Eyles

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
INSTRUMENTATION LABORATORY
CAMBRIDGE, MASS. 02139

C. S. DRAPER
DIRECTOR

AG# 370-69
August 4, 1969

National Aeronautics and Space Administration
Manned Spacecraft Center
Houston, Texas 77058

Attention: Mr. W. Kelly
Project Office PP7
Guidance and Navigation

Mr. Christopher Kraft, Jr. FA

Through: NASA/RASPO at MIT/IL

Subject: Exegesis of the 1201 and 1202 Alarms Which Occurred During the Mission G Lunar Landing

Gentlemen:

Background

During the mission G lunar landing five LGC Executive overflow alarms occurred. They were with approximate times of occurrence,

1202 at PDI + 316 seconds
1202 at PDI + 356 seconds
1201 at PDI + 552 seconds
1202 at PDI + 578 seconds
1202 at PDI + 594 seconds

The first two alarms occurred in P63; the last three occurred in P64. A later section of this letter gives more detail on the other DSKY activity during landing.

These alarms arose because of a computer overload caused by a high frequency train of "counter increment requests" from the RR ECDUs. Each "counter increment request" requires one LGC central processor memory cycle. Consequently, the other computational tasks and jobs of the LGC are effectively slowed down. The slow-down was so bad at times that some repetitively scheduled jobs (like SERVICER, the navigation and guidance job which is done every two seconds) were still running when the time arrived for their succeeding cycle of computations to begin.

This phenomenon (requesting the EXECUTIVE program to start some particular job before the previous request of the same job has been completed) is a feature of the way the LGC guidance cycle is structured and the way sucessive repetitive jobs are scheduled. The execution of jobs may be slowed down by RR ECDU memory cycle robbery but the clock which schedules jobs like SERVICER and LRHJOB (the job which reads LR range) is scarcely affected. The clock (TIME3, the WAITLIST clock) we use to schedule these jobs ineluctably counts down to the time for the next repetition of a job to begin whether the previous repetition is complete or not. This is not the only way to structure our programs but it is the simplest and most natural way. There is, of course, danger in this structure, a danger well understood and, we thought, provided against. Some computations may never be finished if the basic cycle period is to short for the load on the central processor. Worse yet, the EXECUTIVE program may be deluged with requests for jobs - the new ones and the old still unfinished ones. The EXECUTIVE must supply a unshared set of erasables for each job's execution from the moment it is started until the moment it is completed. The EXECUTIVE has only so many sets of these erasables at its disposal. After it has allocated all its sets the next requestor of a job asks for the impossible. The EXECUTIVE responds to the impossible request by turning on the PROG light (program alarm), storing the 1201 or 1202 alarm code, and transferring control to the BAILOUT routine which tries to bail the program out of trouble by stopping all jobs and tasks and restarting only the absolutely required (in the programmer's limited fallible viewpoint) jobs and tasks. Thus, certain astronaut requested activity like extended verbs and monitor displays are preemptively terminated (he can call them back). This cleaning out of the temporarily dispensable can cure a temporary overload. A permanent overload is the very dangerous situation.

If there is a danger in this simple natural philosophy the danger can be reduced to an acceptable level. We try to do this by simulating all the known sources of computer load and then to be safe impose a safety margin. We simulate an unknown source or sources of memory cycle loss (we call it TLOSS) and insist that a certain TLOSS be tolerable (no 1201's, no 1202's). The margin has a price - it can be obtained by lengthening the navigation, guidance and display period and by trading off execution speed against storage in the coding technique. We insisted on a tolerable TLOSS of about 10% during landing. The coding and the guidance period were therefore massaged until 10% TLOSS was tolerable (with a monitor verb running). Unfortunately, the RR ECDU's stole about 15% (most of the time).

The balance of this letter gives a simple explanation of the EXECUTIVE program and the job structure during landing, the interesting hardware interface between the RR shaft and trunnion angle resolvers and the LGC when the RR switch is in AUTO TRACK or SLEW, and also what we are doing in LUMINARY 1B about this interface. Finally, it is not this specific problem which we thoroughly understand and have several ways of avoiding that really concerns us now. As you asked me in your office last Thursday, how can we prevent a similar presently unknown thing from happening? The last part of this letter explains some precautions we are taking and some disciplines we are invoking.

The Executive Program and How 1201 and 1202 Alarms Can Arise

The job scheduling and job supervising program in the LGC is called the EXECUTIVE. When the landing programs must schedule a job they call the EXECUTIVE program and give it the address of the job to be run and the priority for running it. Examples of jobs which are scheduled repetitively during landing are SERVICER, LRHJOB, and LRVJOB (the job which reads the velocity measurement from the LR). An example of a one shot job scheduled at HI-GATE and therefore called HIGATJOB, is the LR antenna re-positioning program.

People sometimes talk about jobs running "simultaneously". This cannot literally happen since there is only one central processor in the LGC and only one job's computations can be done at a time. However, jobs can be simultaneously waiting to be run, and these jobs can be waiting in various stages of completion.

How does the EXECUTIVE decide which one of a group of waiting jobs actually gets control of the central processor? That is the purpose of the job priority rating which the caller of the EXECUTIVE program must specify along with the address of the job. If a priority 20 job is running and a priority 32 job request is made to the EXECUTIVE, the EXECUTIVE will temporarily suspend (put to sleep) the priority 20 job (at a convenient point of suspension of the suspended job) and initiate the priority 32 job. Evidently, there must be an inviolate unshared set of erasable registers for each suspended (sleeping) job where its intermediate results are stored until it can be re-awakened to finish its computations. Even the simplest job requires a certain minimum set of erasables - this is called a coreset (twelve erasable registers per coreset). More sophisticated programming jobs, which perform matrix and vector computations and therefore use the INTERPRETER, need an additional set of erasables called a VAC area (43 erasable memory cells per vac area). (VAC stands for vector accumulation).

Therefore, every job needs a coreset and some jobs need a VAC area. Furthermore, every job must be allocated its required set of unshared erasable registers from the time it begins its computations until the time its concludes its computations and does an ENDOFJOB - even if it is temporarily suspended and waiting to be awakened in order to finish its computations. Jobs which use coreset storage only are called NOVAC jobs; those which require a "vector accumulator area" also, are called VAC jobs. A table of the jobs which may run during landing, their priority, and their type (VAC or NOVAC) is given below.

Obviously, the EXECUTIVE program cannot respond to an arbitrarily large number of requests to schedule jobs. For that would require an arbitrarily large number of erasable registers. The LUMINARY 1B EXECUTIVE program has enough erasable memory to provide 8 coresets and 5 VAC areas. (COLOSSUS only has 7 coresets.) If the EXECUTIVE is requested to schedule more than eight jobs altogether or more than five VAC type jobs it cannot comply. There are several actions which the EXECUTIVE could take in this overload condition. One possible action would be for the EXECUTIVE to terminate abruptly the lowest priority waiting job and pre-empt its erasables for the newly requested job (or ignoring the request for a new job if its priority is lower than the priorities of the already waiting job). Of course, in this kind of scheme, a NOVAC job could not be preemptively "de-scheduled" to accomodate a VAC job. This philosophy is not really realistic. The EXECUTIVE program then might be making mission affecting decisions. It could conceivably ignore the execution of a low-priority but nonetheless crucial job. The EXECUTIVE, then, should not make a decision to abort or kill a job.

How, then, can an EXECUTIVE overload reasonably be handled? To understand how we handle EXECUTIVE overloads we have to understand how the computer handles software or hardware restarts. A restart causes, among other things, a cleaning out of the EXECUTIVE program's queue (and also the WAITLIST program's queue). The RESTART program then consults "phase tables" to see what jobs and tasks should be restarted. The phase tables are updated <u>by the mission programs</u> as the computations proceed. The phase tables do not necessarily start a given job over at its beginning; they are more likely to restart the cleaned-out program at some point conveniently prior to the point at which the restart interruption occurred. If we think of the programming job as being composed of blocks or chunks of functional coding, operations and computations, then we see that logical points for a restart to begin (that is, a logical place to insert a phase change) would be at the beginning of the logical block or chunk which the computer is currently executing. Note that the advantages of causing a software restart following an EXECUTIVE overload are twofold.

1) Old jobs which were not finished before their successors were requested are cleaned-out; i. e., the backlog, the logjam, is

cleaned up. Because SERVICER does its navigation first in its cycle and then its guidance computations and then its DAP output computations, the state vector maintenance is not likely to degrade but the DAP outputs may be old if the SERVICERs begin to accumulate. The philosophy of cleaning out the old unfinished SERVICERs and restarting the current one is therefore a good idea.

2) When a restart is executed only the programs, jobs, and tasks which the programmers provided for in the "phase tables" are restarted. (This, of course, is a two-edged sword for the programmer may forget to provide for a crucial job or task.) Temporarily dispensable items like astronaut requested monitor verbs or extended verbs are not automatically restarted. Thus, there is likely to be an immediate reduction in the computer load. The astronaut may subsequently re-request his preemptively terminated monitor or extended verb.

Reference to the Table of Jobs and Priorities below shows that there are only 8 jobs which can run during lunar landing and that only three of these are jobs which use a VAC area. Furthermore, some of these jobs, like HIGATJOB and LRHJOB (or LRVJOB) cannot run simultaneously. Therefore the 1201 alarm (not enough VAC areas available) and the 1202 alarms (not enough coresets available) must have occurred because of multiple scheduling of the same job. Obviously, then, if there was a 1201 EXECUTIVE overflow some job like SERVICER was scheduled two and possibly three or four times. In effect, the tail end of SERVICER was not being finished before a new SERVICER was scheduled.

The software restarts which accompanied the 1201 and 1202 alarms kept the program from becoming hopelessly snarled up. However, the unusually high frequency of unanticipated "counter increment requests" caused the program to require three software restarts within about 40 seconds in P64.

Long-Range Safer and More Flexible Software

It has been proposed by several people at MIT/IL (Peter Adler and Don Eyles) to provide a variable DT SERVICER program which starts a new cycle of computations as soon as the old cycle is finished. Our present SERVICER has a fixed DT of two seconds and the AVERAGE G, FINDCDUW and PIPA and GYRO compensation routines take advantage of this value of fixed integral number of seconds chosen for DT. Therefore, this programming change is not trivial. Such a mechanization has tremendous advantages however. By running all non-SERVICER jobs (like extended verbs and monitor verbs) at a higher priority than SERVICER, we can get every job done that is requested in one SERVICER cycle by making the end of one SERVICER cycle schedule the next SERVICER cycle. The computer is continually busy this way but the programs never overburden the EXECUTIVE job scheduling capacity. In effect, you have the shortest DT possible for the number of tasks and jobs you are trying to do. This also allows new computations (PCRs) to be inserted with the simple consequence of increasing the average DT. The undesirable effects are that the DT varies and verification of stability due to sampling frequency of PIPA reading and output command frequency to the DAP is more difficult.

How the RR Can Rob 15% of the LGC

The 1201 and 1202 program alarms which occurred during the mission G landing were due to a computer overload caused by high frequency shaft and trunnion angle "counter" incrementing from the RR ECDU's. The spurious erratic high frequency output from the RR ECDUs can occur whenever the rendezvous radar control switch is in SLEW or AUTO TRACK (rather than LGC) because then the ATCA provides a 15 volt 800 cps signal to the angle resolver primaries which has a random phase relationship to the 800 cps which the PSA provides for a reference to the RR ECDUs. The consequence is that the ECDUs may count at their maximum rate, 6400 cps, in their futile attempts to null the ECDU "error's". The loss of computation time for the two ECDUs counting at maximum rate is $12.8 \times 10^3 \times 11.72 \times 10^{-6} = 0.15$ seconds/second or 15% of the LGC time since each involuntary "counter" increment takes a memory cycle (one memory cycle = 11.72 microseconds).

The Apollo 11 crew checklist specified that the RR switch be in AUTO TRACK immediately before calling P63. We were unaware of the effect this would have on the computer execution time load.

Preventing Future LGC Memory Cycle Robbery by the RR ECDUs

When the RR mode switch is in either the AUTO TRACK or SLEW positions the RR ECDUs obviously cannot digitize the signal transformed from the shaft and trunnion angle resolver primaries to the input to the ECDUs. The only time the LGC is interested in the ECDU signal is when the RR mode switch is in LGC. Therefore, a PCR has been written, approved, and implemented in LUMIMARY 1B to monitor the discrete (the RR Power On/LGC Mode discrete) which the LGC receives when the RR switch is in the LGC position. When the LGC does not receive this discrete, it sets the RR ECDU zero command which stops the ECDUs' attempt to digitize the analog voltage from the resolvers. This action effectively prevents "counter increment requests" from the ECDUs and prevents central processor execution of these meaningless interrupts.

It should be mentioned that if the RR DC circuit breaker is pulled the LGC does not receive the RR Power On/LGC Mode discrete, and therefore the LGC will zero the RR ECDUs. This action is correct since the LGC is not interested in the shaft and trunnion angle of the RR when its circuit breakers are pulled.

We may modify our all-digital simulator to send 6400 pps from each RR ECDU to the LGC if the RR switch is not in LGC and the LGC has not set the RR CDU ZERO bit. Of course, if the simulator had this feature for our software verification tests we would have detected the computer overload during our testing.

In addition to the above PCR, other actions have been taken. PCR 814, for example, provides for a modified V57 which incorporates a display of N68 and avoids the need for a monitor verb and job (MONDO).

The separate call of MAKEPLAY has been eliminated by incorporating this job into SERVICER.

A Program to Keep Similar Events From Occurring in the Future.

There were folks who knew about the RR resolver/ECDU/LGC interface mechanization. There were also a great many people who knew the effect which a 15% TLOSS would have on the landing program's operation. These folks never got together on the subject. Therefore we need to improve communications.

There are various simulations throughout the country which might have detected this problem if they had simulated the RR resolver/ECDU/LGC interface with enough fidelity. Therefore, we might look at our various simulator configurations to see whether we are leaving anything significant out. We have just started a new Simulator Configuration Control Board at MIT/IL which will function like your own Apollo Software Configuration Control Board. This board will adjudicate the approval of SCRs (simulator change requests). We will encourage the writing of SCRs which can increase the fidelity of the simulator in significant ways. I hope to constitute this board with hardware personnel as well as software personnel so that important hardware/software interface features are not overlooked.

Our anomaly prevention plan includes the following steps:

1. "What if" sessions between hardware and software personnel.

2. Orientation lectures by hardware personnel for the benefit of software personnel so that programmers and mission support personnel better understand the hardware.

3. A review of the adequacy of the simulator we use for mission verification. Perhaps we can get hardware personnel from GAEC, MSC and MIT/IL to review our current models.

4. Scrutiny of the crew checklist by the hardware personnel. (This could have prevented the alarms.)

5. An insistence that anyone who knows anything peculiar about the hardware, the hardware/software interface, or the software document his information in the appropriate place (AOH, GSOP, Program Note, etc.) and tell the appropriate personnel.

Very truly yours,

George W. Cherry

George W. Cherry
Luminary Project Director

cc:

R. Ragan	K. Glick	W. North, CF
D. Hoag	T. Fitzgibbon	D. Cheathan, EG 2
L. Larson	B. McCoy	W. Goeckler, PD 8
R. Battin	C. Schulenberg	E. Kranz, FC
N. Sears	R. Covelli	H. Tindall, FM
F. Martin	D. Eyles	S. Bales, FC
A. Laats	P. Adler	R. Gardiner, EG
G. Silver	K. Greene	F. Bennett, FM 2
P. Felleman	L. Dunseith, FS	J. Hanaway, EG 412
R. Larson	T. Gibson, FS 5	R. Chilton, EG
J. Nevins	T. Price, FS 5	W. Heffron (Bellcomm)
W. Marscher	N. Armstrong, CB	C. Tillman GAEC
M. Hamilton	A. Aldrin, CB	G. Cherry (20)
A. Kosmala		

TABLE OF JOBS AND PRIORITIES

<table>
<tr><th>Name of Job and Type</th><th>Description of Job</th><th>Priority of Job</th></tr>
<tr><td>SERVICER (VAC)</td><td>Performs Navigation, Mass Updating, Guidance Computations, and Throttle and DAP Command Functions</td><td>20 but does a priority change (PRIOCHNG) to 24 while using subroutine to keep HIGATJOB from interferring (potential erasable conflict) NOT TRUE</td></tr>
<tr><td>MAKEPLAY (VAC)</td><td>Display job set up by SERVICER</td><td>20</td></tr>
<tr><td>1/GYRO (NOVAC)</td><td>IMU compensation job. Compensates gyros.</td><td>21</td></tr>
<tr><td>LRHJOB (NOVAC)</td><td>Reads the LR range</td><td rowspan="2">32 but they only run for a milli-second or so and then sleep for about 80 milliseconds while the LR syne pulses are sent out. They are awakened when LR reading is completed and then run for another millisecond or so.</td></tr>
<tr><td>LRVJOB (NOVAC)</td><td>Reads the LR velocities</td></tr>
<tr><td>CHARIN (NOVAC)</td><td>Started at each DSKY keystroke entry for interpretation of keystroke.</td><td>30</td></tr>
</table>

TABLE OF JOBS AND PRORITIES (CONT.)

Name of Job and Type	Description of Job	Priority of Job
HIGATJOB (VAC)	Commands the LR antenna to reposition to position #2 and sets up a waitlist task to check for appearance of the position #2 discrete. Sleeps during monitor. Is awakened after monitor finds position #2 discrete and computes position #2 beam vectors for LR Read Routine.	Runs at 32 for a millisecond or two. Sleeps until position #2 discrete is received. Is awakened at priority 32 but changes to priority 23 before * calling subroutine TRG*SMNB Note that if SERVICER is using QUICTRIG, HIGATJOB cannot interfere because SERVICER is at a higher priority then. Notice that if HIGATJOB is using TRG*SMNB SERVICER cannot make its priority change to 24 and hence cannot interfere.
MONDO (NOVAC)	Monitor Verb Job. Runs during V16 N68.	30

* Note: Both TRG*SMNB and QUICTRIG use the CDUSPOT registers. Hence a potential erasable conflict exists if TRG*SMNB is called while QUICTRIG is executing or vice versa.

IMPORTANT EVENTS OCCURRING
DURING LUNAR LANDING (Continued)

T Time From Ignition (seconds)	H (feet)	HDOT (ft/sec)	CDUY (degrees)	ΔH (feet)	V (ft/sec)	DSKY or Special Activity
538	4019	- 87.8	39	- 29	LPD 58	AUTO
554			33	+ 7		Program Alarm
556			33			V05N09E (R1 = 1201)
568	1927	- 51.8	28	+ 3	47	PRO to FLV06N64
578			26			Program Alarm
580	1387	- 39.1	25	- 12	44	V05N09E (R1 = 1202)
594	768	- 26.7	20	+ 84	35	Program Alarm (1202)
606		- 19.2	13	+ 7	32	ATT HOLD
618	430	- 11.8	6	+ 6	19	P66
666	244	- 2.9	5.7	- 2	VH 19.4	LR Dropout
680	201	- 4.6	5.6		11.1	LR Reacquire
714	74	- 0.9	6.0	+ 2	6.2	LR Dropout
722	60	- 2.4	6.0		3.8	LR Reacquire
744	20	- 0.7	3.9	+ 3	3.9	LR Velocity Reasonableness Fail
756	17	- 2.0	4.5	- 4	4.9	LR Dropout
760	10	- 0.2	4.4		2.7	Touchdown

IMPORTANT EVENTS OCCURRING
DURING LUNAR LANDING

TFI Time From Ignition (seconds)	H (feet)	HDOT (ft/sec)	CDUY (degrees)	ΔH (feet)	V (ft/sec)	DSKY or Special Activity
2	49971	- 2.2	107.25		5559.7	
232	42426	- 78.7	80		3366.4	Begin Yaw Windows Up
262-286						LR Data Good
286	37672	-104.3	79		2745.4	
304	35706	-113.6	77	-2884	2521.7	V16N68
316			77	-2570		Program Alarm
322			76	-2354		V 05N09E (R1 = 1202)
338	31566	-128.8	75	-2655	2086.1	V57E
344	30772		73	-1785		X Axis Over-ride Denied
346			73	-1772		V16N68E
358			71	-1826		Program Alarm (1202)
372	24703	-113.3	66	- 26	1640.2	
374			66	- 78		V16N68E
380			68	- 98		Key Release
384	23393	- 96.7	67	- 86	1481.1	Throttle Down
396	22233	-102.1	65	- 75	1367.0	ENTER
450	15273	-136.9	57	- 48	961.0	ENTER
506	7380	-125.7	55		526.2	P64
512	6636	-121.4	47		LPD 66	LPD in degrees, LR in Pos #2
528	4772	- 98.2	43	- 30	59	ATT HOLD