resettable-spec>  ::=  <length> | <rest> | <pitch> | <octave> |
        <increment-pitch> | <decrement-pitch> | <increment-octave> |
        <decrement-octave>
<length>  ::=  1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<rest>  ::=  R
<pitch>  ::=  C | D | E | F | G | A | B
<octave>  ::=  <octave-char> <octave-number>
<octave-char>  ::=  O
<octave-number>  ::=  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8
<increment-pitch>  ::=  +
<decrement-pitch>  ::=  -
<increment-octave>  ::=  Y
<decrement-octave>  ::=  U

<resettable-spec>  ::=  <set-spec> (<delete-char>)
<set-spec>  ::=  <dots> | <stem-sign> | <tie> | <chord-sign> |
        <accidentals> | <fractioning> | <tremolo> | <fingering> |
        <extra-spec>
```

```
<delete-char>  ::=  ?
<dots>  ::=  <dot> <dot> | <dot>
<dot>  ::=  .
<stem-sign>  ::=  S
<tie>  ::=  I
<chord-sign>  ::=  &
<accidentals>  ::=  <natural> <sharp> <sharp> | <natural> <flat>
        <flat> | <natural> <sharp> | <natural> <flat> | <sharp>
        <sharp> | <flat> <flat> | <accidental>
<fractioning>  ::=  <note-escape> <fraction-char> <fraction-number>
<note-escape>  ::=  =
<fraction-char>  ::=  /
<fraction-number>  ::=  0 | 1 | 2 | 3 | 4 | 5
<tremolo>  ::=  <note-escape> <tremolo-char> <tremolo-number>
<tremolo-char>  ::=  J
<tremolo-number>  ::=  1 | 2 | 3 | 4 | 5
<fingering>  ::=  <note-escape> <fingering-char> <fingers>
<fingering-char>  ::=  F
<fingers>  ::=  <finger> <finger-and> <finger> | <finger> <finger-or>
        <finger> | <finger>
<finger>  ::=  1 | 2 | 3 | 4 | 5
<finger-and>  ::=  ,
<finger-or>  ::=  :
<extra-spec>  ::=  <note-escape> <extra-type>
<extra-type>  ::=  <turn> | <other-ornament> | <other-extra>
<turn>  ::=  (<accidental>) (<accidental>) <turn-type>
<turn-type>  ::=  ~ | I | N | Z
<other-ornament>  ::=  (<accidental>) <ornament-type>
<ornament-type>  ::=  E | M | T | U | X
<other-extra>  ::=  A | D | P | S | 1 | 2 | . | : | - | > | ^ | _

<note-repeat>  ::=  <repeat-char> <repeat-count-char> <repeat-count>
        (<repeat-char>) | <repeat-char>
<repeat-char>  ::=  "
<repeat-count-char>  ::=  *
<repeat-count>  ::=  1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9



<braille-music>  ::=  (<<braille-measure>>) <end-of-input>
<braille-measure>  ::=  (<<braille-item>>) <bar>
<braille-item>  ::=  (<<braille-dot>>) <eod> | <letter>
<braille-dot>  ::=  <dot-number> (<delete-char>)
<dot-number>  ::=  1 | 2 | 3 | 4 | 5 | 6 | 7 | 8
<letter>  ::=  <vowel> | <consonant>
```

Appendix 7. Samples of print and braille music

The figures in this appendix illustrate the input and various forms of output for part of one of the pieces of music used in testing the system, the primo part of the second movement of Mozart's Sonata in D (K.381).

1
SONATA IN D
Second Movement

MOZART (K. 381)

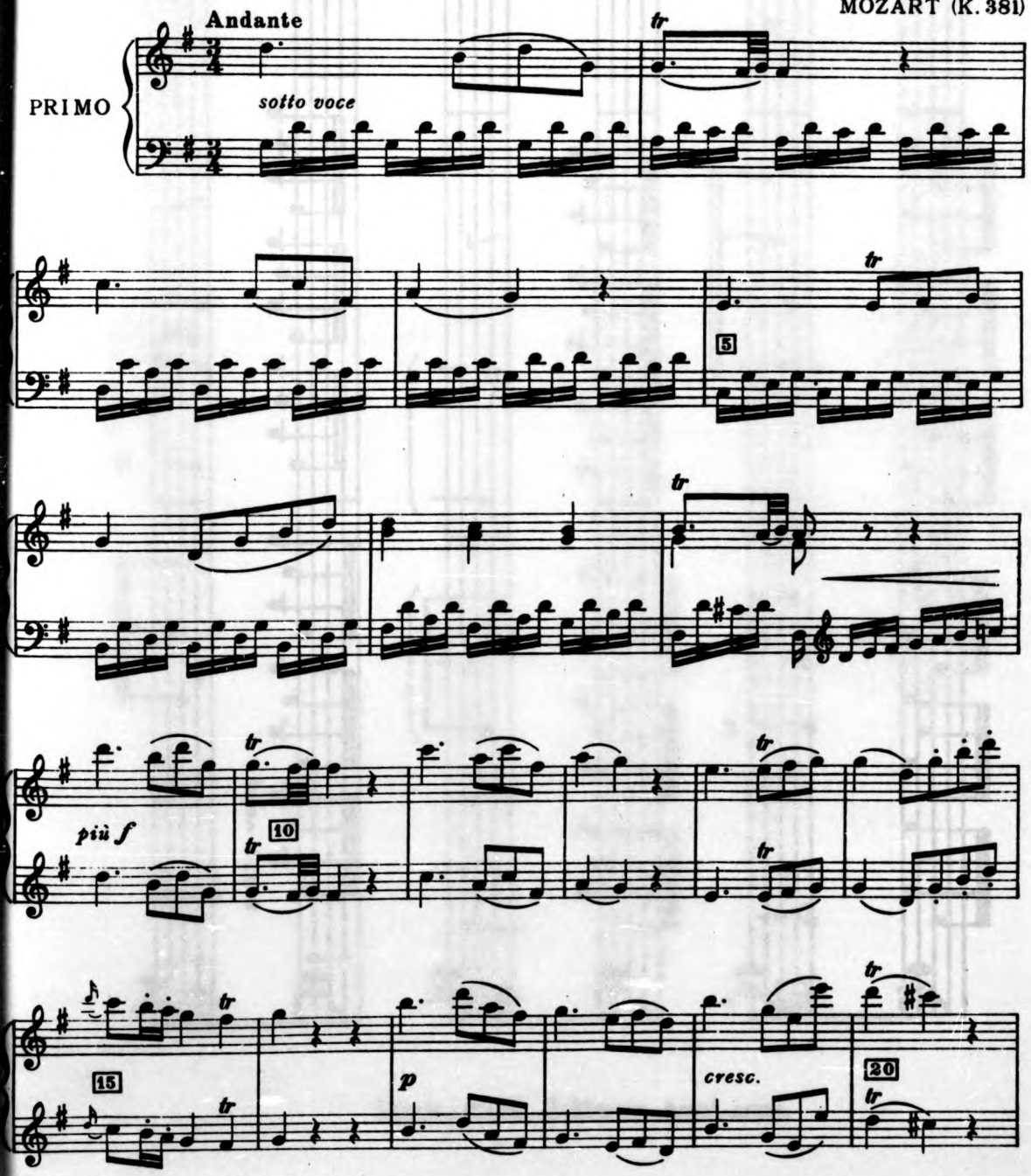


Figure A7:1 Original print
A.B.1199

1
SONATA IN D
Second Movement

MOZART (K. 381)

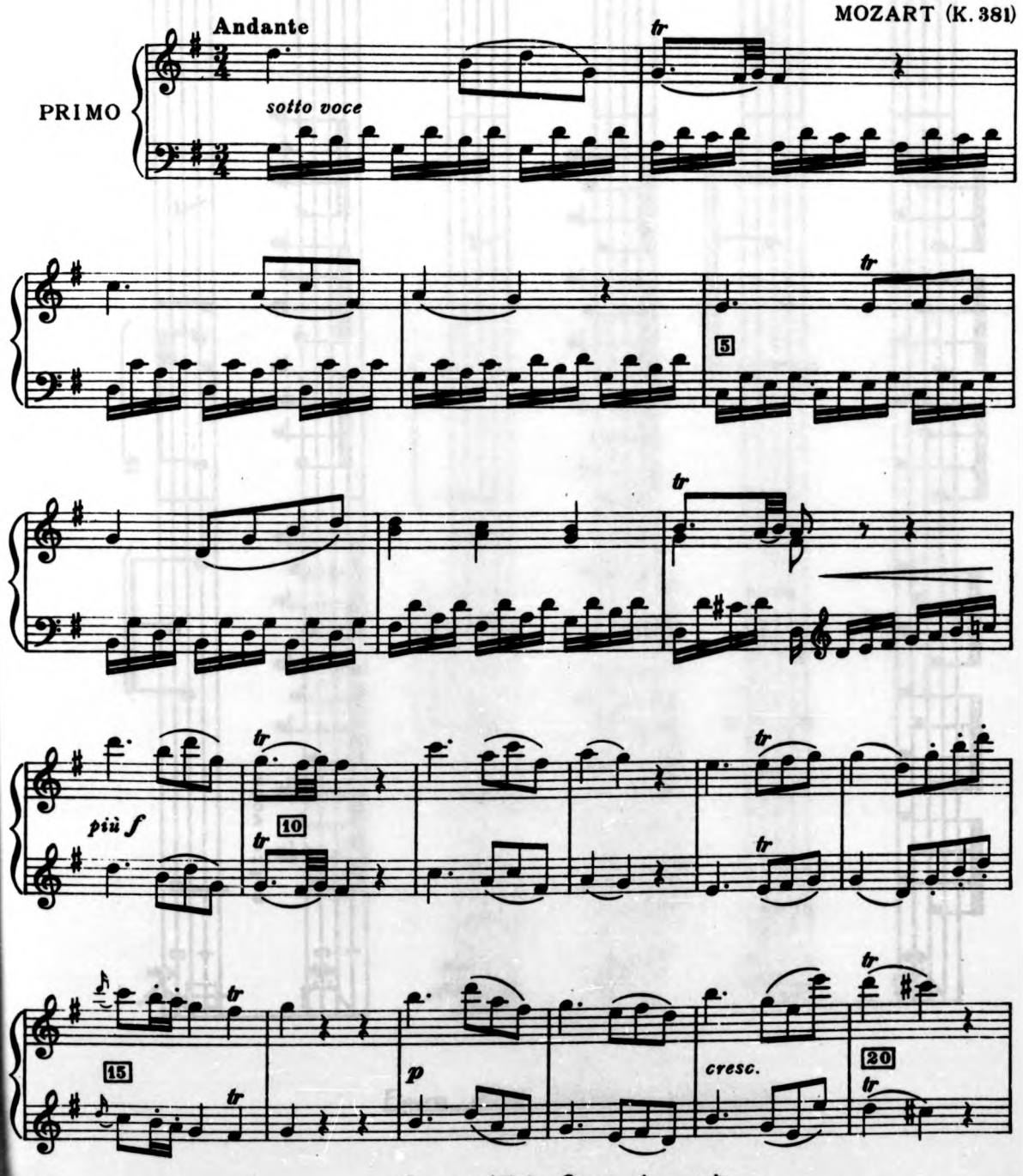


Figure A7:1 Original print
A.B.1199

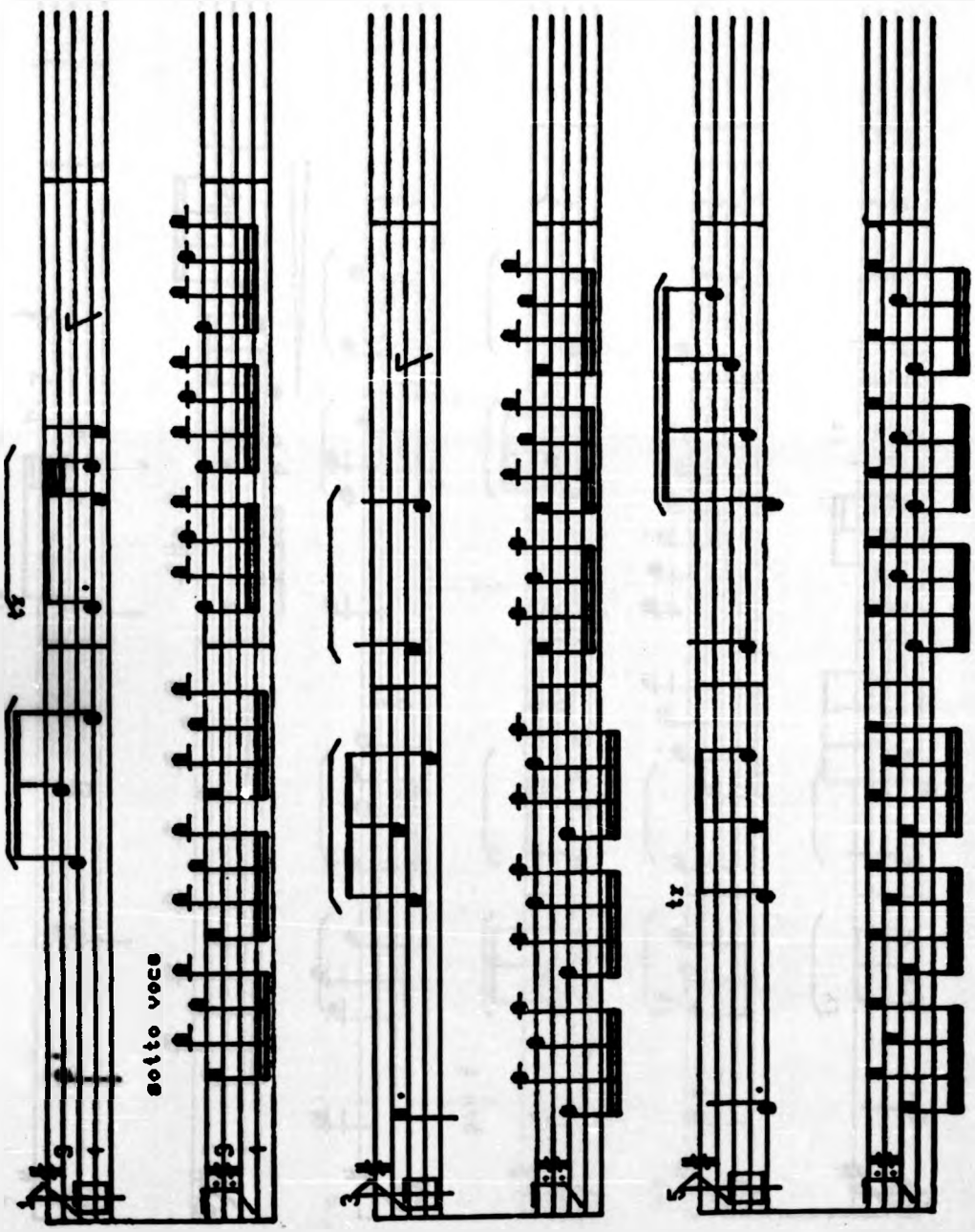


Figure A7:2 Taktronix hard copy.

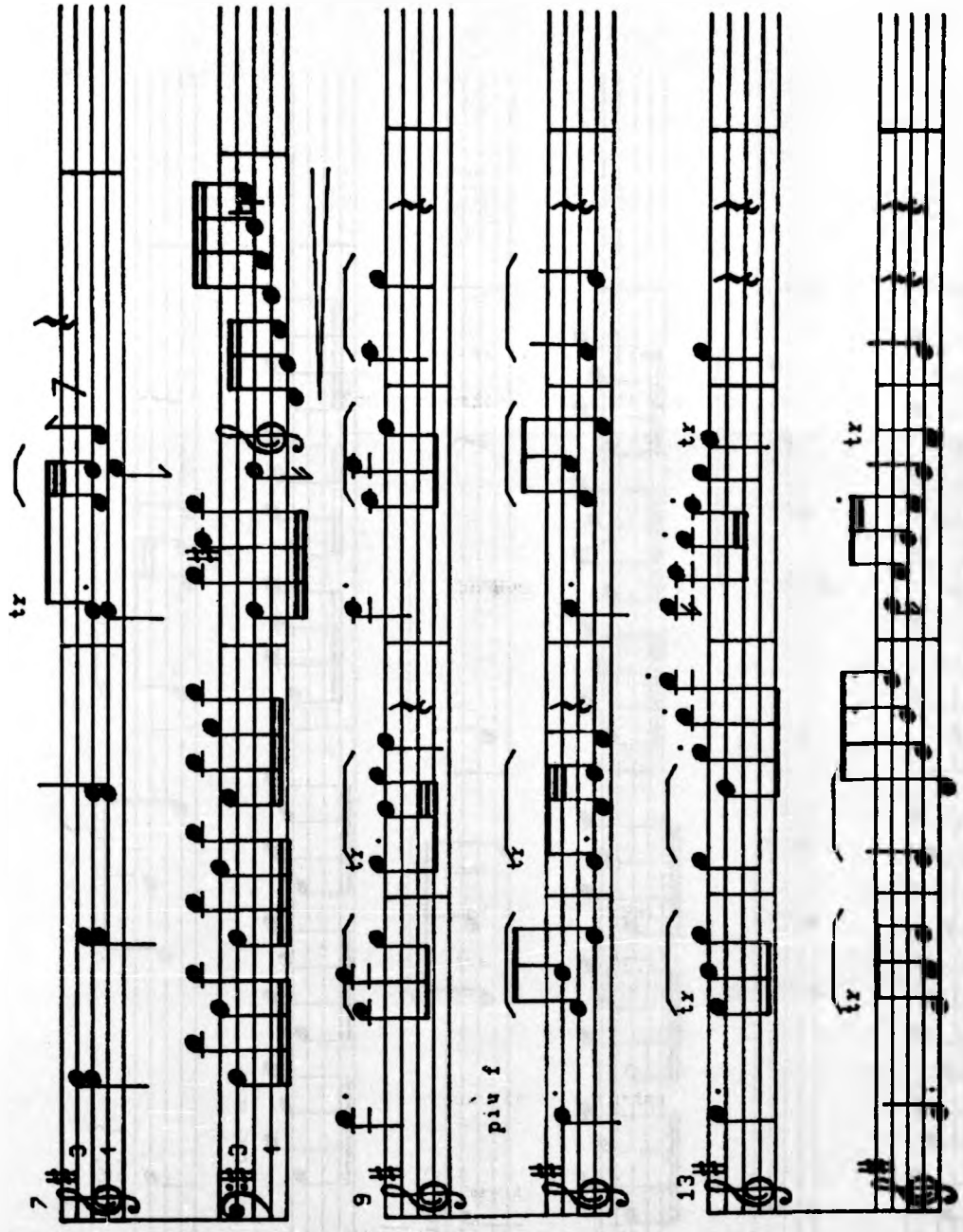


Figure A7:3 Tektronix hard copy with Hershey symbols

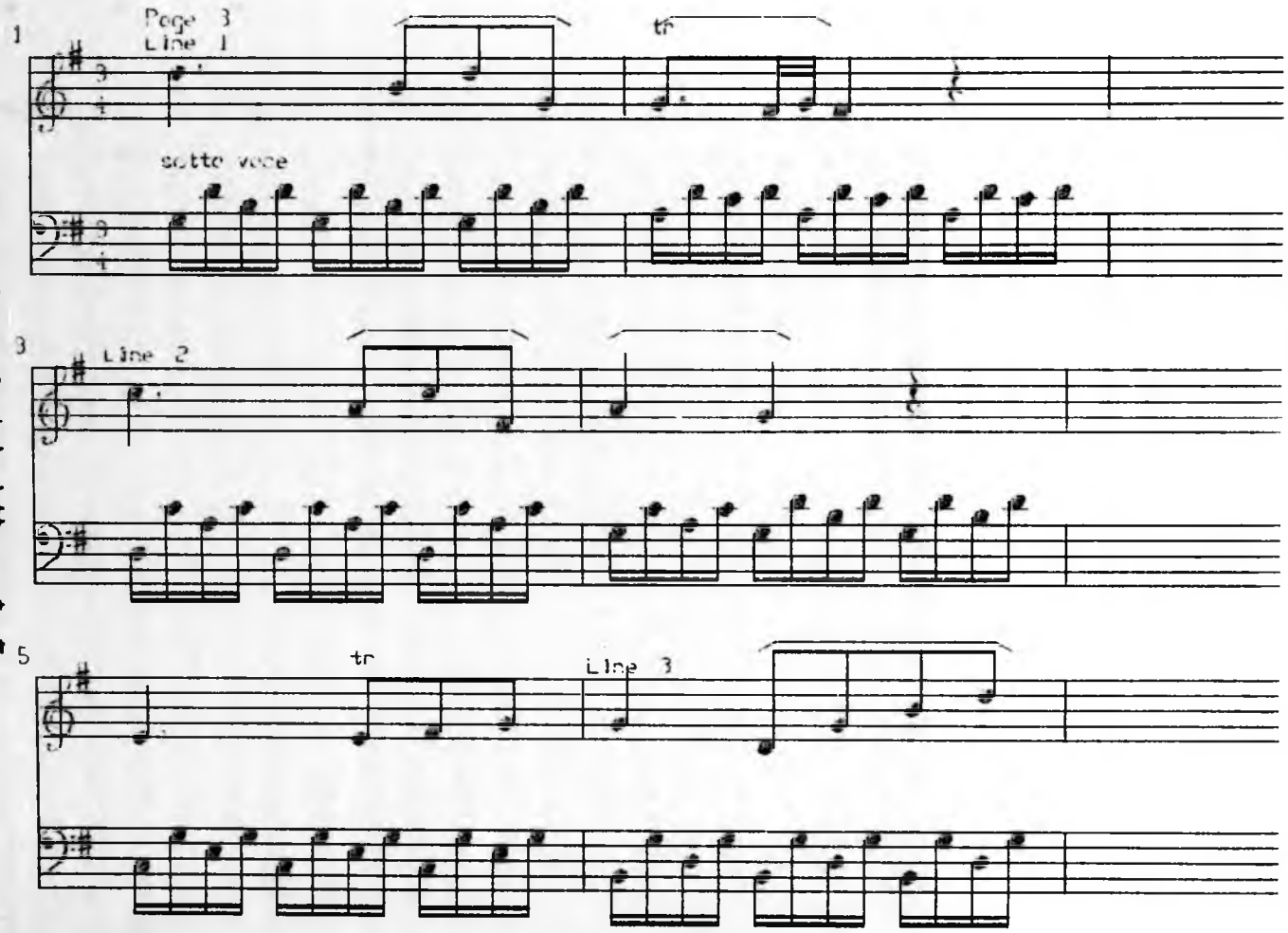


Figure A7.4 Digital plotter output

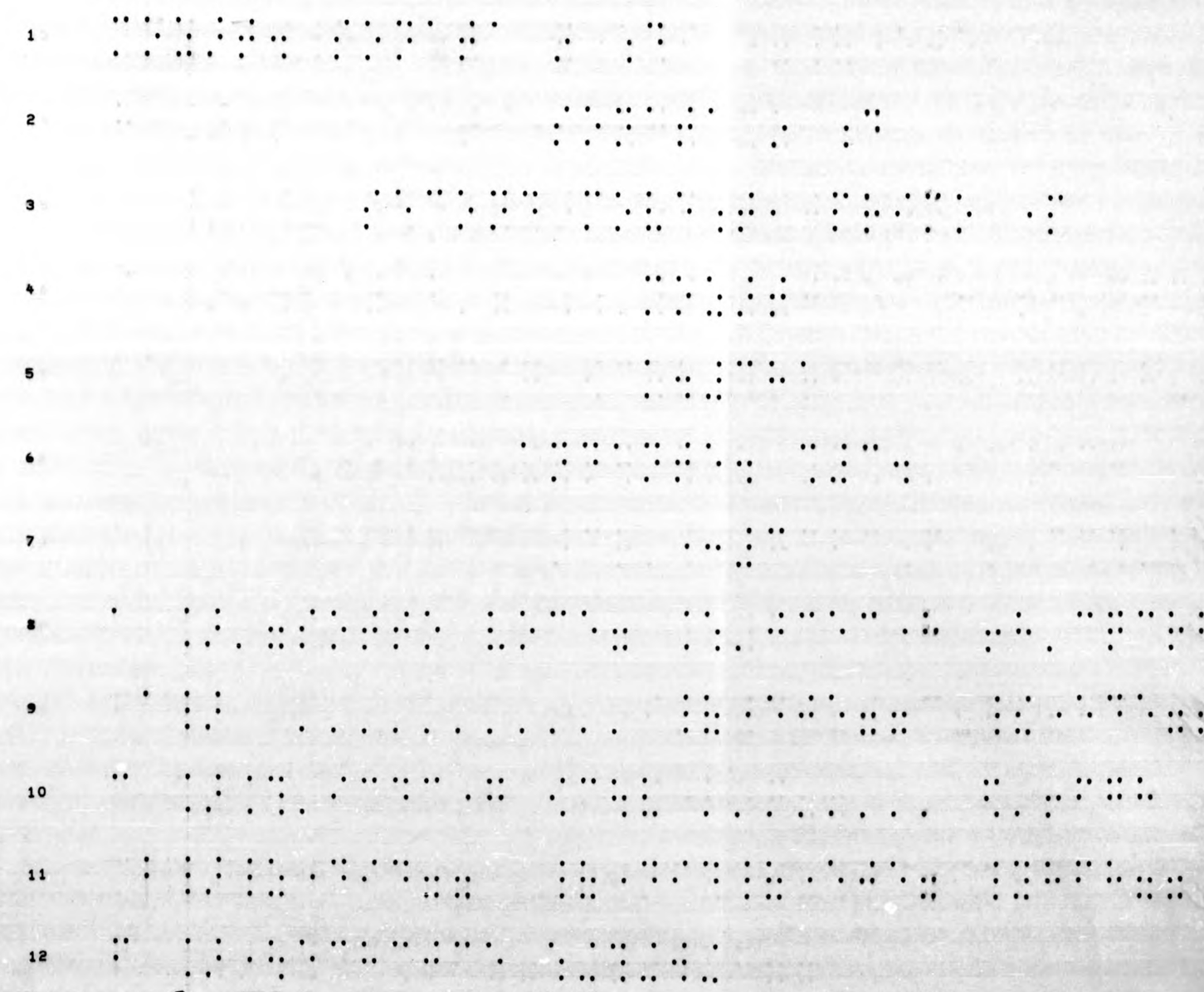


Figure A7:5 Lineprinter proof-listing of braille

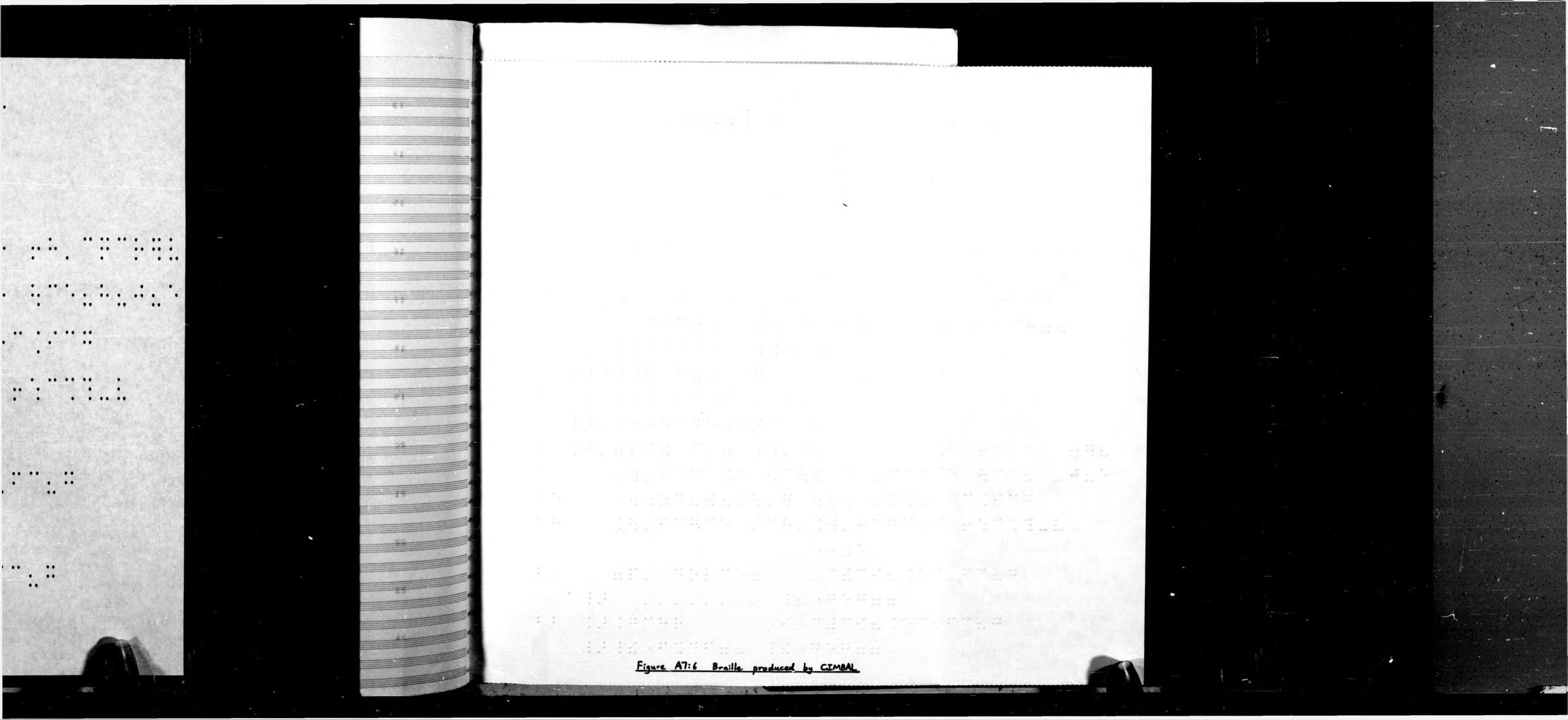


Figure A7:6 Braille produced by CIMBAL