S O M N I A

Design and Development of an Interactive On - Line Data Management System for Biomedical Applications

by

Christian Friedrich-Freksa

RESEARCH PROJECT

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, to partial satisfaction of the requirements for the degree of Master of Sciences, Plan II.

Approval for the Report and Comprehensive Examination:

COMMITTEE: Joe Kamiya, Research Adviser

April 12, 1976 Date

____

______________________ Date

Background: After the development of (off-line) pattern recognition techniques for biophysiological data (e.g. <1>, <2>) and the implementation of single task processing on-line routines (e.g. <3>, currently being tested for routine purposes in clinical applications, <4>) various attempts have been undertaken to more generally solve the problem of on-line processing of biophysiological data on minicomputers and medium-scale lab computers. Especially in the experimental stages flexibility is required to find optimal solutions for routine applications. These activities include the development of special (macro-) languages for biosignals (<5>, <6>), of Real - Time Systems which can be accessed by higher level languages (<7>), and of hardware options providing flexibility in the interrupt mechanism of a computer <8>.

The advantage of a macro language for biosignal processing is the simplicity in programming such a language. The disadvantage is the required assembly time which ranges between 20 and 40 minutes for a typical program (on IBM 1130, 3.6 usec cycle time). A programming error therefore is expensive - unless a large scale computer is available which assembles the program before use on the lab computer.

A Real - Time System (RTS) as developed at the Langley Porter Institute, UCSF, consists of object modules (therefore no overhead due to assembling), the programs which are to be scheduled by the RTS have to be written for each particular problem.

Purpose of SOMNIA: Supervising the Real - Time System of LPNI on the logical level.

Table of Contents

I thank Dr. Joe Kamiya for providing me the great opportunity of joining his stimulating research team.

Comments.

The names of all modules which are specifically designed for SOMNIA start with the letters SO - Sleep Onset.

The routines of the Real Time System begin with the letters RTS.

The term "evaluation" will be used to denote the processing of the physiological data to distinguish it from the organizational processes of SOMNIA.

0 Purpose

At the Langley Porter Neuropsychiatric Institute of the University of California a study about sleep onset is being prepared. This study will include the measurement of several channels of physiological data like electrocardiogram (EKG), electroencephalogram (EEG), electromyogram (EMG), body temperature, breath rhythm, etc. The measurements are to be taken in a certain time period before sleep onset.

In an early stage of the experiment the computer is used to reduce the information of these data and to store the resulting information for further processing.

The described program system is designed for more general use in experiments with different kinds of analog data; the objective of the first implementation however is to serve the sleep onset experiments.

1 Experimental Implications.

1.1 Type of Input Information.

The data that are to be collected are electrical signals which correspond to physiological processes. The information that is to be extracted from these signals may be carried over by

a) amplitude modulation (AM)
b) frequency modulation (FM)
c) DC offset

or by any combination of these three.

Examples:

1) the EEG is usually processed as AM signal. The frequencies of interest for physiologists go up to about 30 Hz.
2) the heart beat rate is an FM signal on the EKG.
3) the body temperature generates in a thermoelement a DC signal.

1.2 Events During the Experiment.

Experiments with living beings frequently take an unexpected course. An electrode may fall off or, in certain circumstances, it might be desired to modify experimental conditions. In the particular example of a sleep onset experiment an additional difficulty appears: we cannot predict the sleep onset time - the experiment may last for two minutes or it may last for one hour.

1.3 Effectiveness of the Experiment.

Physiological experiments are time consuming and expensive. It is desired to make results visible as soon as possible in order to influence the continuation of the experiment according to the information that was obtained during the experiment.

2. Demands of the Program.

Because of uncertainty, during the planning stage, as to which features of the experiment will be interesting, the system should be as flexible as possible.

2.1 Data Collection.

1) The number of data channels should be variable during the experiment.

2) It should be possible to choose freely from the available hardware channels.

3) The data sampling rate must be independently variable on each channel.

2.2 Data Processing.

1) The tasks to be performed on the data should be independently variable on each channel.

2) The time period of data to be processed (hereafter referred to as "window size") should be independently variable for each channel, according to the task that is to be performed on the data of each channel.

3) It might be useful to have the system on-line display some results obtained by the process.

2.3 Data Storage.

1) It should be possible to store the raw data of the experiment as well as the processed information. Then it would be possible to reproduce results later, or to reprocess data, or to compare the information which was obtained by undertaking different processing procedures on the data.

2) In case it would seem useful to get more accurate information from the data than was originally planned, we should allow for storing the raw data with a higher resolution than we actually need for the process.

3) In order to save storage space it should be possible to select the information that is to be stored.

4) For the needs of the sleep onset study we should allow for storing the data to a ring buffer (a data area without logical end in which the oldest data is being overwritten by new data). Thus we save storage and keep only the most recent data which we are interested in.

2.4 Interaction.

1) The system should be capable of accepting interactions by the operator to modify program parameters during the experiment. Thus we can increase the sampling rate for EEG data if we detect unexpectedly high EEG frequencies, add a new task if it seems appropriate for the experiment, or drop all functions of a channel if an electrode falls off. However, the data process of the channels not concerned by such an interaction should not be influenced. This means in particular that no data of the experiment will be lost during an interaction.

2) A change of organizational parameters implies structural changes in the program and/or the data. Because the interactively requested changes in the structure of the process must be obtained quickly, the system should be sophisticated and should be able to calculate quickly the optimal new process conditions

according to the request.

3) Changes in the program structure or the data organization may confuse the interpretation of the results. Therefore, the program must document itself properly: what it is doing, at which point of time, and what the produced data represent. In order to synchronize the data of the program with other observations of the experiment, the program should record the time. It also might issue time signals to the one running the experiment or to registration paper on which signals of the experiment are recorded.

4) It will be possible to request too much from the program in respect to time or to storage capacity. An on-line system which is fed with data steadily must perform its tasks within a certain range of time and must not produce more data than space is provided for. The system should keep control over its capacities and possibly notify the operator about its load upon request.

3 Design of SOMNIA

- Sleep Onset Monitor Notified Inter - Actively.

3.1 Overview.

The CPU has to perform three main tasks in communication with peripheral units:

a) data collection

b) buffer organization and data storage

c) control of interaction

Internally the following three tasks must be performed:

d) calculations for program structure and optimal storage allocation

e) organization of the modules according to the calculations

f) system's modifications after an interaction by the operator

The Real Time System (RTS) by P. Harris and J. Johnston (Appendix B) schedules different tasks on up to 10 different priority levels according to the user's specifications and supervises their execution. We can utilize the supervision of RTS for the modules specified under a), b), c), d), and f). In e) we want to specify what RTS is to supervise. This module therefore must get control of RTS.

3.2 Hierarchy of Tasks.

1) The data collection must get the highest priority in an on-line system in which the data source is an independent variable. It will be performed by the RTS routine --> RTSADS.

2) On the next lower (relative) priority level we must handle the system modification. At the point of time at which the system allows its modification this must happen very fast in order to allow a unique interpretation of the data after a modification. This task will be performed by the module SOFIMO (Fast Instruction MOdifier).

3) Depending on the kind of modification it may become necessary to redetermine program structure and optimal storage allocation. This must happen in the context of all tasks to be performed. For this SOCALC, the Sleep Onset CALCulator, will be responsible.

4) The module that organizes the Real Time System has to come next. First it initiates the higher priority processes and later, after SOFIMO changes the tasks of the

Real Time System, it must modify RTS accordingly. This module also may be scheduled by the Real Time System, because RTS allows its self-destruction and the definition of new tasks during its own run time. The module's name is SORTMO (Real Time MOdifier).

5) The previous modules ensure correct data input. On the next level of the hierarchy these data have to be processed (before they become unavailable). SOBOR, the Buffer Organization Routine will take care of this task.

6) On the lowest level of the hierarchy we may allow human interaction with the scheduled processes. This means, we don't allow external interaction before the program has taken care of all ongoing processes. Because the data transfer rate of a human interaction is very small in comparison to the internal data rate of the CPU, the operator may not even realize the low priority of his actions. SOMMM (Man Machine Message) will replace the RTS idle routine, waiting for external commands which it may analyze in its spare time.

3.3 Data Structures.

We must provide data areas for

a) A-D buffer of RTS

b) selected data to be evaluated

c) transfer buffer for data to be written to disk

The size of a) and b) depends on the particular program configuration and on the space available. It is more efficient, therefore, to specify a large data area to be shared by a) and b). The size of c) depends on the maximal block size on the disk. In order to most efficiently utilize the disk we will always write maximal sized blocks when we have collected enough data to fill a block.

It is a convenient restriction to only allow the specification of powers of 2 for time periods and sizes of data bases. Thus we can easily synchronize our process and efficiently organize the buffer area for the data. Furthermore, we can prevent a systems overload by coincidental demands of several programs. This restriction makes it easy to adapt the data to existing library routines (e.g. Fast Fourier Transformation), which require n=2**m data for input ($m \in N$).

1) Dynamic Buffer Area.

The highest channel number and the lowest sampling period determine the size per time unit for the A-D buffer (RTS samples data for all channels from 1 to the highest channel number specified with the lowest sampling period specified; SOMNIA extracts from this data pool the data it needs.) We should provide space for twice the amount of time that we provide for the extracted and sorted data. Thus we can process one pool of data while we fill the other one (double buffering). The selected data should be ordered according to the channels they belong to. Then they easily can be evaluated channel by channel. Because of the restrictions we defined for the sampling periods, the number of data per channel always will be a power of 2. We therefore can easily compute the entries and current pointers of the channel buffers. However, we can save time by employing seperate pointers for each channel buffer.

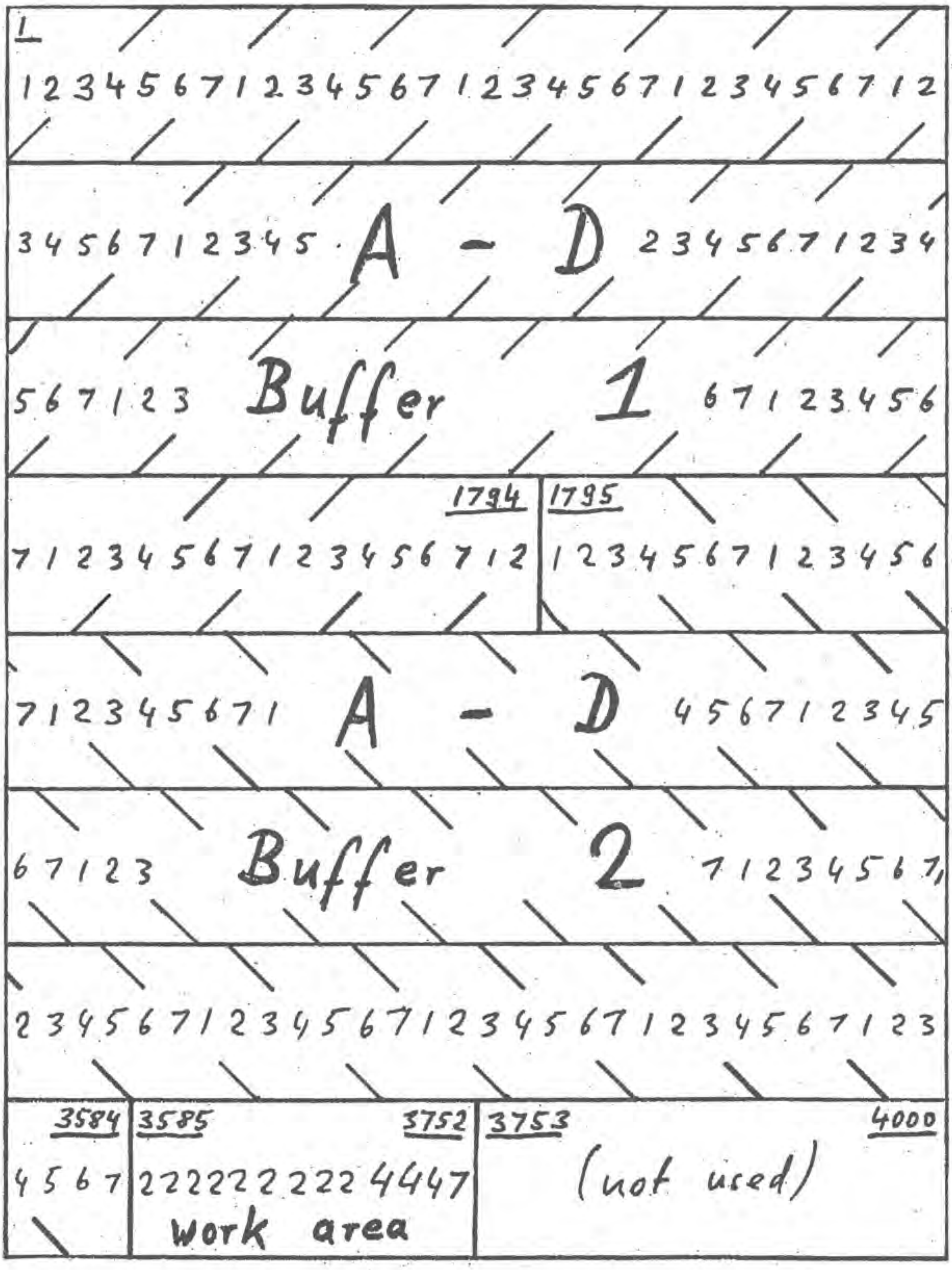

Fig. 1 Dynamic Buffer Area

space distribution as for 3.5 example.

2) Transfer Buffer.

The physical record size on the PDP-15 (hereafter called "block") is 251 words. The amount of data to be stored for each channel will be either zero or a power of 2. Depending on the particular program configuration the sizes and number of logical records will vary. Therefore we shall use variable format records which we shall pack into a block. The records must be identified by the channel number they correspond to. Because the data are to be stored to a ring buffer each block must be identified by a sequence number or better even by a time identifier.

Because we might be interested in a specific selection of records, it is better to have the information about the records along with a location pointer in a directory than to have it at the beginning of each record. Because the number of records in a block may vary, the size of the directory may vary also. When we start the records at the beginning of a block and the directory at the end of a block and let them grow towards each other we can utilize the whole block most efficiently.

Fig. 2 Transfer Buffer Area

1	3	13	248	250
time	data	data	2 4	1 4
	ch.1	channel 2	13 233	3 10

directory

directory contains channel #, sampling period, start address, lenght.

3.4 System Structure.

How do these modules work together ?
We need a table by which we can inform SOMNIA about the tasks to be performed. We could generate this instruction table with the interactive command language in interaction with a dummy process. But usually we shall want to run a series of experiments with the same tasks to be performed. Therefore we shall use a separate program SOINIT which generates a table describing the initial program configuration. This table will be stored on disc and/or tape. SOINIT can be extensively selfdocumenting, because it does not have to be on-line. Eventually it should be able to estimate the load of the system for the issued instruction table. In case of an expected overload it might give suggestions

for a cut back of demands.

All the described modules are to be run under the supervision of RTS. So we need a program which reads the instruction table into a COMMON area, initializes RTS, runs SOCALC to determine the system configuration. SOCALC runs SORTMO to modify the task schedule of RTS accordingly. SORTMO schedules and runs SOFIMO and SOBOR which will become active periodically depending on the calculated time and space schedule.

Last SOMMM gets activated. When SOMMM encountered an instruction by the operator it will prepare the instruction for SOFIMO which becomes active whenever the storage situation is favourable for a schedule modification. If the modification requires a new schedule for the RTS processes SOFIMO will run SOCALC and SOCALC will run SORTMO.

This nested calling structure allows for flexible managing of the RTS processes. Table 1 shows the relation between the different resources of the program system during the initiation.

3.5 Example of a Program Configuration.

We shall assume that we want to evaluate two channels EEG, determine the heart beat rate, and store information about the body temperature. We want to use for this purpose analog channels 1, 2, 4, and 7 respectively. The EEG data are to be collected with a sampling period of t=4 (t=1024 means 1 sec.), the heart beat with t=64, and the body temperature with t=8192. We shall store the raw data of the EEG channels and of the body temperature in the ring buffer.

We further assume that we want to evaluate channel 2 on-line with a sample period (referred to as "evaluation period") of t=8, the heart beat with t=64, and the body temperature with t=8192. Let us choose the window size for the EEG data processing to 128, for the heart beat to 2048, and for the temperature 8192 (i.e., we look only at one temperature value at a time). For simplicity we give the code numbers 1, 2, and 3 for the evaluation routines of channels 2, 4, and 7 respectively.

The generated sleep onset instruction table therefore would look as follows:

channel #	1	2	3	4	5	6	7
sampling periods	4	4	-	64	-	-	8192
store raw data	yes	yes	-	no	-	-	yes
evaluation period	-	8	-	64	-	-	8192
window size	-	128	-	2048	-	-	8192
evaluation mode	-	1	-	2	-	-	3

This instruction table may have been generated off-line. SOCALC has to check it now for compatibility with the on-line system and has to adapt it eventually.

We shall assume the size IBUDIM of the dynamic buffer area to 4000 words. RTS requires for its A-D buffer space for as many channels as the highest number indicates (here: 7) with a sampling period as low as the lowest sampling period indicates (here: 4). In order to make sure that we shall not overwrite unprocessed areas of the A-D buffer we we ought to have double buffering. Now we can determine the maximal size of the A-D buffer to

IADSZE = LENGTH * NCHANS * 2,

where

LENGTH is the highest power of 2 such that

LENGTH =< IBUDIM / (2 * NCHANS).

LENGTH is the size of a single A-D buffer of a single channel.

NCHANS is the highest channel number

IBUDIM is the size of the dynamic buffer area.

In our example we get

LENGTH = 256 words

and therefore

IADSZE = 3584 words.

The time required to fill one A-D buffer is the repetition period

MREPT = LENGTH * MSAMP = 1024,

where

MSAMP = minimal sampling period.

We want to establish our data areas and processes such that they repeat all MREPT milliseconds. Therefore we have to modify the instruction table such that no sampling period exceeds the repetition period.

In case a window size exceeds the repetition period we shall establish a work area with enough space for data of the whole window.

The size of the work area for each channel therefore is

IWA (ICHAN) = MAX (IWINDO(ICHAN), MREPT) / NEVAL(ICHAN) where

IWINDO (ICHAN) is the window size

and

NEVAL (ICHAN) is the evaluation period of the respective channel.

We obtain work area sizes of 128, 32, and 8 for channels 2, 4, and 7 respectively. I.e., we use 3584 + 128 +32 + 8 = 3752 words of the dynamic buffer area.

By this setup we obtain an over all repetition cycle

IOPT = MAX (IWINDO (ICHAN) , MREPT) = 8192,

i.e., all IOPT msec <u>all</u> pointers of the dynamic buffer area will point to their starting position. That is the point of time at which we may allow for modification of the task schedule, because the data in all buffers have been processed.

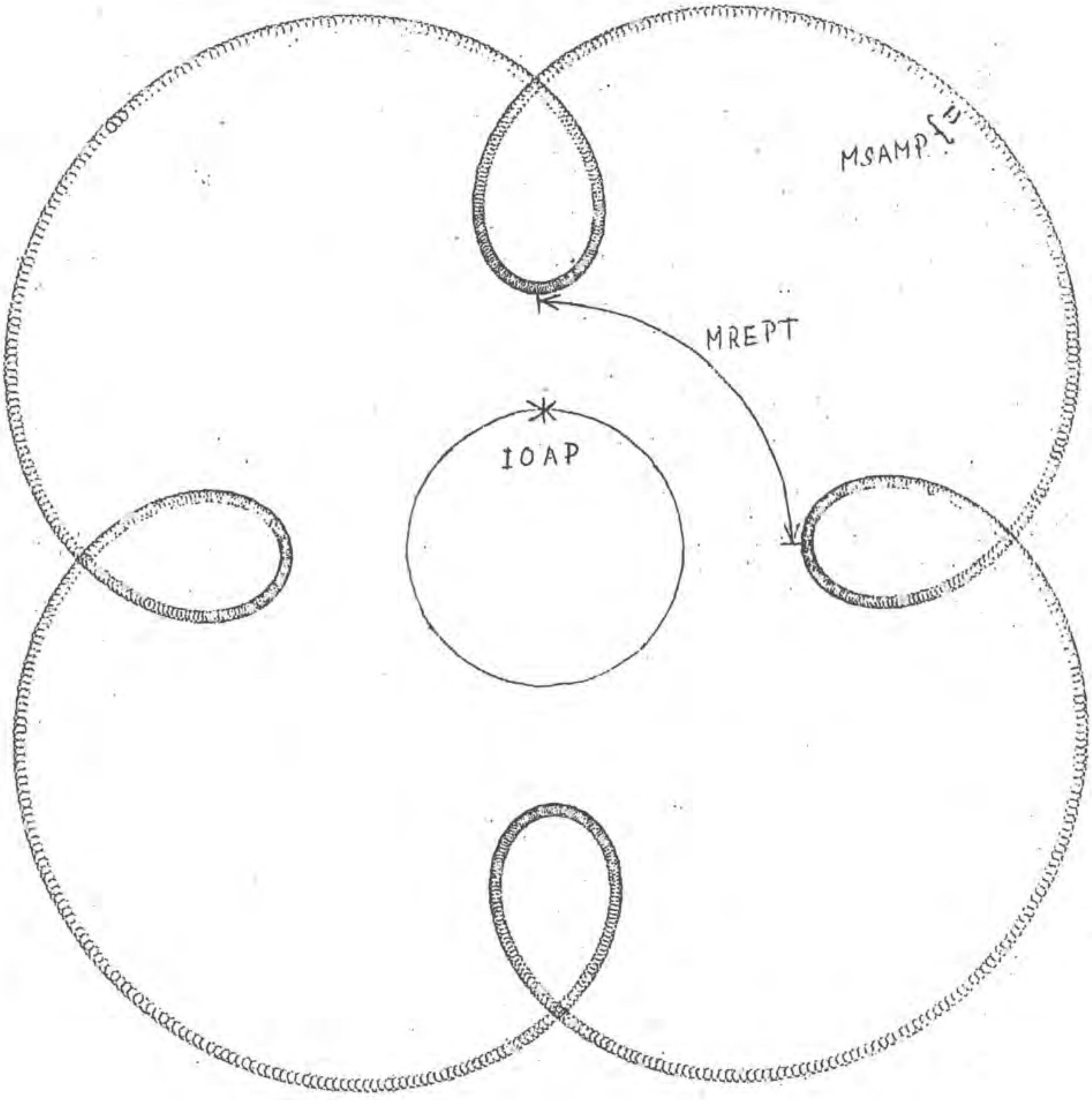

Fig. 3 Scheme of Program Cycles

3.6 The Command Language.

The command language is to be used to interactively influence the program parameters

a) channel numbers
b) sampling periods
c) data storage
d) periods for data evaluation
e) window size for data evaluation
f) evaluation to be performed

The commands should not influence the ongoing process before the validity of the command is checked.

The syntax check should be very fast.

The security must be very high, i.e.
a) meaningless commands must not do any harm to the process.
b) there should be some feedback about the interpretation of the commands before they take effect.

The consequence of a command might be a total re-organization of the storage structure. Therefore it should be possible to concatenate several commands referring to one task in order to prevent changes which might be overthrown by an immediate following command.

Each I/O channel can be considered representing a task in its meaning for operating systems <1>. We may specify the channel number first to denote that the following commands refer to the respective channel. The commands following may simply consist of a parameter specification and a value for that parameter.

After approval of the interpretation of the commands by the operator we then need to tell that the modification actually take place.

BNF - Grammar for the Command Language:

```
<taskmodification> ::= <channel specification>
                       | <instruction> |o
                       <approval>
<instruction>      ::= <parameter specification>
                       <value>
```

We may interpret a command string without an instruction. (i.e., a command string containing only channel number i and approval) as: "forget task i!" That means: deschedule task i if it was specified.

We determine that of two contradicting instructions the last one is significant; thus we may even allow that the same parameter specification appears more than once in a command string. This feature makes it possible to correct an instruction after disapproval of its displayed interpretation.

The channel buffer for the teletype at the PDP-15 may store only one character at a time; we shall implement the language therefore with very short tokens.

Vocabulary of the command language:

<channel specification>	::=	1 |	
		2 |	
		3 |	
		4 |	
		5 |	
		6 |	
		7	
<parameter specification>	::=	S |	(sampling period)
		R |	(raw data storage)
		E |	(evaluation period)
		W |	(window size)
		M	(evaluation mode)

<value> ::= 0 |
1 |
2 |
3 |
4 |
5 |
6 |
7 |
8 |
9 |
10 |
11 |
12 |
13 |
14 |
15 |
16

<approval> ::= G (go)

comment: R0 means no data storage,

RI, where I>0, means raw data are to be stored

In instruction M, <value> denotes the evaluation mode; in all other instructions, <value> denotes the log2(x), where x is the intended parameter value (by definition a power of 2).

Examples of Commandstrings.

	command		meaning
a)	3		select channel 3
		S 8	install sampling period of 2**8=256 units
		R 1	yes, store raw data
		E 8	evaluate data with a period of 256 units
		S 6	correct sampling period to 2**6=64 units
		W 10	set window size to 2**10=1024 units
		M 7	evaluate data with program 7
	G		go on and let foregoing instructions take effect
b)	2		select channel 2
		E 6	change evaluation period to 64
		R 0	don't store anymore the raw data
	G		approve modification
c)	5		select channel 5
	G		and "forget" it

3.7 The Command Language Analyzer.

Human interaction with the program is handled on the lowest level of the process hierarchy. Because the time of interaction is unpredictable for the program we may replace the idle process by a continuous check for interaction. We have to test whether the contents of the channel buffer have changed since the last check. This also works if we want transmit the same symbol twice, because it takes more time to fill up the buffer than to test it; therefore the buffer will be tested before its regeneration is finished and will have a different value at test time. The new value itself may be determined somewhat later when the buffer is fully regenerated. Then analysis of the symbol may take place to find out whether or not it is valid in its context.

If the symbol is valid it will be used to build the command string and the analyzer can be put in its next state. If it is not it must be ignored (eventually even a not yet completed command should be erased from the command string to allow for correction of the entire command after an error occured. I.e., the analyzer should be put back to a previous stage).

When the command string is built up completely the analyzer should not accept further input before it delivered the string to SOFIMO.

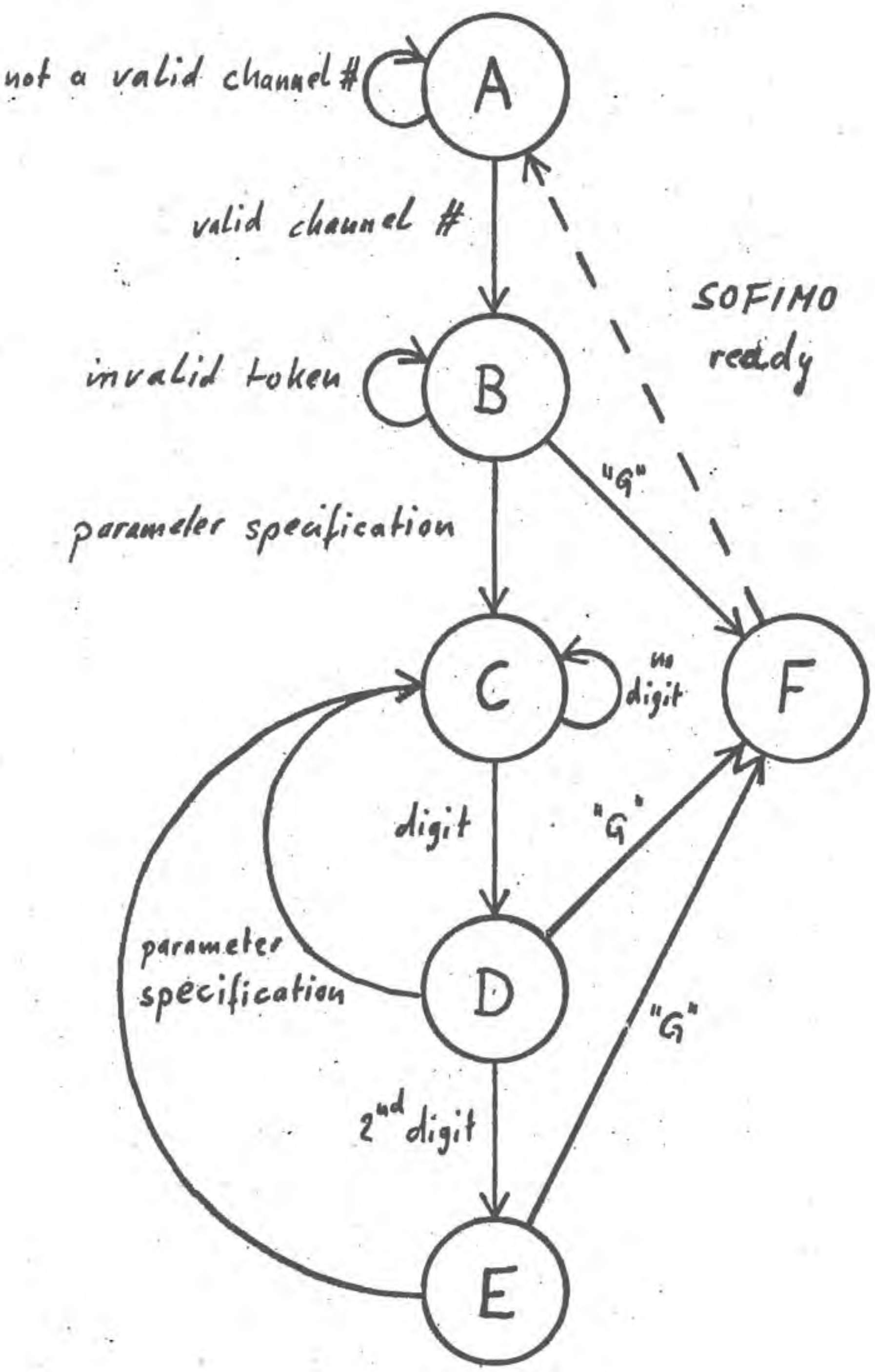

Fig. 4 States of the Analyzer.

4 Description of the Modules.

4.1 SOINIT

is an independent mainline program which creates the instruction table for SOMNIA.
(2302 words)

The sourcelisting of version 1 is selfdocumenting.

4.2 SOSTRT

is the mainline program which starts SOMNIA.
(135 words)

COMMON area IN.

reads instruction table from disk,
initiates file for transfer buffer,
defines size of dynamic buffer area,
initializes the Real Time System,
sets up a process descriptor for
SOCALC on priority level 30,
SORTMO on priority level 20,
SOMMM on priority level 5,
puts SOCALC and SOMMM on the run queue,

turns on the real time clock and starts RTS, i.e., transfers control to SOCALC and SOMMM.

4.3 SOCALC

Initially run by SOSTRT,
In case of modification requests run by SOFIMO.
(461 words)

COMMON areas IN, PT, MO.

determines from the instruction table highest channel # and lowest sampling period and locates the information in the headline to the instruction table,
initializes transfer buffer pointers,
calculates maximal buffer space to be occupied by A-D buffer,
initializes pointers for general data buffer,
determines repetition cycles for operational and organizational routines,
adjusts instruction table to program conditions,
determines required work area for each channel and puts information into the trailer of the instruction table.

4.4 SORTMO

modifies RTS according to the calculations by SOCALC. Is being run by SOCALC.

(201 words)

COMMON areas GB, IN.

gets time since RTS initialization,

dequeues RTSADS, SOFIMO, and SOBOR if they are in a RTS queue,

sets up new process descriptors for

RTSADS with repetition cycle MSAMP,

SOFIMO with repetition cycle IOAP,

SOBOR with repetition cycle MREPT.

These routines are put back onto the clock queue and are run initially to the appropriate time determined from the running time and the repetition cycle.

4.5 SOFIMO

called periodically all IOAP msec.

(400 words)

COMMON areas IN, MO.

checks whether modification requested and returns, otherwise.

if modification requested:

if channel is to be ommitted: modifies instruction table accordingly and returns instantly

if channel is to be modified or new channel to be added: modifies instruction table and runs SOCALC.

4.6 SOBOR

- buffer organization routine,
called periodically all MREPT msec.
(1347 words)

COMMON areas IN, GB, PT.

gets current run time of the program to store in the headline of the transfer buffer

stores sampling rate and bookkeeping information into the directory of the transfer buffer

updates pointers for A-D buffer, general data buffer, and transfer buffer

stores transfer buffer to disk, if full

calls the data evaluation organizer SOEVAL.

4.7 SOMMM

- man machine message,

endless routine which replaces idle process of RTS (562 words)

COMMON area MO.

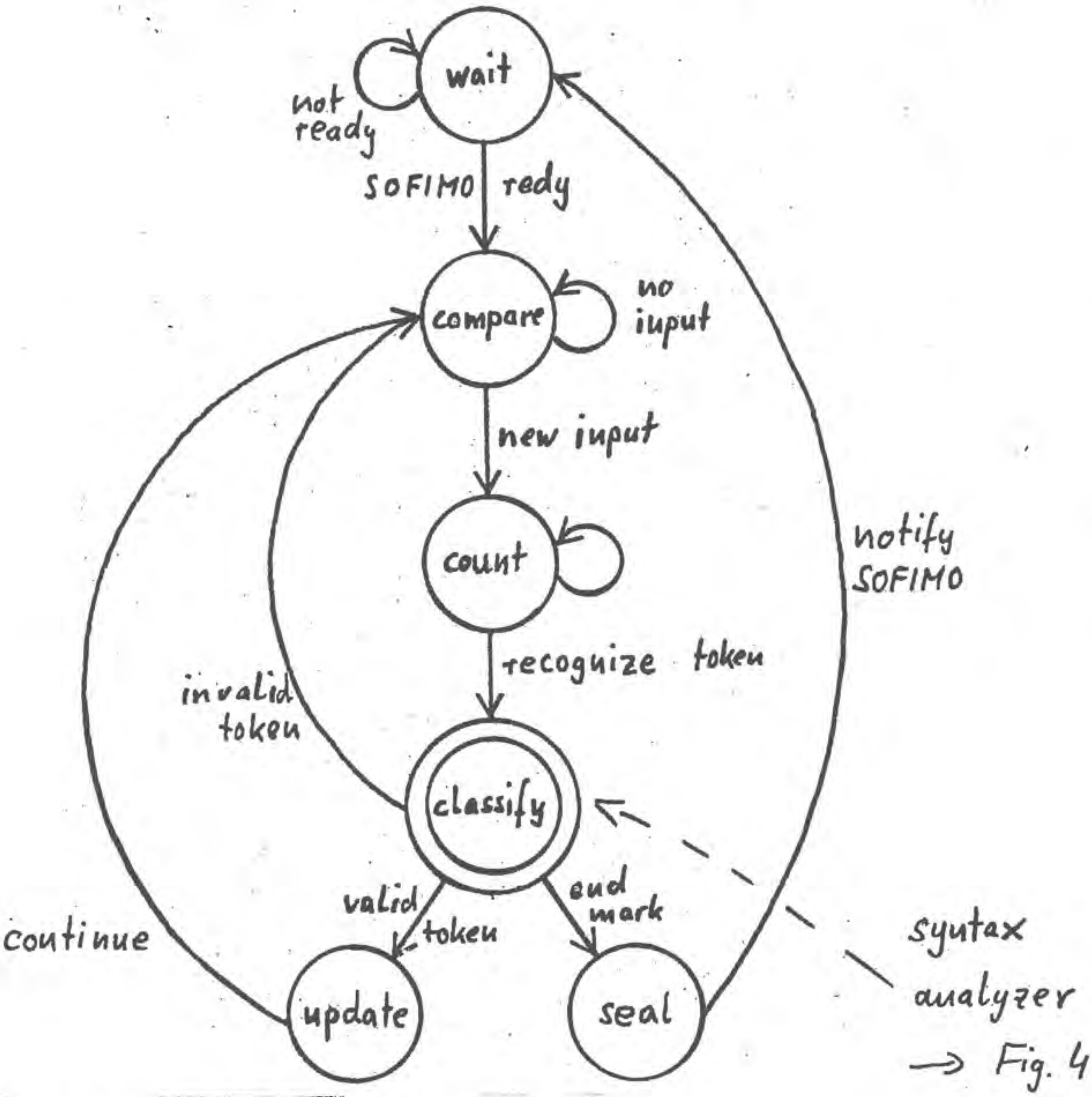

Fig. 5 Transition Diagram of SOMMM.

4.8 SOEVAL

- evaluation organizer
(24 words)

Version 1: dummy routine displaying the channel number of the channel to be evaluated.

APPENDIX A

COMPUTER EQUIPMENT OF THE LABORATORY

A. HARDWARE: PDP-15 COMPUTER

(16 K CORE, 18 BIT WORDS, CYCLE TIME 800 NSEC)

HARDWARE CLOCK

I/O: DISK (APPR. 1 MBYTE)

2 DEC TAPE DRIVES

TELETYPE (10 CHAR/SEC)

PAPERTAPE READ AND PUNCH

ANALOG-DIGITAL CONVERTER

(8 CHANNELS, RESOLUTION 12 BITS)

B. SOFTWARE: DISK OPERATING SYSTEM

MODULAR REAL TIME SYSTEM ORGANIZING DIFFERENT PROCESSORS ON DIFFERENT PRIORITY LEVELS.

(SEE APPENDIX B)

Appendix B

F O R T R A N   C A L L S   T O   T H E
R E A L   T I M E   S Y S T E M

I. FORTRAN CALLING SEQUENCES

A. PRIMARY ROUTINES

```
CALL RTSINT

CALL RTSPR (PROCSR,IPRLVL,IREPT)

CALL RTSAD (NCHANS,LENGTH,IADBUF(1),IPRLVL,IREPT)

CALL RTSBH (IDIBIT,IDELAY,PROCSR,IPRLVL,IREPT)

CALL RTSKED (PROCSR,LDVAL,ISTART)

CALL RTSRUN (PROCSR,LDVAL)

CALL RTSFI (PROCSR,IREPT,ISTART)

CALL RTSDQ (PROCSR)

CALL RTSBDQ (IDIBIT)               (IDIBIT = 0 TO 8)

CALL RTSGO ;   CALL RTSTOP

CALL RTSDON;   CALL RTSDOF         (DION, DIOFF)

CALL RTSTON;   CALL RTSTOF         (DITON, DITOF)

CALL RTSRON;   CALL RTSROF

CALL DIPAUS;   CALL DICONT
```

4. SPECIAL USER'S ROUTINES

```
LPVAL = IACF(LPVAL)

IP = IPULSE(IP)

IPLTIME = IPTIME(IP)

CALL RTSJAM (IBUFFR(1))

CALL RTSCPY (ICHAN,IBUFFR(1),IBUFR2(1))

CALL RTSGET (IBUFFR(1));    CALL RTSGT2 (NDIMEN,IBUFFR(1,1))

CALL GETBF1 (LENGTH,NCHANS,INDEX,IRINGL(1),IUSEBF(1))
        CALL GETBF2 (NDIMEN, LENGTH,....)

CALL ADCONV;    CALL ADCSET (ICHAN,FACTOR)

CALL ADCALB (ICHAN,IADOLD,IADNEW)
```

```
C.  GLOBAL'D VARIABLES WHICH ARE AVAILABLE, USING THE INTEGER
EXTERNAL FUNCTION (IEXTF):

TIME ..... CURRENT TIME (MODULO 262144) (1024 = 1 SECOND)
NCHANS ... NUMBER OF A/D CHANNELS
LENGTH ... NUMBER OF A/D SAMPLES PER CHANNEL
ADSIZE ... SIZE OF A/D BUFFER (NCHANS*LENGTH)
INDXAD ... FORTRAN INDEX TO OLDEST DATA POINT (CHANL 1)
           IN THE A/D BUFFER
STIME .... TIME OF THE LAST A/D SAMPLE
ADBUFR ... ADDRESS OF A/D RING BUFFER
(ICHANL,NEWDAT,OLDDAT)---MODFAD PARAMETERS (NOT FOR F4 USERS)

THESE VARIABLES ARE ACCESSIBLE WITH THE IEXTF & ISEXT
EXTERNAL VARIABLE ROUTINES.
```

SYMBOLS USED IN DESCRIPTION OF ROUTINES

```
SYMBOLS:                                (RTS= REAL TIME SYSTEM)

PROCSR ... SUBROUTINE NAME OF PROCESSOR:
           THIS NAME MUST APPEAR IN AN 'EXTERNAL' STATEMENT.
IPRLVL ... RELATIVE PRIORITY LEVEL, 2 TO 100.  NO MORE
           THAN 10 UNIQUE VALUES OF PRIORITY LEVEL ARE
           ALLOWED BY RTS.
IREPT .... REPETITION PERIOD (MSEC.).  EVERY SCHEDULED
           PROCESS WILL BE RUN PERIODICALLY IF IREPT > 0.
           IF IREPT = 0, THE PROCESS WILL BE RUN ONLY ONCE.
           IF IREPT = -1, THE PROCESS WILL BE "DE-SCHEDULED"
           (REMOVED FROM THE CLOCK QUEUE IN RTS), AT THE
           TIME IT WOULD OTHERWISE HAVE BEEN RUN.
ISTART ... START TIME (MSEC.).  EVERY SCHEDULED PROCESS WILL
           BE RUN INITIALLY AT THE TIME INDICATED BY ISTART.
           (ISTART IS UPDATED BY RTS.)
LDVAL .... VALUE LOADED INTO THE AC WHEN A PROCESS IS STARTED
           BY RTS.

NCHANS ... NUMBER OF A/D CHANNELS TO BE SAMPLED.
LENGTH ... NUMBER OF SAMPLE POINTS, EACH CHANNEL.
IADBUF ... NAME OF A/D RING BUFFER.

IDIBIT ... BIT NUMBER FOR DIGITAL INTERFACE (DI) PULSE.
IDELAY ... TIME (MSEC.) AFTER DI PULSE FOR RUNNING PROCESS.

NDIMEN ... 2ND DIMENSION OF 2-DIMENSIONAL ARRAY.
                (E.G., DIMENSION (4,128): NDIMEN=128)
```

II. DESCRIPTION OF ROUTINES

A. PRIMARY ROUTINES

RTSINT
INITIALIZES THE REAL TIME SYSTEM (RTS & RTS.F4) AND DI. THE REAL TIME SYSTEM AND DI ARE 'QUIESCENT' AFTER THIS CALL. RTSINT SHOULD BE CALLED BEFORE ANY TELETYPE INPUT/OUTPUT. A CALL TO RTSINT SETS THE CURRENT RUNNING PRIORITY LEVEL TO 1.

RTSPR (PROCSR,IPRLVL,IREPT) (EXTERNAL PROCSR)
SETS UP A PROCESS DESCRIPTOR (PD) FOR THE NAMED SUBROUTINE. IF A "PD" ALREADY EXISTS FOR THE NAMED SUBROUTINE, THE VALUES OF THE PARAMETERS FROM THE PRESENT CALL ARE SUBSTITUTED FOR THOSE ALREADY EXISTING. (IF IT IS NECESSARY TO RUN THE SAME PROCESS "SIMULTANEOUSLY" UNDER DIFFERENT PRIORITY LEVELS OR TIMING CONDITIONS, THE PROCESS WILL HAVE TO BE LOADED TWICE AS DIFFERENT SUBROUTINES IN ORDER TO SCHEDULE BOTH OF THEM; THIS WILL PRECLUDE RE-ENTRANCE PROBLEMS.)

RTSAD (NCHANS,LENGTH,IADBUF(1),IPRLVL,IREPT) (EXTERNAL RTSADS)
SETS UP PARAMETERS AND PD FOR THE STANDARD A/D SAMPLING ROUTINE (RTSADS) CONTAINED IN RTS.F4. THE A/D SAMPLER (RTSADS) MUST BE SCHEDULED BY THE USER. (A/D PARAMETERS ARE AVAILABLE TO THE USER; SEE SECTION I.C.)
***NOTE: A/D OUTPUT IS A 12-BIT 2'S COMPLEMENT INTEGER!

RTSBH (IDIBIT,IDELAY,PROCSR,IPRLVL,IREPT) (EXTERNAL PROCSR)
SET-UP ROUTINE FOR DI INTERRUPT PROCESSORS. SETS UP A PD FOR PROCSR IN THE DI LIST OF PD'S, FOR DI BIT # IDIBIT (IDIBIT = 1 TO 8). (IF NO PD IS SET-UP FOR A PARTICULAR DI BIT, INTERRUPTS ON THAT BIT ARE IGNORED.) IF IDELAY & IREPT ARE BOTH ZERO, THE PROCESS IS RUN IMMEDIATELY AFTER ALL HIGHER PRIORITY PROCESSES ARE COMPLETED. IF IDELAY OR IDIBIT ARE NON-ZERO, THE PROCESS IS PUT ON THE CLOCK QUEUE TO BE RUN IDELAY MILLISECONDS AFTER THE DI PULSE, AND PERIODICALLY (IREPT > 0) MILLISECONDS THEREAFTER.

RTSKED (PROCSR,LDVAL,ISTART) (EXTERNAL PROCSR)
CAUSES PROCSR (RTSADS OR ONE SET UP BY RTSPR) TO BE SCHEDULED (PUT ON THE CLOCK QUEUE), TO BE RUN INITIALLY AT THE TIME INDICATED BY ISTART, AND PERIODICALLY EVERY IREPT MILLISECONDS THEREAFTER IF IREPT > 0. IF IREPT = 0, IT IS RUN ONLY ONCE.

RTSRUN (PROCSR,LDVAL) (EXTERNAL PROCSR)
CAUSES PROCSR (RTSADS OR ONE SET UP BY RTSPR) TO BE PUT ON THE RUN QUEUE AND RUN ONLY ONCE, AS SOON AS ALL HIGHER PRIORITY PROCESSES ARE COMPLETED, INDEPENDENT OF TIMING PARAMETERS.

```
RTSFI (PROCSR,IREPT,ISTART)            (EXTERNAL PROCSR)
 CAUSES PROCSR TO BE RUN FROM THE DI HANDLER, SAVING
 QUEUEING & DE-QUEUEING TIME.  PROCSR WILL BE RUN INITIALLY
 AT TIME ISTART, AND PERIODICALLY (IREPT > 0) IREPT
 MILLISECONDS THEREAFTER.  THIS ROUTINE RUNS AT "HARDWARE
 LEVEL".  HOWEVER, THE PC, AC, MQ, LK, XR & LR ARE SAVED
 AND RESTORED FOR THE USER.  THE DI IS DISABLED DURING THE
 TIME THAT THIS (FASTI) PROCESSOR IS RUNNING, AND IT RUNS
 AT AN EFFECTIVELY INFINITE PRIORITY LEVEL.

RTSDQ (PROCSR)                                (EXTERNAL PROCSR)
 CAUSES PROCSR TO BE DEQUEUED (FROM THE CLOCK AND RUN QUEUES).
 (NO ACTION IS TAKEN IF THE PD IS NOT PRESENT.)

RTSPDQ (IDIBIT)              (IDIBIT = 0 TO 8)
 CAUSES THE DE-QUEUEING OF A BIT PROCESSOR, OR OF THE
 FASTI ROUTINE (IDIBIT = 0; SEE RTSFI).

RTSGO
 ARMS THE REAL TIME SYSTEM, TURNS ON THE DI (& REAL TIME
 CLOCK), WHICH CAUSES A CLOCK INTERRUPT TO THE REAL TIME
 SYSTEM EVERY MILLISECOND (1/1024 SECONDS), AND GOES TO A
 USER SUPPLIED (OR DEFAULT) IDLE PROCESS CALLED 'RTSIDL'.
 THE PD FOR THIS IDLE PROCESS IS SET-UP INTERNALLY, AND RUNS
 AT PRIORITY LEVEL 1.  CONTROL REMAINS IN THE IDLE PROCESS,
 EXCEPT WHEN IT IS NECESSARY TO RUN A HIGHER PRIORITY LEVEL
 PROCESS.

RTSTOP
 TURNS OFF THE REAL TIME CLOCK, AND RE-INITIALIZES THE
 REAL TIME SYSTEM (RTS & RTS.F4).  THE ROUTINE WHICH CALLS
 RTSTOP TAKES ON THE SAME STATUS AS THE MAIN PROGRAM USED
 INITIALLY--A NON-SCHEDULED PROCESS RUNNING AT PRIORITY LEVEL 0
 (PRIORITY LEVEL 1 AFTER RTSINT IS CALLED).
```

RTSDON (RTSDOF) (OR, DION (DIOFF))
ENABLES (DISABLES) THE DIGITAL INTERFACE (DI) AND CLOCK, ALLOWING (PREVENTING) DI INPUT PULSES TO CAUSE INTERRUPTS. NOTE: DI INPUT BITS ARE CLEARED WHEN THE DI IS TURNED ON.

RTSTON (RTSTOF) (OR, DITON (DITOF))
ALLOWS (DOESN'T ALLOW) THE REAL TIME SYSTEM TO "SEE" THE CLOCK PULSE FROM THE DI CLOCK. THESE ROUTINES ALLOW ONE TO TURN TIME ON AND OFF WITHOUT DISABLING THE DI.

DIPAUS (DICONT)
DIPAUS TURNS THE DI OFF, AND "REMEMBERS" THE STATE OF THE DI, SO THAT THE DI MAY BE RETURNED TO ITS ORIGINAL STATE BY A CALL TO DICONT. NOTE: ANY INPUT (INTERRUPT) BITS SET DURING THE PAUSE WILL REMAIN SET & WILL CAUSE AN INTERRUPT IF THE DI IS RETURNED TO AN ENABLED STATE. THUS, A SINGLE CLOCK TICK, AND SINGLE INTERRUPTS ON ANY INPUT BIT WILL NOT BE LOST, BUT ONLY DELAYED BY THE PAUSE.

RTSBON (IDIBIT) (RTSBOF(IDIBIT) (IDIBIT= 9-17)
TURNS DI OUTPUT BIT # IDIBIT ON (OFF).

B. SPECIAL USER'S ROUTINES

LDVAL = IACF(LDVAL) (.LIBR5 ROUTINE)
FUNCTION WHICH GIVES THE VALUE OF THE AC. TO GET THE VALUE OF LDVAL SPECIFIED IN THE PD, THIS MUST BE THE FIRST EXECUTABLE STATEMENT IN THE PROCESS.

IP = IPULSE(IP)
FUNCTION WHICH GIVES THE DI BIT NUMBER WHICH CAUSED THIS PROCESS TO BE RUN. THIS MUST BE THE FIRST EXECUTABLE STATEMENT IN THE BIT PROCESSOR. (IPULSE IS IACF FUNCTION.)

IPLTIM = IPTIME(IP)
FUNCTION WHICH GIVES THE TIME OF THE LAST INTERRUPT CAUSED BY DI BIT # IP.

ITIME = IEXTF(TIME) (.LIBR5 RTN) (EXTERNAL TIME)
INTERRUPT PROTECTED VERSION OF LIBRARY ROUTINE FOR GETTING THE VALUE OF .GLOBL'D VARIABLES. (SEE SECTION I.C.)

RTSJAM (IBUFFR(1)) (NOT RE-ENTRANT)
STORES DATA FROM THE A/D RING BUFFER IN USER'S BUFFER (IBUFFR). THE DATA IN IBUFFR STARTS WITH THE OLDEST DATA POINT FIRST, AND IS MULTIPLEXED (X(N,T)=X(CHANNEL N,TIME T)):
X(1,T1),X(2,T1),...X(NCHANS,T1),X(1,T2),X(2,T2),...
TIME REQUIRED IS ROUGHLY 5.6 USEC. PER WORD IF BOTH BUFFERS ARE IN THE SAME PAGE, 7.2 USEC. PER WORD OTHERWISE.

RTSCPY (ICHAN,IBUFFR(1),IBUFR2(1)) (NOT RE-ENTRANT)
STORES ONE CHANNEL (ICHAN) OF DATA IN IBUFR2. INTENDED FOR USE AFTER RTSJAM; DATA IN IBUFFR IS ASSUMED TO BE TIME-ORDERED AND MULTIPLEXED.

RTSGET (IBUFFR(1))
RTSGT2 (NDIMEN,IBUFFR(1,1))
STORES DATA FROM A/D RING BUFFER IN IBUFFR, TIME-ORDERED AND DE-MULTIPLEXED. TIME REQUIRED IS ROUGHLY 10 USEC. PER WORD. THIS ROUTINE IS RE-ENTRANT, AND CAN HANDLE NO MORE THAN SEVEN CHANNELS OF DATA.

GETBF1 (LENGTH,NCHANS,INDEX,IRINGB(1),IUSEBF(1))
GETBF2 (NDIMEN,LENGTH,NCHANS,INDEX,IRINGB(1),IUSEBF(1,1))
GETS TIME-ORDERED, DE-MULTIPLEXED DATA FROM A MULTIPLEXED RING BUFFER. (THIS ROUTINE IS USED BY RTSGET & RTSGT2, TO GET DATA OUT OF THE A/D RING BUFFER.)

ADCONV
CAUSES THE A/D RAW DATA TO BE MULTIPLIED BY A CONVERSION FACTOR. A DEFAULT VALUE OF 1.Ø IS USED UNLESS THE USER CALLS:

ADCSET (ICHAN,FACTOR)
SETS THE CONVERSION FACTOR FOR THE A/D DATA ON CHANNEL # ICHAN TO THE INTEGER EQUIVALENT OF THE REAL VARIABLE 'FACTOR'. (SEE ADCONV.) THE MAGNITUDE OF FACTOR MUST BE LESS THAN (NOT =) 16.Ø. A NUMBER AS SMALL AS .125 WILL BE REPRESENTED WITH FULL 12 BIT ACCURACY.

ADCALB (ICHAN,IADOLD,IADNEW)
CALIBRATE A/D OUTPUT: IADOLD IS THE CURRENT (2'S COMPL) OUTPUT OF THE A/D SAMPLER (CHANNEL # ICHAN); THE CONVERSION FACTOR IS MODIFIED SO THAT IADNEW WILL BE THE NEW A/D OUTPUT FOR THE SAME INPUT SIGNAL. IT IS SUGGESTED THAT PEAK-TO-PEAK VALUES OF AN AC SIGNAL BE USED FOR CALIBRATION.

C. USER SUPPLIED ROUTINES

MODFAD -- MACRO SUBROUTINE
FUNCTION TO MODIFY THE RAW A/D DATA BEFORE IT IS STORED IN THE A/D RING BUFFER. ICHANL IS THE CHANNEL NUMBER, NEWDAT IS THE NEW (RAW) DATA JUST SAMPLED, OLDDAT IS THE OLD (MODIFIED) DATA ABOUT TO BE DISCARDED. NEWDAT (RAW) IS IN THE AC ON ENTRY; THE MODIFIED FORM OF NEWDAT MUST BE IN THE AC ON EXIT. A DEFAULT VERSION OF MODFAD (WHICH SIMPLY RETURNS) IS ON THE USER'S LIBRARY. MODFAD IS CALLED FROM THE A/D SAMPLER -- NAIVE USAGE OF MODFAD CAN CAUSE CATASTROPHIC ERRORS WHICH ARE DIFFICULT TO DETECT! DON'T WRITE YOUR OWN VERSION OF MODFAD WITHOUT FIRST CONSULTING A SENIOR PROGRAMMER!!!!!
THE INDEX AND LIMIT REGISTERS MUST BE PRESERVED BY THE USER!

RTSAVE -- USER'S ROUTINE TO SAVE DATA (IN CASE OF CRASH)
THIS SUBROUTINE SHOULD CONTAIN THE CLOSING OF FILES AND WHATEVER ELSE IS NEEDED TO RECOVER DATA AFTER AN UNRECOVERABLE ERROR OCCURS DURING A RUN. NO INPUT/OUTPUT SHOULD BE ATTEMPTED FROM THE ROUTINE. THIS ROUTINE IS AUTOMATICALLY CALLED IN THE STANDARD RECOVERY PROCEDURE DESCRIBED LATER (POST-MORTEM DUMP). THE SYSTEM LIBRARY CONTAINS A DEFAULT VERSION WHICH SIMPLY RETURNS.

RTSIDLE -- IDLE ROUTINE. THE DEFAULT VERSION OF RTSIDL SIMPLY JUMPS TO ITS OWN MEMORY LOCATION--"SITS THERE IDLING". CONTROL IS TRANSFERED HERE WHEN THERE IS NOTHING ELSE TO DO. USER'S VERSIONS MAY BE USED TO TEST DATA SWITCHES, ETC. USER'S VERSION MUST NOT RETURN!

III. EXPLANATIONS

THE DIGITAL INTERFACE (DI) IS AN 18 BIT REGISTER WHICH ACTS, IN A SENSE, LIKE AN INPUT/OUTPUT DEVICE. THE LEFT HALF OF THE REGISTER (BITS 0 TO 8) IS RESERVED FOR INPUT SIGNALS, THE RIGHT HALF (BITS 9 TO 17) IS RESERVED FOR OUTPUT SIGNALS. WHENEVER A PULSE IS RECIEVED ON ONE OF THE INPUT LINES (0 TO 8), THE CORRESPONDING BIT IN THE DI IS SET AND A PROGRAM INTERRUPT IS GENERATED.

(NORMALLY, PROGRAM INTERRUPTS ARE GENERATED BY STANDARD I/O DEVICES. THE INTERRUPT CAUSES IMMEDIATE TRANSFER OF CONTROL TO THE MONITOR <EQUIVALENT TO JMS* (0)> IN LOW ORDER MEMORY. THE MONITOR DETERMINES WHICH I/O DEVICE CAUSED THE INTERRUPT, GIVES CONTROL TO AN APPROPRIATE "DEVICE HANDLER", AND RETURNS CONTROL TO THE INTERRUPTED PROGRAM (RESTORING ALL MODIFIED REGISTERS) AFTER THE I/O INTERRUPT HAS BEEN PROCESSED. D.E.C. HAS MADE ALLOWANCES FOR THE ADDITION OF 3 USER'S DEVICES; OUR "HOME-MADE" DIGITAL INTERFACE IS CONNECTED TO THE COMPUTER AS AN I/O DEVICE.)

OUR REAL TIME CLOCK IS PERMANENTLY CONNECTED TO BIT 0 OF THE DI. IT GENERATES A PULSE EVERY 1/1024TH SECOND, SETTING BIT 0 AND CAUSING A PROGRAM INTERRUPT.

WHEN THE MONITOR DETERMINES THAT THE INTERRUPT WAS GENERATED BY THE DI, IT TRANSFERS CONTROL TO THE DI "HANDLER" (FILE NAME: GENDIG). THE DI HANDLER FIRST READS THE DI REGISTER TO DETERMINE WHICH INPUT BIT (OR BITS) IS SET--IE, WHICH INPUT LINE CAUSED THE INTERRUPT. IF BIT 0 IS SET (BY THE REAL TIME CLOCK), THE TIME VARIABLE 'TIME' IS INCREASED BY 1. IF SOME OTHER INPUT BIT IS SET, THE BIT PROCESSOR FOR THAT BIT IS PUT ON THE RUN OR CLOCK QUEUE. (MORE ON THIS LATER.)

ON ALL PROGRAM INTERRUPTS, DI OR OTHER, THE REAL TIME SYSTEM (RTS) TAKES CONTROL AFTER THE INTERRUPT HAS BEEN PROCESSED, AND DETERMINES IF ANY NEW PROCESSES NEED TO BE RUN, BEFORE CONTROL IS RETURNED TO THE INTERRUPTED PROGRAM. THE INTERRUPT JUST DESCRIBED (PROGRAM INTERRUPT) IS CALLED AN "HARDWARE INTERRUPT"; IT IS BUILT INTO THE COMPUTER. THE REAL TIME SYSTEM MAY DETERMINE THAT IT IS TIME TO RUN A HIGHER PRIORITY PROCESS, AND INTERRUPT THE CURRENTLY RUNNING PROCESS UNTIL THE HIGHER PRIORITY PROCESS IS COMPLETED. THIS IS CALLED A "SOFTWARE INTERRUPT"; ALL THE BOOKKEEPING, DECISION MAKING, ETC. REGARDING SOFTWARE INTERRUPTS IS THE FUNCTION OF THE REAL TIME SYSTEM, DESIGNED BY PETE HARRIS.

THE RTS.F4 PACKAGE IS DESIGNED TO TAKE CARE OF MANY OF THE FUNCTIONS REQUIRED OF THE USER, AND PARTICULARLY TO MAKE THE RTS AVAILABLE TO THE FORTRAN USER.

THE RTS HAS CONTROL AFTER EVERY HARDWARE INTERRUPT AND AFTER ANY PROCESS STARTED BY IT IS COMPLETED. THE DECISIONS MADE BY IT ARE BASED PRIMARILY ON THE STATUS OF THREE QUEUES: CLOCK QUEUE (CLKQ), RUN QUEUE (RUNQ), AND INTERRUPT STACK (ISTACK).

THE CLKQ IS A LIST OF PROCESSES WHICH ARE SCHEDULED TO BECOME ACTIVE AT A GIVEN CLOCK TIME. CLKQ IS SORTED ACCORDING TO RUN TIME (WITH TIES GOING TO THE HIGHER PRIORITY PROCESS). WHEN THE CLOCK TIME (CURRENT TIME) IS EQUAL OR GREATER THAN THE RUN TIME OF THE PROCESS AT THE TOP OF THE CLKQ, THAT PROCESS IS TAKEN OFF THE CLKQ AND PUT ON THE RUNQ. THE PROCESS IS NOW ACTIVE IN THE SENSE THAT IT WILL BE RUN AS SOON AS ALL EQUAL OR HIGHER PRIORITY PROCESSES ARE COMPLETED. IF THE RE-PITITION PERIOD (IREPT) IS > 0, IREPT IS ADDED TO THE RUN TIME IN THE PD, AND THE PROCESS IS RESCHEDULED (PUT BACK ON THE CLKQ), TO BE RUN AT THE NEW RUN TIME. (THE INITIAL RUN TIME IS ESTART.) IF THE PROCESS TAKEN OFF THE CLKQ HAS A REPETITION PERIOD OF -1, IT IS NOT PUT ON THE RUNQ, NOR PUT BACK ON THE CLKQ: IT IS "DE-SCHEDULED".

THE RUNQ IS A LIST OF "ACTIVATED" PROCESSES, SORTED ACCORDING TO PRIORITIES (TIES GOING TO THE FIRST ENTERED). THE PROCESS AT THE TOP OF RUNQ WILL BE RUN AS SOON AS ALL HIGHER & EQUAL PRIORITY PROCESSES ALREADY STARTED HAVE BEEN COMPLETED.

ISTACK IS A LIST OF PROCESSES WHICH HAVE BEEN INTERRUPTED BY RTS. ISTACK IS ARRANGED IN CHRONOLOGICAL ORDER--WHICH IS ALSO IN ORDER BY PRIORITY, SINCE A PROCESS CAN BE INTERRUPTED ONLY BY A HIGHER PRIORITY PROCESS, AND INTERRUPTED PROCESSES MUST BE COMPLETED BEFORE OTHER PROCESSES OF EQUAL PRIORITY ARE RUN.

IN SUMMARY, THE CLKQ IS A LIST OF "INACTIVE" PROCESSES ("LADIES IN WAITING") WHICH WILL BE "ACTIVATED" (PUT ON THE RUNQ) AT THE TIME INDICATED BY THEIR RUN TIME. (SEMANTICS: A "SCHEDULED" PROCESS IS ONE THAT HAS BEEN PUT ON THE CLKQ.) THE "ACTIVE" PROCESSES -- THOSE ON RUNQ WAITING TO BE STARTED, THOSE IN ISTACK WHICH WERE INTERRUPTED, AND THE ONE CURRENTLY RUNNING -- ARE RUN ACCORDING TO PRIORITY LEVEL. TIES GO FIRST TO THE CURRENTLY RUNNING PROCESS, NEXT TO AN INTERRUPTED PROCESS, AND FINALLY IN RUNQ TO THE FIRST PROCESS ENTERED.

THE MAIN FORM OF COMMUNICATION WITH RTS INVOLVES THE USE OF PROCESS DESCRIPTORS ("PD'S"), WHICH CONTAIN THE ADDRESS OF A PROCESSOR (SUBROUTINE) ITS PRIORITY LEVEL, REPETITION TIME, RUN TIME, AND A CODE WORD TO BE LOADED INTO THE AC WHEN IT IS STARTED.

THE INITIALIZATION (RTSINT) OF RTS.F4 INVOLVES: PRINTING THE REAL TIME SYSTEM VERSION NUMBER, INITIALIZING SOME PARAMETERS FOR THE POST-MORTEM DUMP ROUTINES, INITIALIZING (BUT NOT ENABLING) THE DI AND RTS, SETTING THE PRIORITY LEVEL FOR THE CURRENTLY RUNNING PROCESS TO 1, AND SETTING UP THE PD FOR THE IDLE PROCESS (RTSIDL). (THE PATCHING OF THE MONITOR SO THAT RTS GETS CONTROL AFTER HARDWARE INTERRUPTS IS NOT DONE UNTIL RTSGO IS CALLED.)

THE ADDRESS OF A PROCESSOR, ITS PRIORITY LEVEL, AND ITS REPETITION TIME ARE PUT INTO A PD BY SUBROUTINE RTSPR. SUBROUTINE RTSADS IS USED TO SET UP A/D PARAMETERS AND PUT THE PRIORITY LEVEL & REPETITION TIME INTO THE PD FOR THE STD. A/D SAMPLER (RTSADS). (THE USER HAS TO SCHEDULE RTSADS.) SUBROUTINE RTSKED IS USED TO PUT THE START TIME AND AC WORD (LDVAL) INTO THE PD, AND TO PUT THE PROCESSOR ON THE CLOCK QUEUE. SUBROUTINE RTSRUN IS USED TO PUT LDVAL IN THE PD, AND PUT THE PROCESSOR ON THE RUN QUEUE. RTSRUN AND RTSKED CAN BE USED ONLY FOR PROCESSORS WHICH HAVE BEEN SET UP BY RTSPR (OR RTSAD, IN THE CASE OF RTSADS).

A SINGLE PROCESS ("FASTI") CAN BE RUN AT "HARDWARE LEVEL". THIS MEANS IT RUNS AS PART OF THE DI INTERRUPT HANDLER, AND IS NOT AFFECTED BY RTS PRIORITY CONSIDERATIONS. THIS SPECIAL FEATURE WAS INCLUDED FOR THE VERY SPECIAL CASES WHERE SOME KIND OF "FAST" PROCESS NEEDS TO BE RUN, AND USING "ORDINARY" PROCEDURES WOULD INVOLVE TOO MUCH OVERHEAD TIME. THE PC, AC, LK, MQ, XR, & LR ARE SAVED AND RESTORED BY THE HANDLER. ANY OTHER REGISTERS (E.G., THE ENTRY WORD TO RUN OR SCHEDU, OR AUTO-INCREMENT REGISTERS) WHICH ARE USED BY THE "FASTI" ROUTINE MUST BE RESTORED BEFORE LEAVING.

***NOTE: THE DI IS DISABLED DURING THE TIME THAT FASTI IS BEING RUN. CLOCK TICKS AND INPUT BIT INTERRUPTS WILL BE DELAYED UNTIL FASTI IS FINISHED.

THE USE OF DI PULSES MAKES IT POSSIBLE TO SYNCHRONIZE DATA PROCESSING WITH EXTERNAL EVENTS -- SUCH AS THE STIMULUS AND RESPONSE SIGNALS IN A REACTION TIME EXPERIMENT. PROCESSORS WHICH ARE TO BE SCHEDULED OR RUN AS A RESULT OF DI PULSES, HAVE TO BE SET UP BY SUBROUTINE RTSBH. THE PD'S WHICH ARE SET UP BY RTSBH ARE DISTINCT FROM THE PD'S SET UP BY RTSPR (OR RTSAD).

WHEN THERE IS MORE THAN ONE INPUT (INTERRUPT) BIT SET ON THE DI (AS MAY BE THE CASE AFTER A PAUSE), THE LOWEST BIT SET IS PROCESSED FIRST (GIVING THE CLOCK & FASTI HIGHEST PRIORITY). FOR INTERRUPT BITS (2 TO 8), THE CURRENT TIME IS SAVED & IS AVAILABLE TO THE USER THROUGH THE IPTIME ROUTINE. I.E., THE TIME OF THE LAST PULSE ON ANY INPUT BIT IS ALWAYS AVAILABLE. IF THE DELAY TIME FOR THE PROCESSOR ASSIGNED TO BIT CAUSING THE INTERRUPT IS ZERO, THE PROCESSOR IS PUT ON THE RUN QUEUE TO BE RUN IMMEDIATELY. IF THE DELAY TIME IS NOT ZERO, THE PROCESSOR IS PUT ON THE CLOCK QUEUE, TO BE RUN INITIALLY AT A LATER TIME--CORRESPONDING TO THE INDICATED DELAY. (IN THIS CASE, IF THE REPETITION TIME IS NOT ZERO, THE PROCESS WILL BE RUN PERIODICALLY.) IF THE DELAY TIME IS ZERO (THE PROCESS IS PUT ON THE RUN QUEUE), AND THE REPETITION TIME IS NOT ZERO, THE PROCESSOR IS ALSO PUT ON THE CLOCK QUEUE, TO BE RUN AGAIN AT A LATER TIME--CORRESPONDING TO THE REPETITION TIME.

IV. MISCELLANEOUS NOTES

PART OF THIS PACKAGE IS CONTAINED IN A SPECIAL FILE ON DK0. THE LOAD STRING SHOULD BE AS FOLLOWS (USING LOADER V0B): (ALSO SEE DESCRIPTION OF POST-MORTEM DUMPS.)

```
>←MAIN,SUBR1,SUBR2,...,SUBRN;RTS.K0
```

ALL PROCESSOR NAMES WHICH ARE USED IN ANY CALLS TO RTS.F4 MUST APPEAR IN AN EXTERNAL STATEMENT IN THE PROGRAM OR SUBROUTINE WHICH MAKES THE CALL. IF A SUBROUTINE HAS ITS OWN NAME IN AN EXTERNAL STATEMENT, ITS EXTERNAL (.GLOBL) LIST WILL BE THOUROUGHLY "MESSED UP" -- DON'T DO IT!

DDT USES AUTO-INCREMENT REGISTER 17, WHICH IS ALSO USED BY RTSGET & RTSGT2, WHEN 7 CHANNELS OF DATA ARE USED. DDT MAY NOT BE USED WHEN RTSGET OR RTSGT2 ARE USED WITH NCHANS = 7.

THERE IS A SPECIAL FILE ('ADKEYS') ON DK0, WHICH CONTAINS A SPECIAL VERSION OF THE A/D SAMPLER. ADKEYS TAKES THE VALUE ENTERED IN THE LEFT HALF OF THE DATA SWITCHES AS THE SAMPLE VALUE FOR ODD CHANNELS, AND THE VALUE IN THE RIGHT HALF FOR THE EVEN CHANNELS. THE VALUES ARE TREATED AS 1'S COMPLEMENT INTEGERS IN MULTIPLES OF 8 (LOW ORDER 3 BITS MISSING). (NEGATIVE VALUES ARE EASILY ENTERED BY SETTING ALL SWITCHES IN THE APPROPRIATE HALF OF THE KEYS (-0), AND SETTING SWITCHES OFF WHICH CORRESPOND TO THE MAGNITUDE.) E.G., +25 & -25 (OCTAL) ARE SET BY 000 010 101 & 111 101 010, RESPECTIVELY. THESE VALUES ARE READ AS +250 & -250 (OCTAL), RESPECTIVELY. LOAD THIS AS FOLLOWS:

```
>←MAIN,...,SUBRN;ADKEYS,RTS.K0
```

V. ERROR CODES & POST-MORTEM DUMPS

IOPS 67 ERROR CODES:

```
000000  FILE RTSABS NOT LOADED (SHOULDN'T HAPPEN)
0NNNNN  NNNNN>7: (RTSGET OR RTSGT2)
        NNNNN=NCHANS= TOO BIG.
1AAAAA  RTSPR: TOO MANY PD'S.
        AAAAA=ADDRESS OF PROCSR IN CALL TO RTSPR.
2AAAAA  RTSPH: DI BIT # (IDIBIT) TOO LARGE.
        AAAAA=ADDRESS OF CALL TO RTSBH
3AAAAA  RTSKED OR RTSRUN: PD NOT IN LIST.
        AAAAA=ADDRESS OF PROCSR IN CALL.
```

RT SYSTEM-DETECTED ERRORS: IOPS 70 XXXXXX

```
1--SYSTEM OVERLOAD (QUEUES TOO BIG)
2--TOO MANY PRIORITY LEVELS
3--SYSTEM ERROR: RUNQ EMPTY ON PRE-EMPT
4--SYSTEM ERROR: ISTACK EMPTY ON RESTORE
5--SYSTEM ERROR: CLKQ EMPTY ON CLOCK DEQUING
6--SYSTEM ERROR: RUNQ EMPTY ON EXIT FROM SUB. "RUN"
7--SYSTEM ERROR: CLKQ EMPTY ON EXIT FROM SUB. "SCHEDULE"
10--FORTRAN PROCESS RUN ABOVE FORTRAN LEVEL
```

THE POST MORTEM DUMP PROGRAM MAY BE LOADED WITH THE USER'S PROGRAMS (RESIDENT), OR LATER, AFTER A Q-DUMP (NON-RESIDENT). BOTH APPROACHES WILL BE DESCRIBED AT ONCE.

AFTER A "CRASH" IN WHICH THE COMPUTER "HANGS" (E.G., AN ILLEGAL MEMORY REFERENCE), THE RESET BUTTON WILL HAVE TO BE PUSHED IN ORDER TO GET THE COMPUTER GOING AGAIN. IN THIS CASE, ALL THE FOLLOWING REGISTERS SHOULD BE RECORDED -- BEFORE THE RESET BUTTON IS PUSHED!

```
(000  001  010  011  100  101  110  111 : BINARY)
(  0    1    2    3    4    5    6    7 : OCTAL)
```

#DIGITS	REGISTER	VALUE (OCTAL)	
(2)	INSTRUCTION	......	(ALWAYS)
(6)	PC	....	(ALWAYS)
(6)	AC	....	(ONLY IF RESET IS NEEDED)
(6)	LR	....	(")
(6)	XR	....	(")
(6)	MQ	....	(")

LOADING THE PROGRAMS (GLOAD: LOADER V9P)

```
>+MAIN,SUBR1,...,SUBRN ;RTS.M0          (NON-RESIDENT)

>+MAIN,SUBR1,...,SUBRN ;RTS.M0,RTS.PM   (RESIDENT)
```

WHEN IN A LOOP, AFTER A "CRASH", ETC. -- ANY TIME YOU WISH TO TAKE A POST-MORTEM DUMP,

1. PUSH THE STOP BUTTON
2. RECORD THE INSTR, PC, AC, LR, XR, & MQ REG'S, AS NECESSARY
3. PUT 00045 (OCTAL) IN THE ADDRESS KEYS, PUSH STOP (OR STOP/RESET) & THEN START.

RESIDENT VERSION:

4. THE POST MORTEM DUMP WILL BE PRINTED IMMEDIATELY, FOLLOWED BY A ↑Q. TYPE A '1' -- ONE CHARACTER ONLY. (THE Q-DUMP MAY BE USED FOR FURTHER DEBUGGING.)

NON-RESIDENT VERSION:

4. TYPE A '1' FOLLOWING THE ↑Q. (ONE CHARACTER ONLY)
5. WHEN THE MONITOR RETURNS, LOAD THE DUMP PROGRAM (GLOAD:)

```
>+ ;RTS.PM
```

(NOTE: THE DISK SHOULD HAVE A Q AREA SAVED.)

J.R. JOHNSTON

```
PIP V13A

>T \\L VP←DT2


>T TT (A)←DT2 TAPE#S'003


SYSTEM TAPE #003
SOURCE FILES OF
DOCUMENTATION.


PIP V13A

>T TT (A)←DT2 .LIBR5 DOC
```

7 DEC 1973
(FILE NAME IS FIRST NAME LISTED, UNLESS OTHERWISE NOTED.)
(REAL TIME SYSTEM ROUTINES DOCUMENTED IN FILE RTSVAX DOC)

FORTRAN (MACRO) ROUTINES:

```
DELAY        (23 WORDS) TIME DELAY -- MULTIPLES OF 10 MICROSECONDS.

     CALL DELAY (IT)
          WAITS 10*IT MICROSECONDS  (20 US. MINIMUM)


DSPLAY       (304 WRDS)  SCALED DISPLAY OF SEVERAL CHANNELS OF INTEGER
DSINIT       DATA ON THE SCOPE (WITH PRINT OF MIN & MAX VALUES).

     CALL DSINIT
          INITIALIZES (1ST TIME ONLY) & ERASES SCOPE.

     CALL DSPLAY (IRAY(1),NELM,ICHN,NCHN,ITAB)
          DISPLAYS NELM # OF INTEGER DATA (STARTING WITH
          IRAY(1)) ON THE ICHN-TH SECTION OF THE SCOPE
          (FROM THE TOP), WITH VERTICAL SIZE EQUAL TO
          (FULL SIZE)/NCHN.  A DECIMAL GRID IS DISPLAYED
          AT THE ZERO VALUE OF THE DATA FOR EACH CHANNEL
          --TO MARK THE POSITION OF EACH ARRAY ELEMENT.
          ITAB (OPTIONAL): -1, NO PRINT; 0 TO 3, PRINT MIN & MAX
          VALUES AT BOTTOM OF DISPLAY, ITAB/4 DISTANCE FROM LEFT.


DUMP         PRINT OCTAL AND POSSIBLY INTEGER AND/OR FLOATING
             VALUES OF VARIOUS REGISTERS, AND SELECTED MEMORY
             LOCATIONS.  (250 WORDS + PRINT ROUTINES)

     CALL DUMP (MCODE,A,B,I,...)

          MCODE   REGISTERS
          0       LK,AC,MQ,SW'S (OCTAL)
          1       AC,MQ(INTEGER); SC,XR,LR(OCTAL)
          2       (AC,MQ)(FLOATING); .AA,.AB,(.AA,.AB)
                                        (OCTAL & FLOATING)
          (MCODE OPTIONS ARE "OR-ED".  EG, 3 GIVES ALL OPTIONS)

          A,B,I,... ARE ADDRESSES OF MEMORY LOCATIONS TO BE
          DUMPED--IN THE MODES INDICATED BY MCODE.
          (SEE ADDITIONAL DOCUMENTATION, PAGE A1.)

MINA         FIND MINIMUM VALUE OF AN INTEGER ARRAY.
MAXA         FIND MAXIMUM VALUE OF AN INTEGER ARRAY.

     IMIN = MINA(IRAY(1),NELM)

     IMAX = MAXA(IRAY(1),NELM)

          IRAY(1) = 1ST ELEMENT OF ARRAY, NELM = # ELEMENTS.

     (FILE NAME: MINMAX, 44 WORDS)
```

FORTRAN (&MACRO) ROUTINES (CONTINUED)

SWITCH
ALLSW
ANYSW
(45 WRDS)

LOGICAL FUNCTIONS, RETURN "TRUE" (-1) IF ANY, OR ALL, OF THE TESTED SWITCHES ARE ON. THE SWITCHES ARE NUMBERED FROM THE RIGHT, STARTING WITH 1; AC17=1,AC16=2,...,ACØ=18.
IF THE STATED CONDITIONS ARE NOT MET, A VALUE OF "FALSE" (Ø) IS RETURNED.

TYPICAL USAGE:

```
LOGICAL  SWITCH,ALLSW,ANYSW

IF  (.NOT. SWITCH(1))    ---
          DO ---, IF SW.1 IS OFF.

IF  (ALLSW(1,2,5))    ---
          DO ---, IF SWITCHES 1,2, AND 5 ARE ON.

IF  (ANYSW(1,3,6))    ---
          DO ---, IF SWITCHES 1,3, OR 6 ARE ON.
```

IEXTF
ISEXT
FEXTF
FSEXT

GET OR SET VALUE OF INTEGER OR FLOATING VARIABLE WHICH IS .GLOBL'D IN A MACRO ROUTINE. THE NAME OF THE VARIABLE NEED NOT CONFORM TO THE FORTRAN MODE CONVENTION.

TYPICAL USAGE:

```
EXTERNAL  INUMBR,FNUMBR

I = IEXTF(INUMBR)

A = FEXTF(FNUMBR)

CALL ISEXT (INUMBR,<INTEGER EXPRESSION>)

CALL FSEXT (FNUMBR,<FLOATING EXPRESSION>)
```

(FILE NAMES: IEXT.F, 17 WORDS; FEXT.F, 37 WORDS)

FORTRAN (AMACRO) ROUTINES (CONTINUED)

EXTADJ (51 WRDS) ADJUST (EQUIVALENCE) AN ARRAY TO START AT THE ADDRESS CONTAINED IN A .GLOBL'D VARIABLE. THIS ALLOWS DYNAMIC ALLOCATION OF BUFFERS AT RUN TIME. (25 WORDS)

TYPICAL USAGE:

```
      DIMENSION ITEMP(1)
      EXTERNAL  POINTR

      CALL EXTADJ (ITEMP,POINTR,N1,N2,N3)
```

THE DIMENSIONS N1,N2,N3 ARE OPTIONAL ARGUMENTS; N # OF DIMENSIONS MUST BE INCLUDED FOR AN N-DIMENSIONAL ARRAY -- OR NONE FOR NO CHANGE.

THIS MAY ALSO BE USED WITH NON-EXTERNAL POINTERS. THE FOLLOWING EXAMPLE ALLOWS THE ALLOCATION OF UP TO 10 BUFFERS:

```
      DIMENSION  IBUFF1(1),IBUFF2(1),ITEMP(1)
      COMMON  IBUFR(4096),IPNTR(10),NEXT,NEXTAD

      NEXT = 1
      CALL EXTADJ (NEXTAD,IBUFR)

      CALL EXTADJ (IBUFF1,NEXTAD,ISIZE1)
      IPNTR(NEXT) = NEXTAD
      NEXTAD = NEXTAD + ISIZE1
      NEXT = NEXT + 1

      CALL EXTADJ (IBUFF2,NEXTAD)
      IPNTR(NEXT) = NEXTAD
      NEXTAD = NEXTAD + ISIZE2
      NEXT = NEXT + 1

      CALL EXTADJ (ITEMP,NEXTAD)
C     (TEMPORARY BUFFER: NOT ALLOCATED)
      ...
```

THE BUFFERS ALLOCATED HERE ARE AVAILABLE TO OTHER ROUTINES, USING EXTADJ & THE ADDRESSES IN THE IPNTR ARRAY. ADDITIONAL BUFFERS MAY BE ALLOCATED IN OTHER ROUTINES, USING EXTADJ & THE PARAMETERS IN COMMON.

FORTRAN (& MACRO) ROUTINES (CONTINUED)

FAST FOURIER TRANSFORM ROUTINES

(FILE NAME ISFFTL.W1A, OR HIGHER [WEDGED FILE])

```
ISFFT      INTEGER, SUPER FAST FOURIER TRANSFORM  (444 WRDS + SINES)
ISFFTR     OF COMPLEX OR REAL SERIES

    NSHFTS = ISFFT (IRRAY(1),MPOWR,ICODE)
    NSHFTS = ISFFTR (IRRAY(1),MPOWR,ICODE)

           LET N = 2**MPOWR,

         IRRAY IS AN ARRAY OF:
           N COMPLEX INTEGERS (ISFFT),
             X0R,X0I,X1R,X1I,X2R,...,XNR,XNI,
           OR, N NON-COMPLEX INTEGERS (ISFFTR)
         MPOWR IS THE POWER OF 2 INDICATING THE NUMBER OF
           (COMPLEX OR REAL) VALUES IN THE SERIES.
         ICODE INDICATES DIRECTION OF TRANSFORM:
           +, DIRECT;  -, INVERSE.
         NSHFTS IS THE NUMBER OF SHIFTS MADE TO AVOID OVERFLOW,
           AND FOR NORMALIZATION.  THE ACTUAL VALUE OF THE
           TRANSFORM IS THE OUTPUT SERIES TIMES 2**NSHFTS.
```

THE TRANSFORMS ARE DEFINED:

A(K) = SUM,J [X(J) * EXP(-2*PI*I * J*K/N)],

X(J) = SUM,K [A(K) * EXP(2*PI*I * J*K/N] / N, (INVERSE),

WHERE I = SQRT(-1), & J,K = 0,1,2,...,N-1

FOR ISFFTR, THE OUTPUT IS ORDERED:
A(0),A(N/2),AR(1),AI(1),AR(2),AI(2),...,AI(N/2-1)
WHICH IS ALSO THE REQUIRED FORM OF INPUT FOR INVERSE.

ISFFT & ISFFTR ARE CONSIDERABLY FASTER (ISFFT TAKES ROUGHLY 87% AS LONG; ISFFTR ROUGHLY 45% AS LONG), CONSIDERABLY SHORTER (ISFFT & ISFFTR, WITH FLOATING CONVERSION AND AN EQUIVALENT SINES TABLE REQUIRE 692 WORDS), AND CONSIDERABLY MORE ACCURATE (ROUGHLY 1/2 AS MANY BITS IN ERROR) AS COMPARED TO THE INTEGER ROUTINE (IFFT) IN THE UFFT PACKAGE. UFFT REQUIRES 1334 WORDS (1078 WITH THE SUPERFLUOUS LAST TWO QUADRANTS OF THE SINES TABLE REMOVED). UFFT REQUIRES LOADING OF RELEAE, ISFFT DOES NOT.

THIS PACKAGE USES A PRECOMPILED SINES TABLE. (THE F4 PROGRAM 'SINGEN' GENERATES 4 MACRO COMPILABLE SINES TABLES: SINES1, SINES2, SINES3, & SINES4. THE TABLES ARE 514, 258, 130, & 66 WORDS LONG; AND MAY BE USE WITH TRANSFORMS OF LENGTH 2048,1024, 512, 256 WORDS RESPECTIVELY.) THE SINES TABLE MAY BE LARGER THAN NEEDED. (SINES2 IS ON LIBRARY. ANOTHER TABLE MAY BE USED BY INCLUDING IT IN THE LOAD STRING.) AN IOPS00 ERROR IS GENERATED IF THE SINES TABLE IS TOO SMALL. NOTE: A SINES TABLE LARGE ENOUGH FOR 2*N VALUES MUST BE USED WITH THE AUTOCORRELATION ROUTINE ISAUTR (DESCRIBED BELOW).

FORTRAN (&MACRO) ROUTINES (CONTINUED) (FFT ROUTINES)

*** NOTE: FOR A REAL SERIES, A(0) & A(N/2) ARE SPECIAL CASES
OF A(N-K) = A(K)** : (E.G. N=8) ***

```
A0  A1  A2  A3  A4  A5  A6  A7  A8=A0
I   I   I   I   II  I   I   I   I
I   I   I   I.......I   I   I   I
I   I   I...............I   I   I
I   I.......................I   I
I...............................I
```

ISPOWR GET POWER FROM OUTPUT OF ISFFTR (169 WORDS)
ISPOW GET POWER FROM OUTPUT OF ISFFT

```
NSHFTS = ISPOWR (IA(1),IP(1),MPOWR)

NSHFTS = ISPOW (IA(1),IP(1),MPOWR)
```

THE POWER COEFFICIENT
MAGNITUDE[A(K)]**2 / 2**(NSHFTS+MPOWR)
IS STORED IN IP(K) AS (ISPOWR & ISPOW):
P(0),P(1),P(2),....P(N/2),P(N/2-1),...,P(1)
P(0),0, P(1),0, ..., P(N-1),0 (COMPLEX)
NSHFTS > -MPOWR, ONLY IF REQUIRED TO AVOID OVERFLOW.

IA IS THE ARRAY OF FOURIER COEFFICIENTS COMPUTED BY ISFFTR.
IP IS THE OUTPUT ARRAY FOR THE POWER COEFFICIENTS, WHICH MAY BE THE SAME AS THE INPUT ARRAY (IA).
MPOWR IS THE POWER OF 2: N = 2**MPOWR.

ISAUTR NON-CYCLIC AUTOCORRELATION FUNCTION OF A REAL SERIES (154 WORDS)

```
NSHFTS = ISAUTR (IX0(1),IX(1),MPOWR,ICODE,IXAVG,IG0)
```

IX0 IS THE INPUT SERIES, N WORDS LONG. IX0 MAY BE THE SAME AS THE OUTPUT ARRAY (IX) BELOW.
IX IS THE OUTPUT ARRAY, 2N WORDS LONG: THE FIRST N WORDS CONTAIN THE CORRELATION FUNCTION, THE SECOND N WORDS ARE WORKING STORAGE.
MPOWR IS THE POWER OF 2: N = 2**MPOWR.
ICODE NOT= 0 INDICATES THAT THE MEAN OF THE SERIES IS TO BE SUBTRACTED FROM THE SERIES, FOR COMPUTATION.
IXAVG IS THE AVERAGE (MEAN) OF THE SERIES.
IG0 * 2**NSHFTS IS THE ZERO-LAG VALUE OF THE CORRELATION FUNCTION.

THE FIRST N VALUES OF THE OUTPUT ARRAY CONTAIN THE NORMALIZED AUTOCORRELATION FUNCTION:
IX(L+1) = [G(L) / G(0)] * 131072.0, L = 0,1,2,...,N-1
(1.0 IS REPRESENTED BY 377777 (OCT), OR .999992)

THE NON-CYCLIC AUTOCORRELATION FUNCTION IS DEFINED:
G(L) = SUM,J [X(J)*X(J+L)] / (N-L), J = 0,1,2,...N-1-L.

FORTRAN (& MACRO) ROUTINES (CONTINUED)

```
IFIXA        CONVERT A FLOATING ARRAY TO MAXIMUM SIZE INTEGER.
FLOATA       CONVERT AN INTEGER ARRAY TO FLOATING.  (119 WORDS)

    NSHFT = IFIXA (IX(1),FX(1),N,NSHFT)

    CALL FLOATA (IX(1),FX(1),N,NSHFT)

            IX(I) / 2**(-NSHFT) = F(I)
            F(I) = 2.0**NSHFT * FLOAT(IX(I))

            IX AND FX MAY BE EQUIVALENT ARRAYS.

            NSHFT IS INPUT FOR FLOATA, OUTPUT FOR IFIXA

       THE 'FIXED' INTEGER ARRAY HAS MAXIMUM LEFT JUSTIFICATION,
       SO THAT THERE IS MINIMUM TRUNCATION--UNLIKE THE STANDARD
       FORTRAN INTEGER FUNCTION.


ILOG2        INTEGER LOG BASE 2   (26 WORDS)
IEXP2        INTEGER EXPONENTIATION, BASE 2

    M = ILOG2(N)             INTEGER PART OF LOG2[ABS(N)]
    N = IEXP2(M)             2**M


SFSIN        SUPER FAST SINE FUNCTION .      (97 WORDS)
SFCOS        SUPER FAST COSINE

    FS = SFSIN(X)
    FC = SFCOS(X)

       THESE FUNCTIONS ARE EVALUATED BY LINEAR INTERPOLATION
       OF THE SINES TABLE.  USING SINES2, THE MAXIMUM ABSOLUTE
       ERROR IS <= .000008


IGAUSS       MULTIPLY AN INTEGER ARRAY BY A SYMMETRIC GAUSSIAN
(206 WRDS)   (NORMAL) FUNCTION.  (MAY NOT BE ON .LIBR5(?))

    CALL IGAUSS (IRAY(1),N,SIGMA,ANORM)
            IRAY(1) IS THE FIRST ELEMENT OF THE ARRAY,
            N IS THE NUMBER OF ELEMENTS (EVEN OR ODD),
            SIGMA IS THE STD. DEVIATION DIVIDED BY THE LENGTH
            OF THE ARRAY (& IS DIMENSIONLESS),
            ANORM IS A POSITIVE NORMALIZATION FACTOR.

            COMPUTATION: IRAY(I) = IRAY(I) *
            ANORM * EXP(-.5*((N+1-2*I)/(2*N*SIGMA))**2)
            (TWO INTEGER MULTIPLICATIONS PER POINT)
```

FORTRAN (&MACRO) ROUTINES (CONTINUED)

F4READ (48 WRDS) FORTRAN CALL TO MACRO READ PACKAGE WHICH ALLOWS FOR CONDITIONAL LOADING OF ONLY THOSE ROUTINES REQUIRED.

CALLING SEQUENCE:

```
        EXTERNAL  READO,READI,READF  (ONLY THOSE REQUIRED)

 100    ASSIGN 100 TO LOC
        CALL RDNEW
        CALL F4READ (LOC,READI,IV1,IV2,...)
 200    ASSIGN 200 TO LOC
        CALL RDNEW
        CALL F4READ (LOC,READF,V1,V2,...)
```

F4PRNT FPRNTX (65 WRDS) FORTRAN CALL TO THE MACRO PRINT PACKAGE, WHICH ALLOWS CONDITIONAL LOADING OF ONLY THOSE ROUTINES WHICH ARE REQUIRED.

CALLING SEQUENCE:

```
        EXTERNAL  PRNTNO,PRNTNI,PRNTNF,PRNTFR,PRNTFX,
                  PRNT6B, OR PRNT7B, AS NEEDED

        CALL F4PRNT (ISP,PRNTNI,N,IV1,IV2,...)
        CALL PRNTCR
        CALL F4PRNT (ISP,PRNTNF,N,V1,V2,...)
        CALL PRNTCR
        CALL FPRNTX (IWIDTH,PRNTFX,NF,V1,V2,...)
        CALL PRNTCR
```

ISP IS THE NUMBER OF LEADING BLANKS.
N IS THE NUMBER OF SIGNIFICANT DIGITS, OR CHAR'S.

IWIDTH IS THE WIDTH ALLOWED FOR THE FIXED POINT #.
NF IS THE NUMBER OF FRACTIONAL DIGITS.
(IF THE NUMBER IS TOO LARGE TO FIT INTO THE ALLOTED WIDTH, THE NUMBER OF FRACTIONAL DIGITS IS DECREASED TOWARDS 0. IF THIS DOESN'T WORK, THE VALUE IS PRINTED IN FLOATING FORMAT. ALLOW 2 SPACES FOR SIGN & POINT.)

THE CALL TO PRNTCR CAUSES PRINTING OF THE LINE JUST SETUP.
NOTE: PRNTFX & FPRNTX ARE TO BE USED WITH EACH OTHER ONLY!

(RDINIT) SET & INITIALIZE THE INDICATED DAT SLOT, AND
(PRINIT) INITIALIZE THE READ OR PRINT ROUTINE.

```
          EXTERNAL  IODEV1            [ASSURE LOADING OF HANDLER]
          ...
     120  CONTINUE
          ...
          ASSIGN 120 TO LOCP          [↑P ADDRESS]
          CALL RDINIT (4,LOCP)        [READ FROM 4 (TTY)]
          CALL PRINIT (4,LOCP)        [PRINT ON 4 (TTY)]
          ...
          CALL RDINIT (1,0)           [READ FROM 1 (DK1)]
```

LOCP IS THE ↑P ADDRESS--WHICH IS SET ONLY WHEN THE DAT SLOT # (4, SAY) REFERS TO THE TTY. IF LOCP=0, THE PREVIOUS VALUE OF LOCP IS USED.

IODEVN (0 WRDS) EXTERNAL REFERENCES TO ASSURE LOADING OF A HANDLER. THE USE OF ONE OF THE FOLLOWING IN AN EXTERNAL STATEMENT WILL ASSURE THE THE LOADING OF THE HANDLER FOR THE DAT SLOT INDICATED IN PARENTHESIS:

```
IODEV1,IODEV2,IODEV3,IODEV4,IODEV5,IODEV6,IODEV7,IODEV8,IODVM8
  (1),   (2),   (3),   (4),   (5),   (6),   (7),   (8),  (-8)
```

USAGE:

```
EXTERNAL  IODEV1,IODEV8,IODVM8
```

CAUSES LOADING OF HANDLERS FOR DAT SLOTS 1, 8, & -8

MACRO ROUTINES:

```
GETARG       GET ARGUMENTS OR ADDRESSES (IN AC) FROM FORTRAN
GETADR       CALLING SEQUENCES, ONE AT A TIME.  THIS ROUTINE
(18 WRDS)    USES .GLOBL'D VARIABLES .GETAD & .GOTAD.

             TYPICAL USAGE:

             JMS      SUBR
             JMP      .+1+N    /N=# ARGUMENTS
             .DSA     ARG1
              .        .
              .        .

             .GLOBL  GETARG,GETADR,.GETAD,.GOTAD
     SUBR    0
             LAC      SUBR
             DAC*     .GETAD
             JMS*     GETARG
             (1ST ARG IN AC, ADDR IN .GOTAD)
             JMS*     GETADR
             (ADDRESS OF 2ND ARG IN AC)
             JMS*     GETARG
             (3RD ARG IN AC, ADDRESS IN .GOTAD).

READ1        READ A LINE FROM TTY, AND RETURN A CHARACTER AT A
RD1NEW       TIME TO THE USER (IN AC).  A NEW LINE IS
RDNEW        READ ON ENTRY FOLLOWING RETURN OF CARRIAGE RETURN
(140 WRDS)   OR ALTMODE.  AN ENTRY TO RD1NEW FORCES THE READING
             OF A NEW LINE EVEN IF THE OLD LINE WAS NOT ALL USED.
             RDNEW SETS A FLAG FOR USE OF NEW LINE ON NEXT CALL
             TO ANY READ ROUTINES.

             TYPICAL USAGE:
             .GLOBL  READ1,RD1NEW
    JMS*     RD1NEW
    (NEW LINE READ & 1ST CHARACTER IN AC.)
    JMS*     READ1
    (2ND CHARACTER IN AC.)
    JMS*     RD1NEW
    (1ST CHARACTER OF NEW LINE IN AC.)

             READ6B,READ7B,READI,READO,READF  ARE A SET OF
             FREE FORM CONVERSION ROUTINES FOR TEXT, INTEGER
             OCTAL, AND FLOATING INPUT;
             READ6X & READ7X ARE FIXED FORM TEXT ROUTINES.

             F4READ ALLOWS FORTRAN CALLS TO THE READ PACKAGE.

    (SEE ADDITIONAL DOCUMENTATION ON PAGE A7.)
    (FILE NAMES: READ W2A [WEDGED] & .DPF V1B)
```

MACRO ROUTINES (CONTINUED)

PRINT MACRO ROUTINES FOR CONVERTING TEXT OR DATA AND PRINTING FROM AN INTERNAL LINE BUFFER. BUFFERED I/O ALLOWS USE OF THE COMPUTER WHILE THE LINE IS BEING PRINTED. OCTAL, INTEGER, FLOATING, AND TEXT CONVERSION ROUTINES ARE AVAILABLE.

F4PRNT ALLOWS FORTRAN CALLS TO THE READ PACKAGE.

(SEE ADDITIONAL DOCUMENTATION, PAGE A2.)

(FILE NAMES: PRINT W2R [WEDGED] & .DPF V1B)

DUMP PRINT VALUES OF VARIOUS REGISTERS. DUMP USES THE MACRO PRINT PACKAGE & IS INDEPENDENT OF THE FORTRAN ARITHMETIC AND I/O PACKAGES. ALL REGISTERS ARE RESTORED AFTER THE DUMP IS TAKEN.

CALLING SEQUENCE:

```
CALL DUMP (MCODE,A,B,I,...)
```

SUGGESTED MACRO USAGE:

```
.DEFIN  ZDUMP,M,N                         ZDUMP   M,N
JMS*    DUMP                              .DSA    A1
JMP     .+2+N                             .DSA    A2
.DSA    (M                                  .       .
.ENDM                                       .       .
                                          .DSA    AN
```

MCODE IS A CODE TO SPECIFY THE TYPE OF PRINTOUT DESIRED:

MCODE	PRINTOUT
0	LINK,AC,MQ,DATA SWITCHES (OCTAL)
1	AC & MQ (INTEGER); SC, XR, LR (OCTAL)
2	(AC,MQ) & FLOATING AC (FLOATING), & F.AC (OCTAL)

THESE OPTIONS MAY BE "OR-ED". (ZERO OPTION IS ALWAYS USED.) THE STRUCTURE OF THE PRINTOUT IS SHOWN FOR CODE 3:

```
*[L LLLLL]  L AAAAAA MMMMMM  SSSSSS
 (-AAAAAA) (-MMMMMM)   SC   XXXXXX LLLLLL
 (-F.FFFF±E)  EEEEEE MMMMMM (-F.FFFF±E)
  C(LLLLL)= OOOOOO (-IIIII)  MMMMMM (-F.FFFF±E)
  C(LLLLL)= ...
      .
      .
      .
```

(- IS USED TO INDICATE BLANK OR MINUS,
+ IS USED TO INDICATE PLUS OR MINUS.)

1ST LINE (CODE 0): LOCATION OF JMS TO DUMP, OCTAL VALUES OF LINK, AC, MQ, & DATA SWITCHES.

2ND LINE (CODE 1): INTEGER VALUES OF AC & MQ, OCTAL VALUES OF STEP COUNTER, INDEX REGISTER, LIMIT REG.

3RD LINE (CODE 2): FLOATING VALUE OF COMBINED AC,MQ (WORD CONTAINING EXPONENT IN AC), OCTAL VALUES OF 1ST TWO WORDS IN THE FLOATING AC, FLOATING VALUE OF THE FLOATING AC (FIRST TWO WORDS ONLY).

ADDITIONAL LINES: OCTAL VALUE OF MEMORY LOCATION, OCTAL VALUE OF ITS CONTENTS; & IF CODE 1, THE INTEGER VALUE, AS WELL; & IF CODE 2, THE OCTAL VALUE OF THE SUCEEDING WORD, AND THE FLOATING VALUE OF THE TWO WORDS TAKEN TOGETHER.

PRINT MACRO PRINT PACKAGE:

CONVERTS & PRINTS TEXT OR DATA, DOUBLE BUFFERED & INDEPENDENT OF FORTRAN ARITHMETIC AND I/O PACKAGES.

PRINTC THE MAIN SUBROUTINE IS PRINTC, WHICH ADDS THE .ASCII CHARACTER IN THE AC TO AN INTERNAL LINE BUFFER (LINBUF). (ALL REGISTERS EXCEPT THE AC ARE PRESERVED.) WHENEVER A CARRIAGE RETURN (15 OCT) OR ALT MODE (175 OCT) IS ENTERED, THAT CHARACTER IS ADDED TO LINBUF, A .WAIT IS EXECUTED TO WAIT FOR COMPLETION OF PREVIOUS I/O, LINBUF IS MOVED INTO ANOTHER INTERNAL BUFFER (IOLINE), AND IOLINE IS PRINTED ON THE UNIT SPECIFIED BY THE .GLOBL'D VARIABLE PRUNIT (PRESET TO DAT SLOT 4: TTY). CONTROL IS RETURNED TO THE USER IMMEDIATELY AFTER THE INITIATION OF OUTPUT FROM IOLINE. THUS, COMPUTATION BY THE USER, INCLUDING SETTING UP A NEW PRINT LINE, MAY PROCEED WHILE THE PRINTING OF THE CURRENT LINE TAKES PLACE. WHENEVER THE MAXIMUM NUMBER OF CHARACTERS (73 DEC) IS ADDED TO LINBUF, A CARRIAGE RETURN IS AUTOMATICALLY ADDED TO THE LINE, AND IT IS PRINTED. THUS NO CHARACTERS ARE EVER LOST BY OVERPRINTING ON THE RIGHT EDGE.

A FAIRLY COMPLETE SET OF SUBROUTINES (ALL OF WHICH USE PRINTC) IS INCLUDED FOR CONVERTING AND ADDING VARIOUS KINDS OF DATA TO LINBUF. ONLY ROUTINES IN SECTION I (FOLLOWING) CAUSE ACTUAL PRINTING; THE REST SIMPLY ADD CHARACTERS TO LINBUF. THE CONVERSION ROUTINES ARE PARTITIONED INTO SEPARATE FILES.

PRUNIT
PRMXCH .GLOBL'D VARIABLE WHICH CONTAINS THE DATSLOT FOR THE OUTPUT UNIT. PRUNIT IS PRESET TO 4 (TTY). NOTE: THE DAT SLOT IS INITIALIZED ONLY ON THE FIRST CALL TO (ANY) ONE OF THESE ROUTINES. IF THE DAT SLOT IS CHANGED AFTER THAT, THE USER MUST MAKE SURE THAT THE NEW DAT SLOT HAS BEEN INITIALIZED. PRMXCH IS A .GLOBL'D VARIABLE WHICH CONTAINS THE MAXIMUM NUMBER OF CHARACTERS PER LINE (73), NOT COUNTING THE CARRIAGE RETURN.

PRINIT F4 ROUTINE TO SET DAT SLOT, INITIALIZE IT, & SET THE ↑P ADDRESS IF THE DAT SLOT REFERS TO THE TTY.

```
      EXTERNAL  IODEV3          [TO ASSURE LOADING OF HANDLER]
      ...
120   CONTINUE
      ...
      ASSIGN 120 TO LOC         [↑P ADDRESS = STATEMENT # 120]
      CALL PRINIT (4,LOC)       [PRINT ON D. S. 4 (TTY)]
      ...
      CALL PRINIT (8,0)         [PRINT ON D. S. 8 (VP)]
```

*** IN ALL THE ROUTINES THAT REQUIRE A CHARACTER OR DIGIT COUNT IN THE AC ON ENTRY, EITHER A POSITIVE OR NEGATIVE VALUE MAY BE USED (SO THAT LAW -N MAY BE USED IN PLACE OF LAC (N)).

PRINT (CONTINUED)

USER'S SUBROUTINES

1. CARRIAGE CONTROL & INITIATION OF OUTPUT

CALLING SEQUENCES: JMS* SUBR

```
PRNTCR ....  PRINT LINE WITH CARRIAGE RETURN

PRNTAM ....  PRINT LINE WITH NO CARRIAGE RETURN

PRNTEJ ....  PRINT LINE, FORM FEED, & CARRIAGE RETURN

PRINTZ ....  PRINT LINE IN LINBUF -- IF THERE IS ANY -- WITH A
             CARRIAGE RETURN, AND WAIT FOR COMPLETION OF OUTPUT.
             (THIS ROUTINE SHOULD BE USED BEFORE RETURN TO THE
             MONITOR -- OTHERWISE THE LAST LINE MAY NOT BE
             PRINTED.)
```

2. FIXED FORMAT CONVERSION ROUTINES

```
PRINTO ....  OCTAL NUMBER IN AC;  JMS*  PRINTO
             PRINT 6 OCTAL DIGITS.  USES PRNTNO: MQ NOT SAVED.

PRINTI ....  INTEGER NUMBER IN AC;  JMS*  PRINTI
             PRINT SIGN AND 6 INTEGER DIGITS.  USES PRNTNI:
             MQ NOT SAVED.

PRINTF ....  TWO WORD FLOATING NUMBER IN AC,MQ;  JMS*  PRINTF
             PRINT DECIMAL: ±F.FFFF±E .  USES PRNTNF: MQ NOT SAVED.
             (WORD CONTAINING EXPONENT IN AC.)
```

3. TEXT ROUTINES

```
PRNT6B ....  LAC  N;  JMS*  PRNT6B;  .DSA  BUFFR
             PRINT THE LEFTMOST N .SIXBT CHARACTERS OF THE STRING
             STORED IN BUFFR.  (THREE CHARACTERS PER WORD.)
             MQ IS NOT SAVED.

PRNT7B ....  LAC  N;  JMS*  PRNT7B;  .DSA  BUFFR
             PRINT THE LEFTMOST N .ASCII CHARACTERS OF THE STRING
             STORED IN BUFFR.  (FIVE CHARACTERS PER WORD PAIR.)
             MQ IS NOT SAVED.

PRNTNC ....  LAC  N;  JMS*  PRNTNC;  JMP  .+1+N ;
             LAC  (CHAR1;  LAC  (CHAR2;...  LAC  (CHARN
             PRINT THE N CHARACTERS INDICATED IN THE CALLING
             SEQUENCE.
```

4. VARIABLE FORMAT ROUTINES

```
PRNTMS .... N IN AC;  JMS*  PRNTMS
            PRINT N SPACES (BLANKS)

PRNTNO .... OCTAL NUMBER IN MQ; N IN AC;  JMS*  PRNTNO
            PRINT N HIGH ORDER OCTAL DIGITS.  IF N>6, PRINT
            N-6 LEADING BLANKS.  IF N<6, MQ CONTAINS THE UNUSED
            LOW ORDER DIGITS, LEFT JUSTIFIED.

PRNTNI .... INTEGER NUMBER IN MQ; N IN AC;  JMS*  PRNTNI
            PRINT SIGN AND N INTEGER DIGITS.  IF N>6, PRINT
            N-6 LEADING BLANKS.  LEADING ZEROS ARE BLANKED;
            FINAL ZERO IS PRINTED.  THE SIGN (BLANK OR MINUS)
            PRECEEDS THE FIRST PRINTED DIGIT.  N+1 CHARACTERS
            ARE PRINTED;  IF THE INTEGER IS LARGER THAN N DIGITS,
            EXTRA DIGITS ARE PRINTED TO PRODUCE THE FULL VALUE.
            MQ IS ZERO ON RETURN.

PRNTNF .... LAC  N;  JMS*  PRNTNF;  .DSA  FN
            PRINT DECIMAL VALUE OF FLOATING NUMBER (FN) WITH
            N SIGNIFICANT DIGITS:    -.FF...F+EE
            WIDTH IS N+5 CHARACTERS;  IF THE MAGNITUDE OF THE
            EXPONENT (EE) IS LARGER THAN TWO DIGITS, EXTRA DIGITS
            ARE PRINTED TO PRODUCE FULL VALUE.  (IN THE FORMAT
            ABOVE, - REPRESENTS A SIGN: BLANK OR MINUS,
            + REPRESENTS A SIGN: PLUS OR MINUS.)  THE FLOATING
            VALUES ARE INTENTIONALLY NOT ROUNDED OFF.
            MQ IS NOT SAVED.

PRNTFX .... LAW  -N;  JMS*  PRNTFX;  .DSA  WIDTH;  .DSA  F
            PRINT FIXED POINT VALUE OF FLOATING NUMBER (F).  N
            IS THE NUMBER OF FRACTIONAL DIGITS;  WIDTH IS THE
            WIDTH OF THE FIELD.  IF THE VALUE IS TOO LARGE TO
            FIT IN THE SPECIFIED WIDTH, THE VALUE IS PRINTED WITH
            A REDUCED NUMBER OF FRACTIONAL DIGITS, IF POSSIBLE;
            IF THE INTEGER VALUE IS TOO LARGE TO FIT IN THE
            SPECIFIED WIDTH, THE VALUE IS PRINTED IN FLOATING
            FORMAT.  MQ IS NOT SAVED.

PRNTFR .... INTEGER FRACTION IN MQ; N IN AC; JMS* PRNTFR
            PRINT SIGNED FRACTION OF N SIGNIFICANT DIGITS.
            200000 (OCT) IS PRINTED AS (+).5000 (N=4, SAY).
```

5. FORTRAN CALLS

```
FPRINT .... CALL FPRINT;  FORMAT (NNH TEXT...)
            PRINTS THE CONTENTS OF A HOLLERITH FIELD GENERATED
            BY A FORMAT STATEMENT WHICH MUST BEGIN WITH A TWO
            DIGIT NUMBER (25, SAY).  THE H & BLANK ARE IGNORED.
            THE NN-1 CHARACTERS FOLLOWING THE BLANK ARE PRINTED.

            TYPICAL USAGE:

            CALL FPRINT
       1    FORMAT (07H SAMPLE)

F4PRNT .... CALL F4PRNT (ISP,PRSUBR,N,VAR1,VAR2,...)
            FORTRAN CALL TO PRINT PACKAGE, USABLE WITH
            PRNTNO,PRNTNI,PRNTNF,PRNTFR,PRNTGB,PRNT7B .

FPRNTX .... CALL FPRNTX (IWIDTH,PRNTFX,NF,VAR1,VAR2,...)
            FORTRAN CALL FOR FIXED POINT PRINT.

PRINIT .... CALL PRINIT (NUNIT,LOCP)
            SET PRINT UNIT & INITIALIZE.  (SEE PAGE A2)

            (SEE DOCUMENTATION ON PAGES 1.7 & 1.8)
```

6. STORAGE REQUIREMENTS:

FILE	WORDS	OTHER FILES	ROUTINES
PRINT	[illegible]	NONE	PRINTC,PRINTZ,PRNTCR,[illegible],[illegible],[illegible], PRNTNS; (PRSET[illegible])
PRINT.	58	PRINT	PRNTNC; (PRICON)
PRINTO	[illegible]	PRINT	PRINTO,PRNTNO
PRINTI	[illegible]	PRINT PRINT.	PRINTI,PRNTNI
PRINTF	183	PRINT PRINT. & .DPF	PRINTF,PRNTNF,PRNTFX ; (PRFCON)
PRNTFR	26	PRINT	PRNTFR
PRINT6	32	PRINT	PRNT6B
PRINT7	35	PRINT	PRNT7B
.DPF	139	NONE	(DPM.LD, DPM. ST, DPF.GM, DPF.GF, DPFNUL, DPF8.5, DPF5.4, MPY5.4, MPY8.5, DPNORM)
FPRINT	31	PRINT PRINT7	FPRINT
F4PRNT	65	ANY	F4PRNT,FPRNTX

[illegible] READ BY [illegible]:

READS & CONVERTS FREE FORM TEXT OR DATA, INDEPENDENT OF FORTRAN

THESE ROUTINES ARE PARTITIONED SO THAT ONLY THE ROUTINES REQUIRED ARE LOADED.

READ1
RDINEW
RDNEW

THIS IS THE BASIC READ ROUTINE. WHEN READ1 OR RDINEW IS CALLED, & THE INTERNAL BUFFER IS EMPTY, A [IS TYPED, AND INPUT READ FROM THE TELETYPE UNTIL A CARRIAGE RETURN OR ALT MODE IS TYPED; THE FIRST CHARACTER TYPED IS RETURNED IN THE AC. FURTHER CALLS TO READ1 CAUSES SUCCESSIVE CHARACTERS TO BE RETURNED IN THE AC, UNTIL A CARRIAGE RETURN OR ALT MODE IS RETURNED. THE NEXT CALL THEN CAUSES THE READING OF A NEW LINE FROM THE TELETYPE. A CALL TO RDINEW ALWAYS CAUSES THE READING OF A NEW LINE AND RETURNS THE FIRST CHARACTER IN THE AC. A CALL TO RDNEW CAUSES A NEW LINE TO BE READ THE NEXT TIME ANY READ ROUTINE IS USED.

```
JMS*     SUBR      /READ1,RDINEW,RDNEW
(CHAR RETURNED IN AC (READ1,RDINEW ONLY))
```

FOR ALL THE NUMERICAL ROUTINES, NUMERICAL VALUES MAY BE SEPARATED BY BLANKS, HOR. TABS, A COMMA, A CARRIAGE RETURN, OR AN ALT MODE. ANY COMBINATION OF BLANKS, HOR. TABS, PLUSES, OR MINUSES MAY PRECEED THE NUMERICAL INPUT. THE CONVERSION ROUTINES ARE PARTITIONED INTO SEPARATE FILES.

RDCHAR
RDVALU
RDVAL2

.GLOBL'D VARIABLES WHICH CONTAIN THE LAST CHARACTER READ, AND THE NUMERICAL VALUE FROM THE LAST CALL TO READ0, READ1, OR READF. (RDVALU+1=RDVAL2 CONTAINS THE HIGH ORDER MANTISSA FOR READF.)

RDUNIT
RDMARK

.GLOBL'D VARIABLES WHICH CONTAIN THE DAT SLOT NUMBER OF THE UNIT TO BE READ (PRESET TO 4: TTY), AND THE .ASCII CHARACTER + 300 FOR THE MARK TO BE MADE ON THE TELETYPE (DAT SLOT 4 ONLY) INDICATING NEED FOR INPUT (PRESET TO "[" <333>).

RDINIT

F4 ROUTINE TO SET DAT SLOT, INITIALIZE IT & READ RTN, & SET ↑P ADDRESS IF DAT SLOT REFERS TO TTY.

```
      EXTERNAL  IODEV1          [ASSURE LOADING OF HANDLER]
      ...
120   CONTINUE
      ...
      ASSIGN 120 TO LOCP        [↑P ADDRESS = STATEMENT # 120]
      CALL RDINIT (4,LOCP)      [READ FROM D. S. 4 (TTY)]
      ...
      CALL RDINIT (1,0)         [READ FROM D. S. 1 (DK1)]
```

PREVIOUS VALUE OF LOCP IS USED IF LOCP = 0.

[illegible] .... [illegible]
[illegible] (+ - 10) [illegible]
(- 10) [illegible]
RIGHT [illegible] THE [illegible] IN THE LEFT HALF OF THE AC -- WITH [illegible]
7-BIT [illegible] ONLY.
LEADING [illegible] ARE [illegible]
[illegible] CHARACTERS (EXCEPT + OR -) [illegible]
[illegible] (<,,,>) [illegible]
[illegible]
[illegible]
-, [illegible] IS IGNORED [illegible]
STRING. [illegible] CHARACTER STRING IS [illegible]
[illegible]
[illegible]

[illegible] ARE [illegible]
THE CHARACTER STRING IS TERMINATED BY A [illegible]
[illegible] MODE.

```
[illegible]:
        JMS*    [illegible]     /[illegible],[illegible],[illegible],[illegible]
        JMP     .+[illegible]
        .DSA    BUFFER  /USER'S BUFFER
        (CHAR [illegible], [illegible] RETURNED IN AC.)


        [illegible]     [illegible] (BUFFER(I))
                        [illegible] = [illegible]([illegible]),[illegible])
```

[illegible] .... [illegible] INPUT:

```
    CALL [illegible]:
            JMS*    [illegible]
            (ERROR RETURN - ILLEGAL CHAR, [illegible])
            (NORMAL RETURN, VALUE IN AC)

    ([illegible] USED FOR [illegible].)
```

[illegible] .... OCTAL INPUT.

```
    CALL [illegible]:
            JMS*    [illegible]
            (ERROR RETURN - NON OCTAL CHAR IN AC)
            (NORMAL RETURN, VALUE IN AC)

    (MORE THAN 6 DIGITS ENTERED CAUSES ERROR RETURN.)
```

[illegible] .... [illegible]
[illegible] (EXPONENT [illegible]). THE [illegible] MAY BE PRECEDED BY ANY COMBINATION OF BLANKS, [illegible], [illegible], OR HORIZONTAL TABS. THE EXPONENT, IF [illegible], MUST BE PRECEDED BY AT LEAST ONE [illegible] BLANK, +, OR -. THE FLOATING VALUES MAY BE SEPARATED BY [illegible], HORIZ.TABS, A CARRIAGE RETURN [illegible]. IF [illegible] THE VALUE MUST BE TERMINATED BY A COMMA, [illegible], OR ALT MODE. FOR INSTANCE, THE VALUE [illegible] MAY BE [illegible] 1 [illegible], [illegible], [illegible], [illegible]-1. 1.[illegible], [illegible], ETC. [illegible] IS USED WHEN [illegible] OCCURS. [illegible] VALUES [illegible].

CALL [illegible]
[illegible]
(ERROR RETURN - ILLEGAL CHAR, OR [illegible])
(NORMAL RETURN)

[illegible] FORTRAN CALLS:

[illegible] .... FORTRAN CALL FOR READC, READI, OR READF WHICH STILL INVOLVES CONDITIONAL [illegible] THOSE [illegible] ERROR.

TYPICAL USAGE:

```
        EXTERNAL READI

  10    ASSIGN 10 TO NOC
        CALL [illegible]
        CALL [illegible](NOC,READI,INTGER,[illegible],....)
```

AN ERROR RETURN FROM THE READ ROUTINE WILL CAUSE A RETURN TO THE STATEMENT [illegible] (10) ASSIGNED TO NOC.

[illegible] .... [illegible] READ UNIT & INITIALIZE.
(SEE PAGE 47)

([illegible] PAGES 1.7 & 1.[illegible] [illegible].)

STORAGE REQUIREMENTS

FILE	WORDS	OTHER FILES	ROUTINES
READ1	157	NONE	RDNEW, READ1, RDINEW, RDINIT
READ.	77	READ1	(RGTNC1, GRTDGT, GRTEND, RDTTST, RSGNVL)
RDPAK	71	READ1	(RDPAK, RDPK6S, RDPK7S)
READO	34	READ1 READ.	READO
READI	53	READ1 READ.	READI
READF	121	READ1 READ., READI, .DPF	READI
READT	56	READ1 RDPAK, READO	READ6B, READ7B
READTX	38	READ1 RDPAK	READ6X, READ7X
.DPF	139	NONE	(DPM.LD, DPM.ST, DPF.GM, DPF.GF, DPFMUL, DPF8.5, DPF5.4, MPY5.4, MPY8.5, DPNORM)
F4READ	48	ANY	F4READ

```
$$$

PIP V13A

>T TT (A)←DT2 WATSUP DOC


THIS FILE WILL CONTAIN MISCELLANEOUS NOTES FOR PROGRAMMERS.
LIKE:  AUTO-INCREMENT REGISTERS ACT AS 18 BIT REGISTERS--EVEN
UNDER INDIRECT ADRESSING.


PIP V13A

>T TT (A)←DT2 .LOAD DOC


        .LOAD V1 :

THIS LOADER HAS BEEN MODIFIED TO ACCEPT PART OF THE LOAD STRING
FROM DAT SLOT -5 (DK0
```

=

V

M

Z

[illegible]

[illegible] PART OF [illegible]
[illegible] (. . .). THE[RE] ARE SEVERAL [illegible]:

1) [illegible] SYSTEM PROGRAMS WHICH [illegible] LIBRARY (.LIB . .). THE [illegible] OF [illegible] THE [illegible] BY THE LOADER.
2) [illegible] VERSION OF [illegible] THE [illegible] SYSTEM. THESE [illegible] BY THE [illegible].

[illegible] ";" OCCURS IN THE [illegible] STRING, [illegible] (AS [illegible] IS [illegible]), [illegible]

[illegible] FOLLOWING [illegible]:

1) ANY [illegible] IS IGNORED.
2) [illegible] ";" IS [illegible] IN THE [illegible], A [illegible] LINE (OR [illegible] = RIGHT JUSTIFIED ";") IS [illegible] TO THE LIST OF FILE NAMES. THE ";" IS [illegible] TREATED AS A COMMA.
3) IF THE 'ENTRY' LINE IS [illegible], THE [illegible] UNIT IS CHANGED TO [illegible] -5, [illegible] FILES ARE CONTINUED.

[illegible] TYPICAL [illegible] ARE:

>←ALPHA,ALPHI,[illegible];RTS.KO

(RTS.KO IS THE FILE REQUIRED BY THE REAL-TIME-SYSTEM.)

>←;ADTEST;RTS.KO

(ADTEST IS AN A/D DISPLAY PROGRAM, USING THE REAL-TIME-SYSTEM, WHICH RESIDES IN RAM.)

>←ZEN4;RTS.KO,RTS.PM

(PROGRAM USING REAL TIME SYSTEM, & USING A RESIDENT POST MORTEM DUMP PROGRAM.)

>←ZEN4;RTS.KO

\\
.
.
IOPS00 027206 ↑Q1
@.
.
.
(GLOAD)
>← ;RTS.PM

(USER GETS POST MORTEM DUMP FROM @-DUMP.)

J.R. [illegible], [illegible]

Appendix C

S O M N I A .

SLEEP ONSET MONITOR NOTIFIED INTERACTIVELY

VERSION 1 CHRISTIAN FRIEDRICH-FREKSA, SPRING 1974

SYMBOLTABLE
===========

NAMED COMMON AREAS:

```
IN:                     INSTRUCTIONS
   INSTRU(35)              INSTRUCTION TABLE
      IHEAD(7)                HEADLINE OF INSTRUCTION TABLE
         IOAP                    OVER ALL REPETITION PERIOD
         LIMO                    SIGNAL FOR MODIFICATION
         LENGTH                  SIZE OF 1 A-D BUFFER OF 1 CHANNEL
         MREPT                   REPETITION CYCLE FOR ORGANIZ. ROUT.
         NCHANS                  HIGHEST CHANNEL # SPECIFIED
         MSAMP                   MINIMAL SAMPLING PERIOD SPECIFIED
         IBUDIM                  DYNAMIC BUFFER SIZE

      NSAMPI(7)               SAMPLING PERIODS
      MODE(7)                 CODE FOR EVALUATION MODE
      NEVAL(7)                EVALUATION PERIODS
      IWINDO(7)               WINDOW SIZES

   NGDS(7)                 REQUIRED WORK AREA FOR EACH CHANNEL
```

```
MO:                     MODIFICATION
   ICHAN                   CHANNEL #
   INSTR                   INSTRUCTION CODE
   IDAT(4)                 SPECIFICATIONS FOR CHANNEL MODIFICATIONS

PT:                     POINTERS
   KPOINT(27)              POINTER VECTOR
      IGBPT(8)                GENERAL BUFFER POINTER FOR 7 CHANNELS
                                 AND A FICTIVE 8TH CHANNEL
      ITRAPO                  TRANSFER BUFFER POINTER
      ITRALI                  TRANSFER BUFFER LIMIT
      IGBST(7)                POINTERS TO THE START POINTS OF GB
      NADST(7)

GB:                     GENERAL BUFFER
   IGDB(4000)              GENERAL DATA BUFFER
         ↑ WILL BE CHOSEN AS LARGE AS POSSIBLE

FILE NAME VARIABLES:
   FILRAW(2)                CONTAINS NAME OF RAW DATA FILE
   FILEN(2)                 CONTAINS NAME OF INSTRUCTION TABLE FILE

LOGICAL VARIABLES:
   IFFIL    =   .TRUE., IF INSTRUCTION TABLE ALREADY EXISTS ON DISK
                .FALSE., OTHERWISE
   IPOWR2(I)=   .TRUE., IF I IS A POWER OF 2
                .FALSE. OTHERWISE
   SWITCH(I)=   .TRUE., IF CONSOLE SWITCH I IS ON
                .FALSE. OTHERWISE
```

OTHER SYMBOLS:

ITRANS(251)	TRANSFER BUFFER
RAWST(7)	'YES', IF RAW DATA ARE TO BE STORED
	'NO' OTHERWISE
IDKI	LOGICAL USER DISK UNIT
IDSPL	LOGICAL # FOR VIDEO DISPLAY
ITT	LOGICAL TELETYPE UNIT
LOC	RETURN ADDRESS
JANEL	NUMBER OF CHANNELS SPECIFIED
INST5 = 5	)NDICATES TO SOFIMO THAT SOMMM FINISHED
	ASSEMBLING A COMMANDSTRING
IDOMOD > 0	DO MODIFICATION
IADPO	A-D BUFFER POINTER
IADCO	A-D BUFFER COUNTER
IADOLD	OLDEST DATA POINT IN A-D BUFFER
INSAM	INCREMENT FOR SAMPLE POINTER IN A-D BUFFER
ITIME	PROGRAM RUN TIME
MISEC	RUN TIME IN SEC/1024 MODULO 1024
LATIME	RUN TIME AT LAST PROGRAM CALL
ISEC	RUN TIME IN SEC MODULO 60
MIN	RUN TIME IN MINUTES
IREC	RECORD #
INEVAL	INCREMENT FOR EVALUATION POINTER
IPOINT	ACTUAL DATA POINT IN GENERAL DATA BUFFER

```
C
C              S  O  I  N  I  T  .
C
C        GENERATES SLEEP ONSET INSTRUCTION TABLE.
C
C        VERSION 1,  CFF 5/10/74.
C
         LOGICAL IFFIL,IPOWR2
C
         DIMENSION FILEN(2),IHELP(7),IHEAD(7),NSAMPI(7),MODE(7)
         1,NEVAL(7),IWINDO(7),RAWST(7),INSTRU(35)
C
         EQUIVALENCE(INSTRU(1),IHEAD(1)),(INSTRU(8),NSAMPI(1))
         1,(INSTRU(15),MODE(1)),(INSTRU(22),NEVAL(1))
         2,(INSTRU(29),IWINDO(1))
C
         EXTERNAL READI
C
         DATA FILEN(1),FILEN(2)/5HINSTR,3HUCT/
         DATA YES,HNO/3HYES,3H NO/
         DATA IDK1,IDSPL,ITT/2,8,4/
C
C
C
C        DOES INSTRUCTION TABLE ALREADY EXIST ON DISK?
         CALL ENTER(IDK1,FILEN)
         CALL RENAM(IDK1,FILEN,FILEN,IFFIL)
         IF(IFFIL) GO TO 1000
C
C
C
C        INSTRUCT - FILE WAS NOT FOUND.
C        INITIALIZE SCOPE.
         CALL DSINIT
         WRITE(IDSPL,1)
    1    FORMAT(6X
         1,52 H S L E E P O N S E T I N S T R U C T I O N T A B L E)
C
         WRITE(IDSPL,2)(I,I=1,7)
    2    FORMAT(//8H0CHANNEL,9 X,7I6)
C
C
C
         WRITE(ITT,3)
    3    FORMAT(46H ENTER SAMPLING PERIODS (POWERS OF 2 OR ZERO).)
         WRITE(ITT,24)(I,I=1,7)
   24    FORMAT(7I5)
C
  100    ASSIGN 100 TO LOC
C
C        INITIALIZE READ ROUTINE.
         CALL RDNEW
C
         DO 120 ICHAN=1,7
C * * *  INITIALISATION: NO RAW DATA STORING IN ICHAN.
         RAWST(ICHAN)=HNO
C
         CALL F4READ(LOC,READI,NSAMPI(ICHAN))
C
C        UNLESS PRESCRIBED DIFFERENTLY, EVALUATION RATE EQUALS SAMPLING
C        RATE.
         NHELP=NEVAL(ICHAN)=NSAMPI(ICHAN)
         IF(NHELP)900,120,110
C
```

```
C         CHECK, WHETHER SAMPLING PERIOD A POWER OF 2.
  110     IF(.NOT.IPOWR2(NHELP)) GO TO 930

          NCHANS=ICHAN
  120     CONTINUE
C
  130     WRITE(IDSPL,4) NSAMPI
    4     FORMAT(18H0SAMPLING PERIODS ,7I6)
C
C
C
          WRITE(ITT,5)
    5     FORMAT(/37H0ENTER CHANNEL #'S, THE DIGITIZED RAW
         1/32 H DATA OF WHICH ARE TO BE STORED.)
C
  200     ASSIGN 200 TO LOC
          CALL RDNEW
C
          DO 210 ICHAN=1,NCHANS
          CALL F4READ(LOC,READI,IHELP(ICHAN))
          IF(IHELP(ICHAN).EQ.0) GO TO 220
C
C         CHECK, WHETHER REQUIRED CHANNELS DO EXIST.
          IF(IHELP(ICHAN).LT.0.OR.IHELP(ICHAN).GT.NCHANS) GO TO 910
C
C         ATTRIBUTION OF THE DATA TO THEIR CHANNELS.
          DO 210 JANEL=1,7
          IF(IHELP(ICHAN).EQ.JANEL.AND.NSAMPI(JANEL).NE.0)RAWST(JANEL)=B
  210     CONTINUE
C
  220     WRITE(IDSPL,6) RAWST
    6     FORMAT(15H0STORE RAW DATA,3X,7(3X,A3))
C
C
C
          WRITE(ITT,7)
    7     FORMAT(/49 H0ENTER PEARS OF CHANNEL#'S AND EVALUATION PERIODS
         1/52 H (IF DIFFERENT FROM SAMPLING PERIODS) - POWERS OF 2.)
C
  300     ASSIGN 300 TO LOC
          CALL RDNEW
C
          DO 320 I=1,NCHANS
          CALL F4READ(LOC,READI,ICHAN)
          IF(ICHAN)910,330,310
          IF(ICHAN.GT.NCHANS) GO TO 910
  310     CALL F4READ(LOC,READI,NHELP)
          IF(NHELP.LT.NSAMPI(ICHAN)) GO TO 900
          IF(.NOT.IPOWR2(NHELP)) GO TO 930
          NEVAL(ICHAN)=NHELP
  320     CONTINUE
C
  330     WRITE(IDSPL,8)NEVAL
    8     FORMAT(19 H0EVALUATION PERIODS,7I6)
C
C
C
C         STORE USED CHANNEL#'S IN IHELP.
          JANEL=0
          DO 400 ICHAN=1, NCHANS
          IF(NSAMPI(ICHAN).EQ.0) GO TO 400
          JANEL=JANEL+1
          IHELP(JANEL)=ICHAN
  400     CONTINUE
C
          WRITE(ITT,9)
```

```
     ... FOR THE EVALUATIONS UNDER THE
      1,9 H CHANNEL#)
C
C
  410 ASSIGN 410 TO LOC
      WRITE(ITT,19)(IHELP(ICHAN),ICHAN=1,JANEL)
C
   19 FORMAT(1 X,7I4)
C
      CALL RDNEW
C
      DO 420 I=1,JANEL
  420 CALL F4READ(LOC,READI,IWINDO(I))
C
      JANE=JANEL
      DO 450 I=1,NCHANS
      ICHAN=NCHANS+1-I
      IF(ICHAN.GT.IHELP(JANE)) GO TO 440
C
C     VALID VALUE FOR WINDOW SIZE?
      NHELP=IWINDO(JANE)
      IF(NHELP.LT.NEVAL(ICHAN)) GO TO 920
      IF(.NOT.IPOWR2(NHELP)) GO TO 930
      IWINDO(ICHAN)=NHELP
      JANE=JANE-1
      IF(JANE.EQ.0) JANE=7
      GO TO 450
  440 IWINDO(ICHAN)=0
  450 CONTINUE
C
      WRITE(IDSPL,11)IWINDO
   11 FORMAT(12 H0WINDOW SIZE,5X,7I6)
C
C
C
      JANEL=0
      DO 500 ICHAN=1,NCHANS
      IF(NSAMPI(ICHAN).EQ.0) GO TO 500
      JANEL=JANEL+1
C
C     IHELP CONTAINS USED CHANNEL #'S.
      IHELP(JANEL)=ICHAN
  500 CONTINUE
C
      WRITE(ITT,12)(IHELP(ICHAN),ICHAN=1,JANEL)
   12 FORMAT(/45H0TYPE EVALUATION MODE UNDER PRINTED CHANNEL #/1X,7I
C
  510 ASSIGN 510 TO LOC
      CALL RDNEW
C
      DO 520 I=1,JANEL
  520 CALL F4READ(LOC,READI,MODE(I))
C
      JANE=JANEL
      DO 550 I=1, NCHANS
      ICHAN=NCHANS+1-I
      IF(ICHAN.GT.IHELP(JANE)) GO TO 540
  530 IF(RAWST(ICHAN).EQ.HNO) MODE(JANE)=-MODE(JANE)
      MODE(ICHAN)=MODE(JANE)
      JANE=JANE-1
      IF(JANE.EQ.0) JANE=7
      GO TO 550
  540 MODE(ICHAN)=0
  550 CONTINUE
C
```

```
   13   FORMAT(12 HØEVALUATIONS,6X,716)
C
C
C
C       WRITE INSTRUCTION TABLE ON DISK.
        CALL ENTER(IDK1,FILEN)
        WRITE(IDK1)INSTRU
        WRITE(IDSPL,14)
   14   FORMAT(//30HØTHIS TABLE IS STORED ON DISK.)
        GO TO 1Ø1Ø
C
C
C
C       ERROR MESSAGES.
  9ØØ   WRITE(ITT,9Ø1) NHELP
  9Ø1   FORMAT(I8,17H TOO SMALL VALUE.)
        GO TO 94Ø
C
C
  91Ø   WRITE(ITT,911)IHELP(ICHAN)
  911   FORMAT(I7,23H NOT A VALID CHANNEL #.)
        GO TO 94Ø
C
C
  92Ø   WRITE(ITT,921)
  921   FORMAT(52 H WINDOW SIZE MAY NOT BE SMALLER THAN EVALUATION RAT
        GO TO LOC
C
C
  93Ø   WRITE(ITT,931)NHELP
  931   FORMAT(I8,16H NOT POWER OF 2.)
C
  94Ø   WRITE(ITT,941)
  941   FORMAT(18H ENTER NEW STRING!)
        GO TO LOC
C
C       MESSAGE FOR TEST PURPOSES.
  998   FORMAT(36H P R O G R A M L O G I C E R R O R !)
  999   WRITE(ITT,998)
        PAUSE 21Ø421
C
C
C
 1ØØØ   CALL SEEK(IDK1,FILEN)
        READ(IDK1) INSTRU
C
C****** TEST.
        WRITE(8,1ØØ9)INSTRU
 1ØØ9   FORMAT(19 H INSTRUCTION TABLE./5(7I7/))
C******
 1Ø1Ø   CALL CLOSE(IDK1)
        STOP
        END

PIP V13A

>
```

```
T TT←DT2 SOSTRT SRC

C                S        T        A        R        T     .
C
C
C        VERSION 1,  CFF 5/10/74.
C
         DIMENSION FILEN(2),FILRAW(2)
C
         COMMON /IN/INSTRU(35),NGDS(7)
C
         EQUIVALENCE(INSTRU(7),IBUDIM)
C
         EXTERNAL SORTMO,SOCALC,SOMMM
C

         DATA IDKI/2/
         DATA FILEN(1),FILEN(2)/5HINSTR,3HUCT/
         DATA FILRAW(1),FILRAW(2)/5HSLRAW,4HDATA/
C
C        READ INSTRUCTION TABLE FROM DISK.
         CALL SEEK(IDKI,FILEN)
         READ(IDKI) INSTRU
C
C        OPEN DISK FOR TRANSFER BUFFER.
         CALL ENTER(IDKI,FILRAW)
C
C        GENERAL BUFFER DIMENSION MUST NOT BE BIGGER THAN THE
C        CORRESPONDING VALUE IN THE COMMON STATEMENTS.
         IBUDIM=4000
C
         CALL RTSINT
         CALL RTSPR(SOCALC,30,0)

         CALL RTSPR(SORTMO,20,0)
         CALL RTSRUN(SOCALC,0)
         CALL RTSPR(SOMMM,5,0)
         CALL RTSRUN(SOMMM,0)
         CALL RTSGO
C
         PAUSE 1
         STOP
         END

PIP V13A

>
```

T TT.DT2 SOCALC SRC

```
C
      SUBROUTINE  S O C A L C
C
C     DISTRIBUTES BUFFER SPACE.
C
C     VERSION 1,  CFF 5/10/74.
C
      DIMENSION NSAMPI(7),NEVAL(7),IWINDO(7)
     1,IGBPT(8),IGBST(7),NADST(7)
C
      COMMON /IN/INSTRU(35),NGDS(7)/PT/KPOINT(27)
      COMMON /MO/IDUMMY,INSTR,IDAT(4)
C
C
      EQUIVALENCE(INSTRU(1),IOAP)
     1,(INSTRU(3),LENGTH),(INSTRU(4),MREPT),(INSTRU(5),NCHANS)
     2,(INSTRU(6),MSAMP),(INSTRU(7),IBUDIM),(INSTRU(8),NSAMPI(1))
     3,(INSTRU(22),NEVAL(1)),(INSTRU(29),IWINDO(1))
     4,(KPOINT(1),IGBPT(1)),(KPOINT(9),ITRAPO),(KPOINT(10)
     5,ITRALI),(KPOINT(11),IGBST(1))
     6,(KPOINT(21),NADST(1))
C
      EXTERNAL SORTMO,TIME
C
C
C
C***  ID.
      WRITE(8,2)
    2 FORMAT(7H SOCALC)
C***
C
C     FILLWORD OCT. 377777
      MSAMP=131071
C
C     GET HIGHEST CHANNEL # AND LOWEST SAMPLING PERIOD.
      DO 10 ICHAN=1,7
      IF(NSAMPI(ICHAN).LE.0) GO TO 10
      NCHANS=ICHAN
      IF(NSAMPI(ICHAN).LT.MSAMP) MSAMP=NSAMPI(ICHAN)
   10 CONTINUE
C
C     SET TRANSFER BUFFER POINTER.
      ITRAPO=3
      ITRALI=250
C
C     CALCULATE REPETITION PERIOD AT BEST FOR ACTUAL INSTRUCTION TAB
      LENGTH=IBUDIM/(4*NCHANS)
C
      I=1
   20 I=I*2
      IF(I.LT.LENGTH) GO TO 20
C
C     INCREMENT LENGTH TO NEXT HIGHER POWER OF 2.
      LENGTH=I
C
C     IGBPT(1) POINTS TO THE FIRST WORD AFTER A-D BUFFER (RIGHT NOW)
      IGBPT(1)=LENGTH*NCHANS*2+1
C
   25 MREPT=LENGTH*MSAMP
      IOAP=MREPT
C
      DO 30 ICHAN=1,7
```

```
C         ... SAMPLING PERIOD SHOULD NOT BE LARGER
C         THAN REPETITION PERIOD. IF NECESSARY, CHANGE.
          IF(NSAMPI(ICHAN).GT.MREPT) NSAMPI(ICHAN)=MREPT
C
C         CALCULATE NUMBER OF DATA FOR TRANSFER BUFFER PER CHANNEL.
          NADST(ICHAN)=MREPT/NSAMPI(ICHAN)
C
C         COMPUTE POINTER FOR GENERAL BUFFER.

          IGBPT(ICHAN+1)=IGBPT(ICHAN)+MAX(IWINDO(ICHAN),MREPT)/NEVAL(ICH
          IGBST(ICHAN)=IGBPT(ICHAN)
C
C         OVER ALL PERIOD OF SLB EQUALS MAX(MREPT,WINDOW SIZES)
          IF(IWINDO(ICHAN).GT.IOAP) IOAP=IWINDO(ICHAN)
C
C         EVALUATE REQUIRED BUFFER FOR EACH SLB CALL.
     30   NGDS(ICHAN)=MREPT/NEVAL(ICHAN)
C
C         IS ENOUGH SPACE AVAILABLE IN GENERAL BUFFER?
          MGBS=IGBPT(8)-1-IBUDIM
          IF(MGBS.LE.0) GO TO 40
C
C         NOT ENOUGH SPACE AVAILABLE IN GB.
          LENGTH=LENGTH/2
          GO TO 25
     40   CALL RTSRUN(SORTMO,0)
C
          RETURN
          END

PIP V13A

>
```

T TT←DT2 SORTMO SRC

```
C
        SUBROUTINE  S  O  R  T  M  O
C
C       REAL TIME MODIFIER.
C
C       VERSION 1,  CFF 5/10/74.
C
        COMMON /GB/IGDB(4000)
        COMMON /IN/INSTRU(35),NGDS(7)
C
        EQUIVALENCE(INSTRU(1),IOAP),(INSTRU(3),LENGTH)
        1,(INSTRU(4),MREPT),(INSTRU(5),NCHANS),(INSTRU(6),MSAMP)
C
        EXTERNAL RTSADS,SOFIMO,SOCALC,SOBOR,SOMMM,TIME
C
        DATA ICOUNT/0/
C
        IT=IEXTF(TIME)
C
C***    T.
        WRITE(8,1)
    1   FORMAT(6H SORTMO)
C***
        IF(ICOUNT.EQ.0) GO TO 10
        CALL RTSDQ(RTSADS)
        CALL RTSDQ(SOFIMO)
        CALL RTSDQ(SOBOR)
C
C       THE SAMPLER BUFFER IS TWICE AS LONG AS THE USED BUFFER.
C       (VALUES ARE FOR ONE CHANNEL).
   10   LNGTAD=LENGTH*2
C
C
        CALL RTSAD(NCHANS,LNGTAD,IGDB(1),60,MSAMP)
        MSAST=IT+MSAMP
        CALL RTSKED(RTSADS,0,MSAST)
C
        CALL RTSPR(SOFIMO,40,IOAP)
        IFIST=IT+IOAP+MREPT
        CALL RTSKED(SOFIMO,0,IFIST)
C
        CALL RTSPR(SOBOR,10,MREPT)
        MREST=IT+MREPT
        CALL RTSKED(SOBOR,0,MREST)
C
C

        ICOUNT=1
        RETURN
        END
```

PIP V13A

>

```
C
          SUBROUTINE  S  O  F  I  M  O
C
C         FAST INSTRUCTION TABLE MODIFIER.
C
C         VERSION 1, CFF 5/10/74.
C
          LOGICAL SWITCH
C
          DIMENSION NSAMPI(7),MODE(7),NEVAL(7),IWINDO(7)
C
          COMMON /IN/INSTRU(35),NGDS(7)
          COMMON /MO/ICHAN,INST5,IDAT(4)
C

          EQUIVALENCE
          1 (INSTRU(8),NSAMPI(1)),(INSTRU(15),MODE(1))
          2,(INSTRU(22),NEVAL(1)),(INSTRU(29),IWINDO(1))
C
          EXTERNAL SOCALC,SOMMM
C
C
C***      ID.
          WRITE(8,1)
        1 FORMAT(5H SOFIMO)
C***
C
C         NO MODIFICATION REQUESTED BY MMM ?
          IF(INST5.NE.5) RETURN
C
C

          IDOMOD=0
C         TREAT THIS CHANNEL.
          DO 110 I=1,4
          IF(IDAT(I).NE.0) GO TO 150
      110 CONTINUE
C
C         FORGET THIS CHANNEL.
          DO 140 I=7,28,7
          K=I+ICHAN
      140 INSTRU(K)=0
          GO TO 200
C
C         MODIFY THIS CHANNEL.
      150 DO 170 INST5=1,4
          K=INST5*7+ICHAN
          IF(IDAT(INST5).EQ.0.OR.IDAT(INST5).EQ.INSTRU(K)) GO TO 170
          INSTRU(K)=IDAT(INST5)
          IDOMOD=1
      170 IF(NSAMPI(ICHAN).EQ.0) INSTRU(K)=0
C
C         CHECK, WHETHER VALUES FOR ICHAN MAKE SENSE.
          IF(IWINDO(ICHAN).LT.NEVAL(ICHAN)) NEVAL(ICHAN)=IWINDO(ICHAN)
          IF(NEVAL(ICHAN).LT.NSAMPI(ICHAN)) NSAMPI(ICHAN)=IWINDO(ICHAN)
C
C         CALCULATE BUFFER REQUIREMENTS.
      200 IF(IDOMOD.EQ.1) CALL RTSRUN(SOCALC,0)
C
C         TEST.
          IF(SWITCH(3)) CALL RTSRUN(SOMMM,0)
C
C         SWITCH OFF INSTRUCTION FOR INSTRUCTION TABLE MODIFICATION.
```

```
        RETURN
        END

PIP V13A

>
```

T TI.DT2 SOBOR SRC

```
      SUBROUTINE  S  O  B  O  R
C
C     BUFFER ORGANISATION ROUTINE.
C
C     VERSION 1,  CFF 5/10/74.
C
      LOGICAL SWITCH
C
      DIMENSION ITRANS(251),NSAMPI(7),MODE(7),NEVAL(7)
     1,IGBPT(8),IGBST(7),NADST(7),FILRAW(2)
      DIMENSION FTEST(2)
C
      COMMON /IN/INSTRU(35),NGDS(7)
      COMMON /GB/IGDB(4000)/PT/KPOINT(27)
C
      EQUIVALENCE(INSTRU(2),LIMO),(INSTRU(3),LENGTH)
     1,(INSTRU(4),MREPT),(INSTRU(5),NCHANS),(INSTRU(6),MSAMP)
     2,(INSTRU(7),IBUDIM),(INSTRU(8),NSAMPI(1)),(INSTRU(15),MODE(1)
     3,(INSTRU(22),NEVAL(1)),(KPOINT(1),IGBPT(1)),(KPOINT(9),ITRAPO
     4,(KPOINT(10),ITRALI),(KPOINT(11),IGBST(1),IADIM1)
     5,(KPOINT(21),NADST(1))
C
      EXTERNAL STIME
C
      DATA LATIME,ISEC,MIN/3*0/
      DATA IDISK/2/
      DATA FTEST(1),FTEST(2)/5HTESTF,4HILE1/
      DATA FILRAW(1),FILRAW(2)/5HSLRAW,4HDATA/
      DATA IADOLD/0/
C
C***  TEST.
      WRITE(8,3)
    3 FORMAT(4H SOBOR)
C***
C
C
C
C     GET TIME OF CURRENT SAMPLE IN MIN,SEC, AND MILLISEC.
C     (MUST BE CALLED AT LEAST ALL 2 MINUTES.)
      ITIME=IEXTF(STIME)
      MISEC=ITIME-LATIME
    5 IF(MISEC.LT.1024) GO TO 10
      LATIME=ITIME
      ISEC=ISEC+1
      MISEC=MISEC-1024
      GO TO 5
C
   10 IF(ISEC.LT.60) GO TO 15
      MIN=MIN+1
      ISEC=ISEC-60
      GO TO 10
C
C
C
C     MAIN LOOP OVER ALL CHANNELS.
   15 DO 500 ICHAN=1,7
C
C     CHANNEL NOT USED?
      IF(NSAMPI(ICHAN).EQ.0) GO TO 500
C
C     INITIALIZE IADPO TO OLDEST DATA POINT OF CURRENT CHANNEL.
```

```
      IADPO=IADOLD+IOHN..
      IF(IADPO.GE.IADIM1) IADPO=IADPO-IADIM1+1
C
C     (RE) SET A-D BUFFER COUNTER.
      IADCO=1
C
C     RAWDATA STORING?
      IF(MODE(ICHAN).LT.0) GO TO 400
C
C     CALCULATE INCREMENT FOR SAMPLE POINTER.
      INSAM=NCHANS* NSAMPI(ICHAN)/MSAMP
C
C     YES. ENOUGH FREE SPACE IN TRANSFER BUFFER?
  100 IF(ITRAPO.GT.ITRALI-2) GO TO 200
C
C     YES. STORE ACTUAL SAMPLING PERIOD INTO ADDRESS BOOK.
      ITRANS(ITRALI+1)=NSAMPI(ICHAN)
C
C     GENERATE CODE FOR BOOKKEEPING.
      ITRANS(ITRALI)=ICHAN*256+ITRAPO
C
C     MOVE DATA LIMIT FOR TRANSFER BUFFER.
      ITRALI=ITRALI-2
C
C     SIGN BORDER BETWEEN DATA AND ADDRESS BOOK WITH OCT. 377777.
      ITRANS(ITRALI+1)=131071
C
C     STORE RAW DATA INTO TRANSFER BUFFER.
C     LOOP RUNS FROM ACTUAL DATA POINT TO THE END OF ITRANS.
  105 IF(ITRAPO.GT.ITRALI) GO TO 210
C
C     ARE STILL DATA OF CURRENT CHANNEL AVAILABLE IN A-D BUFFER?
      IF(IADCO.GT.NADST(ICHAN)) GO TO 300
C
C     YES, STORE DATA INTO TRANSFER BUFFER.
      ITRANS(ITRAPO)=IGDB(IADPO)
C
C     INCREMENT TRANSFER BUFFER POINTER.
      ITRAPO=ITRAPO+1
C
C     INCREMENT A-D BUFFER COUNTER.
      IADCO=IADCO+1
C
C     INCREMENT A-D BUFFER POINTER.
      IADPO=IADPO+INSAM
C
  110 IF(IADPO.GE.IADIM1) IADPO=IADPO-IADIM1+1
      GO TO 105
C
C
C     NOT ENOUGH SPACE FOR DATA IN ITRANS.
C     FILL FREE BUFFER UP WITH LIMITER.
  200 ITRANS(ITRAPO)=ITRANS(ITRALI)=131071
C
C     TRANSFER BUFFER FILLED UP. STORE ON DISK.
  210 WRITE(IDISK)ITRANS
      IF(SWITCH(1)) CALL CLOSE(IDISK)
C
C     INCREMENT RECORD NUMBER BY 1.
      IREC=IREC+1
C
C     STORE CURRENT SAMPLE TIME INTO TRANSFERBUFFER.
      ITRANS(1)=MIN*64+ISEC
      ITRANS(2)=MISEC
C
C     RESET TRANSFER BUFFER POINTER.
```

```
      ITRALI=250
      GO TO 100
C
C
C
C     THERE ARE NO MORE DATA FROM THIS CHANNEL IN A-D BUFFER.
C     RESET A-D BUFFER POINTER TO BEGIN OF CURRENT CHANNEL.
  300 IADPO=IADOLD+ICHAN
C
C     STORE REQUIRED DATA FOR EVALUATIONS INTO GENERAL DATA BUFFER.
  400 IPOINT=IGBST(ICHAN)
C
C     CALCULATE INCREMENT FOR EVALUATION POINTER.
      INEVAL=NCHANS*NEVAL(ICHAN)/MSAMP
      IEND=IPOINT+NGDS(ICHAN)-1
C
C     LOOP RUNS FROM ACTUAL DATA POINT TO THE END OF CURRENT
C     BLOCK OF THIS CHANNEL.
      DO 410 IPT=IPOINT,IEND
      IGDB(IPT)=IGDB(IADPO)
C
C     INCREMENT A-D BUFFER POINTER.
      IADPO=IADPO+INEVAL
C
  410 IF(IADPO.GE.IADIM1) IADPO=IADPO-IADIM1+1
C
C     ALL DATA FOR EVALUATIONS IN GENERAL BUFFER?
      IF(IPT.EQ.IGBPT(ICHAN+1)-1) GO TO 450
C**** T.
      WRITE(8,411)IPT
  411 FORMAT(I6)
C****
C
C     NO, INCREMENT GENERAL BUFFER POINTER.
      IGBST(ICHAN)=IEND+1
      GO TO 500
C
C     CALL EVALUATION ORGANISATOR.
  450 CALL SOEVAL(ICHAN)
C
C     RESET GENERAL BUFFER POINTER.
      IGBST(ICHAN)=IGBPT(ICHAN)
C
  500 CONTINUE
C
C     EVALUATE (IADOLD INCREMENTED BY USERLENGTH) MODULO
C     A-D BUFFERLENGTH.
      IADOLD=IADOLD+LENGTH*NCHANS
      IF(IADOLD.GE.IADIM1) IADOLD=IADOLD-IADIM1+1
C
C     GREEN LIGHT FOR MODIFICATIONS.
      LIMO=1
C
      RETURN
      END

PIP V13A

>
```

T TTT←\\←DT2 SOMMM SRC

```
C
      SUBROUTINE  S  O  M  M  M
C
C     MAN - MACHINE - MESSAGE.
C
C     VERSION 1,  CFF 5/10/74.
C

      DIMENSION IOPT(5)
C
      COMMON /MO/ICHAN,INST5,IDAT(4)
C
      DATA NOPT/5/
      DATA INSTR/5/
      DATA IOPT(1),IOPT(2),IOPT(3),IOPT(4),IOPT(5)/
     1          35,      29,      21,      39,      23/
C
      ICHAR=-29
C***  ID.

  200 WRITE(8,2) ICHAN,INST5,IDAT
    2 FORMAT(6H SOMMM,6I8)
C***
      INST5=INSTR
C     WAIT, UNTIL FAST INSTRUCTION TABLE MODIFICATOR SETS INSTR=0.
  100 IF(INST5.EQ.5) GO TO 100
      ICHAN=0
      DO 5 INSTR=1,4
    5 IDAT(INSTR)=0
  105 IDAT(INSTR)=0
      INSTR=0
  110 ISIGN=1
C
C     NEW INPUT?
   10 IF(ITTCAR(IDUMMY)-176.EQ.ICHAR) GO TO 10
      DO 120 I=1,100000
  120 CONTINUE
      ICHAR=ITTCAR(IDUMMY)-176
C*****T.
      WRITE(8,6)ICHAR
    6 FORMAT(10X,I3)
C*****
C
C     CHANNEL# ALREADY KNOWN?
      IF(ICHAN.NE.0) GO TO 20
C
C     IF NOT A VALID CHANNEL #, IGNORE!
      IF(ICHAR.LT.1.OR.ICHAR.GT.7) GO TO 10
      ICHAN=ICHAR
      WRITE(8,1)ICHAN
    1 FORMAT(8H CHANNEL,I3)
      GO TO 10
C
C     INSTRUCTION ?
   20 DO 30 I=1,NOPT
C
C     ATTRIBUTE CHARACTER TO ITS OPTION.

      IF(ICHAR.EQ.IOPT(I)) GO TO 40
   30 CONTINUE
C
```

```
C     INSTRUCTION NOT YET KNOWN?
      IF(INSTR)50,10,50
C
C     FINISH LAST INSTRUCTION.
   40 IF(INSTR.EQ.0.OR.IDAT(INSTR).EQ.0) GO TO 45
      IF(INSTR.NE.2) IDAT(INSTR)=2**IDAT(INSTR)
      WRITE(8,4)IDAT(INSTR)
    4 FORMAT(10H NEW VALUE,I8)
C
   45 INSTR=I
      WRITE(8,3) INSTR
    3 FORMAT(12 H  INSTRUCTION,I3)
C
C     IS INSTRUCTION A G (GO)?
      IF(INSTR-5)10,200,10
C
C     IS CHARACTER A MINUS SIGN?
   50 IF(ICHAR.EQ.-3) GO TO 60
C
C     IF CHARACTER NOT A DIGIT, IGNORE.
      IF(ICHAR.LT.0.OR.ICHAR.GT.9) GO TO 10
C
C     NOT THE ONLY DIGIT OF THE NUMBER?
      IDAT(INSTR)=IDAT(INSTR)*10+ISIGN* ICHAR
C
C     IF NUMBER OUT OF RANGE, IGNORE WHOLE INSTRUCTION.
      IF(INSTR.EQ.2) GO TO 110
      IF(IDAT(INSTR).LT.0.OR.IDAT(INSTR).GT.16) GO TO 105
      GO TO 10
C
C     IF NOT A MODE CHANGE INSTRUCTION, IGNORE MINUS SIGN.
   60 IF(INSTR.EQ.2) ISIGN=-ISIGN
      GO TO 10
C
      END

PIP V13A

>
```

T TT←DT2 MAX SRC

```
        .TITLE MAX
        .GLOBL MAX,GETARG,GETADR,.GETAD,.GOTAD
        /
        /
        /RETURNS THE BIGGER VALUE OF TWO ARGUMENTS
        /CALLING SEQUENCE: K=MAX(I,J)
        /
        /
        /VERSION 1,  CFF 5/10/74.
        /
        /
MAX     XX                  /ENTRANCE,EXIT
        LAC     MAX
        DAC*    .GETAD      /STORE STARTADR.
        JMS*    GETARG
        DAC     I#          /FIRST ARGUMENT
        JMS*    GETARG
        DAC     J#          /SECOND ARGUMENT
        TCA                 /-J
        TAD     I           /I-J
        SMA                 /NEGATIVE?
        JMP     POS         /NO.
        LAC     J           /YES, I.E. J=MAX
        SKP
POS     LAC     I           /I=MAX
        JMP*    MAX         /RETURN
        /
        .END MAX
```

PIP V13 A

>

```
T TT←DT2 IPOWR2 SRC

        .TITLE IPOWR2
        .GLOBL IPOWR2,.GETAD,GETADR,GETARG,.GOTAD
        /
        /
        /VERSION 1,  CFF 5/10/74.
        /RETURNS AC=-1, IF ARGUMENT A POWER OF 2,
        /         AC=0 OTHERWISE.
        /CALLING SEQUQENCE: I=IPOWR2(N).
        /
IPOWR2  XX                  /ENTRANCE,EXIT
        LAC     IPOWR2
        DAC*    .GETAD
        JMS*    GETARG
        TCA                 /COMPLEMENT
        DAC     INCOMP#     /STORE COMPLEMENT OF ARGUMENT
        LAC     (1          /LOWEST NONNEGATIVE POWER OF 2
        DAC     I#
/
/
COMPA   TAD     INCOMP      /DIFFERENCE: ARGUMENT-CLIMBING ↑2
        SMA                 /NEGATIVE?
        JMP     DECIDE      /NO.
        LAC     I
        RCL                 /NEXT POWER OF 2
        DAC     I
        JMP     COMPA       /CONTINUE LOOP
/
DECIDE  SZA!CLC                     /ZERO?
        CLA                 /.FALSE.=0
        JMP*    IPOWR2      /.TRUE.=-1
/
        .END IPOWR2

PIP V13 A

>
```

```
T TT←DT2 ITTCAR SRC

        .TITLE ITTCAR
/
/ITTCAR PUTS LAST TYPED CHARACTER (BITS 10-17)
/INTO AC.
/
/VERSION 1,  CFF 5/10/74.
/
        .GLOBL ITTCAR
/

KRB=700312
/
ITTCAR  XX
        KRB
        JMP*    ITTCAR
/
        .END

PIP V13A


>
```

Appendix D

References:

<1> Okajima, M., Stark, L., Whipple, G., and Yasui, S., "Computer Pattern Recognition Techniques: Some Results with Real Electrocardiographic Data", IEEE Transactions on Bio-Medical Electronics, July, 1963.

<2> Martin, W.B., Johnson, L.C., Viglione, S.S., Naitoh, P., Joseph, R.D., and Moses, J.D., "Pattern Recognition of EEG-EOG as a Technique for All-Night Sleep Stage Scoring, Amsterdam, 1972.

<3> Probst, W., Schulz, H., Dirlich, G., Schuh, H., and Friedrich-Freksa, C., "On-Line Classification of Sleep Stages with a Lab Computer", In: The Nature of Sleep, Stuttgart, 1973.

<4> Dirlich, G., Friedrich-Freksa, C., and Schulz, H., "A Time-Dependent Model of the Sleep EEG", In: Sleep, Basel, 1973.

<5> Gerster, F., Max-Planck-Institute for Psychiatry, Munich: ABS (General Biosignal Language), unpublished.

<6> Raimondi, IBM Research Laboratory, San Jose, unpublished.

<7> Harris, P., Johnston, J., Langley Porter Neuropsychiatric Institute, UCSF: RTS (Real-Time System), unpublished.

<8> Systems Development Group, IBM Research Laboratory, San Jose, unpublished.

<9> Martin, J., "Design of Real-Time Computer Systems", Prentice-Hall, 1967.

<10> Knuth, D.E., "The Art of Computer Programming", Vol 1 (Fundamental Algorithms), Addison-Wesley, 1969.

<11> Martin, J., "Programming Real-Time Computer Systems", Prentice-Hall, 1965.

<12> Coffman, E.G. and Denning, P.J., "Operating Systems Theory", Prentice-Hall, 1973.