To: Distribution

From: R.H. Morrison & D.A. Kayden

Date: April 11, 1975

Subject: GCOS Simulator Restart at the Activity Level

1.0 Introduction.

This project fulfills the requirement of SF-7440 (730010) TASK E. The requirement is to provide the ability under Multics to restart all GCOS jobs at the activity which was interrupted by system failure.

The minimum functional specifications for the Simulator activity restart are assumed to be the functions performed by real GCOS under SR 1G in batch mode. In the Multics GCOS Environment this is equivalent to GCOS jobs submitted via the GCOS Daemon or by absentee. The functions should also be available to the interactive user since they will be controlled and implemented within the Simulator.

2.0 Real GCOS: Restart Functions, Control and Implementation

2.1 Functions and Control

Restart refers to the automatic re-starting of jobs that were in execution at the time of a system interruption. Jobs can be restarted either at the beginning of the activity that was in execution or at the beginning of the job. (Jobs can also be restarted from a program checkpoint; however, this capability is outside the scope of the this project). Restart is conditional upon successful system recovery from the system interruption by a warm boot. If it is necessary to fall back to a cold boot, restart is not attempted.

Primary control for the restart functions are the options REST/NREST and JREST/NJREST on the activity defining control cards in the GCOS job deck. REST/NREST specifies whether or not to attempt restart at the beginning of this activity if a system interruption occurs during the execution or termination of this activity. Similarly, JREST/NJREST specifies whether or not to attempt restart at the beginning of the job after a system interruption, or to attempt job restart if the specified activity restart cannot be attempted. Thus the programmer, based on his knowledge of the job, makes the principal determination on an activity by activity basis of whether or not the job can be successfully restarted.

Secondary controls on restart consist of various checks made by GCOS system programs (mainly the Peripheral Allocator) that no conditions exist that prevent a successful restart. For example, restart is not attempted if a peripheral or a temporary file saved from a preceding activity was released in the current activity prior to the system interruption.

If a tape is required and for some reason it cannot be mounted, the operator can terminate the job.

2.2 Implementation

The job flow through the Simulator is much the same as the job flow through real GCOS and therefore the implementation of restart in real GCOS serves as a model for the implementation of restart in the Simulator. We will briefly trace the real GCOS job flow relevant to restart to point out how restart is implemented in it, and to note the extensions needed to the Simulator for a similar implementation. This section, then, is the background for the proposed Simulator implementation described in section 4.

In the Simulator, the equivalent of these functions are performed in gcos_run_activity_.

2.2.4 System Recovery

GCOS maintains a (hard core) System Recovery Table to save selected queue pointers, queue bodies, and other important tables over a warm boot, and to save restart data for the privileged system programs. At the end of the start-up portion of the warm boot, control is given to POPM for rollcall and initialization. If the operator requests a restart, POPM divides the jobs that were in the system at disaster time into three groups:

1. Those that can potentially be restarted.
2. Those that had finished execution, but still have output in the system.
3. Those that were interrupted during loading and must be reloaded.

Jobs that can be restarted are re-entered in the input queue for either the Peripheral Allocator or the Core Allocator. The job status in the Peripheral Allocator Job Control Table is set to "restart".

Finally, when the Peripheral Allocator is next enabled, it processes each queue entry and makes the final determination of whether or not the job can be restarted.

3.0 Simulator: Restart Functions and Control

3.1 Functions:

Restart functions in the Simulator, as far as the GCOS job is concerned, will be the same as in the real GCOS. An absentee job with the restart option will be automatically restarted in the new process at the beginning of the activity in execution at the time of system interruption, or at the beginning of the job, as specified by the options on the activity definition cards of the GCOS job deck. The Simulator saves the job deck and reads it to define each new activity in the job flow as does GCOS so this control data is available.

The control data not available to the Simulator as it is in real GCOS is a flag that, in effect, would inform the simulator that a system interrupt had occurred, and it will be necessary to

restart the job. The Simulator, in lieu of such a flag, will have to maintain a record of progress of the job through the job activities, and at each invocation will have to search for such a record. If the record exists for a particular job, the job is being restarted; if no record exists, the job is a new job.

This record can be kept along with other necessary activity initiation data in some segment that will be saved through a system interruption. In concept, this segment is equivalent to the GCOS Initiation/Restart Record.

Restart will not be automatic for jobs when the Simulator has been invoked interactively from a terminal. However, if gcos is invoked for the same job name in a new process after a system interruption, the restart mechanism will operate as it would when gcos is re-invoked by the absentee processor after a system interruption. If the job is not to be restarted, the saved data segment would need to be deleted, or a control argument to the Simulator used to specify no restart, when the Simulator is invoked in the new process.

In addition to the scan for a saved data segment, there will need to be an internal restart procedure to reset conditions to those existing at the start of the execution of the interrupted activity. There are conditions that may have occurred which make restart impossible. These are in general the same as those in real GCOS. These conditions and the activity restart options will be checked before restart is actually attempted.

4.0 Proposed Implementation of Restart

The most straightforward implementation of restart in the Simulator is to implement the different restart functions in the job flow as they are sequenced in real GCOS. The extensions to the Simulator, then, would mainly be in the following procedures.

4.1 gcos

This is the entry procedure to the Simulator which processes all control arguments to the Simulator. The extensions would be:

1. Recognize and process a control argument "-nosave". The default would be to restart jobs or activities. A control flag would be set to direct later procedures to create a save_segment, GCOS system file segments, and temporary file segments saved across activities in a directory that will be saved through a system interruption (currently these files, by

default, are created in the process directory).

2. Scan the directory containing the save_segment to see if one exists for this GCOS job. If one exists, this invocation of GCOS is a restart and process control should be sent to the restart procedure.

4.2 gcos_gein_

This procedure performs the GCOS functions of System Input, Peripheral Allocation, and Core Allocation. The extensions would be:

1. Create the segments to be saved in an appropriate directory other than the process directory if the restart flag is true. This directory must be standarized so that GCOS will know where to create the segments and therefore where to look for them (especially the save_segment) during restart. Three main standardization possibilities are:

    1. A directory associated with the person.project ID.

    2. A directory associated with the GCOS Environment. Such a directory is >ddd>GCOS>pool_dir assuming that the GCOS Daemon will be installed if the Simulator is installed (if not, create this directory anyway).

    3. A directory specified by another argument to the Simulator (which will therefore be known at each invocation).

    The preferred directory is the working directory at the time the Simulator was invoked.

2. Initialize the save_segment with job data.

3. Maintain a restart control word with the current activity number and flags for the activity control options of REST/NREST and JREST/NJREST. This word would also contain another field as described in the next section on activity restart. Note that this control word is equivalent to the GCOS first Status Table entry for the job, but in the Simulator it must be kept in the save_segment in lieu of the HCM.

4.3 Restart

gcos_gein_ could contain the restart procedure, although the restart procedure could also be an independent procedure. In either event, the restart procedure's functions are:

1. Recover the save_segment and examine the Restart Control word. If REST is true, reconstruct gcos_ext_stat_ as it was at the beginning of the interrupted activity and restart the activity. If NREST is true (i.e., REST is false) examine JREST/NJREST and either restart the job or abort it. The activity restart function defines the contents of save_segment as those variables in gcos_ext_stat_ and pathnames of GCOS system files and temporary files that must be restored instead of re-initialized.

   The Restart Control word in the save_segment will need to be updated during the execution of an activity if any events (such as a release of a saved temporary file) occur which make an activity restart impossible. Another situation with several alternatives arises when the activity has successfully completed execution, but a system interruption occurs in the activity termination (cleanup) phase. In principle, it should not be necessary to restart at the activity level or job level (and it may often be impossible); the problem becomes that of re-starting or continuing only the activity termination (and perhaps the SYSOUT activity). The functions in real GCOS in this case, are currently evolving, and the best thing to do for the moment may be to defer a decision temporarily. However, we can define a field in the Restart Control word to flag whether or not activity termination had started. We also note that the save_segment might need to be updated when termination begins.

   If the restart functions are successful in re-establishing the initial conditions, then control is returned to the normal job flow control.

4.4 gcos_run_activity_

Activity termination and cleanup is performed in this procedure. The extensions would be:

1. After the termination activities are completed and the job is ready to be returned to gcos_gein_ for the next activity, update the save_segment.

2. Update the Restart Control word when an activity enters termination. Possibly write an updated save_segment with the end of activity data.

4.5 gcos_mme_rels_

This procedure releases peripherals and mass storage files. It must be extended to update the Restart Control Word in the save_segment when files essential to activity restart are released.

4.6 gcos_ext_stat_

As the extensions to the major procedures are made, it is expected that the changes will ripple into some other procedures, either from necessity or convenience. One such change is to re-arrange gcos_ext_stat_ to facilitate saving and restoring the data in it needed for restart.