MASTER

KMSF-U 970
COPY 1

.kms fusion inc.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

KMSF-U970

CONF-800414--1

THE KMS FUSION SYSTEM RESOUCE ACCOUNTING AND PERFORMANCE MEASUREMENT SYSTEM FOR RSX11M V3.2 *

James G. Downward
KMS Fusion, Inc.
Ann Arbor, Michigan 48106

To be presented at Spring DECUS meeting in Chicago in April 1980.

*Work supported by the U.S. Department of Energy under contract DE-AC08-78DP40030.

"By acceptance of this article, the publisher and/or recipient acknowledges the U.S. Government's right to retain a nonexclusive, royalty-free license in and to any copyright covering the article."
NOTE: This notation need not appear in the published article.

DISCLAIMER

This book was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

OF THIS DOCUMENT IS UNLIMITED

THE KMS FUSION SYSTEM RESOURCE ACCOUNTING AND PERFORMANCE MEASUREMENT SYSTEM FOR RSX11M V3.2

James G. Downward
KMS Fusion, Inc
Ann Arbor, Mich. 48104

Past History of Accounting for RSX11M

Since Multi-User protection was implemented on RSX11M, the user community has felt a need for tools with which to measure system performance and to account for the system's usage. Various users have implemented accounting schemes of one form or another over the past few years to satisfy their local needs. Several of these packages have found their way into the DECUS library. An accounting package developed at McGill University that is aimed primarily at charge back accounting is now marketed by GEJAC, Inc. A system for terminal usage accounting and run privlege allocation was developed by KMS Fusion and has been used by many systems not needing accounting for CPU time usage. A limited task accounting scheme was developed by Greg Basset(DEC). It was aimed at measuring the performance parameters (CPU time, QIO count) for a list of tasks input from the console.

RSX11M-PLUS has an accounting system which provides hooks for measuring performance of the system, and also provides enough statistics for providing chargeback accounting if it is so desired. However, the M-PLUS accounting scheme is not suitable for use on an RSX11M system. Firstly, if it is selected as a SYSGEN option, a small system overhead is imposed. Secondly, its design was based on eventually having to implement task accounting. This has made the scheme substantially more complicated than one which was aimed at providing performance measurement tools for tuning the system as its primary goal. Thirdly, it uses up a lot of pool. Pool space (and core space) is much more limited on a standard RSX11M system than on an M-PLUS system running on an 11/70. Fourthly, it creates rather large log files.

The KMS Fusion package is designed to incorporate most of the features of past accounting systmes and to provide statistics suitable for monitoring system performance in a manner consistent with the limitations of an RSX11M system.

System Overview

Version 3.2 of the KMS FUSION accounting system is aimed at providing the user of RSX11M V3.2 with a versatile tool for measuring the performance of the operating system, tuning the system, and providing sufficient usage statistics so that the system manager can implement chargeback accounting if it is required by the ins-

tallation. Sufficient hooks are provided so that the intrepid user can expand the system substantially beyond what is currently provided.

The basic system is implemented, incorporating only minor changes in the EXEC data base (SYSCM) and TDSCH. The user is able to select on an individual basis, CPU time accounting, global performance parameter logging (shuffler requests, loader requests, checkpoint requests, system null time tics), total system QIO's issued, user by user QIO totals, logon/logoff statistics for each terminal, memory usage and pool usage logging, CO: statistics loggings, and disk block usage at logoff. Additionally, task accounting for selected tasks is provided. A summary of the data which may be gathered and logged is seen in the following table.

GLOBAL:	Exec Null Time	USER:	Logon/Logoff Times.
	# Run requests		Account # and Terminal ID
	# LDR Requests		CPU Time.
	# checkpoint Requests		Disk Usage
	# SHF Requests		# Illegal Login Attempts.
	# QIO's Issued(optional)		# QIO's Issued(optional).
			# RUN requests.
			# ...AT. runs.
			# INS runs.
			# Pages Printed(optional).
CO:	CPU Time		
# QIO's	Issued(optional)		
	# RUN requests.		
	# ...AT. runs.		
	# INS runs.		
MEMORY:	# Tasks in Core	TASK:	Stop Time.
	Total Memory in Use		Run Time.
	# Tasks out of core.		CPU Time
	Total Memory Checkpointed		# QIO's Issued.
	Total free Pool.		
	Largest fragment in Pool		
	# fragments in pool.		

The system is composed of four tasks; 1) an MCR interface task ACC which will control and reconfigure accounting on-line and display the current status of accounting, 2) the accounting task, SYSLOG, which will insert hooks into the EXEC to implement the accounting code, 3) a logging task which will recieve data packets from SYSLOG, HELLO, BYE, TSKLOG, and UPD, and 4) the task TSKLOG which implements task accounting for a list of tasks input at the user's terminal. The task accounting is desigined primarily as a performance measurement tool.

Accounting information accumulates in two places. Terminal connect time, user CPU time, and user disk blocks in use is updated in the system account file RSX11.SYS at each logoff (for both TT:'s

and VT:'s). For instance if only CPU time accounting is enabled, the system logging file (SYSLOG.DAT) will only accumulate data on when system logging starts and stops. This is compatible with systems which are severely limited in disk storage space and only occasionally want to activate additional accounting features. In addition the user may select that system wide and user statistics be logged to a file(LB:[1,4]SYSLOG.DAT). The data in the log file is readable via a standard F4P task. The format of the data packets sent to LOGTSK is to be found in TABLE II at the end of this document. Sufficient data is logged to provide terminal usage profiling, total system utilization, user disk block usage at each logoff, CPU time used by each account, system total QIO's issued and a user by user total of QIO's issued. In addition, a user by user count is kept of the number of tasks run (activated) by a user, the number of uses of INS, and the number of times ...AT is run.

The task SYSLOG will log the global system statistics every 'DELTA' minutes, issue a mark time, and stop itself until the mark-time is satisfied. By varying the sampling time the user may get as fine or as coarse a picture of the system as is desired. The parameters which are sampled, logged and reset are chosen to provide insight into how the system is performing, ie QIO's issued, SHUFFLER requests, LDR requests, checkpoint requests, and current memory usage statistics(pool, number of tasks in and out of core, and size of tasks in and out of core).

Design Constraints/Goals

Since RSX11M is primarily used by many users for its fast real time response and functionality on a small PDP11 system, a number of constraints are imposed in the beginning on any implementation of a suitable accounting/performance-measurement system.

1. There must be no system overhead if accounting is disabled.
2. Accounting must be reconfigurable on line.
3. Sufficient hooks must be present so that the user can monitor system performance, locate bottlenecks, etc.
4. It should be easily extensible by a user.
5. Sufficient information should be provided such that a user can implement any type of charge back accounting desired.
6. The accounting system's overhead must be an absolute minimum.
7. Accounting must not drastically decrease the system's pool.
8. Accounting must not use undue amounts of disk space.
9. The methods used to implement accounting must not use an additional APR because it is desirable that HELLO be buildable with an FCS resident library.

These goals are met by this implementation of the KMS Accounting System. It should be emphasized that previous features of the KMS Fusion Accounting/Enhancement package for RSX11M are either retained or their functionality is enhanced. System resourse accounting may be addded to these features to provide a system per-

formance measurement tool or charge back accounting on a user by user basis.

In order to be able to maintain the ability of HELLO to build with an FCS resident library and a 16K exec, it was decided to modify the Executive and terminal data bases rather than installing a common partition for accounting. As it turned out, the data base changes were by far the easiest part of the task.

It also would have been possible to have an accounting task which kept all its data internal to its node pool. This method avoids having to do a SYSGEN but has several severe disadvantages from our standpoint. Firstly, if the task is ever aborted by accident, one must reboot to start up accounting. Secondly, no other tasks can know anything about the location of the accounting terminal accounting nodes. This makes it much more difficult to have tasks like MCR... bump counters each time ...AT is run. All such counting must be done by sending messages to the system logging task. On an active system this can pose substantial overhead as well as filling pool with send packets if the logging task is checkpointed out for a while. Also, it is far faster to bump a counter at a known location than it is to scan for an accounting block. Finally, by using storage in a terminal's own UCB, privleged tasks have fast access to both their current QIO count and CPU time. This makes it relatively easy to implement time critical performance measurement tools to monitor a task's throughput and disk access time.

System Requirements

No attempt has been made to make this package universally usable by all users. It is assumed the prospective user has a mapped system and has selected multiuser protection and support for all executive directives. Not all executive directives are used by this package, but such a system is closest to the one the package was developed on. The prospective user will have to do a SYSGEN to incorporate the changes in the data bases for CO:, TT:, VT: and SYSCM. The modification to TDSCH.MAC must be done to define the variable UPTIM as a global symbol. For this SYSGEN only those changes need be done. All other components can be brought up piecemeal as time allows. However, the data structures required must be created via a SYSGEN even if they will not be immediately used. Three SLP files modify the SYSGEN command files to correctly ask questions concerning the options desired, and to generate the correct data bases. Two SLP files modify the EXEC modules SYSCM and TDSCH. Making these modifications will in no way result in a corrupted system or cause flaky crashes.

In addition to changing UPTIM to be a global symbol the TDSCH.SLP file implements a change which allows modifying the ROUND ROBIN schedule interval and disk swapping interval on line. (The task SCH does this). Modifications to SYSCM provide the system ac-

counting tasks with common storage variables and defines a new quantity LIBUIC. This later change was put in because ...AT can use it if it exists (an M-PLUS feature). Modifications to the set parser allow LIBUIC to be displayed and changed. Modifications to INS allow the RUN $FILENAME to search first to SYSUIC and then on LIBUIC. Hence, it is easy to have separate directories for privileged tasks and the common utilities.

While a number of standard privleged EXEC modules and privileged tasks get minor changes, all tasks can test an accounting feature mask, $ACMSK, to see if they are to execute accounting code or not. With as little as the addition of $ACMSK to SYSCM a single object module for a privileged task could be used for creating a task image for any system. Reassembly would not be necessary if all options were initially selected. It is assumed that the user desiring to implement this enhancement has sufficient skill to do a non-standard SYSGEN, apply SLP files and decide on what trade offs his/her system can tolerate to support acounting. A bare bones system with only 2 RK05 disks, minimal free pool, only 32 K words of memory, and lofty goals will have difficulty implementing, and using this package. Other users should have very little difficulty.

It goes almost without saying that nothing is without a cost. While the accounting system is very efficient and imposes a low overhead on the operating system (real time compatible), it does use up a certain amount of pool space. If CPU time accounting, QIO accounting, and run privilege allocation are all desired, about 80(10) words of pool are used up by a node of PIC code. If task accounting is selected the size of SYSLOG's code node increases slightly and of course there is TSKLOG's code node also to consider. An additional 7-9 words are used for each terminal UCB. SYSCM requires 12-25 extra words in its data base depending on the options selected. The SLP file provided creates a few more words than this in SYSCM for future expansion. The symbol $SPARE(4 words) may be commented out. I use it for testing out concepts requiring extra symbols in the Exec, before actually commiting to a SYSGEN. Table I provides a summary of the various size changes, and pool usage requirements imposed by the various options.

As mentioned before, to incorporate this package a SYSGEN must be performed. While the overhead of this package is low, it is not zero. The most significant overhead occurs at each clock tic. If the system is executing a task, each clock tic, 11 instructions are added to the clock ISR code. If the EXEC is in its idle loop 5 instructions are added to the clock ISR. This presents a small overhead for systems with a 60 Hz line clock or a 100 Hz programmable clock. Using this package with a faster programable clock is not recommended. For 60-100Hz clocks perhaps as much as 5% more code must be executed each clock tic (see TDSCH.MAC). This accounting system (or any system for that matter) probably is not well suited for an operating system with tasks doing very time critical interrupt processing.

TABLE I

MEMORY REQUIREMENTS FOR KMS SYSTEM ACCOUNTING

	SYSCM	TT: UCB	SYSLOG CODE	SYSLOG NODE	TSKLOG CODE	TSKLOG NODE
KMS RUN PRIV	2	1				
CPU TIME		3	24			
GLOBAL ACC.	10	3	30			
USER QIO ACC.		2	24			
MEMORY ACC.						
TASK ACC.	6		26		284	
TOTAL WORDS	18	9 PER UCB	124 ALL OPT.		284 +10/TASK	

This table's function is to give a rough idea as to what various options cost in terms of lost pool. Note that the conditionalization of options is not yet complete and some options automatically drag in others. This is the case with the TT: UCB changes. The data block for user global accounting is lumped in with CPU time accounting. If the user is very careful to change all offsets to U.QIO1 and U.QIO2 in all of the supplied programs and SLP files, the section of each terminal's UCB reserved for global user accounting may be removed (3 words).

System Components

The accounting system is composed of a number of non-DEC tasks and SLP files for modifying certain RSX11M source modules. Incorporating the system will involve a system generation (which has become a rather simple matter for V3.2.).

ACC.MAC	The account task dispatcher/controller.
ACTFIL.MAC	New account file offset definitions.
LOGTSK.MAC	The log task.
SYSLOG.MAC	The central accounting task.
TSKLOG.MAC	The task accounting task.
ACNT.SLP	Enable an existing option to ACNT.
BYE.SLP	Modification to DEC bye.
HELLO.SLP	Modification to DEC hello.
INSHD.SLP	Counts usage of INS (please apply all INS SLP files, not just INSHD.SLP).
JOBEND.OBJ	Enables logging of print spooler usage.
MCRDIS.SLP	Counts usage of ...AT., enables usage of PIN and fixes several MCR bugs.
SYSGEN.SLP	If you want all loadable drivers and all loadable user written drivers, you don't have to edit RSXBLD.CMD.
SGNEXEC.SLP	Create new assembly parameters in each RSXMC.MAC.
SGNPER.SLP	Modify CO:'s data base.
SGNTT.SLP	Modify the terminal driver's data base.

In addition, modifications to the standard KMS Fusion enhancement package have been supplied which are compatible with the system resource accounting. Included in this category are updated versions of modifcations to RESET (zero CPU time when clock time zeroed), to ACCLOG (display the used CPU time and % system utilization), WHO (show who is using BATCH), and BATCH (log V7: CPU time and use PIN instead of ...AT.).

Component Description

ACC -- User Interface Task

The task ACC is used to start accounting, stop accounting, enable and disable accounting features on line, and display those features currently active. It has the following functions.

1. Start accounting.
2. Set the account feature mask for the desired options.
3. Change the accounting features desired (except for CPU time, null time and QIO accounting) on line without interrupting accounting.
4. Modify the accounting sample interval.
5. Turn off accounting.

6. Display the current accounting features.
7. Start/Stop task accounting.

command syntax:

ACC /sw1/sw2/sw3 ...

where the switches are

START	- Start up the accounting task SYSLOG.
STOP	- Stop the accounting task SYSLOG
CPU	- Start CPU and null time accounting .
GBL	- Log system global variables at each interval.
QIO	- Activate system wide qio accounting.
UQIO	- Activate system wide and user qio accounting.
DIR	- Activate directive accounting.
MEM	- Log memory utilization variables every sample interval.
FREE	- Log memory utilization variables if core space is tight or if the checkpoint files are in significant use.
POOL	- Log memory utilization variables if pool gets too low.
DLT:N	- Change sample interval time, to n minutes . Defaults to 30 minutes.
LOG	- Log HELLO/BYE/UPD sign on/off times, statistics, and terminals.
DSP	- Display current settings of account mask.
LCO	- Log co: statistics each sample interval.
TASK	- Start up task accounting, requires /UQIO
PRI[NT]	- Print task accounting statistics to TI: (not log file.)
SPO[OL]	- Enables logging of print spooler usage(new spooler).
NUQ[IO]	- Disable terminal qio accounting. (system wide qio accounting continues).
NLO[G]	- Disable logging hello/bye/upd data.
NLC[O]	- Disable logging co: statistics.
NME[M]	- Disable memory logging.
NFR[EE]	- Disable memory logging on low free core.
NPO[OL]	- Disable memory logging on low free pool.
NTA[SK]	- Disable task accounting.
NPR[INT]	- Disable printing task accounting to TI:
NSP[OOL]	- Disable logging print of spooler usage.

* -- All options enable CPU time accounting.

Switch Description

/START

The /START switch must be used on the command line starting up accounting. It must be followed with other switches in order to define the type of accounting configuration desired. When accounting starts up, a start packet of data (start time, sample interval, and $ACMSK) is sent to LOG...

/STOP

The /STOP switch will stop system accounting and task accounting(if active). It should be issued as a command by itself. Before SYSLOG exits a stop data packet (stop time,schedule interval, requesting task, and $ACMSK) is sent to log...

/DSP

The /DSP switch will display all currently active accounting options including the sample interval. It should be used on a command line by itself as it will display the current status as soon as it is decoded.

/CPU

The /CPU switch indicates that CPU accounting(the default) is desired. The /CPU switch need not be used if other accounting confisuration switches.

/GBL

If the /GBL switch is present, every 'DELTA' minutes, various global accounting parameters are scanned and accounting data packets sent to log. At present the global system statistics include run requests, SHUFFLER requests, checkpoint requests, and LDR requests. Until core becomes very overcrowded, these statistics provide a good measure of system activity in any period of time. If too many tasks are competing for core, the Exec will thrash about swapping tasks in and out. SHF will start being requested very often even though there is nothing it can do. If this accounting shows this to be a persistent and serious problem the task SCH is provided which can modify the Round Robin and Executive Disk Swapping intervals on line. By lengthening these times, tasks will be allowed to execute longer and thrashing should decrease to a manageable level.

Run requests(tasks started via RUN, MCR commands, spawning or requesting will be counted globally in SYSCM and also counted in each owning terminal (of CO:) UCB. Tasks already active will not be counted (ie stopped tasks) if they are requested(like MCR...). If user accounting is selected BYE will send off the user run count to LOGTSK with it's send packet. In addition BYE will display a summary of the user's connect time, CPU time, and number of tasks run (just like RSX11M-PLUS).

/QIO, /UQIO, and /NUQIO

If /QIO accounting has been selected as a SYSGEN option (it must for task accounting to work), the /QIO switch will enable SYSLOG to keep a count of the total system QIO's issued. This value is sampled every 'DELTA' minutes. The counter is a 2 word counter, but for a system with a lot of QIO activity some care must be exercised in selecting the value for 'DELTA' to insure that overflow does not wipe out the data.

The /UQIO switch enables user by user qio accounting. A count of the number of QIO's issued at a given terminal is made into each UCB. This counter is zeroed at logon, and at logoff its value is sent to LOGTSK. The switch /NUQIO may be used to turn off user by user QIO accounting unless tasklogging is operative in which case, using the switch is declared to be an error. System wide QIO accounting may not be directly disabled while SYSLOG is running. To turn off system wide QIO accounting, issue a ACC /STOP and when SYSLOG stops, issue another command to start it up with the appropriate option.

/MEM, /NMEM

The /MEM switch enables accounting for memory utilization (if it is a syssen option). A section of code borrowed from RMDEMO of the number of tasks in core, the number of tasks out of core, the size of task in core and the size of tasks out of core (checkpoint files). Additionally, it keeps track of pool (free pool, largest size, and number of fragments). The /NMEM switch disables logging memory statistics.

/FREE, /POOL, /NFREE, and /NPOOL

Memory utilization accounting can use up a lot of disk space quickly. Often all one wants is to keep a record of when the system started to hang up(too little pool, too little memory, too many tasks in and our of core, or too large tasks). The /FREE switch directs SYSLOG to send off a memory utilization data packet only if core space gets tight or if too many files are on checkpoint files. Currently Syslog will send off a memory record if /FREE is implemented, if more than 75.K words of tasks are in core or if more than 10.K words of tasks are out of core. These values should be changed to reflect the size of free core on one's system and the total amount of checkpoint file space available. Edit SYSLOG.MAC and Pf Inval. The switch /NFREE will disable this accounting feature. The POOL switch will do the same thing for pool. If the /POOL command switch is present, a memory utilizatoi. record will be logged every sampling interval during which pool is less than 500(10) words. However if pool is 200(10) words

or lower, no data will be sent, rather a counter (NOPOOL) is incremented to show how many times low pool would have forced logging pool and memory statistics. The value for NOPOOL is sent with each send packet and is then cleared. /NPOOL disables this option.

DLT:n

The /DLT:n switch set the sample interval to be 'n' minutes. At startup if no sample time is specified, a sample time of 30 minutes is assumed. Any time thereafter the /DLT switch may be used to change the sampling interval. Each time the sample interval is changed a change sample interval data packet is logged. This enables a program reading SYSLOG.DAT to keep track of what is happening. LOGTSK is sensitive to the sampling interval. If the samples are more than 5 minutes apart, it exits between times when it receives data. If sample intervals are more frequent, it stops itself and waits around for data to show up. This eliminates the overhead associated with opening/closing the log file.

/LOG, /NLOG

If the /LOG switch is specified, HELLO, BYE, and UPD all send off data packets to LOGTSK. The /NLOG disables this function. HELLO sends to LOGTSK the logon time, the terminal ID, the account number, the UIC and the number of illegal login attempts since the last logon. BYE sends off the logoff time, the terminal id, the acnt number, the UIC, the CPU time used, and the number of QIO's issued by the user. UPD sends off the logoff time, the terminal id, the account number, the user's system device, the number of files in use, and the number of blocks in use. From the data sent by these three tasks a sophisticated chargeback system could be implemented.

/LCO, /NLCO

A large number of tasks run with CO: as their TI:. Some of these are system tasks while others are real time tasks or tasks which have been scheduled. In order to keep track of the system resources used by these tasks, the /LCO switch is provided. The SYSLOG intercept code will log data to CO:'s UCB just as if it were a real terminal (this is why we require the CO: driver data base). Every 'DELTA' minutes the data (CPU time and QIO count) is logged and the values reset to zero. The /NLCO switch disables logging of data from CO:.

/TASK, /NTASK

If TSKLOG is installed, the /TASK switch will start up task accounting. System accounting must already be running with /UQID enabled. The /TASK switch should be a command by itself. Once it is issued, TSKLOG will prompt for the user to input a list of task names which are to be accounted for. TSKLOG will stop requesting tasks if a null line is entered. Only enter tasks which are going to exit. A task accounting packet is not logged until the task exits (task accounting starts when the task becomes active). Tasks like MCR which stop themselves are always active and can not be monitored short of aborting them. Aborting a task will cause a data packet to be sent, but aborting privileged exec tasks to monitor them is highly risky. Task accounting may be stopped by issuing either the /NTASK or /STOP command. THE /NTASK command stops just task accounting whereas the /STOP command stops everything.

/PRINT, /NPRINT

At times it is far more convenient to have data displayed directly on the user's terminal than to have it sent off to a log file. The /PRINT switch will disable sending task logging data to LOGTSK and will enable printing this data on the user's TI:. Note, that while TSKLOG is printing the task could start up again and TSKLOG would miss its start. The /NPRINT switch disables printing to the user's TI:, and the data is again sent off to LOGTSK.

/SPOOL, //NSPOOL

If one is using the new print spooler/Queue-manager, it is possible to log the total number of pages printed and the number of copies made on each print job. Do to the complexity of the Queue-manager/print spooler the user account number will only be logged with the spool packet if the user is logged on at his TI: when the job is printed. A user with print spooler sources and some time to kill could file the user account number to disk along with the print job and then send it off to the print despooler along with the files to be printed instead of sending the terminal ucb address. This is not hard to do but its implementation is deferred until the print spooler has stabilized and I have access to the up-to-date Queue-manager sources. Since the Queue manager sources are not distributed, this feature is implemented by replacing an object module in LPP.OLB. The switch /SPOOL enables logging print spooler statistics, and the /NSPOOL switch disables the logging.

SYSLOG has three primary functions; 1) inserting hooks into the exec for a PIC code node in pool to use, 2) periodically logging system global variables, 3) restoring the EXEC and Pool to virgin condition on exit. When SYSLOG is requested to run by ACC, it allocates a pool block and reads into it the accounting code (PIC). It then intercepts the instructions at various points in the EXEC and replaces them with hooks into (ie branches into) the appropriate locations in the code node. Once the code node has been hooked into the EXEC the system will run even if SYSLOG is aborted. Following an abort, the code node may be removed by starting up SYSLOG again. It will find its code node and start running. At anytime thereafter, SYSLOG may be normally stopped and the dynamic memory it is using reclaimed.

Next, SYSLOG uses the schedule interval passed on from ACC to issue a marktime. It then issues a receive data or stop. ACC can awaken SYSLOG by either sending a change scheduling interval packet, or a stop accounting packet. The marktime also will awaken SYSLOG. When this happens, the variables consistent with the setting of the system accounting mask are sent to the logging task, and then SYSLOG stops itself to wait for something else to happen.

If memory and pool utilization accounting is selected, SYSLOG becomes careful of sending out data packets to LOG.... If total free pool drops below 200(10) words it does not send out the packet. In this case it does increment a counter to show how many times it wanted to send out a packet and there was too little pool. The next time a packet is sent out this value will be sent along with it.

When it is time to exit, SYSLOG will replace in the EXEC those instructions which were overwritten when the hooks were inserted and will remove the code node from pool. When SYSLOG exits the system returns to the identical state it was in before SYSLOG was started.

Inserting and removing hooks into the EXEC is a rather non standard way of using a privileged program. However, it has a significant advantage over any other method, namely that there is absolutely no overhead added to the system unless accounting is started.

If the user selects that task accounting is desired, an additional two instructions are placed in the clock ISR routine, and if /UQIO is enabled the QIO intercept code adds a count to the task account control block QIO counter for every qio (if an accountable task is found). If task accounting is not enabled, only an additional 2 instructions of overhead are executed for a syslog assembled for task accounting.

LOGTSK

LOGTSK(LOG...) is an F4P program which receives data sent from TSKLOG, SYSLOG, ...HEL, HELTnn, ...BYE, BYETnn, ...UPD, and UPDTnn. if the sample interval is less than 5 minutes, it stops itself to wait for data. It the interval is longer, it exits after logging each packet(s).

HELLO

HELLO is modified (above and beyond the normal KMS Fusion modifications) slightly. First it reads the user's account number from the account file (RSX11.SYS) into a word (U.ACN) in the terminal's UCB. The account number is used so that a later report program reading both the SYSLOG.DAT file and RSX11.SYS can determine who was sitting on what terminal (very useful if one wishes ever to do charge back accounting). The account number (and session ID) are inserted into the account file using ACNT and implementing one of the conditionals existing for RSX11M-PLUS. New account file offsets must also be defined in ACTFIL.MAC. Just prior to inserting the account number into U.ACN, HELLO saves the value stored there. The value stored there is the number of illegal or unsuccessful login attempts at that terminal since the last logoff. A very security conscious system might be interested in this number, and it was essentially free. Each time HELLO fails to log on a terminal this number is incremented.

Once U.ACN is inserted, all the counters in the user's UCB are reset to zero, and a data packet sent off to LOG... containing the logon time the account number, the terminal number, the terminal type (TT: or VT:), the number of unsuccessful login attempts, and the login UIC. It must be noted that a data packet is only sent to LOG... if the proper bit is set in $ACMSK. In addition, the account file is marked with the terminal flag to show which account is on which terminal (both for TT: and VT:). This enables BYE to later find out who is logging off.

If the user still wishes to use the HELLO modifications without doing the system generation required for system accounting, all the KMS features except system accounting may be obtained by task-building HELLO with a global definition for $ACMSK. KMSGEN provides information on how to select this option.

BYE

At logoff BYE finds the account entry in RSX11.SYS based on the terminal number logging off and its terminal type. It then updates the account file with the terminal connect time in minutes, and the CPU time in seconds. If the proper bit is set in $ACMSK, it then sends off a data packet to LOG... consisting of the time, the account number, the terminal ID, the logoff uic, the CPU time used, and the number of QIO's issued. If user QIO accounting is not enabled, this last value will be zero. Whether or not it is

enabled can be determined from other SYSLOG.DAT records. Finally, the UCB counters are reset to zero, the terminal logged off and if so requested by a bit in the user's terminal privilege mask word, UPD is requested to update the user's disk block usage. Right before BYE exits it prints out a summary of the user's connect time, CPU time, and the number of tasks run. BYE may also be assembled on a system not wanting system accounting by means of a global definition for $ACMSK.

ACCLOG

ACCLOG provides a readout of the system account file RSX11.SYS. It provides a quick way of viewing the current total logon time, the total CPU time, the account number, the total files in use and the disk blocks in use. It additionally displays the percent of CPU utilization as a percent of the total logon time. The interpretation of this number will vary depending on the number of terminals present and the average number logged on at any one time. The task RESET is used to reset the cumulative time totals at some regular interval so as to start a new accounting period.

UPD

UPD scans through the user's SY: and login uic, tallies the number of files used and the number of disk blocks used, updates RSX11.SYS with this information and then if the proper bit is set in $ACMSK, sends off a packet to LOG... containing the disk block usage statistics.

TSKLOG

TSKLOG uses a lot of the code originally included as part of Greg Basset's ACCLOG. It however, only controls part of the exec intercept code, namely that code involved with catching the start and exit of a task, and in catching each context switch to see if an accountable task is present. Since a linked list of task account blocks must be scanned each context switch, this option can impose a lot of overhead if the list of tasks is long. For this reason, it should be considered primarily as a performance measurement tool.

TSKLOG is currently not as intelligent as SYSLOG. If it is aborted, the system will probably still keep running, but the pool it used for the task accounting blocks and its code node will not be reclaimed by running it again. Only rebooting will recover the lost pool. Hence it is very good practice not to abort TSKLOG. Also note that 10. words of pool are used up for each task entered to be accounted for. This could add up very rapidly if a lot of tasks were entered. Again I emphasize that TSKLOG will degrade system performance if there is a lot of QIO activity.

TSKLOG makes use of the information provided by SYSLOG, and requires SYSLOG to be active to work. However if SYSLOG were aborted TSKLOG would continue merrily on its way, using up CPU time, measuring nothing, but not crashing. However, it is considered very bad practice to remove a task which TSKLOG has been told to account for. When TSKLOG is given the list of tasks, it gets the TCB address for each task. All kinds of things might happen if an installed task were removed and reinstalled with a new TCB address.

The ambitious user will note that TSKLOG implicitly provides hooks for computing the amount of memory used by a task (excluding regions) at each context switch. If one did not worry about overhead, it would be simple to calculate the kilocore-tics each user was using at each context switch. Unfortunately, it would mean inserting the context switching code into SYSLOG and computing kilocore-tics every context switch. It could be done, but I doubt its usefulness.

The ambitious user should also note that while TSKLOG is running two useful side affects are present. A two word tic counter is kept and is readable by any privileged task ($TIME), also the address of the current task accounting block is at $TKBLK. The existence of a two word absolute time counter(if only $ABTIM were two words but that is another exec mod) allows privileged tasks to time critical routines without the overhead of the GTIM$ directive. By storing the current task acounting packet in $TKBLK additional exec intercept code may be implemented substantially increasing those parameters tasks are accounted for.

TABLE II

FORMATS OF DATA PACKETS SENT TO LOG...

	WORD 1											WORD 13
	RECTYP											
START	1	HR	MIN	SEC	YR	MO	DAY	TIC	TPS	DELTA	*	*	$ACMSK
STOP	-1	HR	MIN	SEC	YR	MO	DAY	TIC	TPS	DELTA	TKNM1	TKNM2	$ACMSK
GLB	2	HR	MIN	SEC	NULL1	NULL2	SHFCT	CKPCT	LDRCT	QICT1	QICT2	RUNCT	*
MEMORY	3	HR	MIN	SEC	NTSKS	TKMEM	NCKP	CKMEM	TPOOL	LGFRG	NFRAG	NTSNT	*
DELTA	4	HR	MIN	SEC	YR	MO	DAY	TIC	TPS	DELTA	TKNM1	TKNM2	$ACMSK
HELLO	5	HR	MIN	SEC	TRMID	ACNT	UIC	NILEG	*	*	*	*	*
BYE	6	HR	MIN	SEC	TRMID	ACNT	UIC	CPU1	ATCNT	QICT1	QICT2	RUNCT	INSCNT
UPD	7	HR	MIN	SEC	ACNT	SYSDV(2WDS)	NFILE	NBLKS	ABLKS	*	*	*	*
CO: PKT	8	HR	MIN	SEC	0	0	0	CPU1	CPU2	QICT1	QICT2	RUNCT	*
TSK PKT	9	HR	MIN	SEC	ACNT	CLK1	CLK2	CPU1	CPU2	QICT1	QICT2	*	*
SPL PKT	10	HR	MIN	SEC	ACNT	TOTPG	COPYS	*	*	*	TKNM1	TKNM2	*

RECTYP -1 - 99 RESERVED
100- USER USABLE

SYMBOL LEGEND FOR TABLE II

```
HR       - HOUR OF DAY
MIN      - MINUTE OF HOUR
SEC      - SECOND OF MINUTE
YR       - YEAR OF UNIVERSE
MO       - MONTH OF YEAR
DAY      - DAY OF MONTH
TIC      - TIC OF SECOND
TPS      - NUMBER OF TICS/SEC OF CLOCK

$ACMSK   - SYSTEM ACCOUNT FEATURE MASK
ABLKS    - MAXIMUM BLOCKS ALLOWED ON SYSTEM DEVICE
ACNT     - USER ACCOUNT NUMBER
CKMEM    - TOTAL MEMORY REQUIRED FOR ALL CHECKPOINTED TASKS
CKPCT    - NUMBER OF CHECKPOINT REQUESTS
CLK1     - WALL CLOCK RUN TIME(LOVAL)
CLK2     -    "    "    "    " (HIVAL)
COPYS    - NUMBER OF COPIES PRINTED IN JOB
CPU1     - CPU TIME USED(LOVAL)
CPU2     -  "    "   "  (HIVAL)
DELTA    - SAMPLE INTERVAL IN MINUTES
LDRCT    - NUMBER OF LOADER REQUESTS
LGFRG    - SIZE OF LARGEST POOL FRAGMENT
NBLKS    - TOTAL NUMBER OF DISKS BLOCKS IN USE AT LOGOFF
NCKP     - NUMBER OF TASKS CHECKPOINTED
NFILE    - NUMBER OF USER FILES AT LOGOFF
NFRAG    - NUMBER OF POOL FRAGMENTS
NTSKS    - NUMBER OF TASKS IN CORE
NTSNT    - NUMBER OF TIMES SYSLOG DID NOT SEND DATA BECAUSE POOL LOW
NILEG    - NUMBER OF ILLEGAL LOGIN ATTEMPTS
NULL1    - TIME IN IDLE LOOP(TICS) - LOVAL
NULL2    -   "  "   "    "    "    - HIVAL
QIOCT1   - NUMBER OF QIO'S ISSUED(LOVAL)
QIOCT2   -    "   "   "      "   (HIVAL)
RUNCT    - NUMBER OF RUN REQUESTS
SHFCT    - NUMBER OF SHUFFLER REQUESTS
SYSDV    - USER'S SYSTEM DEVICE(FORMAT TBD)
TKMEM    - AMOUNT OF FREE CORE IN USE BY TASKS
TKNM1    - SENDER/REQUESTER TASK NAME(RAD50)-FIRST PART
TKNM2    -   "      "      "   "   "    -SECOND PART
TOTPG    - TOTAL NUMBER OF PAGES PRINTED IN JOB
TPOOL    - TOTAL SYSTEM POOL
TRMID    - TERMINAL ID(HI BYTE-'T' OR 'V', LO BYTE-UNIT NUMBER)
UIC      - USER'S LOGIN UIC
  *      - RESERVED FOR FUTURE USE
```

APPENDIX I

MEASUREMENTS OF SYSTEM PERFORMANCE WITH AND WITHOUT SYSLOG RUNNING

One of the first things to consider when considering implementing an accounting package is how much system accounting will degrade system throughput. Another way of looking at the question, is to ask how much of the system's total resources is consumed by accounting. Earlier the ammount of pool used by SYSLOG was presented. In this appendix, the results of a number of timing tests are presented. These tests were run on a PDP 11/45 with 124 K words of memory. The system has an ACT CACHE/45 cache memory. The operating system has essentially all SYSGEN options except Connect-To-Interrupt. All tests were run with no other system activity.

Two types of tests were run; CPU bound tests and I/O bound tests. The CPU bound test consists of two virtually identical F4P tasks with the sole exception of the first task requesting the second task to execute. A complicated expression is evaluated a large number of times. The computation time is clocked using the Fortran 'SECNDS(X)' function. Two tasks were chosen to simultaneously execute so that the timing tests would more accurately reflect the use of our system. The tasks were timed with and without SYSLOG running. SYSLOG was activated with the ACC /START/CPU command (only CPU time accounting activated). Since CPU time is updated every clock tic, this was felt to be the potentially single biggest overhead. Under these conditions the test tasks took about .6% longer to execute.

The QIO bound test was almost identical except that the expression evaluation was replaced by a WRITE to NL:. The NULL device was chosen to simulate worst case overhead. Since each QIO to NL: is immediately returned with almost no processing, this test closely approximates the additional overhead incurred in QIO processing. It does not, however, reflect the additional compute time an average task will incur by including QIO accounting, since the actual number of QIO's a task will issue per minute is generally less than 10,000 (the number of QIO's to NL:), and for most devices there is a substantial latency caused by the device response time.

For this case if both CPU accounting (the default) and QIO accounting was active the task test execution time would be 1% greater with accounting active. If user by user QIO accounting was activated the execution time would be 1.6% greater. As a sanity check the CPU bound test was rerun with QIO accounting active. The time tests were essentially identical to the previous tests with just CPU time accounting active.

In summary, running this accounting package will result in .6% or somewhat greater total system overhead and degradation of throughput. QIO throughput to NL: will be decreased by 1 to 1.6%

if QIO accounting is active. Throughput to devices with more complicated drivers will be affected substantially less. As a touchstone for measuring overhead, RMDEMO running at 1/sec and outputting to a 9600 baud VT100 uses about 5% of the system's CPU resource (not to mention the memory). Systems which can tolerate the overhead of running RMDEMO can probably use SYSLOG without too much difficulty. Systems dedicated primarily to real time acquisition and process control for which timing and overhead are critical should carefully consider whether the benefits of running system accounting warrent the overhead.