Massachusetts Institute of Technology
Charles Stark Draper Laboratory
Cambridge, Massachusetts

Luminary Memo #140

TO: Distribution
FROM: Allan Klumpp
DATE: March 4, 1970
SUBJECT: A Collection of the Known Manifestations of Time Loss in Luminary Revision 131 and LM131 revision 001 - Suggested Work-Around Procedures.

INTRODUCTION

Prior to Apollo 11, we felt that a substantial amount of unscheduled counter activity (PIPA's and CDU's) was extremely unlikely. The Apollo 11 incident produced counter activity equivalent to about a 13% demand on the computer duty cycle, or 13% time loss (TLOSS). Programming measures were taken to prevent the Apollo 11 incident from recurring on Apollo 12 and subsequent flights, and telemetry data from Apollo 12 indicates counter activity corresponding to a TLOSS probably not in excess of 2%, though there is an uncertainty of about 2% on top of the 2% estimate. Therefore, in Apollo 13 and subsequent flights, we again feel that TLOSS in excess of 4% is extremely unlikely, although we concede that we may not know of every possible mechanism which may produce an unanticipated demand on the computation duty cycle.

This memo presents some of the problems which are encountered if the TLOSS vastly exceeds the amount which we consider likely. Many cases have been run in programs P63 and P64 which demonstrate flawless performance up to just short of 8% TLOSS. Innumerable tests have been run at this laboratory, at Grumman, and on the LMS, using every imaginable combination of Auto-P66 and Attitude-Hold-P66, exercising every imaginable combination of ROD inputs and attitude maneuvering, and provided the appropriate work-arounds are used prior to initiating P66, I am not aware of any flaws in the performance of P66. The remainder of this memo describes the performance of programs Luminary 131 and LM131 when they are deliberately forced to misbehave by being subjected to TLOSS in excess of 8% in Program P64 and up to 12% in P66.

When subjected to TLOSS in excess of 8%, no known work-around (a procedure which permits the mission to be continued normally) will avoid the adverse consequences in P64 in either Luminary 131 or LM131, and the performance of Luminary 131 and LM131 is identical. At the end of this memo, work-arounds are suggested which will avoid adverse consequences in P66 of TLOSS in P64 in both Luminary 131 and LM131. If the TLOSS exceeds a limit to be determined but which we suspect lies around 10% in Luminary 131, violent throttle behaviour may occur in P66 without warning, and there is no known acceptable work-around. If the TLOSS exceeds a limit to be determined but which we know lies in excess of 12% in LM131, there will be a gradual loss of stability of the throttle and attitude, and the astronauts will be warned of the impending situation by alarm 01466. From every consideration, LM131 is as safe or safer than Luminary 131, and it is definitely safer if one of the proposed work-arounds is followed.

The reader who is not interested in the internal workings of the programs should now skip to the Work-Around Procedures.

The description of the TLOSS manifestations published here are not complete, and I believe they must contain mistakes. Substantial portions of the descriptions had to be constructed without benefit of verification by simulation. Nonetheless, I feel it is important to publish at this point so others can begin to think about these problems without further delay. I respectfully challenge the conscientious student of these programs to find the mistakes and omissions in this memo, and I will be happy to incorporate suggested corrections in a future revision if one is written.

This collection of manifestations of TLOSS is incomplete because:

1. It covers only the manifestations in the descent programs; the ascent, ascent abort, and P40's are not considered in this memo.

2. Although LM131 has been exhaustively tested for TLOSS manifestations, new manifestations have appeared in the last few days; it seems unlikely that further testing would reveal nothing.

3. Luminary revision 131 has been tested for TLOSS, but not to the same degree as LM131. The behaviour of Luminary 131 has been examined externally, including throttle and attitude performance. The behaviour of LM131 has been examined externally and internally including tracking down the source of every glitch and tracing the offending coding. The problems listed for Luminary 131 include only those we have been able to think of without benefit of the simulator; those listed for LM131 have been discovered both by mental reasoning and by simulation, and most have been observed on the simulator. Further testing of Luminary 131 would undoubtedly reveal more additional TLOSS problems than further testing of LM131.

When we decided the TLOSS problems had to be fixed for Auto P66 in Apollo 13, there were two candidate fixes. The first was to fix the Servicer program, which would avoid multiple servicer jobs in all powered flight programs. The second was to fix the P66 program such as to affect no other program. We decided to fix P66 alone because to fix the Servicer program would mean that all powered flight programs would have to be exhaustively re-qualified which was not possible in time for Apollo 13. Also, the level of TLOSS which would cause problems in programs other than P66 was sufficiently high that its occurrence was judged to be less likely than the introduction of some other anomaly in these other powered flight programs as a consequence of the fix. Therefore, the other powered flight programs, including P64, behave identically in Luminary 131 and LM131.

TLOSS MANIFESTATIONS

The TLOSS manifestations will be described in the order P64, P65, and P66.

1. Delayed P64 Attitude Commands

These problems have equal probability of occurrence in Luminary 131 and LM131. They have been observed in simulations. We know no acceptable work-around procedures which avoid the P64 problems, although quick astronaut intervention could reduce the consequences.

If there is TLOSS in excess of 8%, some servicer job will fail to finish before the subsequent servicer job is requested. We assume the most likely case that the servicer job is suspended in the FINDCDUW Routine. Because of the way the Executive program handles job priorities, the new servicer job will generally start without allowing the current job to finish, leaving the current job lurking in the background waiting for some free computer time to finish and issue its commands. Depending upon the TLOSS level and the virtually random computation load of the various programs running, one or both of two possible adverse consequences may occur. If subsequent servicer jobs also fail to finish, the supply of VAC areas will soon be exhausted, a 31201 alarm will be issued and all but the final unfinished servicer job will be discarded. A job request will be made by the Restart routine to finish the final servicer job starting at a restart point in the middle of the throttle routine. Generally, this final servicer job will not be restarted immediately, and if it is restarted in P64, the consequences are thought to be benign, but this has not been demonstrated; however, if this final servicer job is restarted after P66 is initiated, the consequences can be disastrous, as is described in the P66 problems. If the number of unfinished servicer jobs remains insufficient to produce a 31201 alarm, then this generally means that servicer jobs are only occasionally being suspended and occasionally some free time occurs which permits the suspended jobs to finish. Therefore, the suspended jobs, when they finish, issue attitude commands which are valid at a time considerably prior to issuance. There is a critical range of TLOSS, about 8% to $8\frac{1}{2}$%, in which servicer jobs may accumulate for the entire duration of P64 without ever issuing a 31201 alarm and clearing old jobs out. If such very old jobs finally finish when

in the automatic attitude mode, they can return the spacecraft to the attitude which was needed at the time the job was started. Thus, any time in P64 (but most dangerously near the end of P64), with this critical amount of TLOSS, without any forewarning, the LM could suddenly pitch back to around 45^{o}. If the astronaut could recognize the erroneous pitch rate and switch to Attitude Hold in the three or four seconds it would take to reach the final attitude, he could prevent the full transient. In any case, he can return the spacecraft to the proper attitude. Of course, the automatic system will return the spacecraft to the proper attitude when the old P64 commands are exhausted or when they cannot be issued because of insufficient time.

2. Attitude Oscillations in P65 Due to Unfinished P64 Jobs When P65 is Started

These problems can occur only in Luminary Revision 131. No simulation of the problems has been made so there may be other undiscovered problems. The work-arounds should avoid the problems.

The computations in P65 are substantially less than those in P64. Therefore, unfinished P64 jobs will very likely find time to finish when P65 is started. These will probably finish at the rate of one per two seconds, each issuing a command to return to some pitch attitude appropriate to P64. Since up to four P64 jobs can be in limbo, the oscillations could continue for up to eight seconds (or longer if during some P65 servicer cycles no P64 job finishes). Since the P64 and P65 commands will alternate, the spacecraft will pitch rapidly back and forth with a two second period. The P64 commands will be issued within 300 milliseconds following the P65 commands, and will remain in command of the autopilot for the remainder of the two second period. Therefore, the P64 commands will predominate and the spacecraft will return largely to the P64 attitude. The P64 commands will necessarily be issued in the reverse of the order in which they were generated and therefore the spacecraft will pitch back further and further until the P64 jobs are exhausted. At this time the normal P65 commands will take over and, if the spacecraft motion can be nulled in time, a normal landing should be possible.

The return addresses for the throttle and for FINDCDUW are the same in P64 as in P65, and therefore there should be no random branching as a consequence of unfinished P64 jobs.

3. Alarms and Violent Throttle Activity in P66 Due To Unfinished P64 Jobs When P66 is Started in Luminary 131

No simulations of these problems have been made so there may be other undiscovered problems. The work-arounds should avoid the problems.

P66 normally requires one VAC area more than P64 requires. If there are no spare VAC areas in P64, then starting P66 may produce a 31201 alarm because the first priority 22 ROD job may start before any P64 job has finished and the second P66 servicer is normally requested before the first priority 22 ROD job is finished.

An additional possibility is that the completing P64 jobs will increase the computation load to the point where a P66 job will be suspended in the ROD equations and a subsequent P66 job will pass thru the same ROD equations destroying the erasables and causing the same wild throttle behaviour as described in the following item. There are two ways in which a completing P64 job can increase the computation load in P66. First, the P64 job may reach an area of coding in which it provides no opportunity for another job to start, thus delaying the start of a subsequently requested P66 servicer job. Alternatively, and far less likely, a low address VAC area may become free after the start of P66 allowing the subsequent P66 servicer job to be assigned this VAC area. This will cause all remaining P64 jobs with a higher VAC area to finish before the P66 servicer job request is honored.

4. Violent Throttle Activity in Luminary 131 P66 Due To Excessive TLOSS

This problem has not been observed in Luminary 131, but we believe it can occur if there is sufficient TLOSS because it has been observed at a much lower level of TLOSS in LUM131 Revision 009. This problem was the chief reason why LUM131 was discarded and LM131 was manufactured. We believe it cannot occur in LM131. The only known work-around is to permanently turn off the landing radar, and that has been judged unacceptable because of degradation of the state vector, and the consequent introduction of errors into the displayed velocity and altitude.

If the TLOSS is above a certain limit to be determined, (probably around 10%) in P66, then one servicer job will fail to finish before the next servicer job is requested. The new servicer job will generally be started and the old servicer job will be suspended probably in the ROD equations. Before the new servicer job reaches the ROD equations, a priority 22 ROD job will be started and, because it uses erasables already written into by the suspended job, it will very likely produce erroneous throttle commands. The computation load may not be the same for the new servicer job (there are many possible reasons, the most likely is a radar dropout) and both the new servicer job and the suspended job may proceed to completion, both issuing erroneous throttle commands. In simulations of LUM131we have observed the ROD equations computing erroneous thrust acceleration ranging between less than zero (physically impossible) to POSMAX (corresponds to around 60 times the thrust capability of the engine). These results would cause hard-up and hard-down throttle commands, but anything between is also possible.

5. Alarms and Attitude Oscillations in P66 Due to Unfinished P64 Jobs When P66 is Started in LM131

If there are unfinished P64 jobs when P66 is initiated, then, on the first pass in P66, the most recent P64 job will probably resume. If the attitude control mode is auto, the finishing P64 job will issue commands to return to the P64 attitude. In addition, the finishing P64 job will execute the ROD equations which will in turn request an additional ROD job. The consequence of all this is that four ROD jobs will be performed in the first two seconds of P66. There are such a variety of event sequences which can follow that it is hopeless to analyze all of them, let alone verify the analyses by simulation. Therefore, I must be content to describe the most important possible events and assume the problem will be avoided by using a work-around as described at the conclusion of this memo. Possible events are:

1. The servicer is put so far behind schedule that READACCS occurs before COPYCYCL. This causes the loss of two seconds worth of delta V in both the horizontal and vertical channels, and two seconds worth of gravity in the horizontal. The vertical velocity error will generally be small because the loss of gravity compensates for the loss of delta V, but the horizontal error can be significant if the pitch angle is substantial. Simulations have produced these results.

2. A 31201 alarm may occur. Simulations have produced this result.

3. The servicer which was interrupted by the COPYCYCL will be suspended, permitting the new servicer to go thru first. This means that erasables may be over-written, erroneous answers may be produced, and until we can prove otherwise, we must assume return addresses may be destroyed which could cause random branching. Simulations have not produced these results.

4. The restart group of the interrupted servicer will be turned off when the new servicer exits to guidance. Therefore, if a restart occurs, the old servicer will not be restarted. Simulations have produced this result.

5. P64 attitude commands may be released substantially after the start of P66, returning the spacecraft to the P64 attitude. Simulations have produced similar results, and the possibility of this is being verified at the present time.

6. Random Branching in P66 Due to a Restart in P64

This problem has about an equal apriori probability of occurrence on Luminary 131 and LM131. It has been observed in simulations. The work-arounds should avoid the problem.

If there is a restart in conjunction with time loss in excess of 8% in program 64, then it is very likely that two servicer jobs will be requested after the restart. The first servicer job will be requested by READACCS in restart group 5 to start at the beginning of the servicer program (LM131 page 859, location 33,2206) and the second servicer job will be requested in restart group 3 to start in the middle of the throttle program (LM131 page 790, location 31,2242). The first job (in restart group 5) is a new servicer job called by READACCS and the second job (in restart group 3) is the servicer job which was in progress at the time the restart occurred. Both jobs are requested at priority 20, and therefore, according to Phyllis Rye, the job called by READACCS which will be assigned a higher VAC area, will always start first. Thus the new servicer job will always start first, and the interrupted servicer job will lurk in the background awaiting free time to continue. If the time loss persists, the interrupted servicer job may not be restarted before program 66 is entered. The first pass in P66 will call the throttle routine via BANKCALL and the throttle routine will return to P66 via the normal BANKCALL return called SWRETURN. Thus the return address stored by the throttle program in the register called RTNHOLD will be changed to point to SWRETURN and not to the P64 guidance equations. However, the return address used by SWRETURN stored in BUF2 and BUF2 + 1 is destroyed by SWRETURN, is again changed by the P66 program, and may be again altered by almost any subsequent routine. The first pass in P66 has only one computation of vertical commands rather than the two which occur on subsequent passes, and therefore there is more spare time this first P66 pass than there was in the previous P64 guidance. This is just what is needed to unleash the lurking P64 job, which is now restarted in the middle of the throttle equations and proceeds to return via SWRETURN. Since SWRETURN finds an invalid return address in BUF2 and BUF2 +1, it branches the restarted P64 job into an indeterminable location, with almost any imaginable consequence.

The only mechanism which has been found which can produce this problem is a restart in P64 in conjunction with time loss. This is generally recognizable by alarms 31201, 31202, or 31203. If one of the above alarms occurs in P64, or if a restart occurs by another means in P64 and there is a possible

time loss in excess of 8%, then a work-around must be initiated before entering P66. Because the alarm or restart could occur during the last few seconds of P64, too late to initiate the work-around, the work-around should be performed as a matter of course whether or not an earlier indication occurs.

Any subroutine which has a restart point within it and is called using either a BANKCALL or an IBNKCALL and returns via Q can produce random branching when restarted. Therefore, it is suggested that a search be made of current AGC programs to uncover all such conditions, and that programmers be warned to guard against this practice in the future.

P66 turns restart group 3 off at the start of each pass and therefore the restart point in the middle of the throttle routine is benign provided no unsatisfied job request to start at that restart point exists at the time P66 is entered.

7. Loss of Throttle and Attitude Stability in P66 in LM131

This problem has not yet been observed in simulations because testing has so far been done only to 12 percent which is insufficient TLOSS to produce the problem. All paths and branches of the coding have been tested by means of artifices.

If TLOSS is beyond a certain limit to be determined (in excess of 12%), every second P66 guidance pass will be omitted. With the proposed padload (TOOFEW = 3) this condition will repeatedly produce alarm 01466. The consequence of this infrequent servicing of the throttle and attitude should be a gradual divergence of the throttle and the pitch and roll attitude.

With non-zero TLOSS less than the above limit, P66 commands are occasionally omitted with no serious consequence. Below 5% no commands are ever omitted. At 5% P66 is omitted occasionally, e.g., once in a normal descent from 100 ft. At 8% P66 is omitted once every eight or ten seconds. At 12% P66 is omitted once every six seconds. The consequence of omitting P66 is that throttle and attitude rate commands are in effect longer than intended with the possibility of some attitude or altitude rate overshoot when P66 is omitted at the same time that either the attitude or altitude rate is changing.

8. Loss of Two Seconds Worth of ΔV in the Horizontal Channel And Incorrect Time Tag on Downlinked State Vector Due to Pressing the Abort Button During a Narrow Time Window When P66 is Backed Up With Excessive TLOSS in LM131

This problem was brought to my attention by Frank Gerth of TRW Houston. It has been discussed with MSC personnel at a time when the behavior was not fully understood. This description should correct prior information, and has been verified by digital simulation.

The problem occurs when there is a substantial amount of TLOSS in P66 (12% is sufficient) such that P66 is occasionally omitted, and the abort button is pressed immediately before the exit from the servicer program to the guidance equations on a pass in which P66 would be omitted. When the abort discrete is detected, an enema is done which will always move the restart protected servicer job into the lowest VAC area. Next, the abort is targeted which takes about 140 msec at 12% TLOSS. The servicer program is then restarted at the restart point preceding the point where it was interrupted when the abort discrete was detected. Because of the time consumed by the restart program and by the abort targeting, there is a chance the READACCS will occur before the old servicer job reaches COPYCYCL, and because the old servicer job was moved into the lowest VAC area, the new servicer request will always be honored first. Thus the new servicer job will begin, it will write over the temporary state vector registers used by the old servicer job, do COPYCYCL, and terminate the job by transferring to SERVEXIT. The old servicer job will then restart, redo COPYCYCL with the data replaced by the new servicer job, and also terminate at SERVEXIT.

Because the new servicer writes over the temporary registers of the old servicer, the net effect is that the PIPA data read by the first servicer is never used. In the vertical channel, this means that two seconds worth of thrust acceleration is ignored, but also two seconds worth of gravity is ignored. Consequently, the vertical velocity error is the difference between the two ignored quantities, at most about 15 fps based on the ROD throttle limiting, and very probably much less. Assuming the normal case of constant descent rate the error would be zero. However, in the horizontal (forward) channel, the error is the delta V of two seconds, which could be significant if the attitude were not erect at the time of pressing the abort button. Assuming an extreme case of ignoring 10 fps in the horizontal channel, the apoapsis of the abort orbit would be 7.5 nautical miles lower than intended.

An additional consequence of the above events is that on the first pass after the abort button is pressed, the time tag on the downlink will be incorrect, and if anyone tries to divide by the time difference between this tag and the one two seconds later, a divide overflow will result.

Although neither random branching nor fallacious computational results have ever been observed due to interchanging the order of completion of the servicer cycles, it remains to be proven that these consequences cannot occur.

WORK AROUND PROCEDURES

Adverse consequences of TLOSS in P64 in LM131 and Luminary 131 begin at about 8%, and we know of no acceptable work-around. Adverse consequences in P66 can be avoided in LM131 if TLOSS is limited to 12% and in Luminary 131 if TLOSS is limited to an unknown amount around 10%. TLOSS below these respective limits will be called limited TLOSS.

Fortunately, a single work-around will avoid all known adverse consequences in P65 and P66 of limited TLOSS in both Luminary 131 and LM131. All known adverse consequences are due to unfinished P64 jobs at the time P65 or P66 is entered. Therefore, any procedure which causes all P64 jobs to finish before P65 or P66 is entered, will avoid the known problems in P65 and P66. Suggested procedures follow:

1. At least ten seconds prior to entering P66, switch to Attitude Hold. This causes FINDCDUW to omit DAP commands so that the lurking P64 jobs can finish without producing any attitude transients. Since there can be up to four unfinished P64 jobs, and one will finish every two seconds, at least ten seconds of Attitude Hold is required.

2. Introduce a procedure which will eliminate at least 10% of the computation load. Possibilities:

   A. Switch the antenna position.

   B. Pull the radar breakers.

   C. Switch to the AGS.

3. Initiate P66 and wait until P66 appears on the DSKY.

4. Return to the normal computations.

Allan Klumpp

Approved by:

Gerald M. Levine