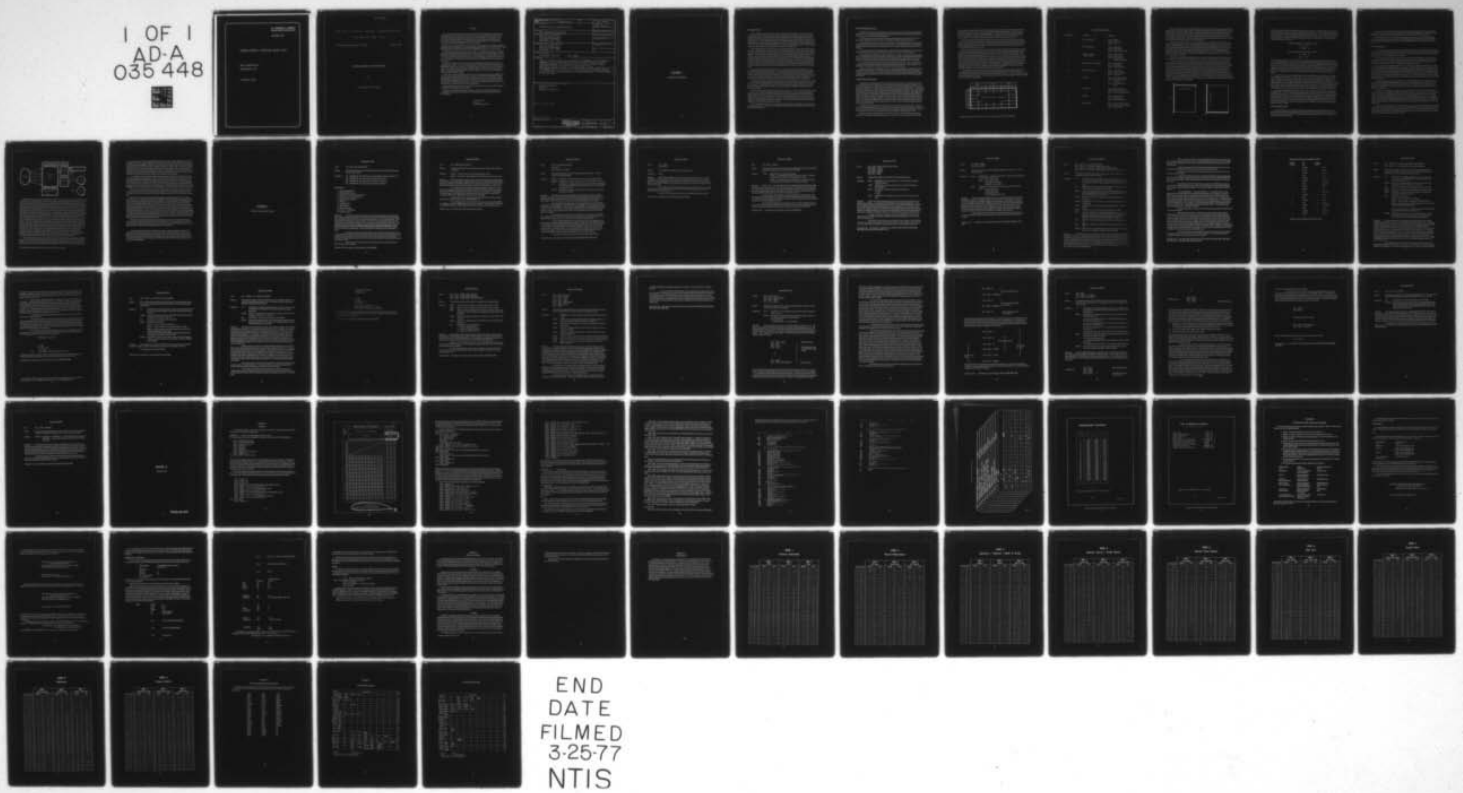
AD-A035 448 NAVAL OBSERVATORY WASHINGTON D C F/G 14/5
FORTRAN AUTOMATIC TYPESETTING SYSTEM (FATS),(U)
AUG 74 P M JANICZEK, G H KAPLAN
UNCLASSIFIED NOBS-CIRC-149 DOD/DF-77/001A NL



1 OF 1
AD-A
035 448

END
DATE
FILMED
3-25-77
NTIS

U.S. DEPARTMENT OF COMMERCE
National Technical Information Service

AD-A035 448

FORTRAN AUTOMATIC TYPESETTING SYSTEM (FATS)

NAVAL OBSERVATORY
WASHINGTON, D.C.

17 AUGUST 1974

AD A 035 448

U N I T E D S T A T E S N A V A L O B S E R V A T O R Y

C I R C U L A R N O. 1 4 9

U. S. Naval Observatory, Washington, D. C. 20390 August 17, 1974

FORTRAN AUTOMATIC TYPESETTING SYSTEM

by

P. M. Janiczek and G. H. Kaplan

Preface

During the year 1940 the Nautical Almanac Office of the U. S. Naval Observatory became the first government organization to introduce punched-card electric accounting machine equipment for scientific computation and the production of printer's copy for its publications. Printer's copy for the Air Almanac 1941 was produced by this equipment and, commencing with the Air Almanac 1942, camera copy for offset printing was automatically prepared on a 401 tabulator ("Air Almanacs" by W. J. Eckert; SKY AND TELESCOPE vol. IV, No. 1, Nov. 1944).

Beginning with the Air Almanac 1946, camera copy of improved quality was produced by a specially designed card-operated typewriter ("The Printing of Mathematical Tables" by W. J. Eckert and R. F. Haupt; MATHEMATICAL TABLES AND OTHER AIDS TO COMPUTATION, vol. II, No. 17, Jan. 1947).

In 1962 the Nautical Almanac Office was the first agency in the Department of Defense to bridge the gap between electronic data processing and automatic photo-composition for offset printing ("Printing of Astronomical and Sight Reduction Tables" by R. L. Duncombe and R. F. Haupt; Navigation, vol. 12, No. 2, Summer 1965). Since that time the Nautical Almanac Office has composed on the Linofilm and/or Linotron equipment of the Government Printing Office over 15,000 pages of astronomical and navigational tables.

The power and flexibility of the Linotron Photocomposing Machine was made more accessible by the appearance in 1968 of "A Linotron System Manual for the Photocomposition of Astronomical and Mathematical Tables" by D. K. Scott (U. S. Naval Observatory *Circular* No. 121). The subroutines presented in that manual allow the preparation of magnetic tapes for Linotron that operate the machine directly. The principal disadvantage of the subroutines, however, is that they require a knowledge of IBM System 360 Assembler Language.

This instruction manual immediately makes the Linotron photocomposition system available to the multitude of programmers who understand and use Fortran and other high level languages. The carefully prepared subroutines presented here mark a milestone in the continuing program of the Nautical Almanac Office to enhance, and improve upon, the methods for efficient and accurate preparation of astronomical and navigational tables. However, the routines were designed with a flexibility that is more than adequate for an extensive variety of tabular presentations.

Inquiries regarding this manual and the availability of the programming subroutines described should be directed to the undersigned or to the authors.

Throughout the course of this development program, the Nautical Almanac Office has received encouragement from the late Public Printer, Mr. A. N. Spense, and from the Deputy Public Printer Mr. John Boyle.

R. L. Duncombe
Director
Nautical Almanac Office

BIBLIOGRAPHIC DATA SHEET	1. Report No. DOD/DF-77/001a	2.	3. F AD A 035 448 o.
4. Title and Subtitle FORTRAN Automatic Typesetting System			5. Report Date Aug. 17, 1974 6.
7. Author(s) P.M. Janiczek and G.H. Kaplan			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Nautical Almanac Office U.S. Naval Observatory Washington, D.C. 20390			10. Project/Task/Work Unit No. 11. Contract/Grant No.
12. Sponsoring Organization Name and Address Nautical Almanac Office U.S. Naval Observatory Washington, D.C. 20390			13. Type of Report & Period Covered 14.

15. Supplementary Notes

For Magnetic Tape, see AD A 035449

16. Abstracts

FATS is a system of subroutines, accessable from FORTRAN and other high-level languages, which generate the control tapes for GPO's Linotron electronic typesetting machine. The routines have been designed for mainly tabular composition, primarily scientific and mathematical tables.
FATS greatly expands the number of Linotron users to include programmers familiar with FORTRAN, PL1, etc., but it may be easily accessed by assembly language. It also minimizes the requirement for an intimate knowledge of Linotron function and control codes.

17. Key Words and Document Analysis. 17a. Descriptors

Tabular Composition
Linotron
Electronic Typesetting

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement		19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 71
		20. Security Class (This Page UNCLASSIFIED	22. Price 4.50

REPRODUCED BY
NATIONAL TECHNICAL INFORMATION SERVICE
U.S. DEPARTMENT OF COMMERCE
SPRINGFIELD, VA. 22161

SECTION I

GENERAL DISCUSSION

INTRODUCTION

Today, a substantial amount of printed matter represents information which, at some time, existed in computer coded form. Since the infancy of computer technology, there has been a recognition of the need to expeditiously transform information from machine recognizable form to printed pages. Some material has been, and is still, reproduced directly from computer printout, but printing from such copy has also introduced problems. It is easily realized that this form of publishing decreases computer efficiency, adds to press and binding costs, sacrifices quality and readability, and results in the handling and storage of greater bulk at every level. The Government Printing Office Electronic Composing System offers a solution to these problems by providing an inexpensive and direct link between computer and printed page.

At the heart of the system is the electronic photo-typesetting machine, Linotron, which can set type at speeds up to 1000 characters per second. Input to the Linotron consists of a magnetic tape containing control characters and formatted data. Output may be in the form of proof pages, opaque positives, or photographic negatives. This device may be considered analogous to a special purpose computer. In that context, it is recognized that realistic and efficient utilization requires a means for translating user-intelligible programming language and data into Linotron control and data code characters. Part of the Electronic Composing System at the Government Printing Office, therefore, is a general purpose computer. Its task is to execute the Master Typography Program, MTP, which performs the functions of a compositor, and constructs the command sequences for input to the Linotron. Because of its flexibility and text orientation, the MTP is a complex series of routines requiring a parametrization of format specifications and the editing of input data prior to its execution.

Many government publications are devoted to the presentation of information in tabular form. U. S. Naval Observatory publications are examples. For this reason, it has become expedient to develop computer subroutines oriented to tabular formats as a means of directly producing Linotron control tapes, including one pass data editing and formatting at the user facility. This direct conversion of user data files into Linotron format, with the automatic insertion of composition control codes, bypasses the MTP. This approach minimizes computer use as well as transportation costs, and reduces the time required for task completion.

One system of computer routines has been described in *U. S. Naval Observatory Circular* No. 121. These are coded in IBM System 360 Assembly Language, and they are optimum in terms of execution time and storage requirements. The Nautical Almanac, and Air Almanac are photo-composed with the aid of this system. The use of these routines, however, requires a familiarity with assembly language, its program layout, and hexadecimal notation.

The Fortran Automatic Typesetting System, FATS, described in this *Circular,* greatly expands the number of Linotron users to include programmers familiar with Fortran, PL1, etc., but it may be easily accessed by assembly language. It also minimizes the requirement for an intimate knowledge of Linotron function and control codes. Therefore, in the description of FATS which follows, specific attention is given only to those details of actual Linotron operation which are essential. Additional details are presented for information purposes only. The presentation is oriented to user viewpoint and logic; detailed optical and electro-mechanical operations of Linotron are beyond the present scope.

FATS is application oriented, but is not a special purpose programming language, and it requires no new vocabulary and syntax. It is a system of interdependent subroutines which are accessible by conventional and easily understood methods.

LINOTRON DESCRIPTION

The decision to employ Linotron composition, and the question of its effective utilization depend upon knowledge of Linotron features and capabilities. The purpose of this section is to provide that information briefly; avoiding, where possible, technical vocabulary.

The maximum area on a page (the frame) which can be composed by Linotron is 8 X 10½ inches. A page may be composed with an 8 inch width or, by changing orientation, with a 10½ inch width.

In the composition process, each character is reproduced from images located on grids. Each grid contains 255 characters, and four grids may be in use at one time. A total of 1020 characters are thus available. Numerous type styles are available, and hence grids are varied by the user according to utility and taste, not necessity.

In printing, sizes are most conveniently given in printers measure, the fundamental unit being the point. With the Linotron, using any grid, the available type sizes are 5, 6, 7, 8, 10, 12, 14 and 18 point.

Other options add to this flexibility. Upon command, the Linotron may produce a pseudo-condensed type. The possibilities are 100%, 83% and 63% of normal width. In addition to existing bold and italic typefaces, any character may be darkened or tilted to produce synthetic bold, synthetic italic, or both. Underscoring or horizontal and vertical rule drawing may be accomplished in .003, .005, and .015 inch widths.

All of the listed features are available at all times during actual composition. Further, the location, on the page, of printed characters and rules is randomly selected through a system of frame coordinates. Thus, the composition sequence will take place according to user logic and convenience.

Obviously every feature and operational mode has a corresponding sequence of commands which must be specified on the input tape. The methods of character, grid, and special effect selection are discussed in connection with FATS routines which construct and transmit the command character sequences. The following section presents the operation of Linotron in logical terms and detail; it is prerequisite to the successful use of FATS.

LINOTRON OPERATIONS

As described in the Introduction, the Linotron machine can be thought of as a special-purpose computer. Input consists of a 7-track magnetic tape containing all the coded commands and text necessary to compose the desired pages. The Linotron input tape is described more fully in the following section. Output from the Linotron consists of the "printed" pages, which actually are photographic reproductions of a high-resolution cathode ray tube screen, upon which the text has been displayed, character by character. These output pages can be returned to the user as either transparent negatives or opaque positives (prints) of normal page size, up to 8 by 10½ inches. On the positives, characters appear uniformly black on a white background. Broken, imperfect, or misaligned characters rarely appear on Linotron output. Either the positives or negatives can be used for reproduction of the pages, depending on the reproduction process and the requirements of the publication. In the following paragraphs, the term "printing" will refer to the process performed by the Linotron machine in obtaining character images, displaying them on the CRT screen, and recording them photographically on the output medium.

The shape of each character printed by the Linotron is determined by a "character image"; the particular character image used depends only on the style of type being set and the particular character to be printed. Thus, all lower-case a's printed in "Spartan Book" type are produced by the same character image. The size of the printed character and its location and orientation on the page are determined electronically.

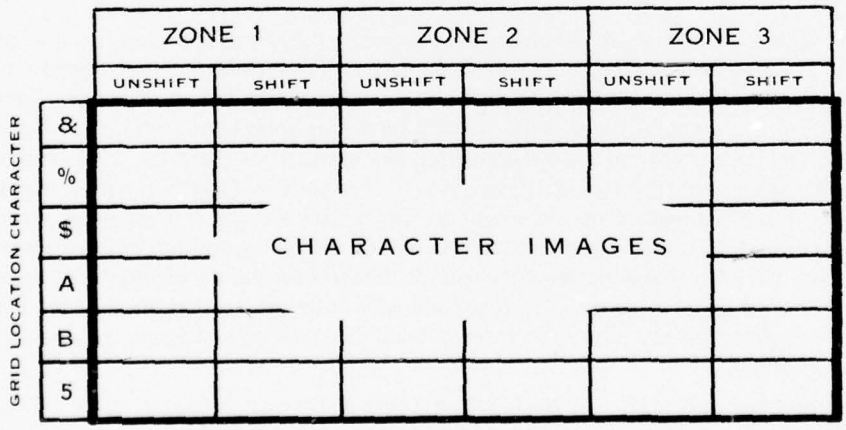
Character images are physically located on arrays called "grids". Each grid contains all the necessary images for three complete styles of type, located in three "zones" on the grid. Each zone is in turn divided

into two sections, one containing "unshift" character images (such as those for lower-case letters and numeric digits), the other containing "shift" character images (such as those for capital letters). The above describes the *logical* arrangement of a grid; the *physical* arrangement of the character images is different, but unimportant. An entire grid can contain up to 255 images. There are a number of character grids available for the Linotron machine. A list of these grids and the type styles contained on each is found on page 7 .

Any Linotron job can access character images from up to four grids. There are four turrets, numbered 1 thru 4, on the Linotron machine which hold the grids needed for a particular job. Any grid may be mounted on any of the four turrets, and any turret may be left empty. Written instructions must be provided with each Linotron input tape indicating to the Linotron machine operators the name of each grid to be used, and the turret number on which it is to be mounted.

Before a line of characters can be printed, the Linotron requires a minimum of five specifications, which it obtains from codes on the input tape. First, the number of the turret containing the appropriate grid must be indicated. Second, the number of the zone where the proper type style is located must be specified. Third, the shift mode must be given, designating whether "unshift" or "shift" characters are to be printed. Fourth, the type size must be specified. Characters can be printed in 5, 6, 7, 8, 10, 12, 14, or 18 point type sizes. (One "point" equals about 1/72 inch, but the type size includes some white space above and below most characters. A convenient rule of thumb is that the height of capital letters in hundredths of an inch equals the type size in points, e.g. capital letters set in 12 point type size are 0.12 inches tall). The last specification required is the page location where the line of print is to begin. Page coordinates will be described more fully in a subsequent paragraph. Once any of the first four specifications are designated, the designation remains in effect until specifically changed by instruction codes further along on the input tape.

Using these parameters, the Linotron machine can electromechanically prepare itself for printing. The characters actually printed depend of course on the character images selected from the designated grid, zone, etc. The selection is governed by what may be called "grid location characters" (or "Linotron characters") which are also contained on the input tape. To aid in visualizing this process, a schematic diagram of the logical arrangement of a single grid is given below:*



*Similar diagrams for each of the actual Linotron character grids are located in Appendix D.

Linotron Character Grids

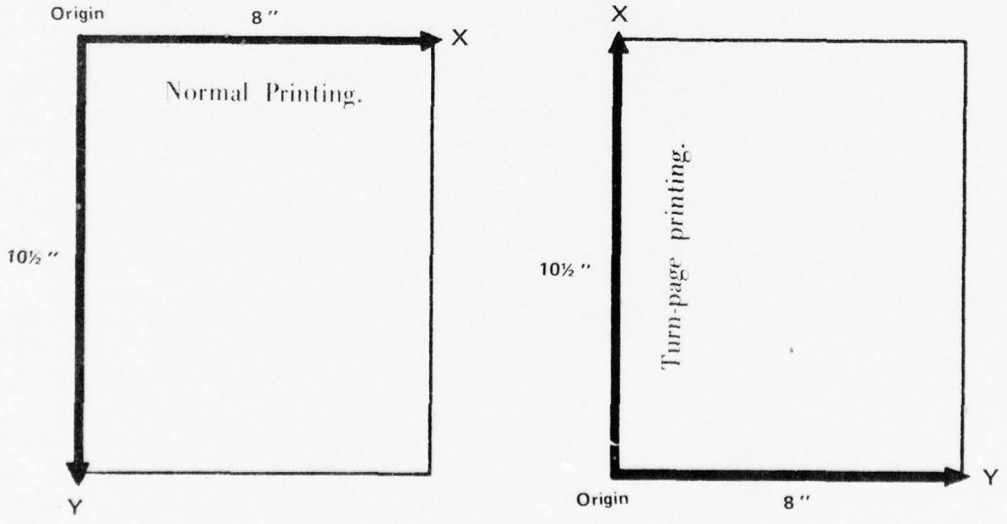
FATS Number	Grid Name	Type Styles
1	Century Expanded	Zone 1: Roman Zone 2: Roman Bold Zone 3: Italic
2	Naval Observatory	Zone 1: Bell Gothic Zone 2: Spartan Book Zone 3: astronomical symbols
3	Superiors / Inferiors / Math & Greek	Zone 1: superior figures Zone 2: inferior figures Zone 3: Greek letters and mathematical symbols
4	Spartan Heavy / Trade Gothic	Zone 1: Spartan Heavy Zone 2: Trade Gothic Zone 3: Trade Gothic Bold
5	Special Times Roman	Zone 1: Times Roman Zone 2: Times Roman Bold Zone 3: Spartan Heavy
6	NRL Grid	Zone 1: Mathematical symbols Zone 2: Times Roman Italic with Greek Zone 3: Times Roman with Greek
7	Crystal Data	Zone 1: Bodoni Book Roman Zone 2: Bodoni Bold Roman Zone 3: inferior figures and Greek
8	Helvetica	Zone 1: Helvetica Roman Zone 2: Helvetica Bold Zone 3: Helvetica Italic Bold
9	Census Gothic	Zone 1: Spartan Book Condensed Zone 2: Spartan Heavy Condensed Zone 3: Spartan Heavy

If the turret containing this grid had been selected, the zone and shift mode specifications would delimit a single column of character images on the grid. A grid location character will then determine which character image in the column is to be used for printing. After printing, the page location coordinates are advanced by the width of the character just printed and the process is then repeated for the next grid location character on the input tape. At any time the Linotron may encounter codes on the tape for changing shift mode, zone, turret, typesize, or page location. Once the indicated operations are performed the Linotron is ready to resume printing, using the grid location character(s) it next encounters.

The Linotron may also encounter codes on the tape that instruct it to electronically thicken, tilt, or squeeze subsequent characters as they are printed, producing pseudo-bold, pseudo-italic, or condensed-width type. Squeezing characters to produce condensed-width type is called changing their *aspect ratio,* which is the ratio of the printed width divided by the normal width, expressed as a percentage. Normal type therefore has an aspect ratio of 100%, but 83% and 63% widths are also available. Tilting or thickening characters (which does not alter their height or width) is called selecting a *special face.* Any of these electronic alterations can be applied to any type style in use in any combination, and can be switched "on" or "off" at any time by codes on the input tape.

The Linotron is capable of drawing horizontal or vertical rules in three thicknesses. Instruction codes on the tape determine the thickness, starting page location, direction, and length of each rule to be drawn.

The Linotron locates points on a page (frame) by means of a left-handed cartesian coordinate system with a positive X axis and a positive Y axis (there are no negative coordinate values). Coordinates are measured in Linotron units; there are 1300 Linotron units per inch, or 18 Linotron units per point. One Linotron unit is the smallest resolvable distance of the Linotron system so that fractions of a Linotron unit are never given. A line of print is always set parallel to the X axis, i.e. the Y coordinate value is fixed while the X coordinate value increases as the printing progresses. There are two possible orientations of the coordinate system relative to what can be considered a fixed page. In the "normal" mode, the origin (Y=0, X=0) is at the upper left corner of the page, the X axis extends to the right 8 inches, the Y axis extends downward 10½ inches, and printing progresses horizontally to the right. In the "turn-page" mode, the origin is at the lower left corner of the page, the X axis extends upward 10½ inches, the Y axis extends to the right 8 inches, and printing progresses vertically upward, as shown below.



The orientation used is determined by codes on the Linotron input tape. For a page set wholly in either mode, the only difference to the user is the different lengths of the Y and X axes. The normal orientation thus permits a "tall" page to be set and the turn-page orientation permits a "wide" page to be set. Otherwise the modes are entirely equivalent. It is possible to change orientations while setting a single page, but in such a case it is important to note that a fixed point on the page has different coordinates in the two modes. The transformation between coordinates is given below:

Turn-page coordinates, given normal coordinates:

$$Y_T = X_N - 165$$
$$X_T = 13950 - Y_N$$

Normal coordinates, given turn-page coordinates:

$$Y_N = 13950 - X_T$$
$$X_N = Y_T + 165$$

In order to avoid coordinate transformations which yield negative-valued coordinates, avoid locations within ¼ inch of the page boundaries. The convention has been adopted that to specify a page location (either mode) the Y coordinate is given first, followed by an X coordinate. The Linotron obtains page coordinates from input tape codes. Other codes are available to increment either coordinate in a positive direction. The Linotron automatically increments the X coordinate after printing each character, so that page coordinates need only be specified for the beginning of a line of characters, not for each character individually. Since the Linotron can locate any page coordinates at random, there is no advantage or requirement in setting a page from top to bottom.

In order that the Linotron can properly advance the X coordinate after printing a character, the width of the character image, measured in "relative units", is coded onto the grid underneath the image itself. To obtain the width of the printed character in Linotron units, the width in "relative units" is multiplied by the type size in points. Thus, a character image 9 relative units wide (a typical value) will produce a printed character 108 Linotron units (0.0831 inches) wide if printed using a 12 point type size. Grid diagrams, showing each character image and its width in relative units, are located in Appendix D. The Y coordinate and X coordinate values of a printed character refer to the lower left corner of the character. (The lower edge of the character is defined to be the "z-line" which is even with the lowermost extension of characters such as g or p.)

After a page (frame) has been completed, codes on the input tape instruct the Linotron to advance the photographic film. If the last page for the job has been printed, a stop code which halts the Linotron should follow the film advance code. If no stop code is present, the Linotron will expect data for a new page to begin at the next block on the input tape. All blocks which start a page must begin with a 6-character job number and a 4-character frame number. The job number is an identification number assigned to Linotron user projects by GPO, and the frame number is simply a page count. When preparing a Linotron tape for GPO, the job number and beginning and ending frame numbers should be marked on the tape reel, along with the total number of blocks on the tape.

THE LINOTRON TAPE

The Linotron tape is a 7-track, 556 CPI, odd parity, unlabeled BCD tape with 2048-character blocks. Each block is packed with contiguous BCD characters without any format or field separators of any kind. The Linotron obtains all its commands from this tape. Generally, each page to be printed will require between 5 and 15 blocks on the Linotron tape.

It is the job of the FATS subroutines to generate the proper command characters and to write them on the tape. For most users, therefore, the Linotron tape can be considered a "black box"; no detailed knowledge of the command codes used on the tape is necessary for the efficient use of the FATS subroutines.

In the remainder of this *Circular,* the characters generated and written onto the Linotron tape by the FATS subroutines will be referred to as *tape codes.* FATS subroutines generate these tape codes in EBCDIC form, and EBCDIC to BCD translation is performed by the 360 Operating System as the tape is actually written.

THE FATS SYSTEM

The Fortran Automatic Typesetting System (FATS) has been designed for users who have little knowledge of printers' terminology, procedures, or measures, and who wish to print mostly tabular data. FATS subroutines could be adapted to print large amounts of textual material, but the MTP is much more suitable for this type of work, especially if text is to be justified at both margins.

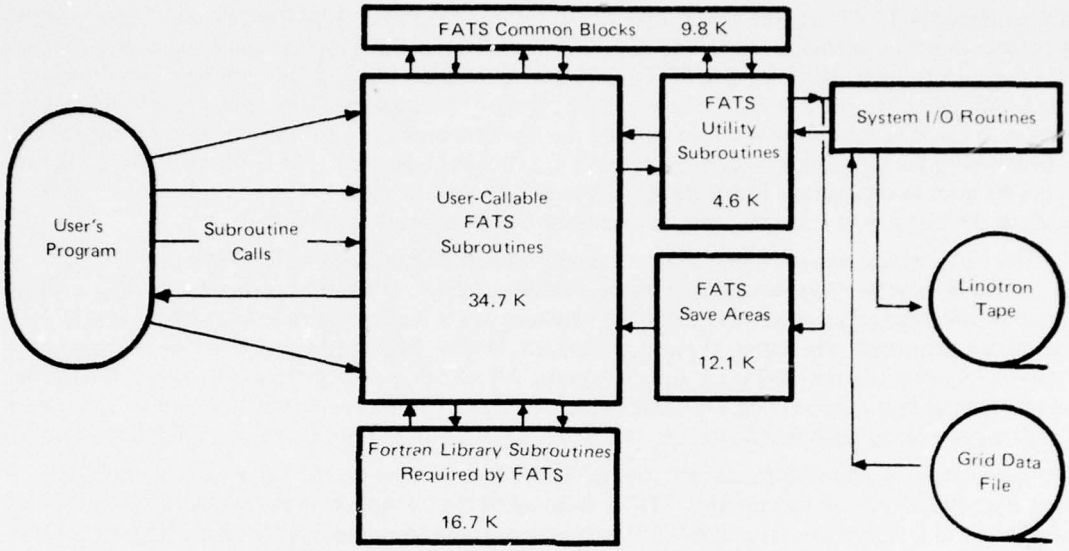
From a user's viewpoint, FATS is simply a group of subroutines which generate the necessary codes for the Linotron and write them onto tape. In the next section of this *Circular,* each subroutine call, its purpose, the arguments required, and a complete discussion of its use and results is given. These subroutines are to be called from a user-written Fortran (or PL1 or Assembler Language) main program.

However, the FATS subroutines described in the next section form only part of the FATS system. Also part of FATS, although not accessable by the user, is a group of utility subroutines and common blocks. Some of the utility subroutines were originally coded in Assembler Language, and these perform vital functions, such as binary-to-decimal conversion, which cannot be efficiently performed in Fortran. The common blocks permit intercommunication and data sharing between FATS subroutines and also provide storage for some of the large arrays required by FATS. The user should not attempt to enter or interfere with any of the utility subroutines or common blocks. (The names of all FATS utility subroutines and common blocks begin with a $, e.g. $READ, $OUTPT, $SAVE, etc.)

The FATS system was originally generated at the U. S. Naval Observatory's G. W. Hill Computing Center, using an IBM System/360 Model 40 with OS release 18.6 and Fortran G and Assembler F softwear. There should be no compatability problems with larger System/360 or 370 models or later OS releases, but recompilation and reassembly of FATS source modules might be advisable. Approximately 128 K bytes is the minimum core size requirement.

Two data sets, one input and one output, must be made available to the FATS subroutines at execution time. The input data set, DDNAME = GRIDFILE, contains data for each of the character grids, and is read only once. The output data set, DDNAME = LINOTAPE is the Linotron tape, and is written more or less continuously. The FATS subroutines completely control I/O operations on these data sets - the user's only responsibility is to supply standard DD statements for them at the execution step of his program. The DD statements should contain only UNIT, VOLUME, DSNAME, DISP, and LABEL parameters, as appropriate; no DCB information should be specified. GRIDFILE can be either a tape or disk data set, but LINOTAPE must be a 7-track tape data set.

The structure of the FATS system as it would reside in core during a user program execution is represented schematically in the diagram below. The arrows indicate the possible flow of data. As indicated, the user's program communicates only with the central part of the FATS system via the subroutine calls. The system as diagrammed requires 78 K bytes of core. Additional storage must be reserved for the resident System, I/O buffers, and the user's program and any subroutines it requires.



When the user issues a call to a FATS subroutine, he has in mind a specific function he wishes the Linotron machine to perform. Of course there is no direct link between the user's program and the Linotron; the user's program calls the appropriate FATS routine, the routine generates the proper tape codes, the tape codes are written onto the Linotron tape, and eventually the tape will be run on the Linotron at GPO. Only then will the function originally desired by the user actually be performed by the Linotron. Since there is no direct connection between the user's program and the Linotron machine, the FATS system in certain respects acts as a surrogate for the Linotron at program execution time. For instance, each time the user's program calls subroutine entry GRID, which generates the tape codes for a turret (grid) change, a common block parameter which keeps track of the "current turret" is changed accordingly. This parameter change within FATS is the analog of the turret change on the Linotron. There are a total of 8 such parameters within FATS, which simulate certain electro-mechanical modes of the Linotron machine. The Linotron machine modes thus simulated are : turret, grid,* and zone selection, shift mode, type size, coordinate system orientation, aspect ratio, and special face selection. This set of modes determines the appearance and orientation of the characters being printed by the Linotron at any time. As tape codes are produced by the FATS subroutines, the values of the 8 FATS parameters will identically represent the corresponding Linotron machine modes when the same tape codes are processed by the Linotron machine. This set of 8 parameters, or their equivalent Linotron machine modes, will be termed the "Linotron configuration". A small group of FATS subroutines is used for establishing or changing the Linotron configuration. Most FATS subroutines, however, either leave the configuration completely unaltered, or make temporary alterations in one or two elements of the configuration but reset them before the return to the user's program. How each FATS subroutine affects, or is affected by, the Linotron configuration is described in the individual subroutine descriptions of this *Circular*.

The first two elements of the Linotron configuration, turret and grid selection, require special attention by the user. As previously described, specific grids are mounted on specific turrets by the Linotron machine operators before a job is run, according to written instructions accompanying the Linotron tape. The FATS system must also be informed as to which grids will be assigned to which turrets, and this is accomplished by

*Note that within a single job, grid selection is determined by turret selection.

a call to subroutine TURT near the logical beginning of the user's program. TURT establishes a turret-grid correspondance in a common block available to all FATS subroutines, and also reads in appropriate information from the grid data file (DDNAME = GRIDFILE). Thereafter, to select a specific grid, the user calls the appropriate FATS subroutine entry (either TYPE or GRID) providing as an argument the *turret number* of the grid he wishes to use. The subroutine will then generate the tape codes necessary to affect the turret change and will update both the turret and grid parameters in the Linotron configuration. The Linotron machine's turret changes are mechanical and slow (requiring about ½ second of time) in addition to being not entirely reliable. Therefore, the user should minimize the number of such changes as much as possible.

The FATS system allows the user to establish his own page measurement units, so that he can specify locations and lengths on the printed page in inches, millimeters, picas, or whatever system he chooses. FATS subroutine DIMN should be called near the logical beginning of the user's program to establish the system of measurement to be used. Thereafter, any page coordinates, lengths, field widths, etc., required as arguments for any FATS subroutine are specified in the user's units. All valid page coordinates must be positive and are measured relative to the current origin of coordinates (see section on page coordinates in *Linotron Operations*). If DIMN is never called, page coordinates, etc. must be given in Linotron Units (1 inch = 1300 Linotron Units).

Except for four job initialization calls (to TURT, DIMN, MARG, and STAB) all FATS subroutine calls must occur between the call to subroutine STRT, which opens the Linotron tape and establishes an initial configuration, and the call to subroutine TERM, which empties the tape output buffers and closes the Linotron tape.

Most FATS subroutines which require page coordinates as arguments allow the use of the "negative coordinate option." A subroutine with this option available will accept either positive or negative coordinate values. Positive-valued coordinates are treated in the usual way, i.e. they are assumed to be valid page coordinates, and the tape codes generated which will change the Linotron's page location before any operations are performed. Negative-valued coordinates, on the other hand, are, of course, *not* valid as page coordinates, and they prevent the subroutine from generating the tape codes to change page location; any operations will therefore be performed at the Linotron's *current page location*, wherever it happens to be. The magnitude of such negative coordinates is irrelevant - their negativity is simply a "code" that prevents a page location change. The negative coordinate option, where available, applies to the Y coordinates and X coordinates independently, so that one coordinate can be left unchanged while the other is varied. It should be mentioned that the current page location would normally be at a point immediately following the last character printed, unless new page coordinates had been specifically established since then.

The FATS system generates no diagnostics, since a diagnostic facility would have required extensive testing of argument values at the beginning of each FATS subroutine - a time-consuming process. The user is urged to familiarize himself with the proper use of each FATS subroutine and any applicable restrictions on its argument values. [Program testing and debugging is described in Appendix C.]

In the next section of this *Circular*, the FATS subroutines are individually described. There are 41 possible calls, but undoubtedly the user will require only a subset of these for each job, depending on the page layout, the form of the input data, and convenience. Many subroutines have functions which overlap those of other subroutines, and the user is thus presented with a choice of possible calls, or sequences of calls, to accomplish a single function. It is hoped that this redundancy allows FATS to meet the needs of a wide range of users and applications.

SECTION II

SUBROUTINE DESCRIPTIONS

Subroutine TURT

Entry: CALL TURT (IG1, IG2, IG3, IG4)

Purpose: This subroutine, called once near the beginning of a job, informs FATS of which grids will go on which turrets.

Arguments:
- IG1 (INTEGER*4) The FATS code number (see below) for the grid on turret No. 1.
- IG2 (INTEGER*4) The code number for the grid on turret No. 2.
- IG3 (INTEGER*4) The code number for the grid on turret No. 3.
- IG4 (INTEGER*4) The code number for the grid on turret No. 4.

Code Numbers:

1 = Century Expanded Grid
2 = Naval Observatory Grid
3 = Superiors / Inferiors / Math & Greek Grid
4 = Spartan Heavy / Trade Gothic Grid
5 = Special Times Roman Grid
6 = NRL Grid
7 = Crystal Data Grid
8 = Helvetica Grid
9 = Census Gothic Grid
10, 11, 12, ... = (Future Use)
0 = No Grid mounted

Discussion: This subroutine must be called once per job, near the beginning of the program, before STRT is called (see subroutine STRT). TURT provides information to the FATS system as to which grids will be mounted on which turrets when the Linotron tape is actually run on the Linotron machine. TURT also reads data for these grids from the grid data file. This is the only time that the grid data file is used; TURT closes this data set before the return to the user's program. In the event that IG1, IG2, IG3, or IG4 designates a grid whose data is not available on the grid data file, the following message will be typed on the console typewriter: END OF FILE ENCOUNTERED WHILE SEARCHING FOR GRID nn DATA where nn is the grid number designated. In this case execution will continue, but with unpredictable results.

Note that this subroutine provides information only to FATS at program execution time; it cannot guarantee that the proper grids will be physically mounted on the proper turrets when the Linotron tape produced by the program is run on the Linotron machine. Instructions to the Linotron machine operators, indicating which grids to mount on which turrets, should be specified in writing with each Linotron tape delivered to GPO.

To help minimize turret changes, have the grid that will be used first mounted on the lowest numbered turret, if possible.

Example of Use: See Appendix A, Sample Program, Statement 0006.

Subroutine DIMN

Entry: CALL DIMN (ZINCHS, ZUNITS)

Purpose: This subroutine allows the user to establish his own units in which page coordinates are specified.

Arguments:

ZINCHS (REAL*4) Any arbitrary length, specified in inches.

ZUNITS (REAL*4) The same arbitrary length, specified in the user's own units.

Discussion: This subroutine may be called once per job, near the beginning of the program, before STRT is called (see subroutine STRT). DIMN establishes "user units" in which all page coordinates and lengths are to specified. Once DIMN is called, all Y or X page coordinates, delta-Y or delta-X lengths, and field widths which are arguments to any FATS subroutine must be expressed in the "user units".

For example, suppose a user wished to express all his page coordinates, etc. in millimeters. An appropriate call would be CALL DIMN (1.0, 25.4). Thereafter, all coordinates or lengths given to any FATS subroutine would have to be in the "user units", i.e. millimeters. For example, an X coordinate value of, say, 176.5 would be understood by any FATS subroutine as meaning 176.5 millimeters = 6.949 inches = 9033 Linotron units to the right of the origin.

In the above example, the same results would have been obtained if the original call had been CALL DIMN (10., 254.) or CALL DIMN (0.03937, 1.0).

If a user wished to use Linotron units as his "user units", he could use CALL DIMN (1.0, 1300.0), since there are 1300 Linotron units per inch. However, in this case the call is unnecessary, since if DIMN is never called, FATS assumes that user units are Linotron units throughout the job.

Example of Use: See Appendix A, Sample Program, Statement 0007.

Subroutine STRT

Entries: CALL STRT (JOBNO, IFRAME)
CALL PAGE
CALL IDEN (Y, X, ITURET)

Purpose: This subroutine takes care of certain bookkeeping functions necessary to properly initialize and identify each page.

Arguments:

JOBNO	(INTEGER*4) The job number for the current job (6 decimal digits).
IFRAME	(INTEGER*4) The frame number for the first frame (page) to be printed.
Y, X	(both REAL*4) The page coordinates, in user units, of the lower left corner of the field in which the job and frame numbers are to be printed. Negative coordinate option valid.
ITURET	(INTEGER*4) The turret number of the grid to be used for printing the job and frame numbers.

Discussion: Each page to be printed must be properly initialized before any subroutines which operate on the page are called. Entries STRT and PAGE perform this page initialization procedure. To initialize the first page to be printed, use entry STRT, providing via the arguments the job number and frame number for the first page. To start the second and all succeeding pages, use entry PAGE.

Entry STRT opens the Linotron tape and types a message on the console typewriter to that effect (a separate message from the 360 System mount message). It also initially sets the Linotron configuration to: page orientation = normal; turret = lowest-numbered turret with grid assigned (normally turret no. 1); zone = 1; shift mode = unshift; typesize = 5 point; aspect ratio = 100%; special type face selection = normal. An internal frame counter within the subroutine is set to the value of the second argument (IFRAME).

Entry PAGE generates tape characters to advance the Linotron film, empties FATS tape output buffers, and initializes a new page. The Linotron configuration is unaltered, except for the page orientation, which is set to normal. The internal frame counter is incremented by 1.

In order to facilitate identification of printed output at GPO, each page should have the job and frame numbers printed somewhere on it, usually in a lower corner. Entry IDEN can be used for this purpose. The job and frame numbers will be printed using 10-point type on zone 2 of the grid on turret ITURET. They will be located in a field approximately 0.1" high and 1.0" wide, with the lower left corner at page coordinates (Y, X). The current value of the internal frame counter is used for the frame number. Turret, zone, shift mode, and typesize may be changed temporarily by this routine but the original configuration will be reset before a return to the user's program.

Do not use entry PAGE to terminate a Linotron job (see subroutine TERM).

Examples of Use: See Appendix A, Sample Program, Statements 0010, 0072, 0073, 0105.

Subroutine TERM

Entries: CALL TERM
CALL PASS

Purpose: This subroutine terminates Linotron processing for a job.

Arguments: (None)

Discussion: Either TERM or PASS should be called once per job, at the logical end of Linotron processing. If the Linotron tape volume in use is to be subsequently used by other jobs for producing additional frames, use entry PASS. If nothing further is to be written on the tape before being sent to GPO, use entry TERM; TERM places "stop code" tape characters on the tape which will physically halt the Linotron machine.

Both entries empty FATS tape output buffers and close the Linotron tape. A message is typed on the console typewriter indicating that the tape has been closed and giving the number of tape blocks written by the current job.

No other FATS subroutines should be called after TERM or PASS has been called.

Example of Use: See Appendix A, Sample Program, Statement 0106.

Subroutine TURN

Entry: CALL TURN (ITURN)

Purpose: This subroutine constructs the sequence of tape codes necessary to orient the coordinate system and printing on a page. Two orientations are possible.

Argument: ITURN (INTEGER*4) The orientation parameter:
ITURN = 1. Coordinate system and printing to be "normal"; this gives an X axis length of 8 inches and a Y axis length of 10½ inches.

ITURN = 2. Coordinate system and printing to be "turn-page"; this gives an X axis length of 10½ inches and a Y axis length of 8 inches.

Discussion: See pp. 8 - 9 in "Linotron Operations" for a complete discussion of page coordinates, coordinate axes, and the "normal" and "turn-page" orientations of the coordinate system. Briefly, since lines of print always run parallel to the X axis, the normal orientation permits a tall page to be composed, while the turn-page orientation permits a wide page to be composed. If the page to be composed does not exceed 8 inches in either height or width, either orientation can be used.

In certain cases it may be necessary to compose a page using both orientations, but in such situations all characters to be printed under one orientation should be composed before the orientation is changed. Before attempting such composition, the user should thoroughly understand the discussion contained in the "Linotron Operations" section.

Calls to STRT and PAGE set the orientation to normal, so that if all pages are to be set wholly in the normal orientation, TURN need never be called. Once turn is called, the coordinate system orientation specified by the call remains in effect until another call to TURN or a call to PAGE.

Examples of Use: See Appendix A, Sample Program, Statements 0098, 0100.

Subroutine TYPE

Entries:
```
CALL TYPE  (ITURT, IZONE, ISHIFT, ISIZE)
CALL GRID  (ITURT)
CALL ZONE  (IZONE)
CALL SHFT  (ISHIFT)
CALL SIZE  (ISIZE)
```

Purpose: This subroutine enables a user to establish or alter typefaces and sizes.

Arguments:

ITURT (INTEGER*4) The turret number (1, 2, 3, 4) containing the grid with the desired typeface.

IZONE (INTEGER*4) The zone number (1, 2, 3) on the grid containing the selected typeface.

ISHIFT (INTEGER*4) The shift mode which is to be effective.
1 = unshift
2 = shift

ISIZE (INTEGER*4) The typesize (5, 6, 7, 8, 10, 12, 14, 18) in points, to be used.

Discussion: A call to the TYPE routine selects the grid, zone, shift mode, and typesize during initialization of composition. It should always be used before any type is actually composed; if it is not, FATS will assign parameters by default (see subroutine STRT). A call to GRID, ZONE, SHFT, or SIZE may be made in order to effect a selective change in one of the specifications. If more than one parameter is to be changed at a given point, TYPE may be used with appropriate parameter values, and zeros supplied for unaffected parameters. Zero should not be supplied as an argument to the single parameter entries, since it produces a no-operation condition. An invalid typesize will yield unpredictable results.

The reader is directed to the description of subroutine COMPOZ for the purpose of making temporary or isolated configuration changes for the sake of initial capitals in text; or the insertion of an unusual symbol.

Some thought, during job planning, should be given to grid changes. Each grid change involves a ½ second delay in composition on the Linotron machine. Greater efficiency will be obtained if all matter requiring a particular grid is composed while that grid is in the immediate access state.

Examples of Use: See Appendix A, Sample Program, Statements 0020, 0023, 0025, 0029, 0033, 0044, 0068, 0074, 0076, 0079, 0080, 0097, 0101.

Subroutine OODD

Entries: CALL SQEZ (NSQZ)
CALL SPEC (NSPC)

Purpose: These routines synthetically alter typefaces by changing the "aspect ratio" or selecting a "special face".

Arguments: NSQZ (INTEGER*4) To indicate the amount by which printed characters are to be condensed (aspect ratio):

NSQZ = 1 gives normal width
2 gives 83% normal width.
3 gives 63% normal width.

NSPC (INTEGER*4) Selects scanning and tilting options (special face):

NSPC = 1 gives normal type.
2 creates pseudo-italic.
3 creates pseudo-bold.
4 creates pseudo-italic and bold.

Discussion: This routine would be entered prior to one or more printing routines. The selected options remain in effect until changed by another use of the routine. Entry SQEZ with arguments 2 or 3 causes characters to be condensed together with the between-character white space. Note that if subroutines FNUM, INUM, ANUM, or COMPOZ are subsequently called, field widths and specified between-word blank spaces are not affected. Condensing of a field must be performed by the user. Between-word spaces to be handled by the COMPOZ subroutine are governed by its space-width parameter which takes precedence over the condensed type option. Shrinking the blank separators must be performed by changing the SPACE parameter in the PRNT calling sequence.

SPEC produces pseudo-bold characters by modifying the electronic image scanning, so that the characters are made darker. Pseudo-italic is accomplished by tilting the characters electronically.

Examples of Use: See Appendix A, Sample Program, Statements 0035, 0037, 0069, 0071, 0102, 0104.

Subroutine COMPOZ

Entries:

```
CALL PRNT  (Y, X, NN, NTEXT, SPACE)
CALL SCAR  (Y, X, ITURET, IZONE, ISHIFT, LGLC)
CALL MLTP  (Y, X, NN, NTEXT, SPACE, YINC, XINC, NTIMES)
CALL MLTC  (Y, X, ITURET, IZONE, ISHIFT, LGLC, YINC, XINC, NTIMES)
```

Purpose: This subroutine generates the tape codes necessary to print a line of alphameric text, including punctuation and special characters.

Arguments:

Y, X	(both REAL*4) Page coordinates, in user units, for the lower left corner of the left-most printed character. Negative coordinate option valid *except* for calls to MLTP or MLTC, in which cases coordinates must be explicitly given.
NN	(INTEGER*4) The number of EBCDIC characters in NTEXT (including "editing characters" and blanks) to be processed.
NTEXT	(Array of A4-formatted words) An array of contiguous EBCDIC characters representing the text to be printed. (The corresponding Hollerith literal can be substituted for an array name).
SPACE	(REAL*4) The width of the space to be inserted between printed words, in user units.
ITURET	(INTEGER*4) The turret number for a grid containing a special character or symbol.
IZONE	(INTEGER*4) The zone number where a special character or symbol is located.
ISHIFT	(INTEGER*4) The shift mode for a special character or symbol.
LGLC	(An A1-formatted word) A word with a single character in its high-order (left) byte. The character is the EBCDIC translation of the grid location character (Linotron character) for a special character or symbol. (A Hollerith literal of one character can be substituted for a variable name).
YINC	(REAL*4) Y coordinate increment, in user units, for multiple-print entries.
XINC	(REAL*4) X coordinate increment, in user units, for multiple-print entries.
NTIMES	(INTEGER*4) For multiple-print entries, the total number of times a character or line of characters is to appear on the printed page.

Discussion: Entry PRNT is used to print a single line of alphameric text, represented by the EBCDIC characters in array NTEXT, beginning at page coordinates (Y, X). The line is printed using the Linotron configuration (turret, grid, zone, typesize, aspect ratio, etc.) in effect at the time of the call. Any changes made to the configuration by this routine are reset before the return to the user's program, so the configuration is unaffected by the call. PRNT should be used only with "standard" zones, i.e. *not* with special-character zones such as zone 3 of grids 2 or 3. Entry MLTP is similar to PRNT but is used to create multiple printings of the same line; the first such printing will begin at (Y, X), the second at (Y + YINC, X+XINC), etc., until the line is printed (NTIMES) times.

NTEXT can have any number of contiguous EBCDIC characters of any of the following types: alphabetic characters A thru Z (hex C1 thru E9), decimal digits 0 thru 9 (hex F0 thru F9), blanks (hex 40), "editing characters" (hex 4F), or any of the punctuation characters listed on the chart on the following page.

Alphabetic characters in NTEXT cause the corresponding letters to be printed, in caps if the shift mode in effect is shift, or lower case if the shift mode in effect is unshift.

Decimal digits in NTEXT cause the corresponding numerals to be printed, *only* if the shift mode in effect is unshift.

Punctuation characters will cause the corresponding characters to be printed, regardless of the shift mode. However, the punctuation character must be one of the ones processed by PRNT (see chart on the following page), and the corresponding punctuation mark must be available on the grid and zone in use.

For each blank in NTEXT, PRNT will insert a space of width SPACE (user units) into the printed line. These blanks can therefore be used for inter-word spacing. Two successive blanks will result in a space of width 2x SPACE to appear in the line, three will result in a space of width 3x SPACE, and so on.

The "editing character" - the logical OR symbol | (12-7-8 punch, hex 4F) - has a special purpose. It informs PRNT that the single character following it (i.e. immediately to its right) in NTEXT is to be processed under the opposite shift mode than that currently in effect for the line as a whole. This is useful when there are a few capital letters (shift) to be printed in a line of predominently lower-case letters (unshift). The line should be printed in the *unshift* mode with the | character preceding each alphabetic character in NTEXT whose corresponding letter is to appear capitalized. It is also useful when a few numerals (unshift) are to appear in a line of capital letters (shift). The line should be printed in the *shift* mode with the | character preceding each decimal digit in NTEXT. The editing character should not precede a blank or punctuation character.

Since generally NTEXT "looks like" the line to be printed, use of PRNT or MLTP should be fairly straightforward.

Occasionally an odd character or symbol must be printed, such as a Greek letter or mathematical symbol. Entry SCAR is used to print such a single special character, at page coordinates (Y, X). The turret, zone, shift mode, and grid location character which the user specifies as arguments to SCAR completely determine a character image on a grid to be used for printing. The Linotron configuration is changed to obtain and print the character, but is reset before the return to the user's program. Note that this can result in two rapid turret changes, which is not recommended. Efficient use of this routine would be obtained only if the special character or symbol is located on the grid in use at the time of the call, so that no turret change would be required. Entry MLTC is similar to SCAR but is used to create multiple printings of the same character; the first such printing will be at (Y, X), the second at (Y+YINC, X+XINC), etc., until the character is printed NTIMES times.

Note that the negative coordinate option can be used for the single-print entries (PRNT and SCAR), but *not* for the multiple-print entries (MLTP and MLTC). The latter entries require that valid page coordinates for the first printing be explicitly specified.

Examples of Use: See Appendix A, Sample Program, Statements 0021, 0022, 0024, 0027, 0036, 0045, 0066, 0070, 0077, 0081, 0082, 0089, 0095, 0099, 0103.

Punctuation Characters Processed by PRNT

EBCDIC character	Card code	Character printed	
.	(12-3-8)	.	period
,	(0-3-8)	,	comma
;	(11-6-8)	;	semi-colon
:	(2-8)	:	colon
(	(12-5-8)	(	open paren.
)	(11-5-8)	)	close paren.
'	(5-8)	'	apostrophe
—	(0-5-8)	-	hyphen
+	(12-6-8)	+	plus
—	(11)	—	minus or dash
*	(11-4-8)	*	asterisk
/	(0-1)	/	slash
$	(11-3-8)	$	dollars
&	(12)	&	ampersand
?	(0-7-8)	?	interrogation
=	(6-8)	=	equals
!	(11-2-8)	!	exclamation
%	(0-4-8)	%	per cent

Characters must be available on grid and zone in use

Subroutine FNUM

Entries: CALL FNUM (Y, X, FLDSIZ, FPN, NODEC, ISIGN, MINDIG)
CALL DNUM (Y, X, FLDSIZ, DFPN, NODEC, ISIGN, MINDIG)

Purpose: This subroutine accepts a single- or double-precision floating point number and constructs the necessary tape codes to print it. The printed number will appear right-justified in a field on the page as specified by the user.

Arguments:

Y, X (both REAL*4) The page coordinates, in user units, of the lower left corner of the field in which the number is to be printed. Negative coordinate option valid.

FLDSIZ (REAL*4) The width (i.e. the size in the X direction) of the field in which the number is to be printed, in user units.

FPN (REAL*4) A single-precision floating-point number to be printed.

DFPN (REAL*8) A double-precision floating-point number to be printed.

NODEC (INTEGER*4) The number of digits to appear to the right of the decimal point in the printed number.

ISIGN (INTEGER*4) Sign option:

ISIGN = 0 Sign of number not to be printed.

ISIGN = 1 Sign of number to be printed left-justified in the field.

ISIGN = 2 Sign of number to be printed immediately to the left of the left-most printed digit ("floating" sign).

If the sign of the number is to be printed, an appropriate sign character must be available somewhere on the grid in use.

MINDIG (INTEGER*4) The *minimum* number of digits to appear to the left of the decimal point in the printed number. If necessary, leading zeros will be added until there are MINDIG digits appearing.

Discussion: This subroutine is used for printing a number right-justified in a field of width FLDSIZE whose lower left corner is at (Y, X) and whose lower right corner is at (Y, X + FLDSIZE). The value of the number to be printed is given to the subroutine in the argument FPN in entry FNUM for REAL*4 (single precision) values, or in the argument DFPN in entry DNUM for REAL*8 (double precision) values. The number is printed using the Linotron configuration (turret, grid, zone, typesize, aspect ratio, etc.) in effect at the time of the call, with two exceptions: (1) if necessary, the subroutine may momentarily switch zones or shift mode in order to obtain and print a decimal point or an appropriate sign character; and (2) the subroutine always sets the shift mode to unshift before printing any digits. In either case the configuration is not permanently changed, and the configuration after a call to FNUM or DNUM is identical to the configuration before the call. FNUM or DNUM should be used only with "standard" zones, i.e. not with special-character zones such as zone 3 of grids 2 or 3.

The number will be printed with NODEC digits to the right of the decimal point, the number having been properly rounded to this precision. To the left of the decimal point, at least MINDIG digits will appear: the subroutine will add leading zeros if necessary to insure that MINDIG

digits are printed. Of course, if the magnitude of the number is large enough, more than MINDIG digits may appear. Regardless of the field size, the maximum number of digits - including leading zeros - that can be printed in the field is sixteen. However, the maximum number of *significant digits* that can be printed in the field is only nine.*

The user can have the algebraic sign of the number printed by specifying ISIGN = 1 or 2. If ISIGN = 1, the sign will be printed at the left extremity of the field. If ISIGN = 2, the sign will be printed immediately to the left of the left-most printed digit, wherever in the field this happens to be. The sign character to be used is determined by the value of FPN or DFPN: negative values will cause a – sign to be printed, and zero or positive values will cause a + sign to be printed. Since many zones do not contain + or – sign characters, this subroutine may switch zones in order to obtain a suitable sign character. However, the subroutine will *not* switch turrets (grids), so the user should make sure that any sign characters required are available somewhere on the grid in use.

In zones without a decimal point (period) character, the subroutine may also switch zones in order to obtain and print it.

The user should make sure that the field size specified in the argument FLDSIZ is wide enough to contain the number as printed, including all digits, the decimal point, and the sign if desired. If the printed number would overflow the designated field by more than half a digit's width, the subroutine will simply not print the number. Of course in many cases the user's designated field will be wider than necessary, and if ISIGN = 1 the field size can be used to determine how much white space will appear between the printed digits and the sign character.

The subroutine is constructed to make it easy to print a column of aligned numbers of varying magnitude. For instance, the call to FNUM or DNUM can be located in a DO-loop where only the Y coordinate and the value of the number to be printed vary:

```
      DIMENSION VALUE (50)
                .
                .
                .
      Y = 800.
      DO 100 I = 1, 50
      Y = Y + 200.
100   CALL FNUM (Y,500., 1000., VALUE (I), 4, 1, 2)
```

This type of coding would produce a column of printed numbers with signs, decimal points, and end-figures vertically aligned, regardless of the individual values of the elements in array VALUE.

Examples of Use: See Appendix A, Sample Program, Statements 0055, 0058, 0063.

*For this purpose, *significant digits* are any digits which remain if the decimal point and all high-order zeros are ignored. Thus, 3.1416000 has eight significant digits and 00.00037401 has five.

Subroutine INUM

Entry: CALL INUM (Y, X, FLDSIZ, INT, ISIGN, MINDIG)

Purpose: This subroutine accepts an integer number and constructs the necessary tape codes to print it. The printed number will appear right-justified in a field on the page as specified by the user.

Arguments:

Y, X (both REAL*4) The page coordinates, in user units, of the lower left corner of the field in which the number is to be printed. Negative coordinate option valid.

FLDSIZ (REAL*4) The width (i.e. the size in the X direction) of the field in which the number is to be printed, in user units.

INT (INTEGER*4) An integer number to be printed.

ISIGN (INTEGER*4) Sign option:

ISIGN = 0 Sign of number not to be printed.

ISIGN = 1 Sign of number to be printed left-justified in the field.

ISIGN = 2 Sign of number to be printed immediately to the left of the left-most printed digit ("floating" sign).
If the sign of the number is to be printed, an appropriate sign character must be available somewhere on the grid in use.

MINDIG (INTEGER*4) The *minimum* number of digits to appear in the printed number. If necessary, leading zeros will be added until there are MINDIG digits appearing.

Discussion: This subroutine is almost identical to FNUM in function, logic, and use, except that in the case of INUM, the number to be printed is an integer, and no decimal point is printed.

See the Discussion for subroutine FNUM.

Example of Use: See appendix A, Sample Program, Statement 0052.

Subroutine ANUM

Entry: CALL ANUM (Y, X, FLDSIZ, NN, NCHAR)

Purpose: This subroutine accepts a number, represented by an array of EBCDIC characters, and constructs the necessary tape codes to print it. The printed number will appear right-justified in a field as specified by the user.

Arguments:

- Y, X (both REAL*4) The page coordinates, in user units, of the lower left corner of the field in which the number is to be printed. Negative coordinate option valid.
- FLDSIZ (REAL*4) The width (i.e. the size in the X direction) of the field in which the number is to be printed, in user units.
- NN (INTEGER*4) The number of characters in array NCHAR.
- NCHAR (Array of A4-formatted words) An array of contiguous EBCDIC characters representing the number to be printed. (The corresponding Hollerith literal can be substituted for an array name).

Discussion: This subroutine is used for printing a number right-justified in a field of width FLDSIZE whose lower left corner is at (Y, X) and whose lower right corner is at (Y, X + FLDSIZE). The number to be printed is given to the subroutine as an array of characters in array NCHAR. The number is printed using the Linotron configuration (turret, grid, zone, typesize, aspect ratio, etc.) in effect at the time of the call, with two exceptions: (1) if necessary, the subroutine may momentarily switch zones or shift mode in order to obtain and print sign characters, decimal points, or commas; and (2) the subroutine always sets the shift mode to unshift before printing any digits. In either case the configuration is not permanently changed, and the configuration after a call to ANUM is identical to the configuration before the call. ANUM should be used only with "standard" zones, i.e. not with special-character zones such as zone 3 of grids 2 or 3.

NCHAR may have any number of contiguous EBCDIC characters of any of the following five types: decimal digits (hex F0 thru F9), periods (hex 4B), commas (hex 6B), plus or minus signs (hex 4E or 60), and blanks (hex 40). Blanks are ignored. The decimal digits, periods, commas, and signs will cause ANUM to select and print the corresponding character images from the zone in use. These characters will be printed contiguously and in sequence, and the group of printed characters thus formed will appear right-justified in the specified field on the page. If necessary, this subroutine may switch zones in order to obtain suitable sign characters. In zones without period or comma characters, the subroutine may also switch zones in order to obtain and print them. In *no* case will the subroutine switch turrets (grids).

The user should make sure that the field size specified in the argument FLDSIZ is wide enough to contain the number as printed. If the printed number would overflow the designated field by more than half a digit's width, the subroutine will simply not print the number.

The subroutine is constructed to make it easy to print a column of numbers with end-figures aligned, and is most suitable for printing numbers read by the user's program from an external input source. For instance, the call to ANUM can be located in a DO-loop where only the Y coordinate varies:

```
      DIMENSION KCHARS (5)
1     FORMAT (5A4)
        .
        .
        .
      Y = 800.
      DO 100 I = 1,50
      Y = Y + 200.
      READ (5,1)  (KCHARS(K),  K = 1,5)
100   CALL ANUM (Y, 500., 2000., 20, KCHARS)
```

This type of coding would produce a column of printed numbers with end-figures vertically aligned, even though the format of the characters read from the input records (hence the format of the numbers printed by ANUM) may vary.

Example of Use: See Appendix A, Sample Program, Statement 0096.

Subroutine RULE

Entries:

```
CALL  RULX  (YORG, XORG, TRM, IWT)
CALL  RULY  (YORG, XORG, TRM, IWT)
CALL  RULD  (YORG, XORG, YTRM, XTRM, IWT)
```

Purpose: This subroutine draws horizontal, vertical, and diagonal rules of optional widths.

Arguments:

YORG	(REAL*4) The Y coordinate, in user units, where ruling is to begin.
XORG	(REAL*4) The X coordinate, in user units, where ruling is to begin.
TRM	(RULE*4) If RULX is called, the X coordinate, in user units, where ruling terminates. If RULY is called, the Y coordinate, in user units, where ruling terminates.
YTRM	(REAL*4) In diagonal ruling, the Y coordinate, in user units, where ruling terminates.
XTRM	(REAL*4) In diagonal ruling, the X coordinate, in user units, where ruling terminates.
IWT	(INTEGER*4) The rule weight code: 1 = a light rule, 0.003 inch width. 2 = a medium rule, 0.007 inch width. 3 = a heavy rule, 0.015 inch width.

Discussion: Horizontal and vertical ruling is a hardware capability of Linotron. Diagonal ruling is a pseudo-operation which creates a rule by overlap printing of periods. The RULD entry normally uses the period from zone 2 of the Naval Observatory grid. If unavailable, the corresponding character from zone 2 of the grid currently in use will be used, but this may produce undesirable results.

Because of Linotron restrictions, the RULE routines may slightly alter the origin and termination coordinates. However, the resulting deviations of position and length will not exceed 0.0046 inch, which should be imperceptible.

The user may be aware that Linotron rules are produced in a 5-point typesize mode. This is automatically accommodated by RULE. When ruling is completed, the typesize is restored to the user specified value.

Examples of Use: See Appendix A, Sample Program, Statements 0011 to 0019, 0032, 0078.

Subroutine SETABS

Entries:
```
CALL STAB (N, FTAB)
CALL MARG (FMRG)
CALL DELY (YINC)
CALL DELX (XINC)
CALL OSET (YORG, XORG)
CALL RTRN (SPACE)
CALL TABB
```

Purpose: This subroutine allows specifying and incrementing of page coordinates without identifying a composing function to be performed at the resulting coordinates.

Arguments:

N	(INTEGER*4) A number less than or equal to 49 giving the length of array FTAB.
FTAB	(REAL*4) An array containing 49 or less X coordinates, in user units, for use in setting material in tabular, columnar, or outline form.
FMRG	(REAL*4) The X coordinate, in user units, of the lefthand margin of composed matter.
YINC	(REAL*4) A positive increment, in user units, to be added to the current Y position coordinate.
XINC	(REAL*4) A positive increment, in user units, to be added to the current X position coordinate.
YORG	(REAL*4) Value in user units of the Y coordinate for subsequent operations.
XORG	(REAL*4) Value in user units of the X coordinate for subsequent operations.
SPACE	(REAL*4) An increment, in user units, to be applied to the current Y position coordinate.

Discussion: These routines were designed to give the user a set of operations analogous to those found on manual composition equipment. The use of STAB is similar to setting or modifying a group of "tab-stops" on a keyboard. The first "tab-stop" is at the value of FMRG which is initialized by MARG, and which is the left margin. RTRN has an effect similar to a carriage return; the X coordinate is set at the left margin, the Y coordinate is incremented by the amount of SPACE, and an indicator is established by FATS enabling the use of "tabs". Thereafter, calls to TABB result in shifts of the X coordinate to the values in FTAB, in a consecutive manner. Initialization of MARG must precede any reference to RTRN. Both MARG and STAB must be called before any use of TABB. Once the margin and "tabs" have been initialized, they remain in effect for the entire composition job, unless altered by the user. Failure to observe the precedence in using these routines will produce unpredictable results.

An increment, XINC, is applied to the X coordinate by DELX. Its effect is similar to repetative use of a typewriter space bar; the Y coordinate is unchanged. DELY leaves the X coordinate unchanged but increments the Y coordinate by YINC in the manner of an "index" key found on some manual equipment. Since XINC is arbitrary, it may cause the X origin to exceed the next "tab-stop"; a subsequent TABB operation may then result in superimposed characters.

Any point within page limits may be specified by OSET. Its operations is similar to rolling and slewing a typewriter carriage. The arguments YORG and XORG are given to the Linotron

as the page coordinates for a subsequent operation. As with DELX, its use may disturb the TABB sequencing.

The use of RTRN and TABB represent a means of positioning and spacing type. OSET, DELX, and DELY represent another. Since mixing of these two means may easily result in superimposed and misaligned type, it is recommended that a choice of methods be made, and adhered to, in the programming planning stage. It is also possible to construct an entire FATS Linotron job with no reference to any SETABS routines, since page coordinates may be specified by the actual printing routines. SETABS has been provided for user convenience, and it presents the possibilities for alternative philosophies and program logic.

Examples of Use: See Appendix A, Sample Program, Statements 0002, 0005, 0008, 0009, 0046, 0050, 0051, 0053, 0056, 0059, 0065.

Subroutine SAVE

Entries:
```
CALL SAVE (ISAV)
CALL SAVE (ISAV, INDCTR)
CALL SPIL (ISAV)
CALL CLER (ISAV)
```

Purpose: This subroutine allows the user to save tape codes generated by a sequence of FATS subroutines, for use later in the program.

Arguments:

ISAV (INTEGER*4) The identification number of one of three "save areas": ISAV = 1, 2, or 3.

INDCTR (INTEGER*4 *variable name* to be given a value by the subroutine). An error indicator *returned* to the user's program:
0 = Save process OK.
1 = Save process error - save area overflow.

Discussion: The one-argument call to SAVE (i.e. CALL SAVE (ISAV), where ISAV = 1, 2, or 3) begins a "save process" which affects the operation of any FATS subroutine subsequently called. The two-argument call to SAVE (i.e. CALL SAVE (ISAV, INDCTR), where ISAV has the same value as in the one-argument call, and INDCTR is an integer variable name) ends this save process. During this save process - that is, between the two calls to SAVE - any FATS subroutine other than STRT, PAGE, TERM, or PASS can be called:

```
      .
      .
      .
CALL SAVE (ISAV)          ┐      Begins save process.
CALL OSET (    )          │
CALL PRNT (        )      │
CALL RULX (      )        │      Any coding except calls
      .                    │      to STRT, PAGE, TERM,
      .                    │      or PASS
      .                    │
    etc.                   │
      .                    │
      .                    │
      .                    │
CALL INUM (      )         ┘
CALL SAVE (ISAV, INDCTR)         Ends save process
      .
      .
      .
```

During the save process, the tape characters generated by FATS subroutines are stored in a "save area" in core, in addition to being written on the Linotron tape as usual. There are three save areas, numbered 1, 2, and 3, each with space for 4096 tape characters (equivalent to two blocks on the Linotron tape). The argument ISAV, which must have the same value in both the one- and two-argument calls, designates in which save area the user wishes the tape characters to be stored. The argument INDCTR is given a

value of 0 or 1 by subroutine SAVE as it ends the save process. If INDCTR=0, then the save process was successfully completed; if INDCTR=1, a save area overflow occured at some point, hence not all of the tape characters generated could be saved. Note that the save process (even if an overflow occurs) does not interfere with the normal output process whereby tape characters generated by FATS subroutines are written onto the Linotron tape, nor does it affect normal program logic in any way. Therefore, from the user's viewpoint, the save process does nothing except slightly increase the execution time of the program segment to which it applies.

At a later point in the program, when the user wishes to write the contents of a save area onto the Linotron tape, CALL SPIL (ISAV) should be used, where ISAV designates which save area is to be "spilled". This call can be repeated as many times as necessary, since SPIL does not clear the save area. Under normal conditions the call to SPIL will produce the same results on the printed page as re-executing all of the FATS subroutines called during the original save process - but with a substantial saving of execution time. For example, if the user's program is to produce 20 pages of some table of data, the page title, column headings, rules, etc. - which are the same on every page - should be generated during a save process when the first page is produced. Then, on succeeding pages, the user has to simply "spill" the appropriate save area and the page title, etc. will appear as on the first page. An important item to note is that SPIL performs an automatic re-set of the Linotron configuration before returning to the user's program, so that the Linotron configuration after the call to SPIL is always identical to the configuration before the call; this is true regardless of any temporary changes in the Linotron configuration resulting from tape characters in the save area.

A single save area can be used at different points in the program to store different groups of tape characters. To clear an already used save area for re-use, use CALL CLER (ISAV), where ISAV designates the appropriate save area. This need not be done initially.

The use of save areas appears simple. The user must, however, keep track of any changes in the Linotron configuration as his program logic progresses. Two important points must be kept in mind. The first is that if the user expects the tape characters in a specific save area to produce the same results on the printed page each time the save area is "spilled", the Linotron configuration must be identical at the time of each call to SPIL, and this configuration must match the Linotron configuration in effect when the save process for that particular save area was started. This point is fairly obvious in theory, but it is easy to overlook when a program is actually coded. The second, more subtle, complication arises when the user changes the Linotron configuration *during* the initial save process (if, for example, TYPE is called between the two calls to SAVE). Since the save process has no effect on normal program logic, any such configuration change, according to the usual rules, continues in effect until the user changes it again. Thus, a Linotron configuration change occuring during the save process may continue in effect beyond the save process. However, when the save area is "spilled" this same configuration change, now coded in the tape characters in the save area, *cannot* continue in effect beyond the call to SPIL since SPIL resets the configuration immediately before returning to the user's program. It can simply be remembered that the Linotron configuration after any call to SPIL is always the same as before the call, but this is not necessarily true for the initial save process. Thus, the user must carefully analyze the Linotron configuration in effect at each point in his program; this is a good rule to follow whether or not save areas are used.

With the above in mind, and with a little care in coding, save areas can have great flexibility. For example, a save area can be filled in segments:

```
CALL SAVE (2)
    .
    .
    .
CALL SAVE (2,INDCTR)      } Save area 2 partially filled.
    .
    .
    .
CALL SAVE (2)
    .
    .
    .
CALL SAVE (2,INDCTR)      } More tape characters added to save area 2.
    .
    .
    .
CALL SPIL (2)             All tape characters in save area 2 "spilled".
    .
    .
    .
```

As long as there is still room in a given save area after one save process (which can be determined by the value of the argument INDCTR), more tape characters can be concatenated onto the initial group by another save process. Then, when SPIL is called, all the tape characters present in the save area go contiguously onto the Linotron tape. Another flexible aspect of save areas is that save processes for different save areas can be nested or overlapped in any combination:

```
    .
    .
    .
CALL SAVE (1)                                  ┐
    .                                          │
    .                                          │
    .                                          │
CALL SAVE (2)                                  │
    .                                          │
    .                  Save process for area 2.│
    .                                          │
CALL SAVE (3)                                  │
    .                                          │
    .                                          │
CALL SAVE (2,IND2)                             │
    .                                          │
    .                                          │
    .                                          │
CALL SAVE (1,IND1)                             │
    .                                          │
    .                                          │
    .                                          │
CALL SAVE (3,IND3)                             ┘
```

Save process for area 3.

Save process for area 1.

Since calls to SAVE do not generate any tape characters themselves, the "inside" calls to SAVE have no effect on the save process(es) already in progress. Whether all this flexibility can be practically applied is dependent only on the user's ingenuity.

Examples of Use: See Appendix A, Sample Program, Statements 0030, 0040, 0075.

Subroutine LINBUF

Entries:
```
CALL BBUF
CALL EBUF (Y, X, IPRNT)
CALL CBUF (IERR, WIDTH)
```

Purpose: This subroutine adds up the total width (in user units) of a line of print, and permits the user to print the line left-adjusted, right-adjusted, or centered at a designated location on the page.

Arguments:

Y, X (both REAL*4) The page coordinates, in user units, of a location on the printed page. Negative coordinate option *not* valid; both Y and X must be explicitly given.

IPRNT (INTEGER*4) Code specifying how line of print is to be adjusted with respect to location Y, X:

1 = Line is to be left-adjusted at Y, X; i.e. the left-most character in the line will have its left edge at x=X.

2 = Line is to be right-adjusted at Y, X; i.e. the right-most character in the line will have its right edge at x=X.

3 = Line is to be centered at Y, X; i.e. the ends of the line will be equidistant from x=X.

In all cases, the bottom edge of the line (the "z-line") will be located at y=Y.

IERR (INTEGER*4 *variable name* to be given a value by the subroutine). Error code *returned* to the user's program:

0 = Line width determination OK.
1 = Line width determination invalid - line is composed of more than 1024 tape codes.
2 = Line width determination invalid - line contains a forbidden tape code.

WIDTH (REAL*4 *variable name* to be given a value by the subroutine). Width of the line, in user units, *returned* to the user's program.

Discussion: The call to BBUF begins a special "buffering process" which affects the operation of any FATS subroutine subsequently called (BBUF = Begin BUFfering). The call to EBUF ends this buffering process (EBUF = End BUFfering). During this buffering process - that is, between the calls to BBUF and EBUF - all calls to FATS subroutines must generate only *one line* of print, progressing from left to right:

```
                .
                .
                .
          CALL BBUF                    Begins buffering process
Left side of line
          CALL PRNT(          )
          CALL DELX(     )
          CALL SCAR(     )             These calls must refer to
                                       one line of print
```

```
                    .
                    .
                    .
                   etc.
                    .
                    .
                    .
                  CALL PRNT(      )
Right side of line CALL INUM(      )
                  CALL EBUF(      )          Ends buffering process
                    .
                    .
                    .
```

This buffering process, supervised by subroutine LINBUF, stores the tape characters generated by FATS subroutines in a buffer area of core, rather than permitting them to be written onto the Linotron tape. As the buffer is filled, LINBUF reads and interprets the tape characters and adds up the width of the line of print they represent. The call to EBUF ends the buffering process and transfers the contents of the buffer onto the Linotron tape, with the line's initial coordinates adjusted so that the line will appear at the location and orientation on the printed page specified by EBUF's arguments Y, X, and IPRNT. Several important restrictions apply to the FATS subroutine calls occuring during the buffering process:

(1) No explicit Y or X coordinates can be used in the calls; the negative coordinate option must be used in *all* calls requiring Y or X coordinates. Therefore, calls to MLTP, MLTC, OSET, TABB, and RULE cannot be used. Spacing within the line can be accomplished by calls to DELX.

(2) Calls to STRT, PAGE, TURN, TABB, RTRN, TERM, or PASS cannot be allowed.

(3) "Nested" calls to BBUF and EBUF cannot be allowed.

Almost anything else can be used: for instance, the type size can be changed in the middle of the line; the aspect ratio can be changed; pseudo-italic or pseudo-bold print can be used for all or part of the line; grids*, zones, and shift modes can be changed within the line, and so on. Any such changes in the Linotron configuration are considered "permanent" in the sense that they continue to apply, even beyond the call to EBUF, unless changed by the user.

Two error conditions can occur during the buffering process. The first is a buffer overflow, meaning that the line is comprised of more than 1024 tape characters; this rarely happens. The second, more common, error condition is that a "forbidden" tape character is present, which results from the user violating one of the restrictions above. In either case the line cannot be printed; the call to EBUF will end the buffering process but nothing will be written onto the Linotron tape, and the line will simply not appear on the printed page. The user's program will continue uninterrupted, essentially as though all the coding between the calls to BBUF and EBUF was not executed at all. In this case the Linotron configuration will be the same as it was before BBUF was called.

The user can check on the error condition and in addition get a numerical value for the line width (in user units) by calling CBUF (CBUF = Check BUFfering). The line width is returned to the user in the argument WIDTH, and the error condition code (= 0, 1, or 2) in the argument IERR. If IERR $\neq$ 0, the width returned is invalid; in addition, the line cannot be printed and will be "lost" when EBUF is called. CBUF can be called anytime after BBUF is called (but before BBUF is called again for another line) but the values returned in WIDTH and IERR apply only to the section of the line generated prior to the call to CBUF. Therefore, in order to obtain the *full* width of the line and the *final* error code, the user must call CBUF immediately before the call to EBUF. These final values can also be obtained

*Frequent grid (i.e. turret) changes are not recommended.

ex post facto by calling CBUF after the call to EBUF.

It should be mentioned that the buffering process will considerably slow the execution time of the user's program if used too frequently. If page headings, titles, etc., which appear on every page are to be centered, the best procedure would be to "buffer" them on the first page only and save the tape characters generated by the BBUF-to-EBUF sequence in a save area (see subroutine SAVE). Then, on subsequent pages all the user has to do is to "spill" this save area. For example, the following coding could be used to center, print, and save the title on the first page:

```
      .
      .
      .
CALL  SAVE (1)
CALL  BBUF

      .
      .
      .
Calls which generate title of page

      .
      .
      .
CALL  EBUF (YTOP, XMIDDL, 3)
CALL  SAVE (1, INDCTR)

      .
      .
      .
```

Then, on subsequent pages, the same centered title could be printed by a single

```
CALL  SPIL (1)
```

Examples of Use: See Appendix A, Sample Program, Statements 0026, 0028, 0034, 0038, 0088, 0090, 0091.

Subroutine ASIS

Entry: CALL ASIS (NN, ITCHAR)

Purpose: This subroutine takes an array of tape codes and writes them "as is" onto the Linotron tape without performing any interpretation, alteration, or editing of any kind.

Arguments:

NN (INTEGER*4) The number of characters in array ITCHAR.

ITCHAR (Array of A4-formatted words) An array of contiguous EBCDIC characters; the characters must be valid tape codes, i.e. coding directly usable by the Linotron machine. (The corresponding Hollerith literal can be substituted for an array name).

Discussion: This subroutine should be used only by programmers familiar with "raw" Linotron tape character coding. ASIS writes the tape characters in array ITCHAR (no limit on size) directly onto the Linotron tape. The only other function ASIS performs is an automatic re-set of the Linotron configuration immediately before the return to the user's program. This insures that the Linotron configuration after the call to ASIS is always identical to the configuration before the call, regardless of any temporary changes in the Linotron configuration resulting from the tape characters in array ITCHAR.

Since ASIS does virtually nothing, it is quite efficient and could be used to cut program execution time. It also has use in unusual situations that could not be efficiently handled by other FATS subroutines.

Subroutine NOOP

Entry: CALL NOOP (NBLNKS)

Purpose: This subroutine places blank characters onto the Linotron tape, for easier readability of tape dumps. Blanks are treated as "no-op" codes by the Linotron machine.

Argument: NBLNKS (INTEGER*4) For NBLNKS $\geq$ 0, the number of blanks to be written. If NBLNKS $<$ 0, the remainder of the current tape block will be filled with blanks.

Discussion: This subroutine can be used to improve the readability of Linotron tape dumps by enabling the user to place blanks at appropriate locations between groups of tape characters generated by calls to FATS subroutines. If NBLNKS is negative (any negative value), the remainder of the current tape block is padded with blanks, and any tape characters generated by subsequently called FATS subroutines will be written in a new block. No function is performed by the Linotron machine when it encounters blanks on the Linotron tape; blanks are therefore effectively "no-op" codes.

For reasons of tape use efficiency, this subroutine should be used with discretion. Also, do *not* call NOOP with NBLNKS $<$ 0 when either a save process (see subroutine SAVE) or a line buffering process (see subroutine LINBUF) is in effect.

Examples of Use: *See* Appendix A, Sample Program, Statements 0031, 0039.

SECTION III

APPENDICES

Preceding page blank

Appendix A

Examples

The following examples are included for illustrating the use of FATS. They have been extracted from programs which have actually composed pages.

EXAMPLE 1: Bureau of Land Management Ephemeris, page 19.
The printed page is reproduced here as page 43. Initialization is performed by the following segment.

```
    CALL TURT(2,0,0,0)
    CALL STRT(JBNO,IFRM)
    CALL TYPE(1,1,1,7)
    CALL SQEZ(1)
    CALL SPEC(1)
    CALL TURN(1)
    CALL DIMN(77./16.,81.)
    CALL MARG(2.)
 50 CALL IDEN (150.,0.,1)
```

The call to TURT establishes the position of the Naval Observatory grid in turret position 1. STRT passes the job and frame numbers to FATS. Turret 1, zone 1, unshift, and 7 point size are selected by TYPE. Normal width, face, and page orientation are specified by the next three routines. The arguments of DIMN create an unusual system of units wherein 81 user units correspond to 4.8125 inches. MARG sets the left margin at X=2 user units. The job and frame numbers are printed on the frame by IDEN, far enough from the images to allow for later trimming.

On the sample page, yearly changes occur not only in the table body, but also in certain arguments. The six horizontal arguments, labelled A on page 43, are computed by the composition program and entered into storage arrays. The following instruction sequence shows how the arguments were composed.

```
    CALL SIZE(7)
    CALL SQEZ (2)
    DO 120 K=1,3
    CALL INUM (18.,61.26+5.5*K,1.35,KM(K),0,1)
    CALL SCAR (-1.,-1.,1,3,2,'T')
    CALL INUM (-1.,-1.,1.35,KS(K),0,2)
    CALL SCAR (-1.,-1.,1,3,2,'Y')
    CALL INUM (27.5,61.26+5.5*K,1.35,KM(K+3),0,1)
    CALL SCAR (-1.,-1.,1,3,2,'T')
    CALL INUM (-1.,-1.,1.35,KS(K+3),0,2)
    CALL SCAR (-1.,-1.,1,3,2,'Y')
120 CONTINUE
    CALL SQEZ(1)
```

Mean Time Hour Angle	AZIMUTH OF POLARIS AT ALL HOUR ANGLES, 1973											CORR. TO AZIMUTH		
	Mean Declination + 89° 08′ 40″											Add for Decl. +89°		
												8′30″	8′20″	8′10″
	Latitude											Subtract for Decl. +89°		
	30°	32°	34°	36°	38°	40°	42°	44°	46°	48°	50°	8′50″	9′00″	9′10″
h m	′	′	′	′	′	′	′	′	′	′	′	′	′	′
5 54.9	...	...	...	...	...	...	...	...	...	...	79.9	0.3	0.5	0.8
55.2	...	...	...	...	...	...	...	...	...	76.7	...	0.2	0.5	0.7
55.5	...	...	...	...	...	...	...	...	73.9	...	...	0.2	0.5	0.7
5 55.7	...	...	...	...	...	...	...	71.4	...	...	...	0.2	0.5	0.7
55.9	...	...	...	...	...	...	69.1	...	...	...	...	0.2	0.4	0.7
56.2	...	...	...	...	...	67.0	...	...	...	...	...	0.2	0.4	0.7
5 56.4	...	...	...	...	65.1	...	...	...	...	...	...	0.2	0.4	0.6
56.5	...	...	...	63.5	...	...	...	...	...	...	...	0.2	0.4	0.6
56.7	...	...	61.9	...	...	...	...	...	...	...	...	0.2	0.4	0.6
5 56.9	...	60.5	...	...	...	...	...	...	...	...	...	0.2	0.4	0.6
57.0	59.3	...	...	...	...	...	...	...	...	...	...	0.2	0.4	0.6
59.0	59.3	60.5	61.9	63.4	65.1	67.0	69.1	71.4	73.9	76.7	79.9	0.2	0.4	0.7
6 09.0	59.2	60.4	61.8	63.4	65.0	66.9	69.0	71.2	73.8	76.6	79.7	0.2	0.4	0.7
19.0	59.0	60.3	61.6	63.1	64.8	66.7	68.7	71.0	73.5	76.3	79.4	0.2	0.4	0.6
28.9	58.7	59.9	61.3	62.8	64.5	66.3	68.4	70.6	73.1	75.9	79.0	0.2	0.4	0.6
6 38.9	58.3	59.5	60.9	62.4	64.0	65.8	67.9	70.1	72.6	75.3	78.4	0.2	0.4	0.6
48.9	57.8	59.0	60.3	61.8	63.4	65.2	67.2	69.4	71.9	74.6	77.7	0.2	0.4	0.6
58.9	57.1	58.3	59.7	61.1	62.7	64.5	66.5	68.7	71.1	73.8	76.8	0.2	0.4	0.6
7 08.8	56.4	57.6	58.9	60.3	61.9	63.7	65.6	67.8	70.1	72.8	75.8	0.2	0.4	0.6
18.8	55.5	56.7	58.0	59.4	61.0	62.7	64.6	66.7	69.1	71.7	74.6	0.2	0.4	0.6
28.8	54.6	55.7	57.0	58.4	59.9	61.6	63.5	65.6	67.9	70.4	73.3	0.2	0.4	0.6
7 38.7	53.5	54.6	55.9	57.2	58.7	60.4	62.2	64.3	66.5	69.0	71.8	0.2	0.4	0.6
48.7	52.4	53.5	54.7	56.0	57.5	59.1	60.9	62.9	65.1	67.5	70.3	0.2	0.4	0.6
58.7	51.1	52.2	53.4	54.7	56.1	57.7	59.4	61.4	63.5	65.9	68.5	0.2	0.4	0.6
8 08.7	49.8	50.8	51.9	53.2	54.6	56.1	57.6	59.7	61.8	64.1	66.7	0.2	0.4	0.5
18.6	48.3	49.3	50.4	51.7	53.0	54.5	56.1	58.0	60.0	62.2	64.8	0.2	0.4	0.5
28.6	46.8	47.8	48.8	50.0	51.3	52.8	54.4	56.1	58.1	60.3	62.7	0.2	0.3	0.5
8 38.6	45.2	46.1	47.1	48.3	49.5	50.9	52.5	54.2	56.1	58.1	60.5	0.2	0.3	0.5
48.6	43.4	44.3	45.3	46.4	47.7	49.0	50.5	52.1	53.9	55.9	58.2	0.2	0.3	0.5
58.5	41.7	42.5	43.5	44.5	45.7	47.0	48.4	50.0	51.7	53.6	55.8	0.2	0.3	0.5
9 08.5	39.8	40.6	41.5	42.5	43.6	44.9	46.2	47.7	49.4	51.2	53.3	0.1	0.3	0.4
18.5	37.9	38.6	39.5	40.5	41.5	42.7	43.9	45.4	46.9	48.7	50.6	0.1	0.3	0.4
28.4	35.8	36.6	37.4	38.3	39.3	40.4	41.6	43.0	44.4	46.1	47.9	0.1	0.3	0.4
9 38.4	33.8	34.5	35.2	36.1	37.0	38.0	39.2	40.5	41.9	43.4	45.1	0.1	0.2	0.4
48.4	31.6	32.3	33.0	33.8	34.7	35.6	36.7	37.9	39.2	40.7	42.3	0.1	0.2	0.3
58.4	29.4	30.0	30.7	31.4	32.2	33.1	34.1	35.2	36.5	37.8	39.3	0.1	0.2	0.3
10 08.3	27.2	27.7	28.3	29.0	29.8	30.6	31.5	32.5	33.7	34.9	36.3	0.1	0.2	0.3
18.3	24.9	25.4	25.9	26.6	27.2	28.0	28.8	29.8	30.8	31.9	33.2	0.1	0.2	0.3
28.3	22.5	23.0	23.5	24.0	24.7	25.4	26.1	27.0	27.9	28.9	30.1	0.1	0.2	0.2
10 38.3	20.1	20.5	21.0	21.5	22.0	22.7	23.3	24.1	24.9	25.8	26.9	0.1	0.1	0.2
48.2	17.7	18.0	18.4	18.9	19.4	19.9	20.5	21.2	21.9	22.7	23.6	0.1	0.1	0.2
58.2	15.2	15.5	15.9	16.3	16.7	17.1	17.7	18.2	18.8	19.5	20.3	0.1	0.1	0.2
11 08.2	12.7	13.0	13.3	13.6	13.9	14.3	14.8	15.2	15.8	16.3	17.0	0.0	0.1	0.1
18.1	10.2	10.4	10.6	10.9	11.2	11.5	11.8	12.2	12.6	13.1	13.6	0.0	0.1	0.1
28.1	7.7	7.8	8.0	8.2	8.4	8.6	8.9	9.2	9.5	9.9	10.2	0.0	0.1	0.1
11 38.1	5.1	5.2	5.3	5.5	5.6	5.8	5.9	6.1	6.3	6.6	6.8	0.0	0.0	0.1
48.1	2.6	2.6	2.7	2.7	2.8	2.9	3.0	3.1	3.2	3.3	3.4	0.0	0.0	0.0
58.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

$$\text{Tan } A = \frac{\sin t}{\cos\phi \tan\delta - \sin\phi \cos t}$$

The product, $\sin\phi \cos t$, is subtracted for hour angles 0° to 90° and added for hour angles from 90° to 180°.

The printed quantities are taken from arrays KM and KS, and set by INUM. Reference to the Naval Observatory grid chart and the description of SCAR will clarify the use of grid location characters "T" and "Y" to select minute and seconds symbols. The numbers are set condensed by invoking SQEZ. This is to allow two-digit "minutes" figures in future years.

The entire table body is set by the following sequences, with the exception of the logic for deciding between numerals and dot leaders.

```
900 CALL OSET (33.,2.)
    DO 240 K=1,IBLOK
    M = INDX(K,KL)
    DO 230 N=1,M
    KT=KT+1
    IF(N.NE.1)GO TO 205
    CALL INUM (-1.,-1.,2.,IH(KT),0,1)
205 CALL FNUM (-1.,0.,7.5,EM(KT),1,0,2)
208 DO 215 J=1,11
210 CALL FNUM (-1.,0.,8.+4.8636*J,BD(KT,J),1,0,1)
215 CONTINUE
    DO 220 J=1,3
    CALL FNUM (-1.,0.,64.5+5.5*J,COR(KT,J),1,0,1)
220 CONTINUE
    CALL RTRN(1.5)
230 CONTINUE
    CALL RTRN(1.)
240 CONTINUE
```

The logic, including indexing, determines vertical and horizontal spacing as well as the inclusion of leading figures in the "h" column. All coordinates, as well as the computations appearing as arguments in FNUM and INUM, are in user units. The flexibility of FATS is emphasized by the fact that data may be accessed through I/O operations while composition proceeds. In this sequence, tabular entries are transferred from storage to the Linotron control tape by the number writing routines FNUM and INUM. It should be noted that the number of digits appearing in each tabular entry is controlled by argument values in FATS routines, and does not require user coded program logic beyond initial selection of the parameters.

The footnote on the sample page was composed by the following sequence.

```
CALL SQEZ(2)
CALL PRNT(127.5,25.,8,'|TAN |A ',1.)
CALL SCAR(-1.,-1.,1,3,2,'9')
CALL RULX(127. ,33.,53.,1)
CALL PRNT(126.5,41.,5,'SIN T',.5)
CALL PRNT(129.,34.,4,'COS ',.38312)
CALL SCAR(-1.,-1.,1,3,2,'2')
CALL PRNT(-1.,-1.,6,'  TAN ',.5)
CALL SCAR(-1.,-1.,1,3,1,'F')
CALL PRNT(-1.,-1.,1,' ',1.)
CALL SCAR(-1.,-1.,1,3,1,'0')
CALL PRNT(-1.,-1.,4,' SIN',1.)
CALL PRNT(-1.,-1.,1,' ',.38312)
CALL SCAR(-1.,-1.,1,3,2,'2')
CALL PRNT(-1.,-1.,7,'  COS T',.5)
```

```
CALL PRNT(131.,17.,12,'(THE PRODUCT',1.)
CALL SCAR(-1.,-1.,1,2,1,',')
CALL PRNT(-1.,-1.,6,'  SIN ',.5)
CALL SCAR(-1.,-1.,1,3,2,'2')
CALL PRNT(-1.,-1.,7,'  COS T',.5)
CALL SCAR(-1.,-1.,1,2,1,',')
CALL PRNT(-1.,-1.,30,' IS SUBTRACTED FOR HOUR ANGLES',1.)
CALL INUM (133.,17.,1.,0,0,1)
CALL SCAR(-1.,-1.,1,3,2,'R')
CALL PRNT(-1.,-1.,4,' TO ',1.)
CALL INUM (-1.,-1.,2.,90,0,2)
CALL SCAR(-1.,-1.,1,3,2,'R')
CALL PRNT(-1.,-1.,32,' AND ADDED FOR HOUR ANGLES FROM ',1.)
CALL INUM (-1.,-1.,2.,90,0,2)
CALL SCAR(-1.,-1.,1,3,2,'R')
CALL PRNT(-1.,-1.,4,' TO ',1.)
CALL INUM (-1.,-1.,3.,180,0,3)
CALL SCAR(-1.,-1.,1,3,2,'R')
CALL SCAR(-1.,-1.,1,2,1,'.')
CALL SQEZ(1)
```

See B on page 43. In this series of statements, the logic is straightforward. The length of coding is due to the number of special characters which require individual handling. The composition of the footnote will be better understood by proceeding through the coding step-by-step, referring to the grid character chart and routine descriptions.

EXAMPLE 2: Sample Program

On the following pages is a sample program which generates two pages of Linotron copy with the aid of FATS subroutines. The Linotron output pages produced by this program are also shown. The following is an outline of the coding logic; for more detailed descriptions of the functions performed by each subroutine call, see the individual FATS subroutine descriptions in this *Circular* or the summarized descriptions in Appendix F. Within the program, Fortran data set 5 refers to the card reader, while Fortran data set 6 refers to the printer. The four digit numbers at the beginning of each paragraph below refer to the internal statement numbers found to the left of each source statement in the program listing.

Job Initialization

0005 - 0009. FATS initialization calls. These calls provide information to FATS, but generate no tape codes. Call to TURT specifies that grids 1, 2, 3, and 4 will be mounted on turrets 1, 2, 3, and 4, respectively (see grid list on page 7). Call to DIMN establishes inches as user's units for page and coordinate measurement. Calls to MARG and STAB inform FATS of left margin of page and array of four "tab stops" to be used later in the program.

Page 1

0010. Call to STRT opens Linotron tape. Job number is 005-827; beginning frame number is 0001. STRT also establishes an initial configuration, with turret 1 (grid 1 mounted) positioned for printing.

0011 - 0019. Rules drawn: three horizontal rules, five vertical rules, and one diagonal rule. RULD will change turrets so that period on the Naval Observatory grid can be used for generating the diagonal pseudo-rule, but original turret (turret 1) is restored before return to user's program.

0020 - 0028. Boxheads and page heading printed. Page heading "TRIGONOMETRIC FUNCTIONS" is automatically centered by the BBUF-to-EBUF sequence of statements 0026-0028.

0029 - 0043. Credit line and accompanying rule are printed near bottom of page. Credit line is printed in pseudo-italic type; the call to SPEC in statement 0035 begins the pseudo-italicizing pricess while the call to SPEC in statement 0037 ends this process and resumes normal printing. Credit line is automatically centered by the BBUF-to-EBUF sequence of statements 0034-0038. All tape codes generated between statements 0030 and 0040 are saved in save area 1; statements 0041-0043 check for save area overflow and stop execution if overflow occurs.

0044 - 0046. Configuration for printing table body is established, degree symbol is printed, and Linotron page location is set.

0047 - 0067. Body of table is printed. Loop cycles at each printed line. Y coordinate is advanced and X coordinate is returned to left margin by call to RTRN in statement 0050. Use of negative coordinates in calls to INUM, FNUM, and DNUM prevents these routines from establishing page coordinates. Instead, page location is controlled by calls to TABB in statements 0051, 0053, 0056, and 0059. Each of these calls sets the X coordinate to the position of the next "tab stop" while leaving the Y coordinate unchanged. Double-precision tangent is used because of extra significant digits needed near 90°. Note that at 90° the tangent is not evaluated or printed; instead, a centered asterisk is printed by statements 0065-0066.

0068 - 0072. Page identification. Note that page number is set in condensed-width type. Call to SQEZ in statement 0069 sets the aspect ratio to 83%, while the call to SQEZ in statement 0071 resets the aspect ratio to 100% (normal printing). IDEN prints the job and frame numbers at the lower right corner of the page.

Page 2

0073. Call to PAGE advances Linotron's photographic film and initializes new page. Configuration is left unchanged. The FATS frame counter is incremented by 1.

0074 - 0078. Credit line and page heading printed. Credit line and accompanying rule are printed by "spilling" the tape codes in save area 1 (statement 0075), stored there when page 1 was generated (see discussion of statements 0029-0043 above). Page heading and horizontal rule for top of page are then printed.

0079 - 0084. Zone and typesize are changed. Multiple-print call MLTP is used to print "km" and "days" several times on the page. Variables Y and X, later to be used as page coordinate arguments, are given initial values.

0085 - 0096. Body of table is printed. Loop cycles at each printed line, which is to begin at X=0.500 inches and end at X=4.950 inches. Variable Y, which designates the Y coordinate of the printed line is incremented, then an input card is read. (a listing of the input cards follows the program source listing). The card contains, in the form of two EBCDIC character arrays, the data to be printed in the line. The first character array, "LABEL", is to be printed left-justified in the line, while the second character array, "IVALUE", is to be printed right-justified in the line. A string of periods ("dot leaders") is to fill the gap in the middle of the line. The BBUF-to-EBUF sequence of statements 0088-0091 prints "LABEL" left justified; the width of the printed version of "LABEL" is obtained by the call to CBUF. This width is then used in statements 0092-0094 to determine the starting X coordinate and number of dot leader periods to be printed. These dot leader periods are printed using the multiple-character-print call MLTC in statement 0095. Finally, "IVALUE", which contains only numeric digits, blanks, decimal points, and commas, is printed right-justified in a field at the end of the line by subroutine ANUM.

0097 - 0100. Disclaimer line is printed, in turn-page mode. Coordinate values in call to PRNT in statement 0099 are turn-page coordinates; i.e. they are measured from the turn-page origin located at the lower-left corner of the page. Call to TURN in statement 0100 resets Linotron to normal mode.

0101 - 0105. Page identification. Similar to corresponding sequence on Page 1.

Job Termination

0106. Call to PASS advances Linotron's photographic film, empties FATS Linotron tape output buffers,

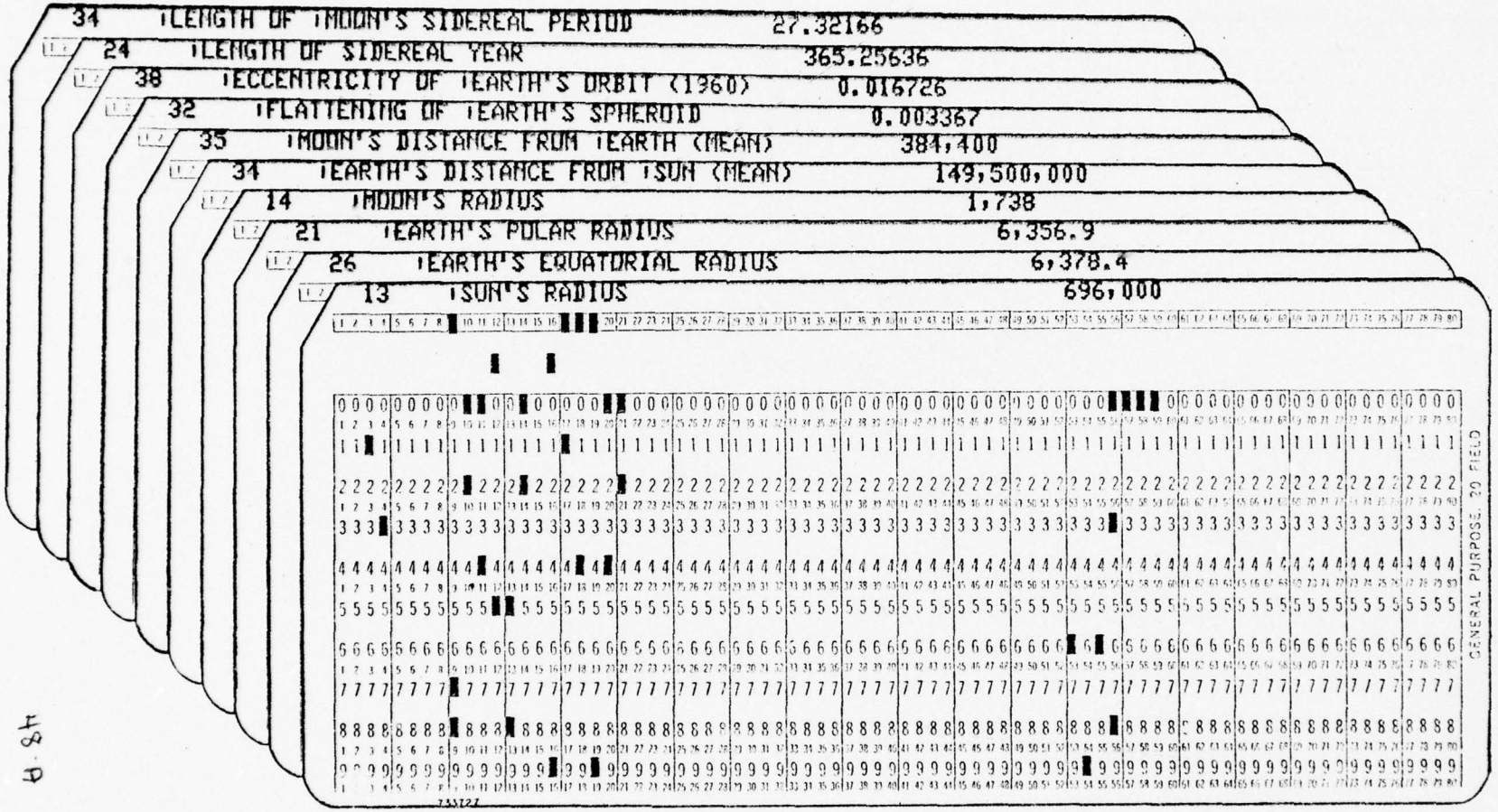
and closes Linotron tape. No stop code for the Linotron is placed on the Linotron tape; therefore, it is assumed that coding for more pages will be added to Linotron tape during a future run.

```
FORTRAN IV G LEVEL  18                  MAIN                  DATE = 73262          01/58/42                PAGE 0001

                C
                C     MAIN PROGRAM TO CREATE TWO PAGES OF LINOTRON COPY.
                C
0001                  REAL*8 DANGLE, TANANG, DFLOAT, DTAN
0002                  DIMENSION XTABS(4), LABEL(10), IVALUE(3)
0003                1 FORMAT (///,' SAVE AREA OVERFLOW ',///)
0004                2 FORMAT (I4,4X,10A4,4X,3A4)
0005                  DATA XTABS/1.4,2.1,3.1,4.1/
                C
                C     INITIALIZATION CALLS.
0006                  CALL TURT (1,2,3,4)
0007                  CALL DIMN (1.0,1.0)
0008                  CALL MARG (0.5)
0009                  CALL STAB (4,XTABS)
                C
                C
                C**** PAGE 1 ***********************************************************************
                C
0010                  CALL STRT (005827,1)
                C
                C     THE FOLLOWING STATEMENTS PRINT THE RULES, BOXHEADS, AND PAGE HEADING.
0011                  CALL RULX (1.6,1.2+5.0,2)
0012                  CALL RULX (2.2,1.2+5.0,2)
0013                  CALL RULX (7.9,1.2+5.0,2)
0014                  CALL RULY (1.6,1.2+7.9,2)
0015                  CALL RULY (1.6,2.0+7.9,2)
0016                  CALL RULY (1.6,3.0+7.9,2)
0017                  CALL RULY (1.6,4.0+7.9,2)
0018                  CALL RULY (1.6,5.0+7.9,2)
0019                  CALL RULD (1.6,1.2+2.2+2.0,2)
0020                  CALL SIZE (7)
0021                  CALL PRNT (1.8,1.5,9,'!FUNCTION',0.0)
0022                  CALL PRNT (2.1,1.4.6,'!ANGLE',0.0)
0023                  CALL TYPE (0,3,0,14)
0024                  CALL PRNT (2.0,2.365,11,'SIN COS TAN',0.7308)
0025                  CALL TYPE (4,1,2,18)
0026                  CALL BBUF
0027                  CALL PRNT (-1.,-1.,23,'TRIGONOMETRIC FUNCTIONS',0.15)
0028                  CALL EBUF (1.0,3.1,3)
                C
                C     THE FOLLOWING STATEMENTS PRINT THE CREDIT LINE AND ALSO SAVE IT FOR
                C     USE ON THE NEXT PAGE.
0029                  CALL GRID (1)
0030                  CALL SAVE (1)
0031                  CALL NOOP (5)
0032                  CALL RULX (8.6,2.5,3.7,1)
0033                  CALL TYPE (1,1,1,10)
0034                  CALL BBUF
0035                  CALL SPEC (2)
0036                  CALL PRNT (-1.,-1.,67,'!PREPARED BY !NAUTICAL !ALMANAC !OFFICE, !U
                     /.!S. !NAVAL !OBSERVATORY',0.07)
0037                  CALL SPEC (1)
0038                  CALL EBUF (8.9,3.1,3)
0039                  CALL NOOP (5)
0040                  CALL SAVE (1,IND)
                C
0041                  IF (IND) 130,130,120
0042              120 WRITE (6,1)
0043                  STOP 2
0044              130 CALL TYPE (2,1,1,10)
0045                  CALL SCAR (2.45,1.63,2,3,2,'R')
0046                  CALL OSET (2.5,1.5)
                C
                C     THE FOLLOWING LOOP PRINTS THE BODY OF THE TABLE.
                C     NOTE USE OF 'TAB STOPS' AND NEGATIVE COORDINATE OPTION.
0047                  DO 170 I=1,181,5
0048                  IANGLE = I - 1
0049                  ANGLE = FLOAT(IANGLE) * 0.017453293
0050                  CALL RTRN (0.138)
0051                  CALL TABB
0052                  CALL INUM (-1.,-1.+0.3,IANGLE,0,1)
0053                  CALL TABB
0054                  SINANG = SIN(ANGLE)
0055                  CALL FNUM (-1.,-1.+0.7,SINANG,5,1,1)
0056                  CALL TABB
0057                  COSANG = COS(ANGLE)
0058                  CALL FNUM (-1.,-1.+0.7,COSANG,5,1,1)
0059                  CALL TABB
0060                  IF (IANGLE-90) 150,160,150
                C     DOUBLE PRECISION TANGENT IS USED.
0061              150 DANGLE = DFLOAT(IANGLE) * 0.017453292519900
0062                  TANANG = DTAN(DANGLE)
0063                  CALL FNUM (-1.,-1.+0.7,TANANG,5,1,1)
0064                  GO TO 170
0065              160 CALL DELX (0.4)
0066                  CALL SCAR (-1.,-1.+2,1.2,'***')
0067              170 CONTINUE
```

```
C
C     PAGE IDENTIFICATION.
0068        CALL ZONE (2)
0069        CALL SQEZ (2)
0070        CALL PRNT (9.7,2.920,7,'(PAGE 1',0.06)
0071        CALL SQEZ (1)
0072        CALL IDEN (9.9,6.0,2)
C
C
C**** PAGE 2 ***************************************************************
C
0073        CALL PAGE
C     THE FOLLOWING STATEMENTS PRINT THE CREDIT LINE, SAVED FROM THE
C     PREVIOUS PAGE, AND THE PAGE HEADING.
0074        CALL GRID (1)
0075        CALL SPIL (1)
0076        CALL TYPE (1,2,1,18)
0077        CALL PRNT (1.0,1.428,29,'LIST OF PHYSICAL CONSTANTS',0.15)
0078        CALL RULX (1.2,2.0,4.2,3)
C
C     THE FOLLOWING STATEMENTS PRINT THE BODY OF THE TABLE.
C     DATA FOR THE TABLE IS READ IN FROM CARDS.
0079        CALL ZONE (1)
0080        CALL SIZE (12)
0081        CALL MLTP (2.0,5.1,2,'KM',0.0,0.2,0.0,6)
0082        CALL MLTP (3.6,5.1,4,'DAYS',0.0,0.2,0.0,2)
0083        Y = 1.8
0084        X = 0.5
0085        DO 220 LINE=1,10
0086        Y = Y + 0.2
0087        READ (5,2)  NUMCHR,  (LABEL(I), I=1,10),  (IVALUE(I), I=1,3)
C     LABEL AND IVALUE ARE CHARACTER ARRAYS FOR CALLS TO PRNT AND ANUM,
C     RESPECTIVELY.  NUMCHR IS THE NUMBER OF CHARACTERS IN ARRAY LABEL
C     TO BE PROCESSED BY PRNT.
0088        CALL BBUF
0089        CALL PRNT (-1.,-1.,NUMCHR,LABEL,0.09)
0090        CALL CBUF (IERR,WIDTH)
0091        CALL EBUF (Y,X,1)
0092        XEND = X + WIDTH + 0.12
0093        XDOTS = AINT (XEND * 10.0) / 10.0
0094        NDOTS = (4.0 - XDOTS) / 0.1 + 1.
0095        CALL MLTC (Y,XDOTS,1.1,1,'.',0.0,0.1,NDOTS)
0096      220 CALL ANUM (Y,4.0,0.950,12,IVALUE)
0097        CALL TYPE (2,2,1,8)
0098        CALL TURN (2)
0099        CALL PRNT (5.573,7.409,23,'(NOT DEFINITIVE VALUES)',0.06)
0100        CALL TURN (1)
C
C     PAGE IDENTIFICATION.
0101        CALL SIZE (10)
0102        CALL SQEZ (2)
0103        CALL PRNT (9.7,2.920,7,'(PAGE 2',0.06)
0104        CALL SQEZ (1)
0105        CALL IDEN (9.9,6.0,2)
C
C
C*****************************************************************************
0106        CALL PASS
0107        STOP 1
0108        END
```

Input Cards for Sample Program



TRIGONOMETRIC FUNCTIONS

Function / Angle	*sin*	*cos*	*tan*
°			
0	+ 0.00000	+ 1.00000	+ 0.00000
5	+ 0.08716	+ 0.99619	+ 0.08749
10	+ 0.17365	+ 0.98481	+ 0.17633
15	+ 0.25882	+ 0.96593	+ 0.26795
20	+ 0.34202	+ 0.93969	+ 0.36397
25	+ 0.42262	+ 0.90631	+ 0.46631
30	+ 0.50000	+ 0.86603	+ 0.57735
35	+ 0.57358	+ 0.81915	+ 0.70021
40	+ 0.64279	+ 0.76604	+ 0.83910
45	+ 0.70711	+ 0.70711	+ 1.00000
50	+ 0.76604	+ 0.64279	+ 1.19175
55	+ 0.81915	+ 0.57358	+ 1.42815
60	+ 0.86603	+ 0.50000	+ 1.73205
65	+ 0.90631	+ 0.42262	+ 2.14451
70	+ 0.93969	+ 0.34202	+ 2.74748
75	+ 0.96593	+ 0.25882	+ 3.73205
80	+ 0.98481	+ 0.17365	+ 5.67128
85	+ 0.99619	+ 0.08716	+ 11.43005
90	+ 1.00000	+ 0.00000	*
95	+ 0.99619	− 0.08715	− 11.43005
100	+ 0.98481	− 0.17365	− 5.67128
105	+ 0.96593	− 0.25882	− 3.73205
110	+ 0.93969	− 0.34202	− 2.74748
115	+ 0.90631	− 0.42262	− 2.14451
120	+ 0.86603	− 0.50000	− 1.73205
125	+ 0.81915	− 0.57358	− 1.42815
130	+ 0.76604	− 0.64279	− 1.19175
135	+ 0.70711	− 0.70711	− 1.00000
140	+ 0.64279	− 0.76604	− 0.83910
145	+ 0.57358	− 0.81915	− 0.70021
150	+ 0.50000	− 0.86603	− 0.57735
155	+ 0.42262	− 0.90631	− 0.46631
160	+ 0.34202	− 0.93969	− 0.36397
165	+ 0.25882	− 0.96593	− 0.26795
170	+ 0.17365	− 0.98481	− 0.17633
175	+ 0.08716	− 0.99619	− 0.08749
180	+ 0.00000	− 1.00000	− 0.00000

Prepared by Nautical Almanac Office, U.S. Naval Observatory

005-827 0001

The above Linotron frame has been reduced to 75% of its original size.

List of Physical Constants

Sun's radius	696,000	km
Earth's equatorial radius	6,378.4	km
Earth's polar radius	6,356.9	km
Moon's radius	1,738	km
Earth's distance from Sun (mean)	149,500,000	km
Moon's distance from Earth (mean)	384,400	km
Flattening of Earth's spheroid	0.003367	
Eccentricity of Earth's orbit (1960)	0.016726	
Length of sidereal year	365.25636	days
Length of Moon's sidereal period	27.32166	days

(not definitive values)

Prepared by Nautical Almanac Office, U.S. Naval Observatory

005-827 0002

The above Linotron frame has been reduced to 75% of its original size.

Appendix B

FATS Use from Other Programming Languages

The FATS subroutine system utilizes the standard OS/360 linkage conventions. When a FATS subroutine has been entered, the following is assumed:

1. Register 1 contains the address of the first byte of an argument list.
2. Register 13 contains the address of the first byte of the calling program's save area.
3. Register 14 contains an address within the calling program for return after execution of the FATS subroutine.
4. Register 15 contains the address of the FATS subroutine entry point.
5. The argument list consists of a list of contiguous addresses, each occupying a full word. Each of these addresses refers to the first byte of a word, array, or character string used as an argument for the subroutine.
6. The calling program's save area is an area 18 full words long to be used by the FATS subroutine as temporary storage for register contents, etc. If the calling program was in turn called by a higher-level program, the second word of the save area should contain the address of the higher-level program's save area. (This is almost always the case, since even the user's main routine is called by the System Supervisor).
7. For calls to subroutine SAVE, the *last address* in the argument list should have a 1 in the highest order (sign) bit.
8. FATS argument types and their internal representation is as follows:

FATS argument type:	Internal representation:	Address in argument list refers to:
INTEGER*4	Full word fixed-point* number (4 bytes)	High order byte
REAL*4	Short floating-point* number (4 bytes)	Characteristic byte
REAL*8 (entry DNUM only)	Long floating-point* number (8 bytes)	Characteristic byte
REAL*4 Array (entry STAB only)	Array of contiguous short floating-point numbers (4 bytes each)	Characteristic byte of first number in array
Array of A4-formatted words	String of contiguous EBCDIC characters (1 byte each)	First character in string
A1- formatted word (entries SCAR and MLTS)	One EBCDIC character followed by 3 blanks (4 bytes total)	First character

*See *IBM System/360 Principles of Operation* for a complete description of the internal representation of fixed and floating-point numbers.

The following gives a brief description of how the above requirements can be met using various programming languages available on the 360.

FORTRAN IV

All requirements and conventions listed above are automatically set up by the compiler when the standard CALL statement is encountered. See *FORTRAN IV (G and H) Programmer's Guide* for further details.

PL/1

Most requirements and conventions listed above are set up by the PL/1 compiler when the standard CALL statement is encountered. The user should utilize the following PL/1 data types:

FATS argument type:	Corresponding PL/1 data type:
INTEGER*4	REAL FIXED BINARY (31,0)
REAL*4	REAL FLOAT BINARY (21) or REAL FLOAT DECIMAL (6)
REAL*8	REAL FLOAT BINARY (53) or REAL FLOAT DECIMAL (16)
A1- or A4 formatted words or arrays	CHARACTER strings

Single-dimensional PL/1 arrays correspond to single-dimensioned FORTRAN arrays of corresponding type, but multi-dimensional arrays are arranged differently by the two compilers.

The PL/1 user must not use array names or character-string names as arguments to FATS subroutines. Use of array or character-string names causes the PL/1 compiler to place the address of a "dope vector", which describes the array or string, into the argument list. Since the FATS subroutine is expecting the address of the first byte of the array or string, *an unpredictable program failure* will occur.

To circumvent this difficulty with arrays, pass the *first element* of the array as an argument rather than the array name. For example:

```
        .
        .
        .
DECLARE TABSTOPS (4) REAL FLOAT DECIMAL (6)
          INITIAL (1.250, 2.655, 4.750, 6.500),
        NUMTABS REAL FIXED BINARY (31) INITIAL (4);
        .
        .
        .
CALL STAB (NUMTABS, TABSTOPS(1));
        .
        .
        .
```

Passing character strings to FATS subroutines requires a little more effort. The best way is to declare a based variable whose pointer variable is set to the address of the first byte in the string. The based variable should then be passed as the subroutine argument. For example:

```
.
.
.
DECLARE (Y, X, SPACE) REAL FLOAT BINARY (21),
        N REAL FIXED BINARY (31,0);
DECLARE NTEXT CHAR (13) INITIAL ( 'TITLE OF PAGE' );
DECLARE CHARSTRING BASED (POINTR );
.
.
.
POINTR = ADDR ( NTEXT );
CALL PRNT ( Y, X, N, CHARSTRING, SPACE );
.
.
.
```

Another, less elegant, way of accomplishing the same thing is to overlay define a variable onto the beginning of the character string, then pass the defined variable as the subroutine argument. For example:

```
.
.
.
DECLARE (Y, X, SPACE ) REAL FLOAT BINARY (21),
        N REAL FIXED BINARY (31,0);
DECLARE NTEXT CHAR (13) INITIAL ( 'TITLE OF PAGE');
DECLARE OVERNTEXT DEFINED NTEXT;
.
.
.
CALL PRNT (Y, X, N, OVERNTEXT, SPACE)
.
.
.
```

The problem with this method is that it causes an IEM1105I compiler error (ERROR level, completion code 8) because of the unlike attributes of the defined variable and the character string. However, if the COND parameter of the link edit EXEC statement is high enough, linkage editing and execution of the procedure can continue.

The linkage editor must have available (in addition to the FATS modules) both the PL/1 and FORTRAN libraries. To accomplish this, include the following JCL statement for the link edit step:

```
//LKED.SYSLIB   DD   DSNAME=SYS1.PL1LIB,DISP=OLD
//              DD   DSNAME=SYS1.FORTLIB,DISP=OLD
```

(The DSNAMEs for the FORTRAN and PL/1 libraries may vary, depending on the installation)

PL/1 users should consult the *PL/1 Reference Manual* for full descriptions of data types, based and pointer variables, the ADDR function, and overlay definition of variables. The *PL/1 (F) Programmer's Guide* contains descriptions of PL/1 linkage conventions, dope vectors, compiler errors and completion codes, and linkage editor processing.

ASSEMBLER LANGUAGE

All of the standard linkage conventions can be set up using the standard CALL macro, subject to a few requirements on the user's part. Assembler language programmers should use the following constant types:

FATS argument type:	Corresponding Assembler Language constant type:
INTEGER*4	F
REAL*4	E
REAL*8	D
A1- or A4- formatted words or arrays	C

A group of F, E, or D constants stored contiguously in core corresponds to a single-dimensioned FORTRAN array of corresponding type.

Assembler language users should use the following procedures within their programs:

Establish a save area within the program 18 full words long. At the entry point of the program, use a SAVE (14,12) macro. Establish the base register, then load the contents of register 13 into the second word of the save area. Then load the address of the first byte of the save area into register 13 and leave it there for the remainder of the program. Whenever a FATS subroutine is to be called, use the standard CALL macro, enclosing the list of argument names in parentheses. For array or character string arguments, the argument name used should refer to the *first byte* of the array or character string. For calling subroutine SAVE, use the VL option of the CALL macro. At the end of the program, re-load register 13 with its original contents (stored in the second word of the save area), then terminate with a RETURN (14,12) macro. The following example outlines the essentials of the procedure:

```
MAIN    START   0
        SAVE    (14,12)
        BALR    10,0
        USING   *,10
        ST      13,SAVEAREA+4
        LA      13,SAVEAREA
                .
                .
                .
        CALL    TURT,(ONE,TWO,THREE,FOUR)
                .
                .
                .
        CALL    STAB,(NOTABS,TABSTOPS)
                .
                .
                .
        CALL    SAVE,(ONE),VL
```

```
                    .
                    .
                    .
          CALL      PRNT, ( Y, X, NCHAR, CHARCTRS, SPACE )
                    .
                    .
                    .
          CALL      SAVE, (ONE, SAVECODE), VL
                    .
                    .
                    .
          CALL      PASS
                    .
                    .
                    .
          L         13, SAVEAREA+4
          RETURN    ( 14,12 )
ONE       DC        F'1'
TWO       DC        F'2'
THREE     DC        F'3'
FOUR      DC        F'4'
                    .
                    .
                    .
NOTABS    DC        F'5'
TABSTOPS  DC        E'1.550, 2.655, 4.800, 6.125, 7.523'
                    .
                    .
                    .
Y         DS        E
X         DS        E
SPACE     DS        E
SAVECODE  DS        F
                    .
                    .
                    .
NCHAR     DC        F'13'
CHARCTRS  DC        C 'TITLE OF PAGE'
                    .
                    .
                    .
SAVEAREA  DS        18F
          END       MAIN
```

The linkage editor must have available (in addition to the FATS modules) the FORTRAN library. To accomplish this, include the following JCL statement for the link edit step:

```
//LKED.SYSLIB   DD   DSNAME=SYS1.FORTLIB,DISP=OLD
```

(The DSNAME for the FORTRAN library may vary, depending on the installation; also, if PARM=(NCAL) is specified on the link edit EXEC statement, it must be overridden).

Assembler language users should consult the *Assembler Language* manual for full descriptions of constant types, *Supervisor and Data Management Macro Instructions* for descriptions of the SAVE, CALL, and RETURN macros, and the *Assembler (F) Programmer's Guide* for further information on linkage to FORTRAN subroutines.

COBOL

The following, somewhat incomplete, information on COBOL has been obtained from various manuals and other sources, but has not been actually tested. COBOL external linkage conventions are identical to those for FORTRAN, so calling FATS subroutines from COBOL should present no special problems. That is, the FORTRAN STATEMENT:

```
CALL   PRNT (Y, X, NUM, NTEXT, SPACE)
```

sets up the same linkage as the COBOL statements

```
ENTER  LINKAGE.
CALL 'PRNT' USING Y, X, NUM, NTEXT, SPACE.
ENTER  COBOL.
```

Proper data item types must be used for the parameters passed as arguments to the FATS subroutine. FORTRAN INTEGER*4, REAL*4, and REAL*8 variables correspond to COBOL COMPUTATIONAL (4 bytes), COMPUTATIONAL-1 (4 bytes), and COMPUTATIONAL-2 (8 bytes) elementary items, respectively. A FORTRAN single-dimensioned array corresponds to a COBOL elementary item with an OCCURS clause. FORTRAN A1- or A4- formatted words or arrays correspond to COBOL alphanumeric items.

Both the FATS modules and the FORTRAN library must be made available to the linkage editor.

COBOL users should refer to the appropriate manuals for further information.

Appendix C
Program Debugging

This Appendix briefly describes two debugging aids which may be helpful to FATS users. One is a subroutine called BYPASS which can be of help in locating the cause of program execution errors. The other is a program called LINSIM which allows users with Cal Comp* plotter capability to obtain a preview of the Linotron's printed output pages before a Linotron tape is sent to GPO. Both BYPASS and LINSIM are independent of FATS and have applications beyond their usefullness for FATS program debugging.

BYPASS

Programs for Linotron photocomposition utilizing FATS generally contain many CALL statements referencing FATS subroutines; therefore, a large fraction of such a program's execution time is spent within the called subroutines. Execution errors frequently occur within FATS routines as a result of miscoded, miscomputed, or otherwise invalid arguments having been passed from the user's program. Such error conditions produce a distasteful diagnostic messages and frequently cause abnormal termination of execution. For many types of error conditions the diagnostic message may seem rather opaque to many Fortran programmers – it may list addresses, register contents, the PSW, or similar data but shows nothing that relates to the original program coding.

In such situations BYPASS can aid the user by indicating the specific subroutine call which lead to the error condition. Once this is established, the arguments appearing in the indicated CALL statement should be carefully checked and corrected where necessary. Usually this simple procedure leads to a rapid resolution of the problem.

To utilize BYPASS, a CALL BYPASS statement is placed near the beginning of the user's main program. This call disables the normal System error facility and substitutes a special error-handling routine. This routine, called $EXIT, is automatically given control whenever errors resulting in program interruptions (e.g. exponent overflows, addressing exceptions, etc.) occur. $EXIT writes a few lines of information regarding the type of error, then calls a System routine which generates and writes a "save area trace". This trace follows the most recently executed subroutine call route, beginning with the user's main program. For each call, the internal statement number (ISN) of the calling statement, the name of the called subroutine, and other data is provided. A quick inspection of this trace usually reveals the FATS subroutine call causing the problem. An investigation of the arguments involved can then begin.

LINSIM

LINSIM is an independent program for checking out Linotron tapes before they are sent to GPO. LINSIM reads and interprets the codes on the Linotron tape and utilizes Cal Comp's basic softwear package to produce a *plot tape*. This plot tape serves as the control tape for a Cal Comp offline digital pen plotter which will then draw a simulation of the Linotron's printed output pages. Naturally, the wide range of type fonts available with the Linotron cannot be adequately represented by the type of line drawing which the Cal Comp pen plotters produce. LINSIM therefore represents each printed character by an appropriately sized and labelled rectangle. The Linotron's horizontal and vertical rules can, of course be easily represented. The plotted simulation therefore indicates well the overall layout and spacing of the page to be composed, but cannot show the appearance of the type.

As LINSIM produces the plot tape, it checks for errors in the coding on the Linotron tape, and writes

*California Computer Products, Inc.

appropriate messages when errors are encountered. For each error message, an indicator will also appear on the plotted simulation of the page, so that the user can more readily locate the source of the error in his original program coding.

Source or object modules for BYPASS and LINSIM are available from the Nautical Almanac Office, U. S. Naval Observatory.

Appendix D
Grid Diagrams

The diagrams on the following pages were designed to assist in the selection of type faces and to facilitate the access of individual characters. All nine grids are accomodated by FATS. The presentation of the diagrams is logically ordered and does not resemble the physical arrangement of the character images on the grids. Linotron composition can proceed once the grid, zone, and shift mode configuration has been established (see subroutines TURT, STRT, and TYPE); individual characters to be printed are then located by means of their grid location characters (automatically generated by subroutine entries FNUM, INUM, PRNT, etc.). The grid location characters are found at the left extremity of each of the grid diagrams. They are presented as EBCDIC characters for which the punch-card code is likewise given (suitable as arguments for FATS subroutine entries SCAR or MLTS). The numeral which appears to the right of each grid character is its relative width. The resulting printed width depends on the type size specification; the printed width in Linotron units is the product of the type size in points and the relative width.

GRID 1

Century Expanded

Grid Location Character (EBCDIC Translation)	ZONE 1 Roman				ZONE 2 Bold				ZONE 3 Italic			
	UNSHIFT		SHIFT		UNSHIFT		SHIFT		UNSHIFT		SHIFT	
(12-3-8)	.	4	)	6	.	5			z	9	)	7
(12-4-8)	;	5	:	5	;	6	:	6	;	6	:	6
(12)	[	6	&	14			&	15	[	7		
(11-3-8)			–	9	..	9	U	12				
(11)	-	6	—	18	-	6			J	8		
(0-1)	]	6							]	7	/	6
(0-3-8)	,	4	(	6	,	5			&	11	(	7
(0-4-8)	$	9	%	15	$	9			$	9		
(0-6-8)	!	6	?	8	!	6	?	8	!	8	?	7
(0-7-8)			*	9								
(12-0)	'	4	‘	4	'	5	‘	5				
(12-1)	a	9	A	14	a	10	A	13	*a*	10	*A*	14
(12-2)	b	10	B	13	b	10	B	14	*b*	9	*B*	13
(12-3)	c	8	C	13	c	9	C	13	*c*	8	*C*	13
(12-4)	d	10	D	14	d	10	D	14	*d*	10	*D*	14
(12-5)	e	9	E	14	e	9	E	14	*e*	8	*E*	14
(12-6)	f	7	F	13	f	7	F	13	*f*	6	*F*	13
(12-7)	g	10	G	13	g	10	G	14	*g*	9	*G*	13
(12-8)	h	10	H	15	h	11	H	15	*h*	10	*H*	15
(12-9)	i	5	I	8	i	6	I	8	*i*	6	*I*	8
(11-1)	j	6	J	9	j	6	J	9	*j*	6	*J*	10
(11-2)	k	10	K	14	k	11	K	14	*k*	10	*K*	14
(11-3)	l	5	L	13	l	6	L	13	*l*	6	*L*	12
(11-4)	m	15	M	16	m	16	M	17	*m*	16	*M*	17
(11-5)	n	10	N	15	n	11	N	14	*n*	11	*N*	15
(11-6)	o	9	O	13	o	10	O	14	*o*	9	*O*	13
(11-7)	p	10	P	12	p	10	P	13	*p*	10	*P*	13
(11-8)	q	10	Q	13	q	10	Q	14	*q*	9	*Q*	13
(11-9)	r	8	R	14	r	8	R	14	*r*	8	*R*	14
(0-2)	s	8	S	11	s	8	S	11	*s*	8	*S*	11
(0-3)	t	7	T	12	t	7	T	12	*t*	6	*T*	12
(0-4)	u	10	U	15	u	11	U	14	*u*	11	*U*	15
(0-5)	v	10	V	14	v	9	V	14	*v*	9	*V*	13
(0-6)	w	14	W	18	w	14	W	18	*w*	13	*W*	17
(0-7)	x	10	X	14	x	10	X	14	*x*	10	*X*	14
(0-8)	y	10	Y	14	y	10	Y	14	*y*	10	*Y*	13
(0-9)	z	8	Z	12	z	9	Z	12	*z*	8	*Z*	13
(0)	0	9			0	9	T	10	*0*	9	Y	11
(1)	1	9	A	11	1	9	K	11	*1*	9	E	11
(2)	2	9	B	11	2	9	L	10	*2*	9	F	10
(3)	3	9	C	10	3	9	M	13	*3*	9	G	11
(4)	4	9	D	11	4	9	N	12	*4*	9	H	12
(5)	5	9			5	9	O	10	*5*	9	I	6
(6)	6	9			6	9	P	10	*6*	9	&	14
(7)	7	9			7	9	Q	10	*7*	9	V	11
(8)	8	9			8	9	R	11	*8*	9	W	15
(9)	9	9			9	9	S	9	*9*	9	X	11

GRID 2

Naval Observatory

Grid Location Character (EBCDIC Translation)		ZONE 1 Bell Gothic				ZONE 2 Spartan Book				ZONE 3 Astr Symbols			
		UNSHIFT		SHIFT		UNSHIFT		SHIFT		UNSHIFT		SHIFT	
.	(12-3-8)	////	18	5	9		4			●	15	○	15
<	(12-4-8)	▬	18	,	9	;	5	:	5	κ	10	{	9
&	(12)	2	9	9	9			9	9	◑	15		
$	(11-3-8)			..	9	(	5	*	8				
-	(11)		9	7	9	-	5			λ	10		
/	(0-1)	3	9							◑	15	◐	15
,	(0-3-8)	▭	18	4	9	,	4			◐	15	◑	15
%	(0-4-8)	₀	9	6	9	/	6			♀	7		
>	(0-6-8)	*	9	0	9	)	5	0	9	♍	10	∓	12
?	(0-7-8)			*	10								
	(12-0)	·	9	‿	18		9	'	4				
A	(12-1)	a	9	A	9	a	10	A	11	α	11	∫	9
B	(12-2)	b	9	B	9	b	10	B	9	π	12	A	12
C	(12-3)	c	9	C	9	c	8	C	11	ξ	9	∫	9
D	(12-4)	d	9	D	9	d	10	D	11	♌	15	Γ	11
E	(12-5)	e	9	E	9	e	9	E	8	♆	18	•	9
F	(12-6)	f	9	F	9	f	6	F	8	δ	9	Δ	12
G	(12-7)	g	8	G	9	g	10	G	12	ϵ	8	⌊	9
H	(12-8)	h	9	H	9	h	9	H	12	ζ	8	⌋	9
I	(12-9)	i	9	I	9	i	4	I	5	♉	15	⁚	9
J	(11-1)	j	9	J	9	j	4	J	7	η	11	}	9
K	(11-2)	k	9	K	9	k	8	K	10	θ	9	Θ	14
L	(11-3)	l	9	L	9	l	4	L	7	⌊	9	⌋	9
M	(11-4)	m	9	M	9	m	14	M	14	♎	9	Σ	12
N	(11-5)	n	9	N	9	n	9	N	12	ρ	10	♎	9
O	(11-6)	o	9	O	9	o	9	O	13	γ	10	″	9
P	(11-7)	p	9	P	9	p	10	P	9	♀	5	♑	7
Q	(11-8)	q	9	Q	9	q	10	Q	13	♄	15	*	9
R	(11-9)	r	9	R	9	r	6	R	10	♇	15	°	9
S	(0-2)	s	9	S	9	s	7	S	9	β	11	{	9
T	(0-3)	t	9	T	9	t	5	T	8	♂	15	′	9
U	(0-4)	u	9	U	9	u	9	U	11	□	15	°	9
V	(0-5)	v	9	V	9	v	9	V	11	ϖ	15	✧	12
W	(0-6)	w	9	W	9	w	13	W	15	♁	15	*	9
X	(0-7)	x	9	X	9	x	8	X	11	ω	12	∂	9
Y	(0-8)	y	9	Y	9	y	8	Y	10	♐	15	″	9
Z	(0-9)	z	9	Z	9	z	8	Z	10	μ	12	√	18
0	(0)	0	9			0	9	7	9	~	12	◐	15
1	(1)	1	9	⁀	18	1	9	[	5	♈	15	·	6
2	(2)	2	9	″	9	2	9	]	5	☾	14	ϕ	12
3	(3)	3	9	1	9	3	9	1	9	☉	15	∘	9
4	(4)	4	9	8	9	4	9	8	9	☿	15	•	6
5	(5)	5	9			5	9	2	9	♀	15	}	9
6	(6)	6	9			6	9	3	9	⊕	15	±	12
7	(7)	7	9			7	9	4	9	♂	15	τ	8
8	(8)	8	9			8	9	5	9	♅	15	ψ	11
9	(9)	9	9			9	9	6	9	+	12	=	13

GRID 3

Superiors / Inferiors / Math & Greek

Grid Location Character (EBCDIC Translation)	ZONE 1 Superiors				ZONE 2 Inferiors				ZONE 3 Math & Greek			
	UNSHIFT		SHIFT		UNSHIFT		SHIFT		UNSHIFT		SHIFT	
(12-3-8)		4		4		4			+	12	7/8	15
(12-4-8)		4		4		4		4	κ	10	'	6
(12)		4	&	9			&	9	1/8	15		
(11-3-8)				9		9		9				
(11)		4		4		4			λ	10		
(0-1)		4							3/8	15	2/3	15
(0-3-8)		4		4		4			1/3	15	5/8	15
(0-4-8)	$	6	%	12	$	6			∓	12		
(0-6-8)	×	9	+	9	×	9	+	9	±	12	3/4	15
(0-7-8)				9								
(12-0)		4		4		4		4				
(12-1)	a	7	A	8	a	7	A	8	α	11	<	15
(12-2)	b	6	B	8	b	6	B	8	π	12	Φ	14
(12-3)	c	6	C	8	c	6	C	8	ξ	9	Ξ	14
(12-4)	d	7	D	8	d	7	D	8	γ	10	Γ	11
(12-5)	e	6	E	8	e	6	E	8	5/8	16	5/8	16
(12-6)	f	4	F	8	f	4	F	8	δ	9	Δ	12
(12-7)	g	6	G	8	g	6	G	8	ε	8	≈	15
(12-8)	h	7	H	9	h	7	H	9	ζ	8	~	15
(12-9)	i	4	I	5	i	4	I	5	1/2	16	1/2	16
(11-1)	j	4	J	6	j	4	J	6	η	11	∞	12
(11-2)	k	7	K	8	k	7	K	8	θ	9	Θ	14
(11-3)	l	4	L	7	l	4	L	7	ι	6	°	6
(11-4)	m	10	M	10	m	10	M	10	σ	12	Σ	12
(11-5)	n	7	N	8	n	7	N	8	ρ	10	Υ	13
(11-6)	o	6	O	8	o	6	O	8	3/4	16	3/4	16
(11-7)	p	7	P	8	p	7	P	8	÷	12	×	12
(11-8)	q	6	Q	8	q	6	Q	8	1/8	16	1/8	16
(11-9)	r	6	R	8	r	6	R	8	7/8	16	7/8	16
(0-2)	s	5	S	7	s	5	S	7	β	11	>	15
(0-3)	t	4	T	7	t	4	T	7	1/3	16	1/3	16
(0-4)	u	7	U	8	u	7	U	8	1/4	16	1/4	16
(0-5)	v	6	V	8	v	6	V	8	ο	9	Ψ	15
(0-6)	w	9	W	11	w	9	W	11	3/8	16	3/8	16
(0-7)	x	8	X	8	x	8	X	8	ν	9	Ω	11
(0-8)	y	7	Y	8	y	7	Y	8	2/3	16	2/3	16
(0-9)	z	6	Z	7	z	6	Z	7	μ	12	≅	15
(0)	0	6			0	6		4	3/4	16	1/4	15
(1)	1	6		9	1	6		9	—	12	χ	10
(2)	2	6	"	9	2	6	"	9	1/8	16	φ	12
(3)	3	6		6	3	6		6	3/8	16	ω	12
(4)	4	6		9	4	6		9	5/8	16	ψ	11
(5)	5	6			5	6	(	4	7/8	16	Λ	13
(6)	6	6			6	6	)	4	1/3	16	1/2	15
(7)	7	6			7	6	[	4	2/3	16	τ	8
(8)	8	6			8	6	]	4	1/4	16	υ	9
(9)	9	6			9	6	%	12	1/2	16	=	12

GRID 4

Spartan Heavy / Trade Gothic

Grid Location Character (EBCDIC Translation)	ZONE 1 Spartan Heavy		ZONE 2 Trade Gothic		ZONE 3 Trade Gothic Bold	
	UNSHIFT	SHIFT	UNSHIFT	SHIFT	UNSHIFT	SHIFT
. (12-3-8)	. 5	) 6	4		– 9	) 5
< (12-4-8)	; 5	: 5	; 4	: 4	; 4	: 4
& (12)	[6	& 11		& 10	[5	
$ (11-3-8)		.. 9	§ 8	 18		
– (11)	- 6	/ 6	- 4		- 4	
/ (0-1)	] 6				] 5	/ 6
, (0-3-8)	, 5	(6	, 4		% 14	(5
% (0-4-8)	$ 10	% 16	$ 8		$ 8	
> (0-6-8)	! 5	? 9	! 6	? 7	! 5	? 7
? (0-7-8)		* 8				
(12-0)	´ 4	´ 4	' 4	' 4		
A (12-1)	a 10	A 12	a 8	A 8	a 8	A 8
B (12-2)	b 10	B 10	b 8	B 9	b 8	B 9
C (12-3)	c 7	C 11	c 7	C 9	c 8	C 9
D (12-4)	d 10	D 11	d 8	D 9	d 8	D 9
E (12-5)	e 9	E 9	e 7	E 8	e 8	E 8
F (12-6)	f 6	F 8	f 5	F 7	f 5	F 8
G (12-7)	g 10	G 13	g 8	G 9	g 8	G 9
H (12-8)	h 10	H 12	h 8	H 9	h 8	H 9
I (12-9)	i 5	I 5	i 4	I 4	i 4	I 5
J (11-1)	j 5	J 8	j 4	J 5	j 4	J 5
K (11-2)	k 9	K 10	k 7	K 8	k 8	K 8
L (11-3)	l 5	L 7	l 4	L 7	l 4	L 7
M (11-4)	m 15	M 14	m 12	M 11	m 12	M 11
N (11-5)	n 10	N 13	n 8	N 9	n 8	N 9
O (11-6)	o 9	O 13	o 7	O 9	o 8	O 9
P (11-7)	p 10	P 9	p 8	P 8	p 8	P 9
Q (11-8)	q 10	Q 13	q 8	Q 9	q 8	Q 9
R (11-9)	r 6	R 10	r 5	R 9	r 6	R 9
S (0-2)	s 7	S 9	s 7	S 8	s 7	S 8
T (0-3)	t 6	T 8	t 5	T 7	t 5	T 7
U (0-4)	u 10	U 12	u 8	U 9	u 8	U 9
V (0-5)	v 9	V 11	v 7	V 8	v 7	V 8
W (0-6)	w 14	W 16	w 10	W 11	w 10	W 12
X (0-7)	x 9	X 11	x 6	X 8	x 7	X 8
Y (0-8)	y 9	Y 10	y 6	Y 8	y 7	Y 8
Z (0-9)	z 8	Z 10	z 6	Z 8	z 7	Z 8
0 (0)	0 10		0 8	/ 6	0 8	— 18
1 (1)	1 10	° 6	1 8	† 9	1 8	' 4
2 (2)	2 10	' 6	2 8	‡ 9	2 8	' 4
3 (3)	3 10	– 9	3 8	– 9	3 8	¶ 9
4 (4)	4 10	— 18	4 8	— 18	4 8	@ 18
5 (5)	5 10		5 8	[5	5 8	* 8
6 (6)	6 10		6 8	] 5	6 8	& 10
7 (7)	7 10		7 8	(5	7 8	. 4
8 (8)	8 10		8 8	) 5	8 8	. 4
9 (9)	9 10		9 8	% 14	9 8	# 14

GRID 5

Special Times Roman

Grid Location Character (EBCDIC Translation)		ZONE 1 Times Roman				ZONE 2 Times Roman Bold				ZONE 3 Spartan Heavy			
		UNSHIFT		SHIFT		UNSHIFT		SHIFT		UNSHIFT		SHIFT	
.	(12-3-8)	.	5	)	6	.	5			.	5	)	6
<	(12-4-8)	;	6	:	6	;	6	:	6	:	5	:	5
&	(12)	[	6	&	15			&	15	½	15		
$	(11-3-8)			—	18	..	9	°	6				
–	(11)	-	6	-	9	-	6			&	11		
/	(0-1)	]	6							¼	15	/	6
,	(0-3-8)	,	6	(	6	,	6			%	16	(	6
%	(0-4-8)	$	9	%	16	$	9			$	10		
>	(0-6-8)	×	15	?	9	"	9	?	9	¾	15	:	9
?	(0-7-8)			*	9								
	(12-0)	'	6	`	6	'	6	`	6				
A	(12-1)	a	9	A	14	a	9	A	13	a	10	A	12
B	(12-2)	b	10	B	13	b	10	B	12	b	10	B	10
C	(12-3)	c	9	C	13	c	8	C	13	c	7	C	11
D	(12-4)	d	10	D	14	d	10	D	13	d	10	D	11
E	(12-5)	e	9	E	13	e	8	E	12	e	9	E	9
F	(12-6)	f	7	F	12	f	6	F	11	f	6	F	8
G	(12-7)	g	9	G	14	g	9	G	14	g	10	G	13
H	(12-8)	h	10	H	15	h	10	H	14	h	10	H	12
I	(12-9)	i	5	I	7	i	5	I	7	i	5	I	5
J	(11-1)	j	6	J	8	j	6	J	9	j	5	J	8
K	(11-2)	k	10	K	15	k	10	K	14	k	9	K	10
L	(11-3)	l	5	L	13	l	5	L	12	l	5	L	7
M	(11-4)	m	15	M	17	m	15	M	17	m	15	M	14
N	(11-5)	n	10	N	15	n	10	N	13	n	10	N	13
O	(11-6)	o	10	O	14	o	9	O	14	o	9	O	13
P	(11-7)	p	10	P	11	p	10	P	11	p	10	P	9
Q	(11-8)	q	10	Q	14	q	10	Q	14	q	10	Q	13
R	(11-9)	r	7	R	13	r	8	R	13	r	6	R	10
S	(0-2)	s	8	S	11	s	7	S	10	s	7	S	9
T	(0-3)	t	6	T	12	t	6	T	12	t	6	T	8
U	(0-4)	u	10	U	15	u	10	U	13	u	10	U	12
V	(0-5)	v	10	V	14	v	9	V	13	v	9	V	11
W	(0-6)	w	14	W	18	w	13	W	18	w	14	W	16
X	(0-7)	x	10	X	14	x	9	X	13	x	9	X	11
Y	(0-8)	y	10	Y	14	y	9	Y	13	y	9	Y	10
Z	(0-9)	z	9	Z	12	z	8	Z	12	z	8	Z	10
0	(0)	0	9			0	9	/	6	0	10	<	15
1	(1)	1	9	+	15	1	9	(	6	1	10	†	9
2	(2)	2	9	=	15	2	9	)	6	2	10	'	4
3	(3)	3	9	–	15	3	9	*	9	3	10	—	18
4	(4)	4	9	±	15	4	9	'	6	4	10	-	9
5	(5)	5	9			5	9	#	14	5	10	*	8
6	(6)	6	9			6	9	α	11	6	10	-	6
7	(7)	7	9			7	9	β	11	7	10	,	5
8	(8)	8	9			8	9	γ	10	8	10	>	15
9	(9)	9	9			9	9	‰	15	9	10	§	9

GRID 6

NRL Grid

Grid Location Character (EBCDIC Translation)	ZONE 1 Math Symbols		ZONE 2 Times Roman Italic		ZONE 3 Times Roman	
	UNSHIFT	SHIFT	UNSHIFT	SHIFT	UNSHIFT	SHIFT
. (12-3-8)	+ 15	⟨ 9	σ 12		. 6	[6
< (12-4-8)	− 15	÷ 15	τ 8	μ 12	(6	) 6
& (12)	= 15	≤ 15		ν 9	- 6	
$ (11-3-8)		⟩ 9	φ 12	π 12		
– (11)	× 15	≥ 15	χ 10		Φ 14	
/ (0-1)	± 15				, 6	: 6
, (0-3-8)	∓ 15	> 15	ψ 11		Ω 14	; 6
% (0-4-8)	‡ 9	< 15	ω 12		‘ 6	
> (0-6-8)	ℓ 12	! 6	→ 18	ξ 9	’ 6	] 6
? (0-7-8)		· 9				
(12-0)	~ 15	° 7	← 18	ρ 10		
A (12-1)	0̊ 7	0 7	*a* 10	*A* 13	a 9	A 14
B (12-2)	1̊ 7	1 7	*b* 10	*B* 12	b 10	B 13
C (12-3)	2̊ 7	2 7	*c* 9	*C* 13	c 9	C 13
D (12-4)	3̊ 7	3 7	*d* 10	*D* 14	d 10	D 14
E (12-5)	4̊ 7	4 7	*e* 9	*E* 12	e 9	E 13
F (12-6)	5̊ 7	5 7	*f* 6	*F* 12	f 7	F 12
G (12-7)	6̊ 7	6 7	*g* 10	*G* 14	g 9	G 14
H (12-8)	7̊ 7	7 7	*h* 10	*H* 15	h 10	H 15
I (12-9)	8̊ 7	8 7	*i* 5	*I* 7	i 5	I 7
J (11-1)	9̊ 7	9 7	*j* 5	*J* 9	j 6	J 8
K (11-2)	' 6	*o* 7	*k* 9	*K* 14	k 10	K 15
L (11-3)	⩾ 15	1 7	*l* 5	*L* 12	l 5	L 13
M (11-4)	⩽ 15	2 7	*m* 15	*M* 17	m 15	M 17
N (11-5)	∝ 15	3 7	*n* 10	*N* 15	n 10	N 15
O (11-6)	| 4	4 7	*o* 10	*O* 14	o 10	O 14
P (11-7)	∞ 12	5 7	*p* 10	*P* 12	p 10	P 11
Q (11-8)	/ 4	6 7	*q* 10	*Q* 14	q 10	Q 14
R (11-9)	′ 4	7 7	*r* 7	*R* 13	r 7	R 13
S (0-2)	1/2 18	8 7	*s* 8	*S* 11	s 8	S 11
T (0-3)	1/2 18	9 7	*t* 6	*T* 11	t 6	T 12
U (0-4)	. 4	, 4	*u* 10	*U* 15	u 10	U 15
V (0-5)	· 4	‘ 4	*v* 9	*V* 13	v 10	V 14
W (0-6)	† 9	¯ 9	*w* 13	*W* 17	w 14	W 18
X (0-7)	Å 14	+ 9	*x* 9	*X* 13	x 10	X 14
Y (0-8)	∂ 9	¸ 7	*y* 9	*Y* 12	y 10	Y 14
Z (0-9)	≡ 15	˚ 7	*z* 8	*Z* 12	z 9	Z 12
0 (0)	∇ 12		*0* 9	*α* 11	0 9	& 15
1 (1)	ℏ 10	{ 9	*1* 9	*β* 11	1 9	? 9
2 (2)	1/2 18	} 9	*2* 9	*γ* 10	2 9	% 17
3 (3)	3/2 18	≠ 15	*3* 9	*δ* 9	3 9	Γ 11
4 (4)	5/2 18	≃ 15	*4* 9	*ε* 8	4 9	Δ 12
5 (5)	7/2 18		*5* 9	*ζ* 8	5 9	Π 14
6 (6)	9/2 18		*6* 9	*η* 11	6 9	Ψ 15
7 (7)	11/2 18		*7* 9	*θ* 9	7 9	/ 6
8 (8)	13/2 18		*8* 9	*κ* 10	8 9	Σ 12
9 (9)	15/2 18		*9* 9	*λ* 10	9 9	* 9

GRID 7

Crystal Data

Grid Location Character (EBCDIC Translation)	ZONE 1 Bodoni Book Roman		ZONE 2 Bodoni Bold Roman		ZONE 3 Inferiors & Greek	
	UNSHIFT	SHIFT	UNSHIFT	SHIFT	UNSHIFT	SHIFT
. (12-3-8)	. 6	] 5	. 6		4	= 12
< (12-4-8)	; 6	: 6	(6	) 6	− 12	≡ 15
& (12)	(5	& 15		/ 6	+ 12	
$ (11-3-8)		½ 9	; 7	: 7		
− (11)	- 5	¼ 9	- 6		9	
/ (0-1)	) 5				4	< 15
, (0-3-8)	, 6	[5	, 6		4	> 15
% (0-4-8)	@ 16	! 6	· 6		9	
> (0-6-8)	/ 6	… 18	[6	] 6	± 9	□ 15
? (0-7-8)		? 9				
(12-0)	' 5	' 5	+ 12	− 12		
A (12-1)	a 9	A 13	a 10	A 13	a 7	α 11
B (12-2)	b 10	B 12	b 12	B 14	b 6	β 11
C (12-3)	c 9	C 13	c 10	C 13	c 6	γ 10
D (12-4)	d 10	D 13	d 12	D 15	d 7	δ 9
E (12-5)	e 9	E 12	e 10	E 14	e 6	ϵ 8
F (12-6)	f 6	F 11	f 7	F 13	f 4	Δ 12
G (12-7)	g 9	G 13	g 11	G 14	g 6	ü 9
H (12-8)	h 10	H 14	h 12	H 16	h 7	ø 9
I (12-9)	i 5	I 7	i 6	I 8	i 4	ä 9
J (11-1)	j 6	J 8	j 7	J 10	j 4	θ 9
K (11-2)	k 10	K 13	k 12	K 15	k 7	κ 10
L (11-3)	l 5	L 12	l 6	L 13	l 4	λ 10
M (11-4)	m 15	M 16	m 17	M 17	m 10	μ 12
N (11-5)	n 10	N 14	n 12	N 14	n 7	ν 9
O (11-6)	o 9	O 14	o 11	O 15	o 6	α 12
P (11-7)	p 10	P 12	p 12	P 13	p 7	β 12
Q (11-8)	q 10	Q 14	q 12	Q 15	q 6	γ 11
R (11-9)	r 7	R 13	r 9	R 15	r 6	δ 10
S (0-2)	s 8	S 11	s 8	S 11	s 5	ϵ 9
T (0-3)	t 6	T 12	t 7	T 13	t 4	μ 13
U (0-4)	u 10	U 13	u 12	U 15	u 7	ν 10
V (0-5)	v 9	V 13	v 9	V 14	v 6	λ 11
W (0-6)	w 13	W 18	w 13	W 18	w 9	κ 11
X (0-7)	x 9	X 13	x 11	X 14	x 8	Δ 13
Y (0-8)	y 9	Y 13	y 10	Y 13	y 7	ł 5
Z (0-9)	z 8	Z 11	z 9	Z 12	z 6	ζ 8
0 (0)	0 9		0 12	0̄ 9	0 6	± 12
1 (1)	1 9	° 6	1 12	1̄ 9	1 6	√ 18
2 (2)	2 9	' 6	2 12	2̄ 9	2 6	~ 15
3 (3)	3 9	" 6	3 12	3̄ 9	3 6	Å 13
4 (4)	4 9	* 9	4 12	4̄ 9	4 6	ë 9
5 (5)	5 9		5 12	5̄ 9	5 6	è 9
6 (6)	6 9		6 12	6̄ 9	6 6	ê 9
7 (7)	7 9		7 12	7̄ 9	7 6	ë 9
8 (8)	8 9		8 12	8̄ 9	8 6	ä 9
9 (9)	9 9		9 12	9̄ 9	9 6	ö 9

GRID 8

Helvetica

Grid Location Character (EBCDIC Translation)		ZONE 1 Helvetica Roman				ZONE 2 Helvetica Bold				ZONE 3 Helvetica Italic Bold			
		UNSHIFT		SHIFT		UNSHIFT		SHIFT		UNSHIFT		SHIFT	
	(12-3-8)	.	5	)	6	.	5			–	9	Ø	10
	(12-4-8)	:	6	:	6	;	6	:	6	;	6	:	6
&	(12)	[	6	&	12			&	13		6		
$	(11-3-8)			....	18	§	10	•	8				
–	(11)	-	7	/	8	-	7			-	12		
/	(0-1)	]	6								6	/	8
,	(0-3-8)	,	5	(	6	,	5			%	16	®	18
%	(0-4-8)	$	10	%	16	$	10			$	10		
>	(0-6-8)	!	6	?	11	!	6	?	11	!	6	?	10
?	(0-7-8)			•	7								
	(12-0)	'	5	'	5	'	5	'	5				
A	(12-1)	a	10	A	12	a	11	A	13	a	11	A	12
B	(12-2)	b	11	B	13	b	11	B	13	b	11	B	13
C	(12-3)	c	10	C	13	c	10	C	13	c	10	C	13
D	(12-4)	d	11	D	13	d	11	D	13	d	11	D	13
E	(12-5)	e	10	E	12	e	11	E	12	e	11	E	11
F	(12-6)	f	6	F	11	f	6	F	11	f	6	F	10
G	(12-7)	g	11	G	14	g	11	G	14	g	11	G	14
H	(12-8)	h	11	H	13	h	11	H	13	h	11	H	13
I	(12-9)	i	5	I	5	i	5	I	6	i	5	I	6
J	(11-1)	j	5	J	9	j	5	J	10	j	5	J	9
K	(11-2)	k	10	K	12	k	10	K	13	k	10	K	12
L	(11-3)	l	5	L	11	l	5	L	11	l	5	L	10
M	(11-4)	m	16	M	16	m	16	M	16	m	16	M	16
N	(11-5)	n	11	N	13	n	11	N	14	n	11	N	13
O	(11-6)	o	11	O	14	o	11	O	14	o	11	O	14
P	(11-7)	p	11	P	12	p	11	P	12	p	11	P	12
Q	(11-8)	q	11	Q	14	q	11	Q	14	q	11	Q	14
R	(11-9)	r	7	R	13	r	7	R	13	r	7	R	13
S	(0-2)	s	9	S	12	s	10	S	12	s	9	S	11
T	(0-3)	t	6	T	11	t	6	T	11	t	6	T	10
U	(0-4)	u	11	U	13	u	11	U	13	u	11	U	13
V	(0-5)	v	9	V	12	v	10	V	12	v	9	V	11
W	(0-6)	w	14	W	17	w	14	W	17	w	14	W	16
X	(0-7)	x	10	X	12	x	10	X	12	x	9	X	11
Y	(0-8)	y	9	Y	12	y	10	Y	12	y	9	Y	11
Z	(0-9)	z	9	Z	11	z	9	Z	11	z	9	Z	11
0	(0)	0	10			0	10	/	8	0	10	—	18
1	(1)	1	10	•	18	1	10	†	10	1	10	'	5
2	(2)	2	10	.	17	2	10	‡	10	2	10	'	5
3	(3)	3	10	–	9	3	10	–	9	3	10	¶	10
4	(4)	4	10	—	18	4	10	—	18	4	10	@	18
5	(5)	5	10			5	10	[	6	5	10	•	8
6	(6)	6	10			6	10	]	6	6	10	&	7
7	(7)	7	10			7	10	(	6	7	10	.	5
8	(8)	8	10			8	10	)	6	8	10	.	5
9	(9)	9	10			9	10	%	16	9	10	#	14

GRID 9

Census Gothic

Grid Location Character (EBCDIC Translation)		ZONE 1 Spartan Book (Cond)				ZONE 2 Spartan Heavy (Cond)				ZONE 3 Spartan Heavy			
		UNSHIFT		SHIFT		UNSHIFT		SHIFT		UNSHIFT		SHIFT	
.	(12-3-8)	.	5	)	5	.	5			.	5	)	6
<	(12-4-8)	$	9	%	16	$	9	%	16	$	10	%	16
&	(12)	-	5	&	12			&	11	-	6		
$	(11-3-8)			:	5	+	18	:	5				
–	(11)	—	18	..	18	—	18			=	18		
/	(0-1)	(	5							!	5	[	5
,	(0-3-8)	,	5	.	9	,	5			,	5	]	5
%	(0-4-8)		6	__	18		6			(	6		
>	(0-6-8)		9	?	9		9	?	18		9	!	5
?	(0-7-8)			_	9								
	(12-0)	–	9	;	5	–	9	;	5				
A	(12-1)	a	9	A	11	a	8	A	10	a	10	A	12
B	(12-2)	b	9	B	10	b	8	B	9	b	10	B	10
C	(12-3)	c	8	C	10	c	7	C	9	c	7	C	11
D	(12-4)	d	9	D	11	d	8	D	10	d	10	D	11
E	(12-5)	e	8	E	8	e	8	E	7	e	9	E	9
F	(12-6)	f	6	F	8	f	6	F	7	f	6	F	8
G	(12-7)	g	9	G	11	g	8	G	10	g	10	G	13
H	(12-8)	h	9	H	11	h	8	H	10	h	10	H	12
I	(12-9)	i	4	I	5	i	4	I	5	i	5	I	5
J	(11-1)	j	4	J	7	j	4	J	7	j	5	J	8
K	(11-2)	k	9	K	11	k	8	K	9	k	9	K	10
L	(11-3)	l	4	L	8	l	4	L	7	l	5	L	7
M	(11-4)	m	14	M	15	m	12	M	14	m	15	M	14
N	(11-5)	n	9	N	11	n	8	N	11	n	10	N	13
O	(11-6)	o	9	O	11	o	8	O	10	o	9	O	13
P	(11-7)	p	9	P	9	p	8	P	9	p	10	P	9
Q	(11-8)	q	9	Q	11	q	8	Q	10	q	10	Q	13
R	(11-9)	r	7	R	10	r	6	R	9	r	6	R	10
S	(0-2)	s	8	S	9	s	7	S	8	s	7	S	9
T	(0-3)	t	6	T	9	t	6	T	8	t	6	T	8
U	(0-4)	u	9	U	11	u	8	U	10	u	10	U	12
V	(0-5)	v	9	V	11	v	8	V	10	v	9	V	11
W	(0-6)	w	13	W	16	w	12	W	14	w	14	W	16
X	(0-7)	x	9	X	10	x	9	X	10	x	9	X	11
Y	(0-8)	y	9	Y	10	y	8	Y	9	y	9	Y	10
Z	(0-9)	z	8	Z	10	z	7	Z	9	z	8	Z	10
0	(0)	0	9			0	9	#	14	0	10	0	7
1	(1)	1	9	'	4	1	9	'	4	1	10	1	7
2	(2)	2	9	"	9	2	9	"	9	2	10	2	7
3	(3)	3	9	=	9	3	9	@	18	3	10	3	7
4	(4)	4	9	/	6	4	9	/	6	4	10	4	7
5	(5)	5	9			5	9	‡	9	5	10	5	7
6	(6)	6	9			6	9	†	9	6	10	6	7
7	(7)	7	9			7	9	¤	7	7	10	7	7
8	(8)	8	9			8	9	'	5	8	10	8	7
9	(9)	9	9			9	9	¢	9	9	10	9	7

Appendix E

FATS Control Section and Entry Names

The following is an alphabetical list of FATS control section and entry names. FATS users should avoid naming any of their own subprograms, subprogram entry points, or common blocks any of these names.

ANUM	PAGE	TYPE
ASIS	PASS	ZONE
BBUF	PRNT	$CLS
CBUF	RTRN	$CORD
CLER	RULD	$DMP
COMPOZ	RULE	$IWD
DELX	RULX	$NUMBR
DELY	RULY	$OPN
DIMN	SAVE	$OUTPT
DNUM	SCAR	$READ
EBUF	SETABS	$RESET
FNUM	SHFT	$SAVE 1
GRID	SIZE	$SAVE 2
IDEN	SPEC	$SAVE 3
INUM	SPIL	$SHFTR
LINBUF	SQEZ	$SPLT
MARG	STAB	$TOBIN
MLTC	STRT	$1
MLTP	TABB	$2
NOOP	TERM	$3
OODD	TURN	$4
OSET	TURT	$5

Appendix F

FATS Subroutine Summary

	Name	Arguments									Pg.
initialize / terminate	TURT	grid 1 [1]	grid 2 [1]	grid 3 [1]	grid 4 [1]						14
	DIMN	stand. length [2]	user length [2]								15
	STRT	job [1]	frame [1]								16
	PAGE										16
	IDEN	Y [2]	X [2]	turret [1]							16
	TERM										17
	PASS										17
configuration	TURN	orient. [1]									18
	TYPE	turret [1]	zone [1]	shift [1]	size [1]						19
	GRID	turret [1]									19
	ZONE	zone [1]									19
	SHFT	mode [1]									19
	SIZE	points [1]									19
	SQEZ	aspect ratio [1]									20
	SPEC	face code [1]									20
edit and compose	PRNT	Y [2]	X [2]	count [1]	chars. [3]	blank size [2]					21
	MLTP	Y [2]	X [2]	count [1]	chars. [3]	blank size [2]	ΔY [2]	ΔX -	repeats [1]		21
	SCAR	Y [2]	X [2]	turret [1]	zone [1]	shift [1]	char. [4]				21
	MLTC	Y [2]	X [2]	turret [1]	zone [1]	shift [1]	char. [4]	ΔY [2]	ΔX [2]	repeats [1]	21
	FNUM	Y [2]	X [2]	field size [2]	number [2]	no. decimals [1]	sign code [1]	min.lead.[1] digits			24
	DNUM	Y [2]	X [2]	field size [2]	number [5]	no. decimals [1]	sign code [1]	min.lead.[1] digits			24

[1] Integer*4
[2] Real*4
[3] Array of characters in A4 - formatted 4-byte words
[4] Single character in an A1 - formatted 4-byte word
[5] Real*8
[6] Array of Real*4 words

FATS Subroutine Summary

	Name	Arguments									Pg.
edit and compose	INUM	Y [2]	X [2]	field size [2]	number [1]	sign code [1]	min. digits [1]				26
	ANUM	Y [2]	X [2]	field size [2]	count [1]	chars. [3]					27
	RULX	Ystart [2]	Xstart [2]	Xstop [2]	weight [1]						29
	RULY	Ystart [2]	Xstart [2]	Ystop [2]	weight [1]						29
	RULD	Ystart [2]	Xstart [2]	Ystop [2]	Xstop [2]	weight [1]					29
print position	STAB	length [1]	coords. [6]								30
	MARG	left *margin* [2]									30
	RTRN	space [2]									30
	TABB										30
	DELY	ΔY [2]									30
	DELX	ΔX [2]									30
	OSET	Y [2]	X [2]								30
repetition	SAVE	area [1]	error code [1]								32
	SPIL	area [1]									32
	CLER	area [1]									32
line position	BBUF										35
	EBUF	Y [2]	X [2]	position code [1]							35
	CBUF	error code [1]	line width [2]								35
misc.	ASIS	count [1]	chars. [3]								38
	NOOP	no. of blanks [1]									39

[1] Integer*4
[2] Real*4
[3] Array of characters in A4 - formatted 4-byte words
[4] Single character in an A1 - formatted 4-byte word
[5] Real*8
[6] Array of Real*4 words