SHARE PROGRAM LIBRARY AGENCY

PROGRAM NUMBER

09/001

University of Miami

1365 MEMORIAL DRIVE - CORAL GABLES, FLORIDA

(305) - 284-6257

SHARE PROGRAM LIBRARY SUBMITTAL FORM

SHARE

SHARE PROGRAM LIBRARY AGENCY
Triangle Universities Computation Center
Post Office Box 12076
Research Triangle Park
North Carolina 27709-2076 USA

SPLA CONTROL NUMBER: _____

This form should be completed and submitted with the program package to the SHARE Program Library Agency at the address shown above. Standards and instructions for submitting programs are in the SHARE Program Library Agency User's Guide and Catalog of Programs, or may be obtained from SPLA.

(1) Program Number (if new program, will be filled in by SPLA) 370D-09.1.001

(2) Title of Program ..... LOGTIME and MAPTIME

(3) System Type(s) (Machine) ..... System 370

(4) Search Keys ..... performance measurement timing

(5) Programming Systems/Languages ..... VM, Fortran

(6) Primary Subject Code 09.1

(7) Minimum System Requirements ..... CMS and Fortran

(8) New (N) or Revision (R) ..... N

(9) Date of Submittal ..... 13 JUNE 1986

(10) Documentation (number of pages of hardcopy submitted) 9

(11) Author's Name and Address ..... RANDOLPH G SCARBOROUGH
IBM SCIENTIFIC CENTER
1530 PAGE MILL ROAD
PALO ALTO CALIF 94304

(12) Name and Address of Contact for Technical Inquiries (if different from author) ..... (same)

(13) Submitter's Installation Membership Code ..... GSN

(14) Abstract (use reverse side of this form)

DISCLAIMER
Users should note that Triangle Universities Computation Center serves solely as the distribution agent for contributed programs and does not test or maintain them. They are distributed essentially in the original form submitted by the author. Neither TUCC nor SHARE Inc. makes any warranty, expressed or implied, as to the documentation, function, or performance of the contributed programs.

SPLA 1001 Revised 12-84

SHARE PROGRAM LIBRARY SUBMITTAL FORM

The abstract should contain sufficient information for a reader to determine the value of the program. Subjects listed below may serve as a guide for information to include.

a. Purpose of Program
b. Programming Language Used
c. Version and modification level or release number
d. Field of application
e. Type of routine (main program, subroutine, etc.)
f. Specific description of machine requirements

LOGTIME and MAPTIME are programs able to discover where an object program spends its CPU time while running under VM/CMS. LOGTIME, which executes in parallel with the user object program to be analyzed, uses the virtual interval timer to observe and record the locations in the object program where CPU time is spent during execution. MAPTIME subsequently formats the recorded information so that it may be analyzed. LOGTIME is written in assembler language and MAPTIME is written in Fortran. They are both complete programs which execute under control of companion EXECs.

(Please attach additional abstract pages if necessary. Total pages attached _____)

An "Acknowledgement of Assistance" statement must be attached to this Submittal Form.

Conveyance of Rights:

"I hereby convey to SPLA all rights to these submitted materials, including, but not limited to, the unqualified right to reprint, reproduce, and publicly distribute this program, except that I expressly retain the rights to utilize, reprint, reproduce, and distribute the materials for my own purposes."

(15) Signature of Submitter and Date Randy B. Scarbrough 14 MAY 1986

(16) Signature of Installation Addressee (if SHARE member) _____

SPLA 1001 Revised 12-84

K120053

LOGTIME and MAPTIME

LOGTIME and MAPTIME are programs able to discover where an object program spends its CPU time. LOGTIME monitors an execution of the object program to record where time is spent, and MAPTIME summarizes the recorded information to report where time is spent. The programs may be run directly or with the assistance of CMS exec files LOGTIME and MAPTIME.

Typically the programs are used in the following manner. LOGTIME is invoked to initialize itself and to construct a table, the LOGDATA table, which will represent a range of object program storage to be monitored. The range of storage to monitor, usually determined by examining the load map for the object program to be monitored, is specified with parameters. LOGTIME is then told to begin timing. The object program to be monitored is invoked. LOGTIME, executing in parallel with the object program, observes the object program and records in the LOGDATA table where the object program is spending its time. The object program is run normally to completion. LOGTIME is then told to cease timing, to write the LOGDATA table into a disk file, and to terminate. MAPTIME is invoked to read, analyze, and format the data in the LOGDATA disk file.

LOGTIME monitors the object program by interrupting it at regular intervals and examining it to determine the location it is executing. These intervals are generated by operating the virtual interval timer in a virtual machine. The timer is set to expire after a standard interval and the object program is given control. When the interval expires, LOGTIME determines the location undergoing execution in the object program and increments a counter for this location in the LOGDATA table. This procedure is repeated until the object program is completed. If enough of these intervals were generated during the execution of the object program then the recorded location counters should indicate where the time actually was spent.

MAPTIME presents in a usable form the data recorded by LOGTIME. MAPTIME reads the LOGDATA table written by LOGTIME and the load map produced when the object program was loaded for execution. Then it produces histograms showing where the time was spent. One of two output formats may be selected. The first format contains one line per control section (a control section is roughly equivalent to a subroutine) summarizing all monitor intervals which expired in that control section. The second format contains one line for each span in the object program in which at least a given percentage of the intervals expired. The size of this span and the range of object program storage to be presented in the histogram may be specified by parameters.

The LOGDATA table is contained in virtual storage while LOGTIME and MAPTIME are executing and is transferred between them in a disk file.

The table consists of four-byte entries each of which represents a span in the storage of the object program under observation. The size of the object program storage span corresponding to an entry is given by parameter and typically ranges from eight to 256 bytes. All monitor intervals which expire in a span are recorded in the single table entry for the span. As the span gets shorter the resolution available with LOGTIME and MAPTIME gets better--it becomes possible to pinpoint individual instructions in the object program--but but the storage required for the LOGDATA table gets larger. The range of object program storage to be represented in the LOGDATA table similarly is specified by parameter. Normally the range might cover only the addresses contained in the load map for a program (excluding the common blocks at the end--no execution occurs in a common block). Sometimes, however, it is useful to monitor the entire virtual storage range of a machine to see if CPU time is burned in system routines. As the storage range increases, of course, the LOGDATA table gets larger.

Given below are explanations of the LOGTIME and MAPTIME parameters. A quick word, however, about the LOGTIME and MAPTIME exec files. LOGTIME will run a complete LOGTIME and MAPTIME sequence for monitoring a single object program. It will produce on the terminal and in the file MAPTIME LISTING a histogram showing the percent of time spent in each control section and the percent of time spent in each storage span in which at least 0.5 percent was spent. You must know the storage range you want monitored, the storage span you want per LOGDATA table entry, and the length of the monitor interval you want LOGTIME to use, and you must know the name of the file containing the load map for the object program you intend to monitor. LOGTIME will prompt for this information. MAPTIME can be used to analyze the result of a LOGTIME in more detail. It will prompt for MAPTIME requests and produce on the terminal and in the file MAPTIME LISTING the result of executing these requests. It can be used to analyze only the LOGDATA produced by the most recent LOGTIME execution.

Please direct questions and comments to Randolph G Scarborough, IBM Scientific Center, 1530 Page Mill Road, Palo Alto, California, 94304 (415-855-3244 or 8-465-3244) (PALOALTO SCARBORO).

LOGTIME

LOGTIME may be invoked with the following parameters:

INIT address0 address1 spansize timeunit: Initialize LOGTIME, obtain storage for the LOGDATA object program timing table, and zero all counts in the LOGDATA object program timing table. Four subparameters (all in hexadecimal) are required:

address0: the lowest virtual storage address to be monitored.

address1: the highest virtual storage address to be monitored.

spansize: the span in bytes of the discrete intervals into which the monitored virtual storage is to be broken for recording. Interrupts taken at all addresses within a span will be recorded together. Two bytes is the smallest possible useful span, eight bytes is the smallest span which guarantees that no span contains storage from more than one control section, and sixty-four bytes (hex 40 bytes) is probably the largest span with which it is possible to identify accurately individual Fortran inner loops.

timeunit: the number of timer units to be used for each interrupt interval during monitoring. Smaller numbers (down to a machine-dependent threshold) mean more interrupts will be generated. Except for extraordinarily long-running programs 1 is fine. (This number is used to initialize the virtual interval timer for each interrupt interval. A value of 0100 formally represents a timer interrupt interval rate of 300 per second. Smaller numbers will probably be treated as if 0100 had been entered.)

TERM: Terminate LOGTIME and release the storage obtained for the LOGDATA object program timing table.

ZERO: Zero all counts in the LOGDATA object program timing table. (Timing is not automatically begun.)

TIME: Begin timing. (The LOGDATA table is not automatically zeroed.)

WAIT: Cease timing. (The LOGDATA table is not automatically filed.)

FILE: Write the LOGDATA object program timing table into the file LOGTIME LOGDATA A1. The table is appended to the file if the file already exists. (The LOGDATA table is not automatically zeroed.)

Several parameters may be specified in a single LOGTIME command. They will be executed in order from left to right. The sequence of commands

```
LOGTIME INIT 20000 5FFFF 40 1 TIME
X
LOGTIME WAIT FILE ZERO TIME
X
LOGTIME WAIT FILE TERM
```

for example will produce two LOGDATA object program timing table images in the LOGTIME LOGDATA A1 disk file and each image will record a separate execution of the command X.

Sometimes it may be desirable to time only a section of an object program. A one-byte flag is available for this purpose. The flag is at storage location X'00005C'. It is interrogated whenever a monitor interval expires between LOGTIME TIME and LOGTIME WAIT. If the flag is X'00' then the interval is recorded; if the flag is X'80' then the interval is discarded. These values may be inserted with CP ST S5C 00 and CP ST S5C 80 commands. The flag should be set to no other values (no checking is performed) and should be manipulated only between LOGTIME INIT and LOGTIME TERM.

LOGTIME depends upon the VM/370 supervisor to simulate the virtual interval timer accurately. How accurate this simulation is is not known, but results for practical purposes seem to be adequate unless privileged instructions occur frequently during execution. The CP SET TIMER ON or CP SET TIMER REAL command must be issued before LOGTIME is executed.

Note that interrupts may be recorded while the virtual machine is waiting for input or output, including input or output from the terminal. The location recorded for interrupts while waiting, however, normally is in the supervisor (in the hex address range 010000-01FFFF) or in the OS simulation routines. If only object program storage is monitored then these waiting interrupts will not be recorded and the interrupts which are recorded will represent better the distribution of CPU time.

MAPTIME

MAPTIME is invoked with no parameters. It obtains its parameters by reading them as terminal input lines. The input lines may specify the following parameters:

SPAN address0 address1 spansize percents: Produce one histogram for each LOGDATA table image in the LOGTIME LOGDATA A1 disk file. The histogram will be governed by four required subparameters (three in hexadecimal and one in Fortran F8.2 format (decimal point required)):

address0: a virtual address in the lowest control section to be displayed. If address0 and address1 are both zero then the lowest storage address represented in the LOGDATA table image will be used.

address1: a virtual address in the highest control section to be displayed. If address0 and address1 are both zero then the highest storage address represented in the LOGDATA table image will be used.

spansize: the span in bytes in the monitored object program each line in the histogram is to represent. This MAPTIME spansize will be rounded upward to an integral multiple of the LOGTIME spansize. If spansize is zero then a summary histogram with one line per control section totalling all time spent within the control section will be produced.

percents: the percent of the total interrupts in the MAPTIME storage range which must be present in a histogram line for the line to be displayed.

NAME routine0 routine1 spansize percents: Produce one histogram for each LOGDATA table image in the LOGTIME LOGDATA A1 disk file. The histogram will be governed by four required subparameters (two in characters (eight characters maximum), one in hexadecimal, and one in Fortran F8.2 format (decimal point required)):

routine0: the name of the lowest-address control section to be displayed.

routine1: the name of the highest-address control section to be displayed.

spansize: same as spansize for the SPAN parameter.

percents: same as percents for the SPAN parameter.

TERM: Terminate MAPTIME. A null line is equivalent to TERM.

All interrupts in a span are considered to have occurred in the control section which contains the last byte in the span. If time is spent near the end of a control section but the span containing the end also contains the beginning of the next control section then the time near the end may be reported in the wrong control section. Fortran routines in particular perform prologue and epilogue processing at the end of the routine. This effect can be eliminated with small span sizes.

MAPTIME requires the following file definitions:

FILEDEF 1 DISK filename filetype filemode (RECFM FB LRECL 100 BLOCK 1000): This the load map for the monitored object program.

FILEDEF 2 DISK LOGTIME LOGDATA A (RECFM V LRECL 1028 BLOCK 1032): This is the LOGDATA object program timing table image produced by LOGTIME.

FILEDEF 3 DISK MAPTIME MAPWORK A (RECFM F LRECL 80 BLOCK 80): This is a one-record work file used during parameter translation.

FILEDEF 5 TERM: This is the source of parameters.

FILEDEF 6 TERM (RECFM FA LRECL 121 BLOCK 121): This is one of two places where the histograms are written.

FILEDEF 7 DISK MAPTIME LISTING A (RECFM FBA LRECL 121 BLOCK 12100): This is one of two places where histograms are written.

Note that MAPTIME is a Fortran program. Since disk space is expensive it is maintained as an object module rather than as a module module. When run it must be run with a LOAD MAPTIME (START) command issued while GLOBAL TXTLIB FORTLIB is in effect.

SAMPLE PROGRAM (LOGTIME INIT 020000 027FFF 2 1)

```
      LOGICAL*1 CARD(80)/80*08/
      I=0
      DO 00001 N=1,1000000
00001 I=I+(N+N)
      I=0
      DO 00002 N=1,1000000
00002 I=I+(N*N)
      I=0
      DO 00003 N=1,1000000
00003 I=I+(N/N)
      A=0
      DO 00004 N=1,0100000
00004 A=A+EXP(LOG(FLOAT(N)))
      DO 00005 N=1,0010000
00005 WRITE (1,00009) CARD
      STOP
00009 FORMAT(80I01)
      END
```

SAMPLE HISTOGRAM (MAPTIME SPAN 0 0 0 1.0)

LOGTIME 03/05/80 08:38:23

INTERRUPTS IN MACHINE RANGE		2786	100.0%
INTERRUPTS IN LOGTIME RANGE		2751	98.7%
INTERRUPTS IN MAPTIME RANGE		2751	98.7%

MAIN	020000	24.65	|****************************
IHOSEXP	020210	14.36	|**********
IHOSLGN	020468	15.09	|**********
IHOECOMH	0206B0	7.31	|********
FIOAP#	0214E0	0.0	|
IHOCOMH2	021B00	0.04	|
IHOERRM	0224A8	0.04	|
IHOFCVTH	022AD0	35.99	|**
IHOEFIOS	0237C0	2.36	|**
IHOFIOS2	024950	0.04	|
IHOEFNTH	024F98	0.0	|
IHOUOPT	025798	0.0	|
IHOCMSS	025AD0	0.15	|
IHOETRCH	025D90	0.0	|
IHOFCONO	026040	0.0	|
IHOFCONI	0268F8	0.0	|
IHOUATBL	026D10	0.0	|
IHOFTEN	027348	0.0	|

SAMPLE HISTOGRAM (MAPTIME SPAN 0 0 8 1.0)

LOGTIME 03/05/80 08:38:23

INTERRUPTS IN MACHINE RANGE			2786	100.0%
INTERRUPTS IN LOGTIME RANGE			2751	98.7%
INTERRUPTS IN MAPTIME RANGE			2751	98.7%

MAIN	020000		24.65	
		000138-000013F	1.20	*
		000150-0000157	1.56	**
		000158-000015F	1.53	**
		000160-0000167	13.30	***************
		000188-000018F	2.29	**
IHOSEXP	020210		14.36	
		000048-000004F	1.09	*
		000088-000008F	3.02	***
		000098-000009F	2.47	**
IHOSLGN	020468		15.09	
		000030-0000037	1.60	**
		000048-000004F	1.71	**
		000058-000005F	1.45	*
		0000A0-00000A7	1.78	**
		0000B0-00000B7	1.71	**
IHOECOMH	0206B0		7.31	
		000290-0000297	2.18	**
		000300-0000307	1.85	**
FIOAP#	0214E0		0.0	
IHOCOMH2	021B00		0.04	
IHOERRM	0224A8		0.04	
IHOFCVTH	022AD0		35.99	
		000480-0000487	1.85	**
		000490-0000497	1.56	**
		0004A0-00004A7	1.38	*
		0004A8-00004AF	2.33	**
		0004B0-00004B7	5.13	*****
		0004B8-00004BF	12.43	*************
		0004C0-00004C7	1.71	**
		0004C8-00004CF	1.85	**
		0004D0-00004D7	1.13	*
		0004D8-00004DF	5.78	******
IHOEFIOS	0237C0		2.36	
IHOFIOS2	024950		0.04	
IHOEFNTH	024F98		0.0	
IHOUOPT	025798		0.0	
IHOCMSS	025AD0		0.15	
IHOETRCH	025D90		0.0	
IHOFCONO	026040		0.0	
IHOFCONI	0268F8		0.0	
IHOUATBL	026D10		0.0	
IHOFTEN	027348		0.0	

SAMPLE HISTOGRAM (MAPTIME NAME IHOSEXP IHOSLGN 8 0.3)

LOGTIME 03/05/80 08:38:23

INTERRUPTS IN MACHINE RANGE	2786	100.0%
INTERRUPTS IN LOGTIME RANGE	2751	98.7%
INTERRUPTS IN MAPTIME RANGE	810	29.1%

```
IHOSEXP  020210                48.77
              000000-0000007    1.48  |******
              000008-000000F    0.74  |**
              000028-000002F    1.11  |*****
              000038-000003F    2.84  |**********
              000040-0000047    3.33  |************
              000048-000004F    3.70  |**************
              000050-0000057    1.23  |*****
              000058-000005F    0.86  |***
              000060-0000067    1.23  |*****
              000068-000006F    2.10  |********
              000070-0000077    2.72  |**********
              000078-000007F    0.86  |***
              000080-0000087    0.99  |****
              000088-000008F   10.25  |****************************************
              000090-0000097    0.74  |**
              000098-000009F    8.40  |********************************
              0000A0-00000A7    0.37  |*
              0000B0-00000B7    0.99  |****
              0000B8-00000BF    0.99  |****
              0000C8-00000CF    0.37  |*
              0000D0-00000D7    0.37  |*
              000100-0000107    0.49  |**
              000108-000010F    0.37  |*
              000110-0000117    1.60  |******
IHOSLGN  020468                51.23
              000000-0000007    1.73  |*******
              000008-000000F    1.98  |********
              000010-0000017    2.10  |********
              000018-000001F    0.86  |***
              000020-0000027    1.36  |******
              000028-000002F    0.62  |**
              000030-0000037    5.43  |*********************
              000038-000003F    1.60  |******
              000048-000004F    5.80  |***********************
              000050-0000057    0.99  |****
              000058-000005F    4.94  |*******************
              000060-0000067    1.98  |********
              000068-000006F    0.86  |***
              000078-000007F    2.84  |**********
              0000A0-00000A7    6.05  |************************
              0000A8-00000AF    0.74  |**
              0000B0-00000B7    5.80  |***********************
              0000C8-00000CF    2.72  |**********
```

TAPE KEY for the LOGTIME and MAPTIME program.

This tape is unlabeled and is in CMS TAPE DUMP format; the data may be loaded with the CMS TAPE LOAD command. The tape physically contains one file. When loaded via TAPE LOAD, seven CMS files will be retrieved.

LOGTIME	ASSEMBLE	A1	F	80	342	36	06/13/86 09:47
LOGTIME	EXEC	A1	V	80	57	6	06/13/86 09:47
LOGTIME	MODULE	A1	V	2008	3	6	06/13/86 09:48
LOGTIME	SCRIPT	A1	V	80	482	26	06/13/86 09:44
MAPTIME	EXEC	A1	V	75	25	6	06/13/86 09:46
MAPTIME	FORTRAN	A1	F	80	268	31	06/13/86 09:45
MAPTIME	TEXT	A1	F	80	172	21	06/13/86 09:45

--- EOF --- 139 BLOCKS, MAX SIZE 805