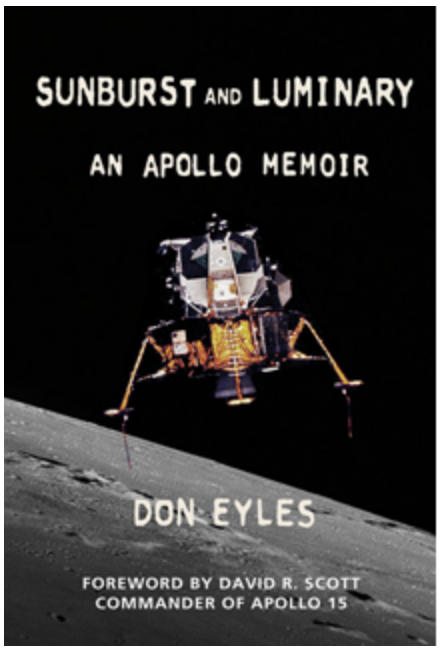
DON EYLES HOME PAGE



APOLLO / MIT ORGANIZATION CHARTS

TALES FROM THE LUNAR MODULE GUIDANCE COMPUTER

Don Eyles

*(A paper presented to the 27th annual Guidance and Control
Conference of the American Astronautical Society (AAS), in
Breckenridge, Colorado on February 6, 2004, and designated
AAS 04-064. This version includes additional illustrations and
comments, and several minor corrections.)*

ABSTRACT: The Apollo 11 mission succeeded in landing on the moon despite two computer-related problems that affected the Lunar Module during the powered descent. An uncorrected problem in the rendezvous radar interface stole approximately 13% of the computer's duty cycle, resulting in five program alarms and software restarts. In a less well-known problem, caused by erroneous data, the thrust of the LM's descent engine fluctuated wildly because the throttle control algorithm was only marginally stable. The explanation of these problems provides an opportunity to describe the operating system of the Apollo flight computers and the lunar landing guidance software.

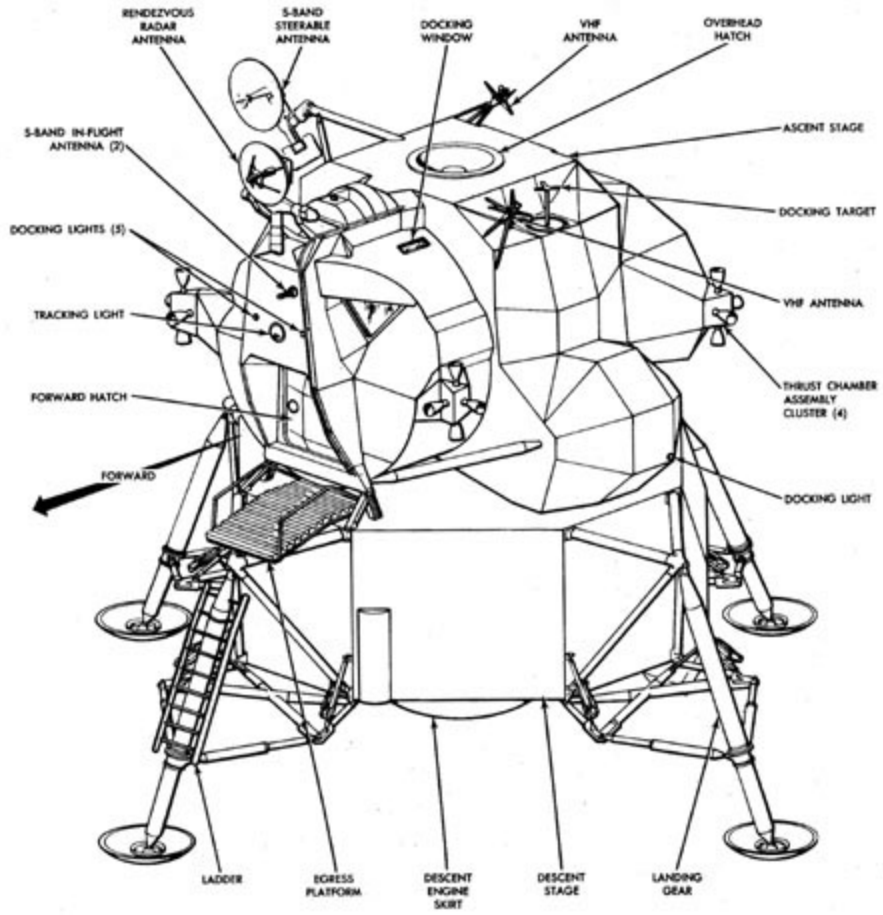


Figure 1: The Lunar Module

LM-1, also known as Apollo 5, was a 6-hour unmanned mission in earth orbit for the Lunar Module (LM) only. The date was January 22, 1968. For those of us who developed the onboard software for the LM Guidance Computer (LGC) it was our first flight. An event that had once seemed impossibly distant was now upon us.

The mission included two firings of the LM's Descent Propulsion System (DPS). For the second "burn" Allan Klumpp, who designed the lunar landing guidance equations[1] based on work by George Cherry[2], had devised an earth-orbit version of the lunar landing guidance. It had three parts, meant to simulate the

"braking" phase, "visibility" phase, and final landing phase of a real descent. But first there was a burn meant to simulate the descent orbit insertion maneuver that preceded the landing. This was to be the first in-flight firing of the LM's descent engine, lasting about 38 seconds.

The LGC was in Phase 9 of the "canned" LM-1 mission, the program for the first DPS burn. (Later missions were organized more flexibly and the first DPS burn was conducted in P40.) The LM had maneuvered to the burn attitude. The computer counted down to ignition. At thirty seconds a "task" called READACCS was executed for the first time. It read the accelerometers in the spacecraft's inertial measurement unit, scheduled a "job" called SERVICER to run immediately, and then scheduled itself to run again two seconds later. Having been initialized with state vectors from the onboard orbital integration software, SERVICER's "average-G" navigation equations began to use accelerometer data to update the position and velocity vectors. READACCS and SERVICER would repeat every two seconds throughout the powered-flight phase. Seven and a half seconds before ignition an "ullage" burn of the Reaction Control System (RCS) jets began, to settle the propellant in the DPS tanks. We leaned closer to the squawk box that connected us to mission control in Houston.

We heard "Engine on"... several seconds passed... "Engine off".

Soon we understood what had happened. A small piece of code in SERVICER called the "delta-V monitor" had concluded that the engine had failed and sent an engine-off command. But why? To give the engine time to come up to thrust, the delta-V monitor always waited some period of time after engine-on before it began to monitor the engine. But this time, at the end of the grace period the engine was still not producing enough thrust to satisfy the monitor's thrust criterion.

Published accounts[3] have attributed the slow DPS thrust buildup to the fact that the LM's tanks were only partially pressurized. The author's investigations show that the problem was elsewhere. For the DPS fuel system, the normal procedure was to open the valve that allowed fuel to enter the propellant manifold at the time the engine was armed, several seconds before ignition. But on LM-1 the control valve that regulated the passage of fuel from the manifold into the engine was suspected of being leaky. To prevent the possible, premature entry of hypergolic propellant into the engine (which could have had explosive consequences) the decision was made, shortly before flight, to delay arming the engine until the time of ignition[4].

The engine was slow to start not because the tanks were less pressurized, but because the propellant had further to travel to reach the engine. It would have been easy for us to adjust the parameter that controlled how long the delta-V monitor waited before testing the engine — but nobody told us.

Houston sent a signal to turn off the onboard computer. The main objectives of the LM-1 mission were achieved under ground control. We who programmed the LM's computer hung our heads in disappointment, and endured a public reaction

that did not distinguish between a "computer error" and a mistake in the data. Yet, this was not the last time that a seemingly innocuous parameter, relating to the performance of the descent engine, would come perilously close to ruining a mission.

* * *

The job of designing the guidance system for the Apollo spacecraft had fallen to the MIT Instrumentation Laboratory in Cambridge, Massachusetts. Under the leadership of its founder "Doc" Charles Stark Draper, the Lab had since 1939 played the preeminent role in perfecting inertial guidance systems. Our contract to design and program the Apollo Primary Guidance Navigation and Control System (PGNCS, pronounced "pings") was the first Apollo contract signed. Doc had volunteered to fly the mission himself.

> *(In 1970 the Instrumentation Laboratory was renamed the Charles Stark Draper Laboratory, and in 1973 became independent from MIT, although the two institutions remain linked. The Draper Laboratory is still deeply involved in NASA's manned spaceflight programs.)*

The flight computer program for LM-1 was called SUNBURST. By the time LM-1 flew we were already working on SUNDANCE, the program that would fly the earth-orbital Apollo 9 mission. SUNDANCE in turn evolved into LUMINARY, the program for Apollo 10 and the lunar landing missions. It was LUMINARY revision 99 that flew the Apollo 11 mission in July, 1969. Revision 116 flew Apollo 12 in December, and so on.

> *(This paper follows nomenclature used during the Apollo Program. Program names, and the names of tags and variables within programs, were usually written in upper case.)*

Informally, the programs were called "ropes" because of the durable form of read-only memory into which they were transformed for flight, which resembled a rope of woven copper wire. For the lunar missions, 36K words of "fixed" (read-only) memory, each word consisting of 15 bits plus a parity bit, were available for the program. In addition there were 2K words of artfully timeshared "erasable" or RAM memory. Allowing for the identical Apollo guidance computer (AGC) in the Command Module (CM), containing a program called COLOSSUS, it is correct to say that we landed on the moon with 152 Kbytes of onboard computer memory.

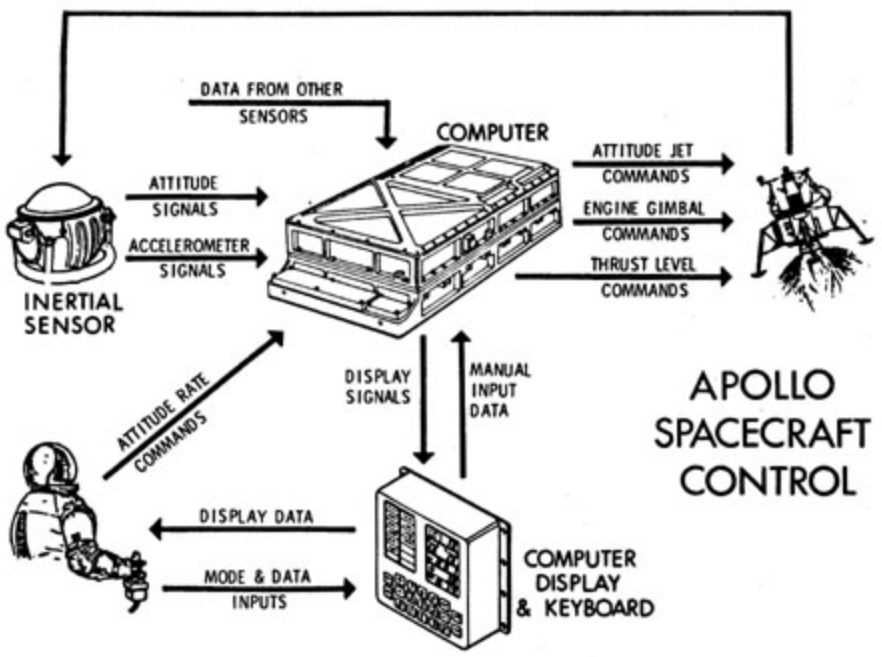


Figure 2: Apollo LM Primary Guidance and Navigation System (PGNS)

The AGC was packaged in a sturdy, sealed, aluminum-magnesium box, anodized in a gold color, that measured about six inches, by one foot, by two feet, weighed 70 pounds and consumed about 55 watts. Its logic was made up of 5600 3-input NOR gates packaged two-each in flat-pack integrated circuits. Eldon Hall, the machine's principal designer, has related the bold decision to use integrated circuit technology for this computer despite its immaturity in the early 1960's[5].

The LGC (with related equipment) was mounted behind the astronauts at the back of the LM cabin. In front of the astronauts was a rigid structure called the "Nav Base" that held an alignment telescope and the Inertial Measurement Unit (IMU) in a fixed geometrical relationship. The computer's Display and Keyboard Unit (DSKY) was mounted like a desk between the two astronauts. Figure 2 illustrates the components and high-level interfaces of the LM's primary guidance system.

The IMU, packaged in a spherical case about a foot in diameter, was the heart of the guidance system. The heart of the IMU itself, enclosed by three nested gimbals, was the "stable member" — a small platform containing three accurate gyroscopes and three accelerometers — that could be "aligned" to an inertial orientation. Any deviation from the inertial alignment would be sensed by the gyros, and the gimbals would move to correct, all happening with such precision that no matter what attitude (orientation) the spacecraft took (almost), the stable member deep inside provided a steady attitude reference. A matrix called REFSMMAT expressed the stable-member alignment with respect to the reference inertial frame. The accelerometers were there to count velocity increments during powered flight in the coordinate system of the stable member.

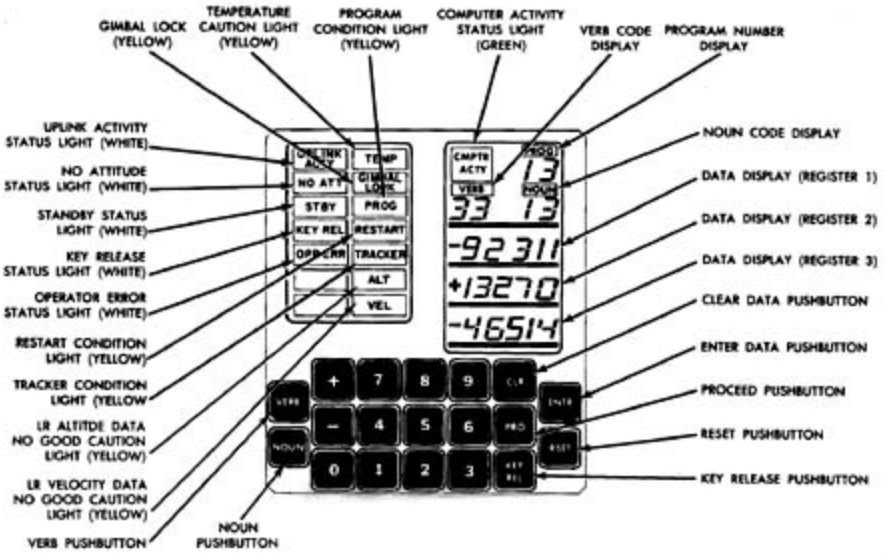


Figure 3: Lunar Module Display and Keyboard Unit (DSKY)

The DSKY (Figure 3) was the principal man-machine interface for the LGC. For display it provided three signed five-digit registers for general-purpose use, three two-digit registers to indicate the current phase (a number between 63 and 68 for the lunar landing), and the current "verb" and "noun". Verbs and nouns provided a primitive language for communication between the crew and the computer. Phases and verb/noun combinations were determined by the software in some cases, and in other cases were entered by the crew on a keyboard of 19 keys. The contents of the three general-purpose registers depended on the current verb and noun. The DSKY also contained an array of indicator lights that were under the control of the computer, and a computer-activity light that lit when the LGC was *not* in its idle state.

The AGCs in the LM and CM were programmed in two languages. The one we called "Basic", but more properly "Yul", was an assembler language of about 40 operations[6], authored by Hugh Blair-Smith. "Interpretive" was a list-processing interpretive language (essentially a set of subroutines) designed to facilitate guidance and navigation calculations involving double precision (30-bit fixed-point) vectors and matrices — at the cost of being very slow[7]. The Interpreter was written by Charles Muntz.

The memory-cycle time for the AGC was 11.7 microseconds. A single-precision addition in the assembler language took two memory cycles. A double-precision vector cross-product programmed in Interpretive took about 5 milliseconds. One of the challenges in programming the AGC was juggling the two languages to obtain the best blend of speed and compactness for the given situation.

The computer programs for Apollo were still small enough to fit into one listing — typically six inches thick on 11x15 inch fan-fold paper. The listing included symbol tables that allowed threads to be traced. With a single listing we always knew that the answer was there, when we had a bug to deal with, but it might be devilish to find.

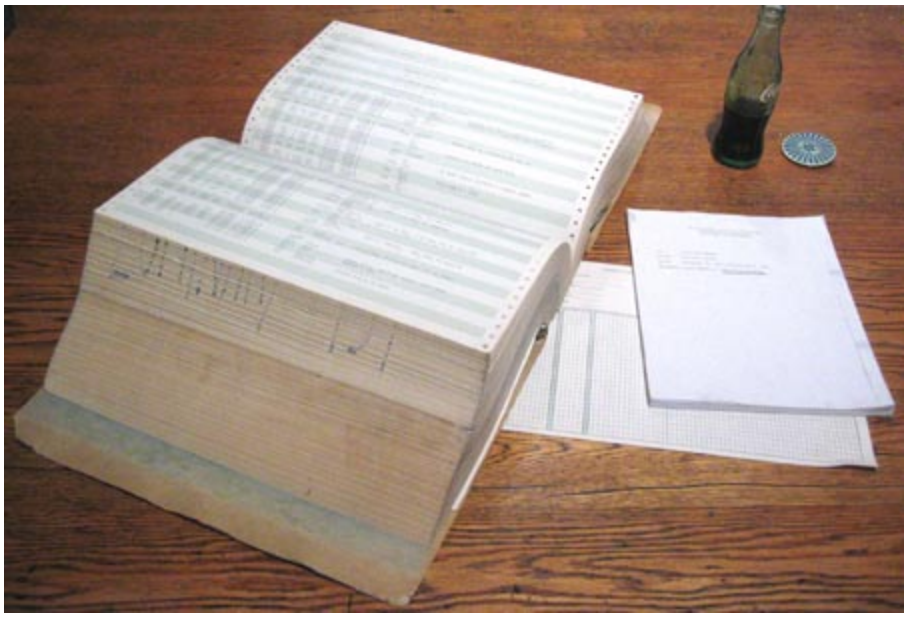


Figure 4: Listing of LM Computer Program LUMINARY 131

With respect to units, the LGC was eclectic. Inside the computer we used metric units, at least in the case of powered-flight navigation and guidance. At the operational level NASA, and especially the astronauts, preferred English units. This meant that before being displayed, altitude and altitude-rate (for example) were calculated from the metric state vector maintained by navigation, and then were converted to feet and ft/sec. It would have felt weird to speak of spacecraft altitude in meters, and both thrust and mass were commonly expressed in pounds. Because part of the point of this paper is to show how things were *called* in this era of spaceflight, I shall usually express quantities in the units that it would have felt natural to use at the time.

* * *

By now the area on the second floor of 75 Cambridge Parkway where we monitored missions had been moved to a larger space, but on July 20, 1969 the room was crowded despite efforts to keep it clear for those of us who were most involved in this phase of the mission. We listened to a squawk box in a nondescript classroom, while a quarter of a million miles away a manned spacecraft emerged from behind the moon and approached its orbital low-point (perilune) of about 50, 000 feet above the cratered surface, where the lunar landing burn would begin.

The crew keyed in Verb 37 to select P63, the phase that controlled the preparations for Powered Descent Initiation (PDI) and stayed in control until the burn achieved its first set of targets. The computer processed an algorithm to compute the exact time for ignition and the attitude the LM should be in at that time. Next the spacecraft maneuvered to that orientation. At the time of ignition the engine bell would be pointed almost dead ahead, directly opposing the spacecraft's orbital velocity.

Now the computer issued code 500. It thought the landing radar antenna was in the wrong position. The crew saw that the relevant switches were already in the right positions, but they cycled them anyway and the warning cleared. This had no connection with the events that would follow, but it nourished our suspicion of "discretes", those signals that told the computer some fact like the position of a switch or an antenna — but sometimes lied.

Control passed to BURNBABY — the master ignition routine that we wrote after LM-1 to save memory by exploiting the similarities among the powered flight phases in the period leading up to ignition. Verb 06 Noun 62 appeared on the DSKY. The middle register contained a time in minutes and seconds that began to count down toward light-up. At 35 seconds the display went blank, and at 30 seconds reappeared. This was a signal that Average-G had started. At seven and a half seconds, the ullage burn began. At five seconds, the display flashed to request a "go" from the crew. Buzz Aldrin, the LM Pilot, standing on the right side of the cockpit, had the main responsibility for working the DSKY. Now he keyed PROCEED.

At Mission Elapsed Time (MET) 102:33:05 self-igniting propellants came together in the descent engine and it lit up at 10% throttle. Armstrong did not even feel the gentle push — less than 1/25 G. The display changed to Noun 63 and the three display registers now showed a total velocity of 5559.7 ft/sec, an altitude-rate of -2.2 ft/sec, and an altitude of 49971 feet[8]. The gimbals that pivoted the descent engine moved to align the thrust vector with the spacecraft's center of mass. Then, 26 seconds into the burn, the software throttled-up the DPS to its maximum thrust of 9870 pounds (43, 900 newtons), 94% of the engine's official rating of 10500 pounds, and at the same time enabled the descent guidance.

P63 was called the braking phase because its only purpose was to shed horizontal velocity. It would end in about eight minutes when the spacecraft reached target conditions known as "high gate" at about 7400 feet altitude. Figure 5 illustrates the phases of the lunar landing.

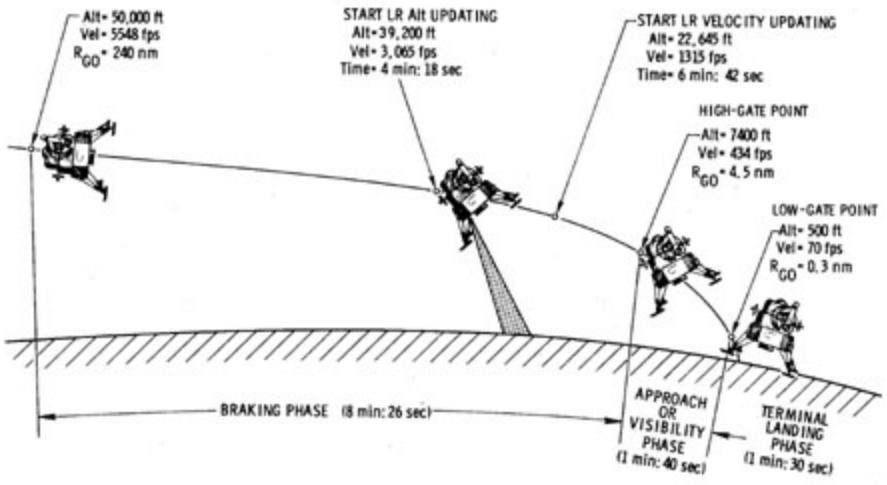


Figure 5: Phases of the Lunar Landing (Numbers Approximate)

At MET 102:36:55 Neil Armstrong, the Commander, standing on the left side of the LM cockpit, used his joystick to spin the spacecraft about its thrust axis so that the windows, which had allowed the astronauts to look down at the surface while hurtling forward feet first, would point out into space, where Earth was visible. But the spacecraft was rotating too slowly. Armstrong realized the autopilot rate switch was at 5 deg/sec and switched it to 25[9]. Just before the maneuver was complete the landing radar signaled "data good".

It was not possible to navigate so accurately as to touch down safely on the lunar surface with no local knowledge of its relative distance or velocity. The landing radar provided this information. Despite the "reasonability check" performed by the software, radar data could not be incorporated into the state vector without crew (and ground) approval. So about five minutes into the burn Aldrin keyed in Verb 16 Noun 68 — a request to monitor a noun whose third register showed the difference between the altitude sensed by the radar and the computed altitude. This number, called DELTAH, was about -2900 feet. This was within the range of expected altitude error. The radar data could gradually be folded into navigation without adversely affecting the shape of the trajectory.

Then we heard the words "program alarm". In Cambridge we looked at each other. Onboard, Aldrin saw the PROG light go on and the display switch back to Verb 06 Noun 63. He quickly keyed in Verb 5 Noun 9. Alarm code 1202 appeared on the DSKY. This was an alarm issued when the computer was overloaded — when it had more work to do than it had time for. In Cambridge the word went around, "Executive alarm, no core sets". Then Armstrong said, with an edge, "Give us a reading on the 1202 program alarm"[10].

From here events moved very quickly, too fast for us to have any input from Cambridge. It was up to Mission Control in Houston. The story of what happened there has often been told — how it fell to a 26-year-old mission control guidance officer named Steve Bales to say "go" or "abort". Bales had participated in a recent review of LGC alarms that had deemed 1202 a "go" unless it occurred too often or the trajectory deviated. He was supported by Jack Garman of NASA and Russ Larson of MIT in the back room. Garman said, "go". Larson gave a thumbs-up. (He later said he was too scared to form words.) So Bales answered, "go", Flight Director Gene Kranz said "go", and capsule communicator Charlie Duke passed it up to the crew. At MIT, where we realized that something mysterious was draining time from the computer, we were barely breathing.

Half a minute elapsed between the alarm and the "go" from Houston. During that time mission control approved the DELTAH, and Aldrin keyed in 57 to allow navigation to incorporate the landing radar measurements. Then he tried Verb 16 Noun 68 again and watched DELTAH decrease to 900 feet. Again a program alarm light. Again Verb 5 Noun 9 — 1202 alarm. Again "go" from the ground.

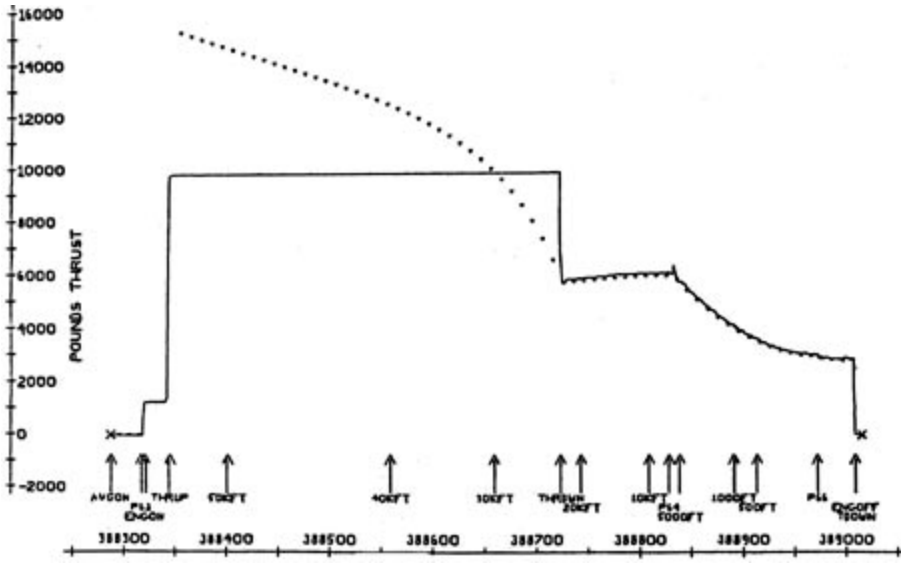


Figure 6: Commanded (dotted line) Versus Actual Thrust (solid line) During Powered Descent (Simulation Data)

At MET 102:39:31 the best possible confidence builder occurred — throttle down, right on time. "Ah! Throttle down... better than the simulator" commented Aldrin, "Throttle down on time!" exclaimed Armstrong, their excitement palpable. In the official transcript of communications between spacecraft and ground during the powered descent, these are the only exclamation points[11].

The descent engine experienced excessive nozzle erosion if operated in the range between 65% and maximum thrust. Throttle down occurred when the thrust required by guidance sank to a level enough below that limit that a gradual increase through the end of the braking phase would not force a return to maximum (see Figure 6). Throttle down was a sensitive indicator of how well the guidance system was doing. It was also true that if the throttle stuck at maximum an abort might soon be necessary, because in about 40 seconds the guidance equations would command the spacecraft to invert.

While the LM was still facing the lunar surface Armstrong had clocked landmarks that indicated the LM was further downrange than desired. He realized now that the computer did not know the lander was going long[12]. Otherwise the engine would have stayed at maximum thrust for longer as guidance tried to stop short.

At MET 102:41:32, as the spacecraft passed through 7400 feet, sinking at 125 ft/sec, high gate was achieved. Guidance began using a new set of targets. The LM pitched forward so that the lunar surface was visible ahead. On the DSKY the mode register changed to 64 indicating the Visibility Phase, and Noun 64 replaced Noun 63. Two two-digit numbers replaced velocity in the top register. One was a "landing point designator" (LPD) angle that indicated where Armstrong should look along a reticle attached to his window to see where the LM would touch down if it were allowed to land automatically. The guidance system controlled yaw to keep the landing site along the line of the reticle. The crew could move a hand controller to shift the site. (Armstrong had stated before

the flight that he planned not to use this capability, but there was apparently one inadvertant redesignation late in the visibility phase.) The second number gave the time remaining during which a redesignation could be input. With the redesignation logic now engaged, this was the busiest period of the landing.

At MET 102:42:17 a 1201 alarm occurred. It was another Executive alarm — "No VAC areas available". About 24 seconds later there was another 1202. Just 16 seconds later, with the lander at 770 feet with a sink rate of 27 ft/sec, yet another 1202 occurred. Mission control in Houston called a "go" in each case. Neil Armstrong, whose heart rate rose from 120 to 150 during this period[13], put it this way:

> Normally, in this time period, that is, from P64 onward, we'd be evaluating the landing site and starting LPD activity. However, the concern here was not with the landing area we were going into, but rather whether we could continue at all. Consequently, our attention was directed toward clearing the program alarms, keeping the machine flying, and assuring ourselves that control was adequate to continue without requiring an abort. Most of our attention was directed inside the cockpit during this time period and in my view this would account for our inability to study the landing site and final landing location during final descent[14].

Nevertheless, Armstrong had time to notice that the LPD indicated "we were landing just short of a large rocky crater with very large rocks covering a high percentage of the surface"[15]. So at MET 102:43:08 (650 feet), after deciding that he could not stop short of the crater, Armstrong flipped the autopilot mode switch from AUTO to ATT HOLD to take manual control of the LM's attitude. He maneuvered to zero pitch to maintain horizontal velocity and skim over the rocky area.

> *(ATT HOLD meant the digital autopilot's Rate-Command Attitude-Hold mode, in which the astronaut could command an attitude rate by deflecting a joystick. After the stick was released the autopilot nulled rates to maintain the present attitude.)*

At MET 102:43:20 (430 feet) Armstrong flicked a spring loaded toggle switch with his left hand, entering the rate-of-descent mode (P66). Now the computer controlled the spacecraft's thrust to maintain a rate-of-descent commanded by the ROD switch. A flick upward slowed the descent by one foot per second; a flick downward increased the descent rate by the same amount. Using the joystick, Armstrong tilted the LM to null out horizontal velocity and bring the LM to a safe area for touchdown. After some "possibly spastic" control motions because dust kicked up by the exhaust plume distorted his perception of translational velocity, at MET 102:45:40, Armstrong landed the spacecraft safely in the Sea of Tranquility.

* * *

Years before Apollo 11, when the guidance system was first being conceived, the onboard software was almost an afterthought — "Hal will take care of it" was the sentiment. In fact it ended up taking scores of people, with hundreds more in support, but to Hal Laning, in the early days, fell the job of figuring out how to organize the numerous software functions that must go on almost simultaneously in a real-time spacecraft control computer — in this case one of limited size and speed.

Hal's design avoided the pitfalls of a "boxcar" executive, in which the computations must be divided up explicitly between time slices. A boxcar executive is painful to implement because computations must be broken up arbitrarily. During development the allocation may need to be revised whenever any of its parts is modified or new functions are added. Worst of all, a boxcar executive is a *brittle* system during operation. It breaks down completely as soon as any function takes longer than the time it is allocated.

Instead, Laning envisioned a system in which software functions were allocated among various "jobs" that could be of any size and shape, as determined by the nature of their function. Each job was assigned a priority. The operating system always executed the job with the highest priority. Thus, if a low-priority job was executing and a high-priority job was scheduled, the low-priority job was suspended while the higher-priority job executed. This system gave the illusion that jobs ran simultaneously, although of course they merely took turns. Such a system was not deterministic in the sense that what executed when could be determined *a priori*, but its operation could be sufficiently understood and verified that in sum it enhanced reliability, safety, flexibility of use, and especially ease of development.

In such a design the Executive function that orchestrated the execution of jobs had to provide each job with a set of registers in which its status could be saved if it was suspended during the execution of a higher priority job. The LGC contained an array of eight such "core sets" of 12 registers each, each register having 15 bits. A core set of this size was sufficient for many jobs, but jobs that used the Interpretive language to do vector and matrix computations required more space. For such jobs an additional area of 43 registers was allocated for the storage of intermediate results. There were five such "Vector Accumulator (VAC) areas" in the LGC.

With a limited number of core sets and VAC areas, the allocation of functions to jobs had to be done thoughtfully. Functions that had a sequential relationship with each other were grouped into the same job. Thus the large SERVICER job that was active during the lunar landing (and other powered flight modes) first performed average-G navigation, then guidance equations, then throttle and attitude output, and then the updating of displays — each part using the outputs of the ones preceding.

The availability of core sets and VAC areas limited the number of jobs that could be in the queue at any time to eight, of which up to five could require VAC areas.

In normal steady-state operation, the number of jobs executed equaled the number being scheduled and therefore the average usage of core sets and VAC areas was more of less steady, although jobs that occurred on a one-shot or asynchronous basis might cause the usage to fluctuate.

However, if more jobs were being scheduled than were being finished, the number of core sets and VAC areas in use must rise. If the debit continued long enough, the resources would be exhausted. The next job request could not be fulfilled.

Cut to a time about a year before Apollo 11, when we software engineers, who thought we already had enough to do, were requested to write the lunar landing software in such a way that the computer could literally be turned off and back on without interrupting the landing or any other vital maneuver! This was called "restart protection". Other factors than power transients also caused restarts. A restart was triggered if the hardware thought the software was in an endless loop, or if there were a parity failure when reading fixed memory, or for several other reasons.

Restart protection was done by registering waypoints at suitable points during the operation of the software such that if processing happened to jump back to the last waypoint, no error would be introduced, as in the following example:

```
NEW_X = X + 1
register waypoint
X = NEW_X
```

It is evident that without the waypoint, going through this code a second time would cause X to be incremented twice.

Following a restart, such computations could be reconstructed. For each job, processing would commence at the last registered waypoint. If multiple copies of the same job were in the queue, only the most recent was restarted. Certain other computations that were not considered vital were not restart-protected. These would simply disappear if there were a restart.

Restart protection worked very well. On the control panel of our real-time "hybrid" simulator in Cambridge was a pushbutton that caused the AGC to restart. During simulations we sometimes pushed the button randomly, almost hoping for a failure that might lead us to one more bug. Invariably, once we got the restart protection working, operation continued seamlessly.

(The hybrid simulator combined SDS 9300 digital and Beckmann analog computers with a real AGC and realistic LM and CM cockpits.)

Restart protection was prompted by the possibility that the hardware could cause a restart, but the software could also initiate a restart if it reached a point where it did not know how to continue. This was done by transferring control to the tag BAILOUT in the Alarms and Aborts software. An error code accompanied this call.

This was the action taken by the Executive program if its resources were exceeded. If a job could not be scheduled because no "core sets" were available, the Executive called BAILOUT with alarm code 1202. If no "VAC areas" were available, BAILOUT was called with alarm code 1201.

Not all the functions executed in the LGC were "jobs". There was also a system of hardware interrupts, which could break in at any point (when not explicitly inhibited) to perform high priority functions. Interrupts were dedicated to particular functions including the digital autopilot, uplink and downlink, and keyboard operation.

Another interrupt could be used to execute any piece of code that had to be executed at a given time. Such functions, called "tasks", were scheduled by calling a subroutine called WAITLIST. A task had to be of very short duration.

Whereas jobs were scheduled to execute immediately at a given *priority*, tasks were scheduled to run at a given *time*. Tasks and jobs were often used together. A task might be scheduled to capture sensor data that needed to be read at a definite time, and the task in turn might schedule a job at an appropriate priority to perform processing based on the measurement.

When Hal Laning designed the Executive and Waitlist system in the mid 1960's, he made it up from whole cloth with no examples to guide him. The design is still valid today. The allocation of functions among a sensible number of asynchronous processes, under control of a rate- and priority-driven preemptive executive, still represents the state of the art in real-time GN&C computers for spacecraft.

* * *

To understand the root cause of the alarms on Apollo 11 during the powered descent, one must first look ahead to the rendezvous with the Command Module that followed the LM's ascent to lunar orbit. Just as it needed the landing radar to measure altitude and velocity with respect to the lunar surface during the landing, the LM, as the active vehicle during rendezvous with the CM in lunar orbit, needed the rendezvous radar (RR) to measure the range, range-rate, and direction of the other spacecraft.

The RR had several modes of operation, determined by the setting of its mode switch. As flown on Apollo 11, the available RR modes were SLEW, AUTO,

and LGC. In SLEW and AUTO modes the radar operated under the control of the crew, independently of the LGC. This was the method that would be used during ascent and rendezvous if the primary guidance system failed. In SLEW mode the rendezvous radar antenna could be steered manually, but otherwise was stationary. Once the antenna was pointed near the target, the AUTO (automatic tracking) mode could be used to acquire and track the target. In these cases the RR range and range-rate, and the shaft and trunnion angles that defined where the RR antenna was pointing, were made available for display on cockpit cross-pointers and tape meters. Range and range-rate were also made available to the abort guidance system (AGS), a computer with only 6144 words of memory that was provided by TRW as a backup for use if the PGNS failed during lunar descent or ascent.

> *(The naming of the three rendezvous radar modes has been a source of confusion for some commentators. Based on crew input the designations were changed between LM-1 and the lunar landing missions. The mode called LGC on Apollo 11 was formerly called AUTO. The mode called AUTO on Apollo 11 was formerly MANUAL. SLEW was unchanged. Although it in no way contributed to the problem on Apollo 11, LUMINARY's internal documentation at this time still referred to the discrete in Channel 33 that indicated that the rendezvous radar was powered up in LGC mode as RR AUTO-POWER ON.)*

If the PGNS was healthy (as it always was) the radar was controlled by the LGC, and in this case the RR mode switch was set to LGC. The RR interface electronics made available to the software the target range and range-rate measured by the radar, and the angles of the RR antenna's shaft and trunnion, from which the direction to the target could be determined. Programs running in the LGC used this information to guide the LM to a favorable rendezvous.

It turned out that the rendezvous radar could also be operated during the powered descent, and this was done during Apollo 11. Crew procedures called for the RR to be switched on just before P63 was selected, and to be kept in SLEW or AUTO mode throughout the landing maneuver.

Many explanations have been offered for why the RR was configured in this way for the lunar landing. For example, a fanciful scheme for monitoring the landing by comparing RR data to a chart of expected readings may have been considered by some people in Houston. However, a simpler explanation is sufficient to explain the facts: The RR was on for no other purpose than to be warmed up if there were an abort, and it was in AUTO (while the LM was in a position to track the CM) or in SLEW (at other times), simply to keep the antenna from moving uselessly.

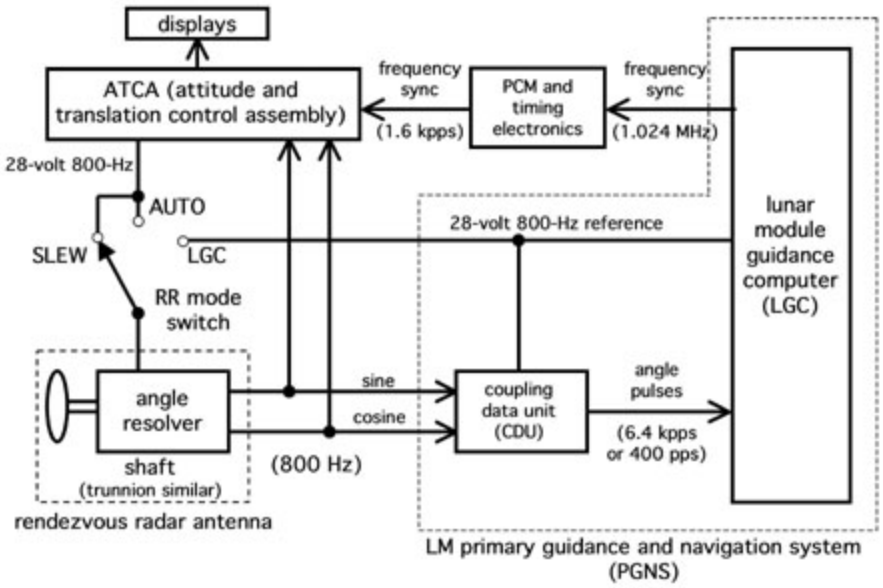


Figure 7: Interfaces Among PGNS, ATCA and the Rendezvous Radar

The problem has also been attributed (including by the author previously) to a "checklist error". This formulation is no more accurate than calling the delta-V monitor's premature shutdown of the engine on LM-1 a "computer error", when it was actually caused by faulty documentation. In fact, the RR switch settings on Apollo 11 should not have caused any problem. That they did so can be traced to another case of... faulty documentation.

Years previously, an interface control document (ICD) had been written to define the electrical interface between the PGNS and an electronic assembly called the attitude and translation control assembly (ATCA) that was provided by Grumman Aerospace, the builder of the Moon lander. The ICD specified that the 28-volt 800-Hz voltages in the two systems be "frequency locked", but did *not* say, "phase synchronized". As built, the two voltages were locked in frequency by a "frequency sync" signal sent by the LGC. They were also locked into a constant phase relationship. However, the phase angle between the two signals was completely random, depending on the instant at which the LGC, which was always powered up *after* the ATCA, began sending the first frequency sync signal. These interfaces are pictured in Figure 7.

The 800-Hz phasing problem was detected during launch site testing of LM-3 and documented — but it was never corrected. As a result, when the RR mode switch was in AUTO or SLEW, the shaft and trunnion resolvers were being excited by an 800-Hz signal from the ATCA that was very likely to be out of phase with the 800-Hz waveform used as a reference by the coupling data units (CDUs) whose job was to make sense of the resolver signals, and in turn increment (or decrement) the counters inside the computer that told the software how the antenna was pointed.

On Apollo 11, however, the CDUs were being asked to comprehend a contradiction. Because they were based on a separately controlled excitation

voltage, the resolver signals as received by the CDUs indicated no known angle. The discomfiture of the CDUs was at its worst when the phase angle between the two 800-Hz waveforms was near 90 or 270 degrees — and Apollo 11 evidently hit one of these sweet spots. The response of the CDUs was to increment or decrement the counters in the LGC, nearly constantly, at the maximum rate of 6400 pulses per seconds for each angle. This phenomenon occurred whenever the RR mode was in SLEW or AUTO, regardless of whether the rendezvous radar itself was powered up.

The CDU interface counters in the LGC were incremented or decremented by means of external commands that were processed inside the computer as increment or decrement *operations* with names like PINC and MINC. Like the LGC's programmable operations, these took time, in this case one memory cycle of 11.7 microseconds, each. Moving at their maximum rate, the RR CDU counters consumed approximately 15% of the available computation time. At the time, conservatively, we assumed the time drain (called TLOSS) was about 13%, which was consistent with the behavior that was observed.

> *Following Apollo 11 Grumman engineers conducted tests in an attempt to duplicate the flight experience. They confirmed that even in the worst case the RR CDUs would, for brief periods, not count at their maximum rate. They arrived at a figure of 13.36% for the maximum TLOSS that could occur. Simulations at that rate experienced alarms similar to those that occurred in flight. This number is the best documentable estimate for the amount of TLOSS experienced by Apollo 11. [Clint Tillman, "Simulating the RR-CDU Interface When the RR is in the SLEW or AUTO (not LGC) Mode in the FMES/FCI Laboratory," August 9, 1969]*

I am indebted to LM guidance systems expert George Silver for his patient explanations of the rendezvous radar interface. Silver's role was pivotal during the Apollo 11 mission. He was at Cape Canaveral for the launch, then flew to Boston to get ready for an assignment to monitor the lunar ascent in Cambridge. On July 20 he watched the lunar landing at home on television. He heard the alarms, grasped that something was stealing CPU time, and remembered the case he had seen during LM-3 systems testing in which the rendezvous radar interface had caused wild counter activity. After some additional analysis by the team monitoring the mission in Cambridge, Silver finally got through to the MIT representatives in Houston, on the morning of July 21, less than one hour before lunar liftoff.

* * *

The lunar landing was the busiest mission phase on Apollo. Landing guidance had to hit targets that were defined in position, velocity, acceleration (so the LM would stay right side up), jerk (the rate of change of acceleration), and one dimension of "snap" — as Klumpp was pleased to dub the rate of change of jerk

(pointing to "crackle" and "pop" for the next two derivatives). During the visibility phase the software permitted the crew to redesignate the landing site. The throttle had to be controlled continuously. Navigation had to incorporate landing radar measurements. (Figure 8 shows the typical duty-cycle profile between the selection of P63 and touchdown.)

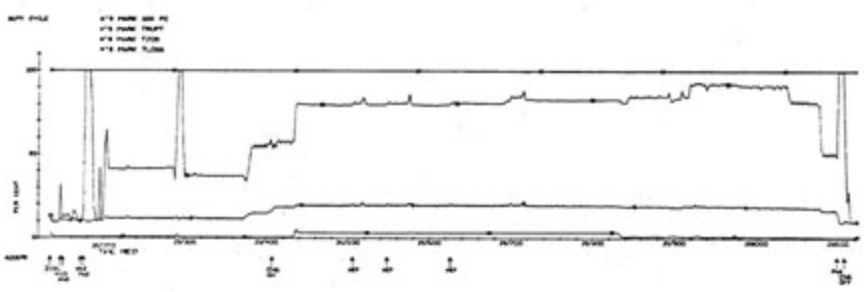


Figure 8: Duty Cycle During Powered Descent (Simulation Data)

Even so, we had tried to make our programs fast enough to preserve some margin against TLOSS from an unknown source. The chief constraint was the two-second period that was built into the average-G navigation used during powered-flight. This was the frequency at which the READACCS *task* read the accelerometers and scheduled the big SERVICER *job* that used those readings as the starting point for a new round of navigation, guidance, throttle, attitude-command, and display. During the lunar descent, duty-cycle simply describes how much time was used in aggregate by jobs, tasks, and interrupts, during each 2-second period.

During the braking phase, up to the time the landing radar locked onto the surface, the duty-cycle margin was over 15%. After the radar acquired, the extra computations involved in converting the body-referenced radar data to the navigation coordinate system lowered the margin to perhaps 13%. When a monitor display such as Verb 16 Noun 68 was added, the margin shrank again, to 10% or less. Buzz Aldrin was perceptive when he said after the second 1202 alarm, "It appears to come up when we have a 1668 up"[16].

With a 10% margin and a 13% drain, the LGC simply did not have enough CPU time to perform all the functions that were required. Thanks to the flexibility of the Executive design — and quite unlike what would have happened with a boxcar structure — there was no collapse.

name	priority	type	function
SERVICER	20	VAC	navigation, guidance, throttle and autopilot command, and display
MAKEPLAY	20	VAC	display job
1/GYRO	21	NOVAC	performs IMU gyro compensation
LRHJOB	32	NOVAC	reads landing radar range
LRVJOB	32	NOVAC	reads landing radar velocity
CHARIN	30	NOVAC	runs to interpret each DSKY keystroke
HIGATJOB	32	VAC	runs once only to reposition landing radar antenna at high gate
MONDO	30	NOVAC	runs when Verb 16 monitor is active

Table 1: Jobs Active During the Lunar Landing

Table 1 lists the jobs that were active during the Apollo 11 powered descent. SERVICER carried the lowest priority, but was also by far the longest. The higher-priority jobs that could break in on SERVICER were all of relatively short duration.

Having a relatively low priority because of its size, SERVICER got last crack at the available computation time. With a negative time margin it was SERVICER that had not yet reached its conclusion when the next READACCS, running punctually, scheduled SERVICER again. Because it had not reached its end, the earlier SERVICER had not released its core set and VAC area — so the next time READACCS called FINDVAC to schedule SERVICER the Executive assigned a new core set and VAC area. That SERVICER also did not finish. After a short span of such operation the Executive exhausted its supply of core sets and/or VAC areas. When the next request was made the Executive, unable to comply, called BAILOUT with a 1201 or 1202 alarm code.

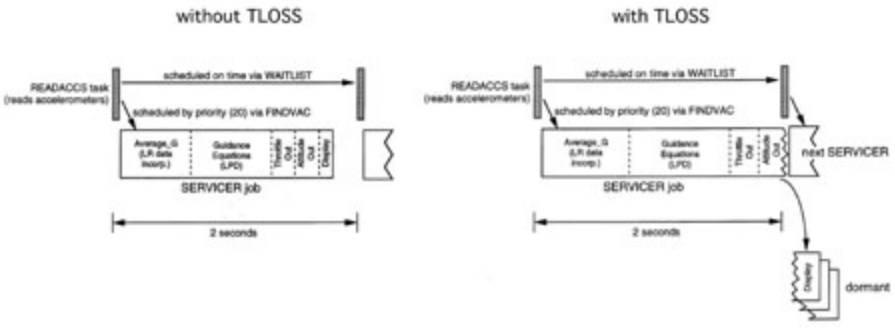


Figure 9: SERVICER Operation, With and Without TLOSS

Figure 9 illustrates how SERVICER behaves in the presence of severe TLOSS, and Figure 10 compares plots of core set and VAC area usage for a normal case, and a high TLOSS case in which restarts occur.

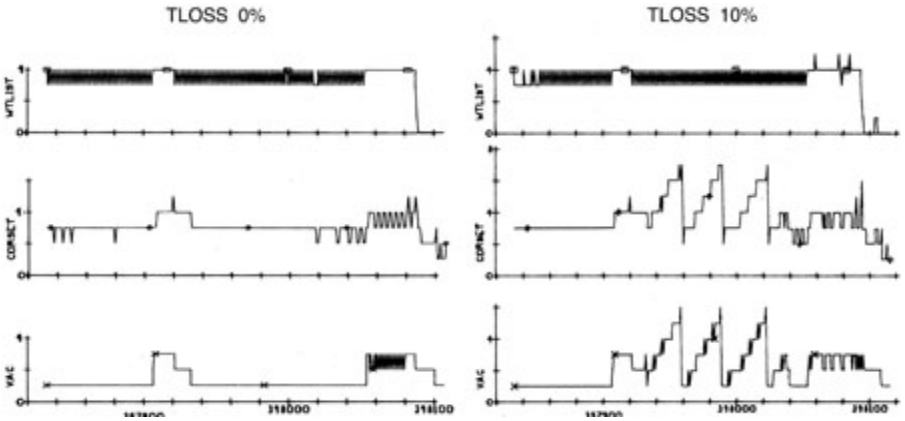


Figure 10: Effect of TLOSS on Executive and Waitlist Resources During Lunar Descent
(Simulation data, starting in P63 before acquisition of radar velocity data, ending at touchdown[17].)

The interesting effect of this train of events, during P63, was that the problem fixed itself. The software restart reconstructed only the most recent incarnation of the SERVICER job, and flushed the uncompleted SERVICER "stubs" that had accumulated. In addition, it terminated functions that had not been restart protected because they were not deemed critical — including the DELTAH monitor Verb 16 Noun 68. This is why, following the two alarms in P63, the display returned from Noun 68 to Noun 63.

Here a system of restart protection that was primarily motivated by the possibility of hardware glitches synergistically provided a means to shed computational load in response to a software logjam caused by TLOSS. We had devised a real-time control system that under certain conditions was "fault tolerant".

During P64 the situation was different. Added to the regular guidance equations was new processing that provided the capability to redesignate the landing site. With this addition, the essential software by itself left a duty-cycle margin of less than 10%. The alarms kept coming. There were three 1201 and 1202 alarms within 40 seconds. Each time, the software restart flushed the Executive queue but could not shed load.

At MET 102:43:08, forestalling the next alarm, Armstrong switched the autopilot from AUTO to ATT HOLD mode, easing the computational burden, and then entered semi-manual mode P66, where the burden was still lighter. After 2 minutes and 20 seconds spent maneuvering in P66 without alarms, the LM landed.

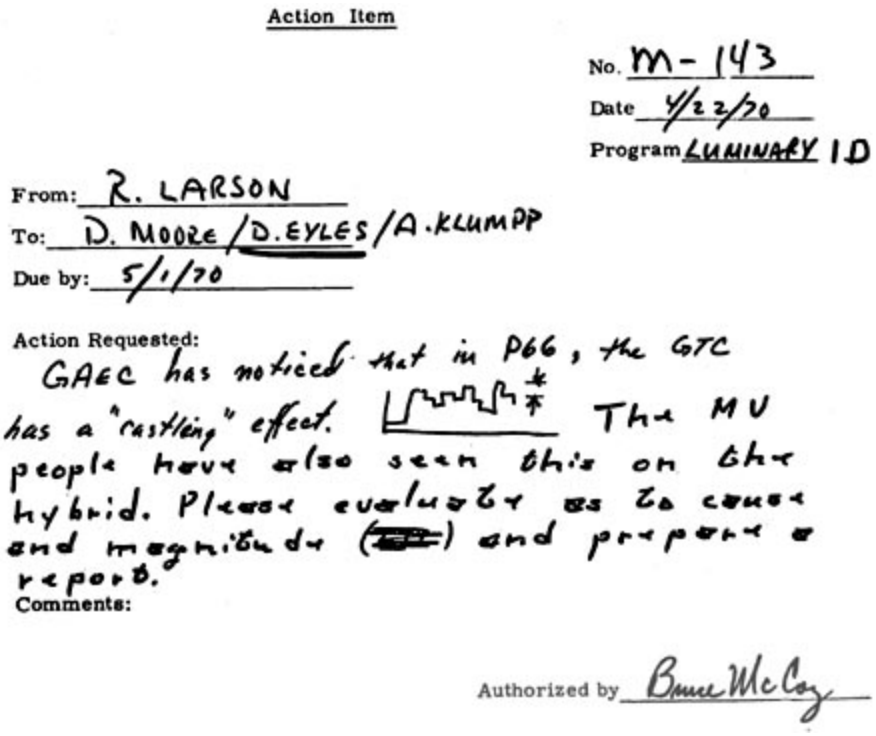
* * *

Five months later Apollo 12 survived a lightning strike during boost and landed on the Moon. Thanks in part to a new noun (69) that we had defined to allow the crew to make position corrections based on ground tracking data during the

braking phase, astronauts Pete Conrad and Alan Bean were able to land the LM within an easy walk of an unmanned Surveyor spacecraft that had landed on the Moon in April, 1967. Apollo 12's pinpoint landing paved the way for landings in more difficult terrain.

It was only after Apollo 12 that we began to understand the other serious problem.

It started when Clint Tillman of Grumman Aerospace (the builder of the Lunar Module) noticed throttle oscillations during simulations of the final descent, on the order of 5% of the DPS thrust. This prompted Tillman to examine telemetry data from Apollo 11 and 12, where he noticed throttle oscillations during the final landing phases that were on the order of 25% peak to peak. (See Figure 12.) This was the period when the Commander was simultaneously using the ROD switch to control altitude-rate and the joystick to maneuver the vehicle. Because plots of this data resembled the battlements and turrets of a castle (or a castellated nut) this problem got to be known as "throttle castellation".



Action Item

No. M-143
Date 4/22/70
Program LUMINARY 1D

From: R. LARSON
To: D. MOORE / D. EYLES / A. KLUMPP
Due by: 5/1/70

Action Requested:
GAEC has noticed that in P66, the GTC has a "castling" effect. [redacted] The MIT people have also seen this on the hybrid. Please evaluate as to cause and magnitude ([redacted]) and prepare a report.
Comments:

Authorized by Bruce McCandless

Figure 11: First Report of Throttle Castellations

Klumpp, in Cambridge, traced the excitation that caused the oscillations to a previously unrecognized phenomenon that came to be called "IMU bob"[18]. The IMU was located above, and about four feet in front of, the center-of-mass of the vehicle. Small but rapid pitch maneuvers, such as those required during final descent, slung the IMU in a way that was interpreted by the accelerometers as a change in the vertical velocity of the *vehicle*. This in turn affected the calculations of altitude-rate, and the estimate of thrust.

But this theory only partially explained the throttle behavior observed in the flight data.

Rocket engines that can be throttled were and still are unusual, but a throttleable engine was a necessity for making a soft landing on the Moon. A fixed-thrust engine and a very simple guidance equation could put a spacecraft *through* a spot on the lunar surface. But to get there right side up, moving slowly, with visibility and the ability to hover while choosing a landing area, required an engne that could balance lunar gravity while varying its thrust as the vehicle's mass decreased, as the vertical component of the thrust vector changed during attitude maneuvers, and as the astronaut requested changes in the descent rate.

The guidance equations determined what acceleration was required, both in magnitude and direction. The autopilot maneuvered the vehicle to satisfy the thrust direction commanded by guidance. It was up to the throttle-control program to control the magnitude. Throttle-control started by computing the LM's mass. Knowing mass, it determined the magnitude of the thrust correction required to change vehicle acceleration from that measured by the accelerometers to that commanded by the guidance equations, converted this to the units used by the throttle assembly (about 2.8 pounds per pulse), and sent it to the hardware.

The accelerometers in the IMU did not really measure acceleration; they merely counted velocity increments since the last reading. Because a throttle change commanded on the previous guidance pass occurred at some time between the accelerometer readings, the measured delta-V did not show the full effect of the most recent adjustment.

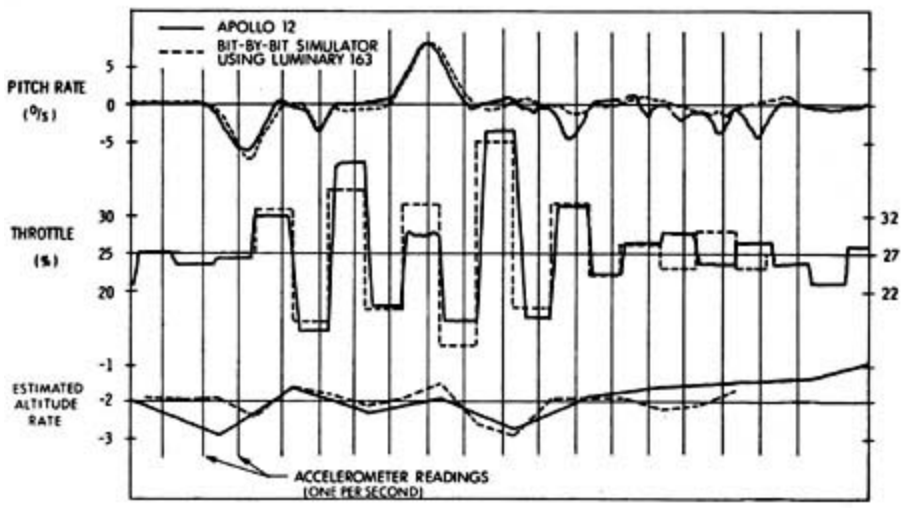


Figure 12: Throttle Excursions During Apollo 12 P66[19]

Throttle control had to compensate for this effect. The amount of compensation depended on when during the guidance period throttle commands were issued, and it also depended upon the rapidity with which the engine followed the throttle command. The applicable ICD stated that the throttle time lag was 0.3 seconds.

It fell to the author to program and test the throttle-control routine. In plots produced by a simulation that accurately modeled the DPS using the time lag of 0.3 seconds, I observed the oscillation that occurred in the actual thrust level after a large throttle change was commanded without compensation for the throttle lag. When I compensated for 0.1 second I saw that the oscillation was reduced. When I compensated for 0.2 seconds the oscillation appeared to be virtually eliminated. There the matter rested. Klumpp remembers me saying, "It's just like medicine, don't give it more compensation than it needs".

Klumpp knew it was *not* "just like medicine", but he never insisted that I program the correct number. Examining his motives 15 years later, Klumpp wrote:

> I thought it was important to nurture self-reliance, to let coworkers' decisions on small matters prevail, even when not optimum. So I withheld my thoughts and let Don's decision stand, at least until he might reconsider it independently[20].

Examining my own motives, I believe that the annoyance I felt toward the compensation terms for cluttering up my throttle logic may have translated into a desire to compensate no more than necessary. Be that as it may, both Apollo 11 and Apollo 12 flew with 0.2 seconds of compensation for a 0.3 second throttle delay.

But now both Klumpp's analysis[21], and an independent report prepared by J. A. Sorensen at Bellcomm[22], concluded that "The oscillatory character of the P66 throttle command was apparently due to the actual value of the descent engine time constant being smaller than that assumed" (Sorensen). Klumpp tracked it down. The performance of the descent engine had been improved, but the ICD was not modified accordingly. The actual time lag for the descent engine was about 0.075 seconds. It turned out we had overcompensated. As a result the throttle was barely stable.

Klumpp's analysis had an even more startling result. It showed that if the software had compensated at 0.3 seconds on Apollo 11, the throttle would have been unstable. The throttle oscillations, instead of settling down, would have become greater. Following throttle-down in P63, or perhaps in P66 under the excitation of IMU bob, the DPS engine would have rapidly oscillated between minimum and maximum thrust. No doubt mission control, quite logically, would have linked the throttle behavior to the 1202 alarms that were occurring for entirely independent reasons.

An abort would have been inevitable. With all modesty, it appears to be the case that if the author had coded the "correct" compensation number in the throttle-control routine, Apollo 11 would not have landed. I invite someone with no personal stake and a grasp of the mathematics to reexamine this theory.

* * *

We fixed IMU bob by removing the velocity changes caused by IMU motion from the acceleration measurements. We corrected the throttle time lag and simulations showed that this indeed fixed the throttle instability. Neither fix was on Apollo 13, but that mission was not able to attempt a lunar landing.

Curiously, a change made *before* the throttle problem came to light, which was on Apollo 13, would have offered a backup if the automatic throttle had failed. A new noun (92) was defined that the crew could select to see the throttle level desired by guidance. Logic that would have terminated automatic guidance if the throttle were (or appeared to be) switched to MANUAL was removed. These changes[23] let the astronaut take control of the throttle during P63 or P64 while guidance continued to command attitude. I do not know whether these difficult procedures were ever practiced.

The problem of the Executive overload alarms was dealt with several times over.

The rendezvous radar mode switch was placed in LGC for ascent. For future missions the descent checklist was changed. Meanwhile we added logic to LUMINARY to check the rendezvous radar mode, and if it was not in LGC, send a signal to zero the rendezvous radar counters[24].

Allan Klumpp studied the Executive problem from another angle. He discovered that under conditions in which TLOSS occurred intermittently, or when the level of computer activity fluctuated in the presence of TLOSS, it was possible for incomplete SERVICER jobs that had been interrupted during the issuance of attitude commands, but had not yet been flushed by a software restart, to be resumed at a later time — with the possibility that inappropriate attitude commands could be issued to the autopilot. In time for Apollo 13 Klumpp devised a fix in which an occasional whole SERVICER job would be dropped to catch up, if necessary.

But for the future, none of these changes provided fundamental relief from the constraint of the fixed, two-second guidance period. A terrain model needed to be added to the landing radar routines to allow landing in difficult terrain. Guidance modifications were waiting in the wings. Where would the time come from?

We developed a concept we called "variable SERVICER", in which the guidance period was allowed to stretch if it needed to. Fears that the two-second interval was built inextricably into the software proved unfounded. It was only necessary to measure the guidance period and use that value explicitly in place of the two seconds that was implicit in a few calculations. We got variable SERVICER working in an offline version of LUMINARY, and demonstrated its immunity to very high levels of TLOSS[25].

Freedom from the two-second straitjacket allowed other ideas to be considered. Astronaut John Young suggested a capability that we called P66 LPD. By now P66 had evolved into an even more flexible program than it was when Armstrong flew it on Apollo 11. One of its new features was that if the crew switched the attitude mode back from ATT HOLD to AUTO, guidance would then control the attitude to null the horizontal velocity. Young's idea was for the LGC to display an LPD angle (as during the visibility phase) that would show the Commander the spot over which the LM would come to hover, if at that instant the autopilot were switched to AUTO[26].

To make P66 LPD accurate, the software had to react instantly when the astronaut switched to AUTO — more quickly than the two-second period, or even the one-second period at which parts of P66 operated, allowed. We coded a version in which a job running every quarter of a second reacted to the change in autopilot mode by immediately issuing attitude and throttle commands, and responded far more quickly and precisely to inputs from the ROD switch as well. In manned simulations run at the LM Mission Simulator (LMS) at Cape Canaveral, with its fabulous terrain models visible in the LM's windows, we showed that this system facilitated very precise landings.

Neither variable SERVICER nor P66 LPD ever flew. NASA had made the decision that Apollo 17 would be the last landing. With so few missions remaining, the software control board made the conservative decision — no major changes to the landing software. By synchronizing the landing radar measurements with the time the accelerometers were read, Robert Covelli gained enough time to squeeze in the terrain model for Apollo 15, 16, and 17.

Apollo 14 brought the author a brief notoriety. The abort switch on the instrument panel was sending a spurious signal that could have spoiled Alan Shepard and Ed Mitchell's landing. I had written the code that monitored this discrete. The workaround simply changed a few registers, first to fool the abort monitor into thinking that an abort was already in progress, and then to clean up afterward so that the landing could continue unaffected. The procedure radioed up and flawlessly executed by the astronauts involved 61 DSKY keystrokes. Perhaps the most interesting part of the Apollo 14 incident has been the number of differing versions that have been offered to history. But Apollo 14 is a story for another day.

In December 1972 I traveled to Cape Canaveral for the launch of Apollo 17. At this moment spaceflight was hip. The writer Tom Wolfe was there with photographer Annie Leibovitz to write the four-part story for Rolling Stone magazine that was the precursor of "The Right Stuff"[27]. It was the only Apollo night launch. The misty Florida sky lit up orange from horizon to horizon as the huge Saturn V ripped downrange on a quarter-mile flame that licked at the end like a blowtorch.

I spent a few days at the LMS testing some procedures that we called "erasable memory programs". These were snippets of code that could be installed in

unused VAC areas to handle certain malfunctions — an idea that was a legacy of the Apollo 14 incident. Then I flew back to Cambridge for the landing itself.

After that came the pleasure of listening in while Gene Cernan and Jack Schmitt, a geologist by training, explored the Moon in the lunar rover, venturing over 3 miles, out of sight of the spacecraft. And that was the last time anyone walked on the Moon.

Figure 13: Some of the People Involved.

Large photo, front row: Vince Megna, "Doc" Charles Stark Draper, the author, Dave Moore, Tony Cook; back row: Phil Felleman, Larry Berman, Allan Klumpp, Bob Werner, Robert Lones, Sam Drake. Small photo, front row: Larry Berman, Peter Volante, the author; back row: Sam Drake, Bruce McCoy. Also involved but not present for either photo were Steve Copps, Romilly Gilbert, Ken Goodwin and Russ Larson.

REFERENCES

[1] Klumpp, A. R.; "Apollo Lunar Descent Guidance"; MIT Charles Stark Draper Laboratory, R-695; June, 1971.
[2] Cherry, G. W.; "E-Guidance — A General Explicit, Optimizing Guidance Law for Rocket-Propelled Spacecraft"; MIT Instrumentation Laboratory, R-456; August, 1964.

[3] Brooks, Courtney G., et al; "Chariots for Apollo, A History of Manned Lunar Spacecraft"; NASA; 1979.
[4] Silver, George; private communication; 2004.
[5] Hall, Eldon C.; "Journey to the Moon: The History of the Apollo Guidance Computer"; AIAA, 1996.
[6] Blair-Smith, Hugh; "Block II Instructions"; MIT Instrumentation Laboratory, AGC4 Memo 9; July 1, 1966.
[7] Muntz, Charles A.; "User's Guide to the Block II AGC/LGC Interpreter"; MIT Instrumentation Laboratory, R-489; April 1965.
[8] Apollo 11 Downlink Data.
[9] "Apollo 11 Technical Crew Debriefing"; NASA, July 31, 1969 [Debriefing].
[10] "Apollo 11 Technical Air-to-Ground Voice Transcription"; NASA, July 1969 [Voice].
[11] Voice.
[12] Debriefing.
[13] "Apollo 11 Mission Report"; NASA, SP-238.
[14] Debriefing.
[15] Debriefing.
[16] Voice.
[17] Klumpp, A.; untitled memo regarding real-time plot for monitoring computer activity; MIT Charles Stark Draper Laboratory, April 9, 1970.
[18] Klumpp, A. and Kalan, G.; "Elimination of Noise and Enhancement of Stability and Dynamic Response of the Apollo LM Rate-of-Descent Program"; MIT Charles Stark Draper Laboratory, E-2543, October 1970 [Noise].
[19] Noise.
[20] Klumpp, Allan; private communication; 1985.
[21] Noise.
[22] Sorensen, J. A.; "Linear Stability Analysis of LM Rate-of-Descent Guidance Equations"; Bellcomm Inc., B70 06074, June 25, 1970.
[23] Tindall, H.W. and Garman, Jack; "Remove check of Auto Throttle Discrete"; LUMINARY 1C Program Change Request (PCR) 285, September 30, 1969.
[24] Eyles, D.; "Prevent RR ECDUs from Stealing LGC Memory Cycles"; LUMINARY 1B PCR 848, July 23, 1969.
[25] Eyles, Don; "Description of Variable Servicer"; MIT Charles Stark Draper Laboratory, Luminary Memo 139, March 3, 1970.
[26] Eyles, Don; "Apollo LM Guidance and Pilot-Assistance During the Final Stage of Lunar Descent"; MIT Charles Stark Draper Laboratory, E-2581; May 1971.
[27] Wolfe, Tom; "Post-Orbital Remorse"; Rolling Stone; January 4, 1973.

DON EYLES HOME PAGE

SEND EMAIL TO DON EYLES