NATIONAL RADIO ASTRONOMY OBSERVATORY
COMPUTER DIVISION
INTERNAL REPORT

PROGRAM LIBRARY FOR THE GENERAL PURPOSE COMPUTER AT THE NRAO

BY

PETER STUMPFF

Report No. 3
December 1967

I. INTRODUCTION

Until now there has been no systematic documentation of computer programs. The programmer or scientist who developed a program also had to maintain it. It was up to him how he documented it. If somebody had a question about a program he had to contact the author. If somebody wanted to find out whether or not a certain type of program had already been written, he had to ask everyone who might possibly have written such a program.

As long as data processing is not a major task of an institution, this "system" (which actually is no system at all) might work well. However, data processing has become one of the major tasks at the NRAO. The number of jobs submitted to the general purpose computer is increasing rapidly. The number of professional programmers will not increase proportionally; even if it would, the present lack of documentation is no longer acceptable. One reason is that many different main programs (in particular, all telescope oriented programs) consist of many little elements which are similar if not identical. They could be written in the form of subroutines and be incorporated in the library. Everyone who writes a new main program could make use of these subroutines. Another reason is that if a programmer leaves without documenting his programs in the standard way, all of the effort spent during his time here might be lost for the NRAO.

In order to improve this situation we will now start a program library and correspondingly, documentation of programs. The documentation system must fulfill the following conditions:

1. It must be as simple as possible.
2. It must be able to cover as many different types of programs as possible.
3. It must be strictly followed.
4. Exceptions must be allowed since no simple documentation system exists without exceptions. If exceptions occur, they must be described sufficiently.

Instead of starting a philosophical consideration of possible documentation systems, I will introduce one which I have developed step by step in the past, always having been led by experience. I have applied it to my own programming work for several years and have seen it working nicely. Physically, the library will consist of the following:

a) INDEX FILE:

Each program is represented by at least one card which shows:

1. NAME of the program.
2. An IDENTIFICATION CODE.
3. A very short description of the purpose of the program.
4. The "LCD-NUMBER", i.e., the Library Card Deck number of the program. If more than one punched card deck exists (for example, a source deck and an object deck), all these numbers are shown.

The INDEX FILE is divided into problem oriented sections and can be used to find out whether a program exists which deals with a certain problem. If a program has relationship to more than one section, its INDEX CARD will be found in all of these sections.

b) DESCRIPTION FILE:

Each program is represented by a hanging folder. A label on top of the folder contains the NAME and the IDENTIFICATION CODE. Inside the folder is a standardized description of the program followed by the listing of the program.

c) I/O EXAMPLE FILE:

Each program which explicitly uses card reader/punch, printer, and/or tapes is represented by a folder which contains typical examples of data (usually these will be printer listings). Exceptions are programs whose I/O functions are so simple that an example would not be necessary.

d) LCD-FILE:

Each program will be represented by at least one complete punched card deck. Each deck has its own number (LCD-NUMBER).

e) LOG BOOK:

This book is maintained by the computer division librarian. It will show the complete record of the "historical" development of every program which is incorporated into the library.

The files b,c,d, will be in sequential order. The sequential order of b and c is given by the MAIN NUMBER of a program. The MAIN NUMBER is part of the IDENTIFICATION CODE and will be assigned to programs in the sequence of submission to the library. The sequential order in file c is given by the LCD NUMBER. LCD NUMBERS are assigned to the card decks in the sequence of their submission to the library. The sequence of MAIN NUMBER and LCD NUMBER is not necessarily related to problems. The problem relationship is given in the INDEX FILE.

II. DEFINITIONS OF PROGRAMS TO BE INCORPORATED IN THE LIBRARY

In general, we will not try to determine what programs in what categories we may need and then write them. We simply do not have enough people to do so. Instead, we will proceed with programming in the same way as we have in the past with only the following difference. If we start a new program we will investigate which subroutines or parts of the program might possibly be used in future programs. We will then try to incorporate these subroutines or parts in the library. In this way the library will grow haphazardly. The following categories of programs can be incorporated:

a) General purpose SUBROUTINES and FUNCTIONS developed by the Computer Division.
b) General purpose MAIN PROGRAMS developed by the Computer Division.
c) Frequently used MAIN PROGRAMS developed by the Computer Division.
d) Programs falling under a-c developed by persons outside the Computer Division, if they have been accepted by the Computer Division for incorporation in the library. Since we do not have enough programmers, we would probably ask the author of the program to make those changes necessary for incorporation. We would also ask him to give us the description of the program, and we will only accept the program if the description is sufficient. What this means is described in Section IV of this report.

III. NAME AND IDENTIFICATION CODE OF A PROGRAM

Each program shall have a mnemonic NAME which should not consist of more than 8 characters for a MAIN program and not more than 6 characters for a SUBprogram. If possible, one should choose a smaller number of characters than these limits to allow for adding numbers to the NAMES (in case of later versions). If possible, when a FUNCTION or a SUBROUTINE is in double precision the NAME should start with a "D". The IDENTIFICATION CODE has the following format:

NRAO m/s x (y)

Explanation:

NRAO: indicates a program in the NRAO library.

m: MAIN NUMBER of the program. This number is assigned to programs in the sequence of their submission to the library.

s: SUB NUMBER. The first version of the program gets the SUB NUMBER s=1. For each new version the SUB NUMBER will be increased by 1.

x: This will be one of the characters M,S, or F:
- M: MAIN program
- S: SUBROUTINE subprogram
- F: FUNCTION subprogram

y: This will be either a D or a blank:
- D: The program is permanently stored on disk in addition to residence in card form.
- Blank: The program must be added to the job in the form of a card deck.

If a program generates data output on the printer it should print its own NAME and the IDENTIFICATION CODE (omitting the "(y)") at least at the beginning of the output, and if possible, at the top of each page. An easy way to document data output can be found in the description of a special page-skipping routine "PAG(K,L,INFPAG)" which will be incorporated in the library with the IDENTIFICATION CODE "NRAO 36/1 S (D)" and which can be used as a prototype for this purpose.

It may be helpful to give some idea about the way a program is submitted to the library and later developed within the library. If a new programming project is started a NAME should be chosen for it. If you think the program will be permanently stored on a disc, you should check the existing NAMES within that category thus avoiding a NAME that already exists. During the testing stage you may print the NAME on every output, but, of course, no IDENTIFICATION CODE exists during this period. If the development of the program has reached the stage where all formal errors are corrected and where test samples seem to be calculated satisfactorily, then write the description and submit the program (i.e., the description and the card deck) to the Computer Division. When the program has been accepted the MAIN NUMBER will be assigned to the program and the SUB NUMBER will receive the value s=1. If the program generates output, the full IDENTIFICATION CODE must be printed; this requires one last change in the program. A note is made in the LOG BOOK, the punched card deck is stored in the LCD FILE, an INDEX FILE CARD will be filled out, and the description and I/O examples will be filed in the DESCRIPTION FILE and the I/O EXAMPLE FILE, respectively. When this is completed the program exists in the library and can be used by everyone. It is very important that from now on every time a correction or change has to be made a new SUB NUMBER is assigned to the program. Correspondingly, the format in the program which prints the IDENTIFICATION CODE on the output has to be changed. A note to this effect must appear in the LOG BOOK. If the new version is a correction to the previous version and does essentially the same task (so that it is clear that the previous version must never be used again), one might remove the previous version from the library (except in the DESCRIPTION FILE). Of course, one has to be careful. For example,

let us assume a subroutine (s=1) exists with a certain calling sequence. Let us assume further that this program is correct but somebody makes a new version which does a better job and involves a change in the argument list. In this case, as far as the numerical results are concerned, the programs are identical. The old version must not be removed from the library because there might be users who do not want to change their calling main programs. However, if the new version of the program differs from the old one by better timing and/or better accuracy, but not in the calling sequence and the name, the old version might simply be replaced by the new one. In general, I would not suggest that this be done even in such a clear case. Removal of an existing program should be done only if a space problem occurs in the various files, and then only after careful investigation of the consequences that might occur.

IV. PROGRAM DESCRIPTIONS

To make documentation as easy as possible, we have developed standard forms which will hopefully cover the majority of cases. Any description will consist of:

I. Standard form, Page 1
II. Standard form, Page 2
III. Standard form, Page n (n=3,4,...)
IV. Standard form for Variable + Constants, Page n
V. Listing of the source deck

In the appendix, an example is given of the description of a time conversion program. In the following pages I will make some comments on the various items in the standardized description.

1. Standard Form, Page 1

This page contains information about the program in a most concentrated form. The concept of page 1 is based upon the assumption that most programs in the library will be SUBROUTINES or FUNCTIONS. Therefore, page 1 already contains a detailed description of the calling sequence of a subprogram, i.e., just the information needed when calling the routine inside a main program. In many cases the user will not even have to look at the other pages. In the case of most main programs, the definition of input and output replaces the calling sequence of a subprogram. However, there is no easy way of standardizing input-output descriptions in a way similar to the calling sequence description of a subprogram. This must be left to an open formatted description inside the document.

Top Line: Contains the IDENTIFICATION CODE, the DATE of submission of the program to the library, and the NAME of the program.

EXPLAN. OF NAME: Explains in a few sentences the purpose of the program. The characters included in the NAME should appear as underlined capitals. In that way two goals are reached: a) the purpose is described; b) the mnemonic meaning of the NAME is explained.

AUTHOR: The name of the author of the particular version of the program. In other words, if version No. 2 of the program has been made by author B, by changing anything in version No. 1 (author A), then on page 1 of the description of version No. 2 the author would be B.

LANGUAGE: For example, "F4" for FORTRAN IV.

NO. OF STATEMENTS: The sum of all the statements which get an internal statement number during compilation.

MACHINE: For example, "IBM 360/50".

TIMING: Average execution time, or lower and upper limit of execution time. Frequently it might be too difficult to specify this, in which case no information will be given. There may be many main programs where one would say something like "in the particular case.... about 30 min"; or "depends upon the number of input cards, about 45 sec per card". In the case of most mathematical subprograms, however, timing could be done during accuracy tests and then this information could be given.

Since time depends on compiler, and perhaps on the operating system in use, information about this should be added. In the example in the appendix for instance the program was compiled with the G compiler under release 11. Therefore, "(G11)" was added to the timing information.

STORAGE: This can be found from the compilation messages. Compiler name and release no. should be added. In the case of a main program, the storage of all subprograms called has to be included. In the case of a subprogram, the length of other subprograms called is not included.

LIBRARY CARD DECK: This is the LCD NUMBER. Information about the language has to be added. For example:
"(F4S)" means that it is a source deck.
"(BG11)" means that it is an object deck generated under G, release 11.
"(BH11,Ø2)" means that it is an object deck generated under H, release 11, with optimization.

Of course, if more than one deck exists to the same version of the program, their LCD numbers should be listed.

If the program calls other programs from cards, those programs should be included in the library card deck.

SUBPROGRAMS CALLED FROM DISC: All subprograms which are used internally should be listed with their names and NRAO numbers, or with their names only in the case of FORTRAN functions (SIN, COS, etc.) and routines belonging to the scientific subroutine package.

SUBPROGRAMS CALLED FROM CARDS: All subprograms which are used internally should be listed with their names and NRAO numbers. If it is a subprogram which is not incorporated into the library, and therefore has no NRAO number, it need not be listed. However, the total number of subprograms of this category should be mentioned.

INPUT DATA (CARDS, TAPE, BOTH): This, in general, concerns
OUTPUT DATA (CARDS, TAPE, BOTH): main programs. Only some
(DETAILS IN DESCRIPTION): general remarks should be made here, such as: reads telescope tape, autocorrelation receiver, 300' telescope, generates FORTRAN readable 9-track tape, punches basic information on cards.

CALL: The NAME of the subprogram (if it is one) should be written followed by the argument list. The arguments are explained in the table. If the subprogram is a FUNCTION, the name has to appear in the table.
Explanation of the various columns:
DIM'N: Dimension. If the NAME belongs to an array, the dimension specification should appear in this column, within parentheses (as at the beginning of the source program). If it is a single variable or a constant, the column is left blank. If one wishes to explain the various elements of the array, one should write:

NAME	DIM'N	EXPLANATION
ARR	(2,2)	array so-and-so
ARR	1,1	cos(e)
ARR	2,1	sin(e)
ARR	1,2	-sin(e)
ARR	2,2	cos(e)

In other words, if the dimension is presented within parentheses it describes the array in general (number of lines and number of columns). If the dimension is without parentheses, a particular element is described.

TYPE: For example: I4 for INTEGER, single precision
R4 for REAL, single precision
R8 for REAL, double precision, and so on.

UNITS: It is important for the user to know which units are required or generated. We will use the following abbreviations:

RAD = radian
REV = revolutions
H = hours
M = minutes
S = seconds
DEG = degrees
MA = minute of arc
SA = second of arc
y = year
m = month
d = day
- = dimensionless

HST = hours of sidereal time
MST = minutes of sidereal time
SST = seconds of sidereal time
HMT = hours of mean time
MMT = minutes of mean time
SMT = seconds of mean time

There will probably be many other abbreviations. We will update a list of these in a special folder in the DESCRIPTION FILE.

G/R: G means that the argument is given.
R means that the argument is a result.

Sometimes the argument might be both G and R, i.e., the argument had a special value before calling the routine and this value changed afterwards. In this case, you will either write "G/R" on one line, or two lines - one for "G" and one for "R".

Usually, the 10 lines in the table will be enough for a description of the arguments. If not, you might either

continue the table in a later part of the description, or avoid writing any description on page 1. In any case, one should notify the user on page 1 that information about the arguments will be given in a later section of the description.

COMMON'S: If the program is a main program it may have COMMON'S but this would not affect the user since the LCD of the program would include all COMMONS and all subprograms using those COMMONS. In general, SUBROUTINES and FUNCTIONS belonging to the library shall not use COMMONS. Therefore, in most cases the "NO" box will be crossed. There might be exceptions, in which case it is very important to warn the user that there are COMMONS to be observed.

2. Standard Form, Page 2

The true description of the program starts here. It is very convenient to first list all possible references, for example:

1. References: (I) some textbook
(II) some other NRAO report
(III) a description of another NRAO program (give name and NRAO number)
. ...
. ...

All following parts of the description can be shorter by referring to the numbers in the reference list.

The next section will almost certainly be: 2. Purpose - What the program is for and what it actually does is described in more detail in this section. After this, section 3. Method - It depends entirely upon the character of the program as to how much detail one has to go into. If it is a mathematical program, a set of equations should be given or some reference should be mentioned. If the program contains organizational structures which are important for the user to know, they should be described here. If the program is in FORTRAN, the source deck listing itself is a description of the method. If the program is relatively short, the source listing might be sufficient. However, there are FORTRAN programs which require a more detailed explanation. The next section would perhaps be: 4. Accuracy - This part is especially important for mathematical subprograms. Sometimes it will be adequate to say "6 significant mantissa decimals". In other cases a more detailed error analysis might follow.

MAIN programs require a description of all input-output functions, formats, etc. They should be handled in one or two

separate sections. Other items are: Range of Values of Variables, Error Messages (if there are any), Application Examples for the Program, etc. It is impossible to say in advance how many other things might be necessary to describe. The following is a list of some items:

1.	References	7.	Output
2.	Purpose	8.	Range
3.	Method	9.	Applications
4.	Accuracy	10.	} Other
5.	Error Conditions	11.	
6.	Input	.	
		.	

In general, one thing has to be said. By the time somebody has developed a certain program he may think that he will never forget anything about it. He may be willing to spend the time for describing the most important features but think that a more detailed description would be a waste of time. Usually this is wrong, because sometime later he will want to develop or change the program and then will have to spend much more time remembering the details than would have been necessary to describe them from the very beginning.

3. Standard Form, Page n

This is a form which can be used for an unlimited continuation of the description. It has the header line on it, but the proper page number has to be filled in.

4. Standard Form for Variable and Constants, Page n

This form can be used for both program development and description. Usually, in short programs it will not be necessary to explain all the local and other variables and constants because the source deck listing will fulfill the same purpose. Occasionally, however, adding this type of explanation to a program can be very helpful. The columns DIM'N, TYPE, and G/R contain the same information already described under I. The column I/C/L is explained at the bottom of this form. Occasionally, one will describe arguments of the calling sequence in more detail. These variables are not input, common, or local, but this is not important since the name shows that it is an argument (if one compares with page 1). In the explanation column you should list the numerical value of a constant. This is very helpful during programming work.

V. LCD FILE

The card decks belonging to the various programs are stored in order of their LCD numbers. We will observe the following rules:

1. Each deck will be a complete unit, i.e., if it is a main program it will contain all subprograms which are called except those on disk.
2. For principle reasons we will punch all FORTRAN source programs on FORTRAN cards.
3. A subprogram will start with a pink card.
4. A main program will start with a brown card. The cards before and after a data input deck will be blue cards.
5. All source decks will be interpreted.
6. A comment card placed after the job cards of a main source program or after the SUBROUTINE or FUNCTION statement of a subprogram will contain the following information:

Col. 1	C
Col. 4-7	NRAO
Col. 9-11	m (MAIN NUMBER), right adjusted
Col. 12	/
Col. 13-14	s (SUB NUMBER), right adjusted
Col. 25-32	NAME (starting in Col. 25)
Col. 36-44	Code for language, like "F4S", "BG11", etc. (always starting in Col. 36)
Col. 46-49	LCD=
Col. 50-53	LCD number, right adjusted

   We will keep duplicates of these comment cards in an extra file which can be used for an updated listing of all programs. This file will also contain cards corresponding to object decks and to programs in the scientific subroutine package.
7. The LCD number will be written on top of each card deck which is thick enough. Since the decks are in sequential order it will not be difficult to find a deck even if it does not have the LCD number on top.

VI. USAGE OF THE LIBRARY

There will be free access to the INDEX FILE, the DESCRIPTION FILE, and the I/O EXAMPLE FILE. These files will be placed in the keypunch room in Charlottesville and somewhere in Green Bank. No index card or folder should be removed from these files. A Xerox copy of the program description is available from the computer secretary.

There will be no access to the LCD FILE. If somebody wishes a card deck, he should submit a job submittal card to the computer

operator which specifies that a deck of the library program is needed.

Of course, there is no free access to the LOG BOOK. This is maintained by the program librarian.

The computer division secretary will always keep a number of standard description forms on file for people to use during development of a program or for writing a library program description.

In the DESCRIPTION FILE a special hanging folder will contain all general definitions used in the descriptions.

VII. FINAL REMARKS

As I said at the beginning, the idea of this type of library is that it grows haphazardly. If somebody starts a programming project, he may check the library to see whether he can find something which is helpful for his project. He may not always find the exact program he needs, but something similar which he can change for his own purpose. On the other hand, he should remember that such a library exists and that he might possibly contribute his own programs. Therefore, he should spend more time in developing a more general program - more general as compared to his present project.

It is not possible to see all the complications in advance. In the future it might be necessary to change some of these rules. In the beginning there will probably not be many programs in the library. As a start, Neil Stoltzfus (my summer student) and I developed a system of astronomical routines which can be useful in data reduction problems. I am preparing a special internal report on this system which will contain an appendix describing all these programs. I will distribute this report to all users after it has been completed. Since there are 36 programs altogether, the report and the appendix will give the users a certain feeling for the documentation of programs in the library as well as for the usefulness of such a library. In the future, new programs will not be distributed automatically. Instead, we will just announce the incorporation of a new program, with NAME, IDENTIFICATION CODE and a short description of the purpose of the program.

APPENDIX

Example of a Program Description

A computer listing of this program is not attached.

NRAO 1 / 1 S (D) DATE: 12/01/67 NAME: SIT

EXPLAN. OF NAME: Calculation of local mean SIdereal Time.
(SHORT DESCRIPTION)

Given are calendar date and zone time, longitude of observer and definition of zone time. The local mean sidereal time and the julian date are computed.

AUTHOR: N. Stoltzfus and P. Stumpff

LANGUAGE: F4 NO. OF STATEMENTS: 11

MACHINE: IBM 360/50
TIMING: 1.97 ms (G 11)
STORAGE 27C bytes (hexadecimal) (G 11)

LIBRARY CARD DECK: 1 (F4S)

SUBPROGRAMS CALLED FROM DISC: DJL(=NRAO 5/1 F); RED(=NRAO 21/1 F)

FROM CARDS: None

INPUT DATA (CARDS, TAPE, BOTH): None
OUTPUT DATA (CARDS, TAPE, BOTH): None
(DETAILS IN DESCRIPTION)

CALL: SIT(IY, IM, ID, ZTIM, OBLONG, ZTLONG, STIM, DATJUL)

ARGUMENTS

NAME	DIM'N	TYPE	UNIT	G/R	EXPLANATION	
IY		I4	y	G	calendar year (4 digits!)	on the zone time meridian
IM		I4	m	G	calendar month	
ID		I4	d	G	calendar day	
ZTIM		R4	REV	G	zone time (for example: EST)	
OBLONG		R4	REV	G	geogr.long.of observer	
ZTLONG		R4	REV	G	geogr.long.of zone time meridian	
STIM		R4	REV	R	local mean sidereal time	of the given instant of time
DATJUL		R8	d	R	julian date	

(G = GIVEN , R = RESULT)

COMMON'S: YES ☐ NO ☒ (IF "YES", SEE DESCRIPTION)

NRAO 1/1 S NAME: SIT

PROGRAM DESCRIPTION

1. References: (I) NRAO Computer Division Internal Report No. 2

2. Purpose: A certain instant of time is given by the civil calendar date (IY,IM,ID) and by the civil time (ZTIM). The local mean sidereal time (STIM) and the julian date (DATJUL) have to be calculated which correspond to that instant of time.

Civil date and civil time are common within certain geographic zones; actually, they are the mean solar date and the mean solar time for a certain meridian within that geographic zone (zone time meridian). Therefore, we say that IY,IM,ID is the calendar date on the zone time meridian and that ZTIM is the zone time. Which particular zone time is used depends on ZTLONG, the geographic longitude of the zone time meridian. For example, if ZTIM is Eastern Standard Time (EST), then ZTLONG has to be set equal to 0.208333.. ($= 5^h$ west of Greenwich). If ZTIM is Universal Time (UT), then: ZTLONG = 0.0

To compute STIM, the local mean sidereal time, one has to specify OBLONG, the geographic lomgitude of the observer.

Both ZTLONG and OBLONG are constants which have to be defined in a main program which calls SIT.

In addition to STIM, the program also computes DATJUL. This is the julian date which corresponds to the given instant of time (IY,IM,ID,ZTIM,ZTLONG). DATJUL can be used as the time argument for other routines which compute time depending variables such as, for instance, nutation or aberration.

3. Method and accuracy:

The method is described in Ref.(I), section 6. For any given instant of time, STIM is calculated from the Green-

NRAO 1 /1 S NAME: SIT

wich mean sidereal time for 1967.0 (=beginning of the Besselian year 1967) and from the time interval to that initial epoch. The second-order terms in the definition of mean sidereal time are neglected because they would not exceed $0^s.05$ within the 20^{th} century; the single-precision mode of the program will cause larger errors than $0^s.05$. The variable TAU (see Ref.I, equation 6/10) which is the time interval to 1967.0 will assume maximum values (within the 20^{th} century) of about 67,... in 1900. In single precision, this number would be not more accurate than about 0.00005. When STIM has been computed and has been reduced to the standard interval $0 \leq STIM < 1$ (in revolutions), or $0 \leq STIM < 24$ (in hours), STIM will have an error of this amount (0.00005 REV = $4^s.5$).
In the period 1958 to 1976, however, TAU is less than 10, and the error of STIM is reduced by one order of magnitude. These accuracy considerations lead to the following

Rule: STIM can be applied to any instant of time within the 20^{th} century with an error of not more than about $4^s.5$ in the resulting sidereal time. For applications in the period 1958-1976, the error will not be larger than about $0^s.45$. DATJUL will always be as accurate as ZTIM and ZTLONG are. If the accuracy of STIM is not satisfactory, one should call the routine DSIT(=NRAO 2/1 S) which gives the full precision of the Ephemerides.

4. Applications:

If one has to compute many values of the sidereal time within time intervals of several hours, it would save computer time by computing STIM only for the beginning of the interval and then to add always the sidereal time

equivalent of the interval of zone time elapsed since the beginning. After the addition, one may call the routine RED(=NRAO 21/1 F) in order to reduce the result to the standard interval 0 to 1.

5. Special values for OBLONG and ZTLONG:

Green Bank: OBLONG = 0.2217953 REV (=$5^h19^m20^s.0$)
ZTLONG = 0.2083333 REV (=5 0 0.0; it is assumed that ZTIM = EST)

Kitt Peak: OBLONG = 0.3100371 REV (=$7^h26^m27^s.203$)
ZTLONG = 0.2916667 REV (=7 0 0.000; it is assumed that ZTIM = MST)

NRAO 1 /1 S NAME: SIT

VARIABLES AND CONSTANTS

NAME	DIM'N	TYPE	UNIT	G/R	I/C/L	EXPLANATION(value, if const.)
DJULA		R8	d	G	L	2439491.541=1967.0= JDA (Ref.I, 5/18)
DJUL0		R8	d	R	L	= JD_0(ZDAT) " 6/10
GA		R4	REV	G	L	=0.27777871=G_A " 6/8
CK2		R4	-	G	L	=1.0027379=k_2 " 3/2
						CK2 is the ratio $\frac{\text{mean solar day}}{\text{mean sidereal day}}$
CK3		R4	trop. year per day	G	L	=0.0027379093=k_3 (Ref.I, 3/4) CK3 is the increase of mean sidereal time per mean solar day
RDELTA		R4	REV	R	L	= Δ (Ref.I, 6/8)
GB		R4	REV	R	L	= G_B^r " 6/9
TAU		R4	trop year	R	L	= τ " 6/10

(G = GIVEN , R = RESULT , I = INPUT , C = COMMON , L = LOCAL)