

A DEMONSTRATION OF THE VALUE OF SPACECRAFT COMPUTERS

The Transit Improvement Program (TIP) satellites are navigation satellites equipped with an on-board, general-purpose minicomputer. The ability to reprogram the flight computer in orbit allowed sophisticated alternative techniques to salvage two early TIP missions in the face of serious hardware problems. This article describes how new flight software was implemented after launch to perform some new control functions that achieved a partial mission success on both spacecraft. The experience was a dramatic illustration of the advantages of a central computer on board a complex spacecraft.

INTRODUCTION

The Transit Improvement Program satellites were designed by APL to upgrade the then-existing navigation satellite system. The first three developmental spacecraft (called the TIP series) were built and launched by APL, and the later production versions (called Nova) were built by RCA. Among other improvements and advancements, these satellites were equipped with a general-purpose minicomputer with 64K bytes of memory. Also, each TIP satellite has a hydrazine-fueled orbit adjust and transfer system and an attitude control system that operates in both a gravity-gradient and a spin-stabilized mode. The spacecraft is spin stabilized during the orbit adjust phase and later operates in the gravity-gradient mode as a drag-free satellite.

Although fully reprogrammable on-board computers have become more common, such systems were innovative in 1969 at the starting time of the TIP design, and the lessons learned from that program's experience are more appropriate today than ever before. Some of this experience is presented herein, partially as an interesting historical record and partially as a stimulus to more imaginative uses of such systems. With today's electronic technology, we can extrapolate to the possibility of IBM 3033s and CRAY-1s in orbit. The presence of such devices on board can completely alter the approaches taken to achieve mission goals. Their flexible nature encourages creative and innovative methods and strategies, e.g., achieving fault tolerance by making a spacecraft more "intelligent" and flexible rather than simply more redundant.

This article describes the uses of the TIP/Nova flight computer to overcome some early failures in the spacecraft development that, although later corrected, could have jeopardized the entire program. TIP-II and TIP-III¹ were extremely important to the overall program because they were proof-of-concept and in-orbit verification for all of the new techniques and subsystems

that had been introduced into the improved system. The ability to change the flight software after launch allowed us to overcome some early deployment failures and continue the development to a production status.

The fully deployed TIP or Nova spacecraft is shown in Fig. 1. During the initial orbit adjust phase, the scissors boom is folded, and the hydrazine rocket and tank are attached to the spacecraft. The four solar panels provide a configuration for stable spin about the longitudinal axis (labeled z). Later, after the hydrazine is used up, the boom is extended, with the empty rocket system acting as an end mass for gravity-gradient stabilization.

The solar panels are designed to unfold immediately after the spacecraft achieves parking orbit. When the TIP-II panels failed to erect, the spacecraft was left in a low-power condition and with unfavorable moment-of-inertia ratios for spin stabilization. Without stable spin, the orbit adjustment rocket could not be pointed and controlled as originally planned. Nearly one year later, TIP-III experienced an identical failure. In addition, a boom deployment problem later caused the scissors boom links to break on TIP-II under normal motor-driven deployment.

With hard-wired spacecraft logic, these problems would have precluded the attainment of any of the mission goals or even a modest engineering checkout of the on-board subsystems. However, changes to the flight computer program after launch allowed us to implement a variety of techniques to work around problems and achieve a partial mission success.

We will describe how the flight computer was quickly reprogrammed after launch to

1. Carry out power management to avoid troublesome spacecraft blackouts,
2. Achieve enough spin stability to fire the orbit adjustment thruster,
3. Raise the parking orbit to a workable altitude,

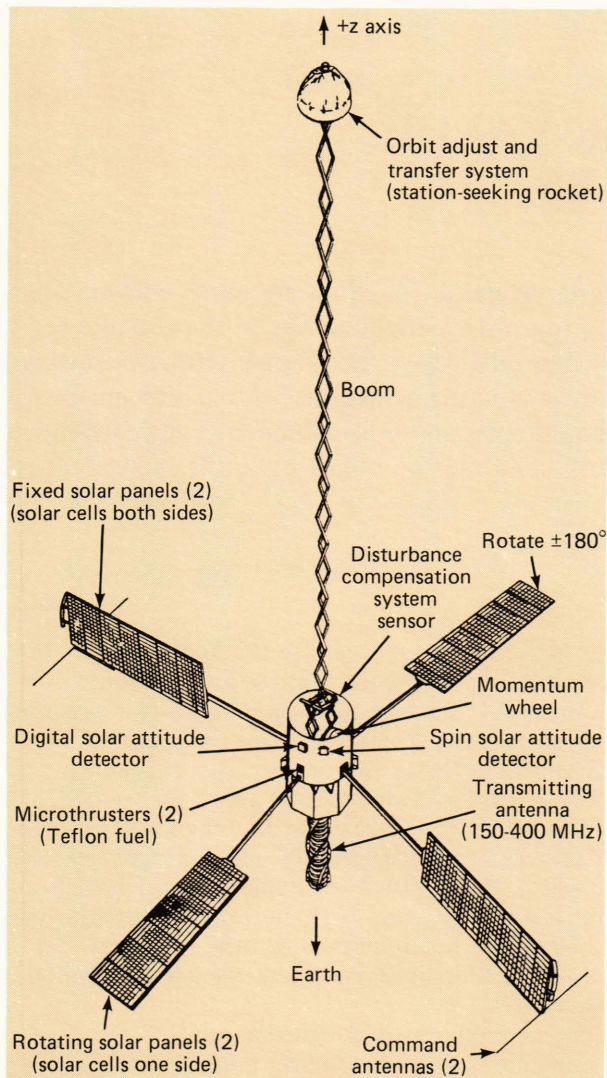


Figure 1—Orbital configuration.

4. Remove a high (45 revolution per minute) tumble rate (an indirect result of one of the failures),
5. Deploy the gravity-gradient boom successfully on TIP-III.

THE FLIGHT COMPUTER AND ITS SOFTWARE

The TIP flight computer is a general-purpose minicomputer that uses specialized input/output logic to perform various spacecraft functions in real time. The computer consists of two redundant central processing units and two magnetic core memories. Each memory provides programmable storage of 32K bytes. The memory cycle time is 4.8 microseconds. There is also a 64-word hard-wired, read-only memory containing a special "bootstrap" loader program for restarting the software in orbit.^{2,3} The complete hardware package has a volume of about 0.2 cubic foot and weighs 14 pounds.

The computer and its ground support system was custom-designed about 1970 for the TIP mission, as

was a complete software operating system that includes an assembler. The flight computer software consists of a system of interrupt-driven real-time programs. These programs perform on-board data management and interact with other hardware subsystems to give the computer far-reaching powers to control and monitor the satellite. The flight computer derives most of its power to perform control functions by virtue of its direct interface to the spacecraft telemetry and command systems. The telemetry system is digital, with 8 bits per channel, 172 channels per frame, and a 4.227-second frame period. The computer/telemetry interface allows the computer to exchange data with the telemetry system under direct software control.

The spacecraft command subsystem contains digital (10 bits per second) logic to perform the remote execution of various types of commands and the loading of the computer memories. Through the computer/command interface, the flight computer has direct access to the front end of the command system. Any command can be issued by the flight software by serially transmitting the command bits through the interface at the required 10 bits per second rate. The length of a relay command bit string requires 2.3 seconds for complete transmission. Any command can be executed at a precisely prescribed future time by allowing the computer to issue the command. This "delayed command" capability results from loading the information for the commands into the computer memory to be processed at a specified time according to the computer-kept clock, which is synchronized to universal time.

The main implications of the input/output interfaces described above are that the computer is limited to a data-sampling rate of 4.2 seconds for any given telemetry channel, and the maximum command rate is one every 2.3 seconds. These constraints became quite important in some of the control functions implemented.

The overall ground system is complicated but very flexible. It allowed us to completely reprogram the flight software after launch, as well as to manage the system in orbit in ways we had not thought of when the software was developed. The backbone of the system is the ground station PDP-11/40, operating through a front-end PDP-8. This computer is used to control all real-time satellite operations and display and for formatting data to be transmitted up to the spacecraft. For certain real-time flight operations, described later, the PDP-11 is interfaced directly to the laboratory central computer.

The main flight computer software is a basic operating system called SYS, which is resident in memory at all times and contains

1. Loading programs that can handle uplink data at 10 or 1000 bits per second,
2. A memory dump program that can read out areas of memory on either a 325 or 1300 bits per second downlink,
3. A status routine that sends computer information to the telemetry system each telemetry frame,

4. A timekeeping routine that keeps a high-precision universal time clock,
5. A time queue program that controls the chronological sequencing of computer events, such as delayed commands.

For details on the complete flight software system, see Refs. 4, 5, and 6.

In addition to SYS, there are many other special programs that are up-loaded into the computer when needed. Two of those programs are worth mentioning because of their importance in the flight operations described later.

Delayed Command Program

Delayed commands are prepared by special card inputs to a PDP-11 program called TIPLOAD. The TIPLOAD processor automatically formats the command bit strings and the appropriate time queue entries for the flight computer. In addition to commands in the time queue, the delayed command program may also be used to send commands when certain events occur on board. This is done in conjunction with the telemetry storage program, TMON, which initiates delayed commands whenever the data in certain telemetry words match prespecified criteria. Since it takes 2.3 seconds to send a relay command, the delayed command program must make sure that the telemetry-triggered delayed commands do not interfere with each other or with the time-ordered commands being controlled by the time queue (or with the commands coming up from the ground, either authorized or unauthorized).

Telemetry Storage Program

TMON is used to sample and store in memory real-time telemetry data. The program expects as inputs a start time (time queue entry), a list of telemetry channels to be monitored or stored, and the rate at which each is to be sampled. TMON allows each telemetry channel to be sampled at its own rate; hence, all channels need not be sampled during the same frame. The program automatically stops storing data when the specified storage area fills up, and the data are later recovered by a memory dump to the ground.

The use of the time queue for delayed commands and the ability to send commands while monitoring telemetry functions proved to be extremely valuable after the TIP failures. In our wildest imagination, we could not have foreseen the use we would make of these programs or the salvation they would provide for the crippled mission.

THE EFFECT OF THE FAILURES

The TIP spacecraft are launched into a polar parking orbit at an altitude of 180 by 400 nautical miles. The orbit adjust and transfer system is then used to change the orbit to a circular one at an altitude of 600 nautical miles. At the same time, the inclination is adjusted to a selected value near 90° to control the nodal precision. An important part of this operation is to

select optimum directions for applying thrust, in order to correct the altitude and inclination simultaneously and minimize the fuel requirements.

The spacecraft is designed to be spin stabilized about its longitudinal symmetry axis (z axis) to provide stable directional control during thruster firing and to compensate for thruster misalignments. To achieve spin, an analog magnetic dipole spin-up system provides continuous torque about the z axis using the earth's magnetic field. Passive nutation dampers on the ends of the solar panels negate the effect of random transverse torques introduced by the spin-up system during this operation. To slew the z axis to a desired firing direction (after spin is achieved), a reversible z dipole coil is aboard to provide precessional torques, using the earth's magnetic field.

After the orbit is adjusted, the remaining hydrazine is vented, and the empty orbit adjust and transfer system tank becomes the end mass on a scissors-type boom for gravity-gradient stabilization (see Fig. 1).

When the solar panels failed to deploy on TIP-II, we were left in the following situation:

1. The spacecraft was generating less than half its normal power,
2. The spacecraft was not stable in spin about the z axis,
3. The nutation dampers were not in the correct position to be effective in damping out torques transverse to the z axis,
4. The 60 pounds of liquid hydrazine at the tip of the spacecraft was an effective mechanism to quickly transfer any spin about the z axis into tumble about the stable transverse axis.

For various reasons, the spacecraft needed to be at an average altitude of at least 400 nautical miles to be able to operate effectively as a navigation satellite. Also, it was necessary to reduce the orbit eccentricity to achieve the gravity-gradient stability required for effective operation of the disturbance compensation system.

As will be described, we managed to achieve this, leaving about half the hydrazine to be vented before erecting the boom. Unfortunately, the venting system was designed for stable z axis spin; with our configuration, a tumble torque was inevitable. We had no way of knowing how bad this would be and were forced to take our chances and vent. The next time we saw the spacecraft, the tumble rate was 45 rpm, and the solar panels had been ripped from their stowed position and forcibly deployed by the centrifugal force. This improved the power situation, but before the boom could be deployed, the tumble motion had to be dissipated. This was done by implementing an interesting digital phase-locked loop and using the spacecraft z coil to work against the earth's magnetic field.

When we finally attempted to run out the scissors boom on TIP-II, the links broke because of an unforeseen problem. Unfortunately, this happened after TIP-III was launched so the problem was not corrected on TIP-III. However, once the problem was

understood, it was possible to work around it on TIP-III by using centrifugal force generated by a tumble motion. We were able to solve the problem by reversing the de-tumble mentioned above.

POST-LAUNCH OPERATIONS

Power Management

The immediate problem after the solar panels failed to deploy on TIP-II was the severely restricted power capability of the spacecraft. The problem was particularly bad when we tried to use the high-voltage magnetic system for spin-up and precession, and it was exacerbated by the fact that we launched into a minimum-sun orbit.

To protect the battery, the power system is equipped with a low-voltage sensing switch that shuts down the main power bus automatically when the battery voltage reaches 13.8 volts. Although this is not a disaster, it was extremely inconvenient when it occurred because the spacecraft had to be restored to its previous state through a complicated series of commands. The spacecraft oscillator shifted frequency dramatically during the first few minutes of warm-up, making it very hard to keep the receivers locked on the signal. Also, the flight computer system had to be reloaded and restarted and the universal time clock reset each time the low-voltage sensing switch tripped. With passes only about 8 minutes long, this led to a hectic operation, with the spacecraft frequently rising silent and not responding immediately to the recovery commands. The spacecraft was "lost" on numerous occasions.

To solve the problem, we modified the telemetry storage program to monitor the battery voltage channel once per minute. When the voltage fell to a threshold level, the program used delayed commands to throw off the magnetic system power. The threshold was usually set to 14.5 volts. The magnetic system draws about 60 to 70 watts, and removing this load when the battery reached 14.5 volts was generally sufficient to prevent the low-voltage sensing switch from tripping.

Even though the battery voltage was monitored carefully, the spacecraft systems had to be duty-cycled to prevent power drain. The time queue feature of the flight software was used to turn systems off and on at scheduled times to effect the duty cycling. However, this proved to be a great deal of work—punching cards, preparing updated time queue files with the TIPLOAD program, and injecting the data into the spacecraft memory. When the time queue software was developed, its primary purpose was to fire the orbit adjustment rocket out of view of a ground station, and it was not designed to handle a large number of delayed commands easily (such as 100 per day). The operations team quickly found most of their time devoted to trying to keep the flight computer fed with duty-cycling data.

It proved rather easy to reduce the workload dramatically. We quickly recognized that nearly all of the chronological delayed commands for duty cycling were

periodic in time. They were generally tied to the geometry (for example, some systems were turned on and off over the equator or the poles, some were turned off in the earth's shadow, etc.). The answer was to make the time queue cycle itself automatically.

A program called CYCLE was quickly written to cause the time queue actions to repeat at a commanded periodicity. This is done by calling CYCLE from the last time queue entry in the list. The CYCLE program sets appropriate pointers to the starting conditions and then restores the original time queue list, adding the input period to the time for each entry. This causes the time queue actions to be periodically repeated until the process is stopped by ground command.

At first, some people were nervous about relinquishing control to the computer and allowing the spacecraft to operate autonomously. However, this simple fix worked beautifully. Even the simplest operation would have been difficult to carry out without CYCLE; as will be seen, it would have been next to impossible to carry out the more complicated operations we eventually undertook.

Spin-Up Operations

The early operations with TIP-II were involved with attempts to spin the spacecraft about the z axis. There were two reasons for these attempts: (a) we hoped that sufficient spin could generate enough centrifugal force to forcibly deploy the solar panels, and (b) stable spin was required to fire the orbit adjustment rocket.

The spin-up system is a feedback control system that uses the earth's magnetic field to torque the spacecraft. Two orthogonal coils (x and y) provide a dipole moment in the spacecraft to supply the torque. The earth's field is continuously sensed, and the x and y coil currents are automatically phased to keep the resultant dipole orthogonal to the component of the earth's field that lies in the x-y plane. (The geometry is shown in Fig. 2.) This results in a torque about z that is always in the same sense, along with random torques trans-

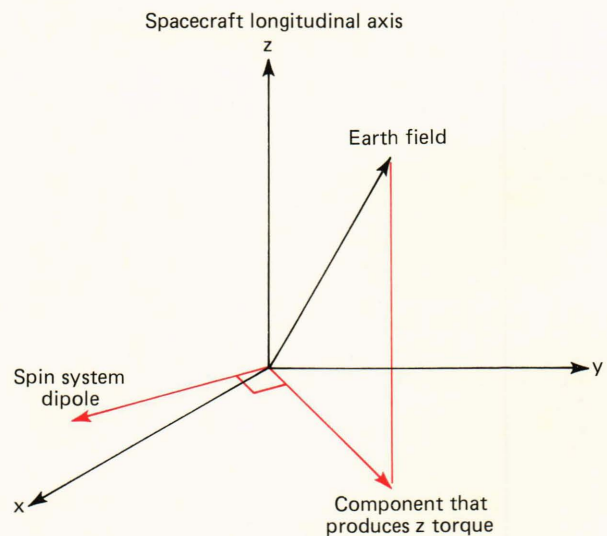


Figure 2—Magnetic spin-up geometry.

verse to the z axis. With a stable spin configuration, the spin about z will gradually build up to the desired level, while passive dampers remove the nutation induced by the transverse torques.

With our unstable configuration, it was a different story. When the spin system was turned on, the liquid hydrazine was sloshed by the transverse torques and acted as an effective mechanism to quickly transfer any spin into a tumble motion about the stable transverse axis. Initial attempts to achieve any spin above $\frac{1}{4}$ rpm were unsuccessful. To solve the problem, we modified the flight software to control the times that the spin system was on in order to minimize the transverse torques.

The spacecraft are equipped with three orthogonal magnetometers measuring the body-fixed x, y, and z components of the earth's field. We modified the telemetry storage program to sample the channel for the z component every frame and use it as the criterion for turning on the spin system. In each telemetry frame, the program determined if the absolute value of the z magnetometer reading was less than a preselected threshold. When that condition was satisfied, the program used a delayed command to turn on the spin system and conversely turned the system off when the z component was out of range. We had to include additional logic to prevent the program from continuously sending commands once the system was in the correct state. By allowing the system to be on only when the z component was near zero, the transverse torques were nearly eliminated.

It happens that there are two relays in series to control the magnetic system power. This was quite useful, since we could let one relay (normally on) be controlled by the battery voltage and the other be controlled by the z magnetometer reading. Thus we could allow both monitoring functions to operate in parallel, to protect against the low-voltage sensing switch tripping during spin-up. This was about the only piece of good luck that we had in the operation.

It took about two days to generate the spin-up program, and, after some trial and error, we settled on a z magnetometer threshold value of about 10% of the full-scale field reading. The program helped quite a bit. We were able to achieve spin rates up to 1 rpm, but then the spacecraft would gradually build up nutations and transfer to tumble with a time constant of about one orbit (90 minutes). This was still not enough spin rate or stability to fire the orbital adjustment rocket.

At this point, the idea arose of using the spacecraft z coil as a device for actively damping the nutational motion to maintain more stable spin. (That coil is normally used to provide precessional torques for pointing the rocket nozzle.) It can be switched by relay command to either the plus (dipole along +z) or the minus (dipole along -z) state.

For clean spin without nutation, the spacecraft magnetometers record a very distinctive pattern from the earth's field vector. Because the local earth field is varying rather slowly due to the orbital motion, the

attitude dynamics dominate the magnetometer's variations. The x and y coils record a sinusoidal variation 90° out of phase, with the period equal to the spin period, and the z magnetometer records a nearly constant value. As nutation (coning) builds up, the z reading begins to show an oscillation with a period equal to the nutation or coning period.

The idea behind the damping program (DAMP) was to let the computer sense the derivative of the z magnetometer reading and then set the polarity of the z coil to produce a transverse torque (using the earth's field) that opposed the derivative. Again we could modify the telemetry storage program, this time to control the z coil polarity based on the monitored z magnetometer reading. The logic was simple:

1. For each frame, compute the difference between the current z magnetometer reading and the previous frame's reading, as an estimate of the derivative.
2. When the difference changes sign from plus to minus, send a delayed command to set the z coil polarity to a minus dipole.
3. When the difference changes sign from minus to plus, send the command to change the z coil to a plus dipole.

We also had to include additional logic to correct the telemetry reading for the z coil effects. The z coil strength is of the same order as the earth's field at these spacecraft altitudes; therefore, it has a large effect on the z magnetometer reading.

Note also that the DAMP program can be used to de-tumble the spacecraft when it is in stable tumble about a transverse axis. In this case, the z magnetometer records a sinusoidal variation. The effect of the program is to adjust the z coil polarity continuously in order to produce a torque opposite to the motion. This proved to be quite handy in reducing the time required to dissipate tumble motion.

As soon as we had the DAMP program written, we found that it worked well. But to be effective, we needed to alternate continuously between the spin-up and damping modes. However, the magnetic system is designed so that either the spin system or the z coil can be in use, but not both modes at the same time. This meant we needed a way of dynamically switching between the SPINUP logic and the DAMP logic in the telemetry monitoring program.

The idea of changing the program logic dynamically while the flight software was actively running was a new feature for our software system. It was not hard to implement, and it provided a very powerful capability.

We compiled both sets of logic (SPINUP and DAMP) into the telemetry monitoring program and added a simple program switch to select which path would be used to evaluate the z magnetometer reading. We then wrote a program (ALTPRO) that was called by a time queue entry to change the switch at a scheduled time. ALTPRO was written to be rather general, driven by an input list of memory addresses

and their corresponding new contents. Each time the program is called at a scheduled time, it works on the next entry in the list, loading the new contents into the specified addresses.

To accomplish a spin-up, then, the scenario was as follows. At a prescheduled time, the night before a pass, with the spacecraft essentially motionless in attitude, the computer would select the magnetic system for spin-up, turn on the telemetry system, and activate the SPINUP logic. Near each equator crossing, the computer would select the z coil by delayed command and switch to the DAMP logic using the ALT-PRO program. After several minutes of damping, the spin system would be reselected and the logic switched back to the SPINUP program. The entire set of time queue entries was then cycled with the orbital period by the CYCLE program for continuous operation. Of course, during the complete scenario, the battery voltage was monitored to prevent the low-voltage sensing switch from tripping. If the battery monitoring program did shut down the magnetic system, the time queue cycle would turn it back on at the beginning of the next cycle.

By starting with a fully charged battery, the above scenario achieved spin rates up to 4 rpm, which was enough to be able to fire the orbit adjustment rocket. However, we still did not have directional control over the spin axis. If we left the spacecraft alone after achieving 3 to 4 rpm, it would maintain its spin stability reasonably well for about one orbit (90 minutes). After that, it would begin coning, and the coning would quickly degenerate into tumble. As soon as we tried to precess the spin axis with the z coil, the induced nutations would make the motion unstable.

We tried alternately precessing and damping, but we rapidly ran out of battery power.⁷ There was no way to get directional control, and without it the orbit adjustment looked bleak. The solution to the problem required ingenuity, hard work, and a willingness to make decisions under severe time pressure.

Firing the Orbit Adjust Rocket

Selection of the time and inertial direction of the firing of the orbit adjust rocket is based on a solution to the problem of minimizing the expended fuel for the required orbit change. Our normal procedure is to solve for the true anomaly at fire and the direction of thrust, from the current orbit and desired orbit parameters. This gives us three degrees of freedom to solve for each firing and thereby optimally correct the semimajor axis, the eccentricity, and the inclination. Without directional control, we could not hope to use this scheme.

However, after some thought, we came up with an idea that we hoped might work. Each time we ran the spin-up scenario we found a distinctly different inertial attitude, and although we could not change it, the spin axis direction remained gyroscopically stabilized for about one orbit. We could make use of this fact in the following way.

Instead of solving for a time and direction of thrust from the orbit parameters, we could accept the direction we had and solve for the optimum time to fire, given that attitude and the orbit. We could then obtain a measure of how effective the thrust would be by comparing the resulting orbit changes to those we would have achieved if we could have chosen the direction. If this measure was reasonably high, we would fire the rocket. Otherwise, we would de-tumble the spacecraft and try again. With the spacecraft coming up in a random attitude each time we spun it up, we would sometimes be lucky and obtain a favorable attitude.

Several problems needed to be solved before the above scheme could be implemented. First, the calculations for the optimum firing point based on the current attitude had to be done in real time during a single pass because the spacecraft would not spin stably longer than about one orbit. Thus, we needed to begin an automatic spin-up scenario four or five hours before our pass so that the spacecraft rose spin-stabilized at 3 to 4 rpm. Then in real time during the pass we would: (a) determine the inertial attitude of the spin axis, (b) calculate from this and the current orbit the optimum time to fire in the next 90 minutes, (c) make the decision whether the thrust would be effective enough, and (d) inject the appropriate time queue and delayed command data into the spacecraft to control the firing at the specified time. We had about 8 to 10 minutes during the pass to accomplish the above operation.

Luckily, the first requirement was already satisfied by an existing capability. The TIP attitude determination software had been designed so that it could be run in real time during the satellite passes. We begin by describing this system.⁸

The attitude calculation (and the thrust calculation as well) is too complex to be handled by the PDP-11 ground station computer, which is fully occupied during the pass in managing the satellite data link. The attitude calculation is done on APL's IBM 3033 computer, which is connected by telephone data link to the PDP-11. The telemetry attitude data are fed in real time to the IBM 3033, where the calculation is done in an interactive session. The system is shown schematically in Fig. 3. Note that we had the capability to operate remotely through a station in Hawaii, as well as through the APL ground station. In this setup, the PDP-11 acts as a "log-on" terminal for the IBM 3033, controlling a second interactive session that receives the raw telemetry data and passes them through a shared disk file to the attitude/thrust time-sharing option session.

To use the system for our scheme, we had to add to the attitude determination software the extra program to handle the thrust calculation described as item b above. The required equations are developed and discussed in the first boxed insert. This turned out to be a nontrivial program. The software was run interactively, and the information about the firing was dis-

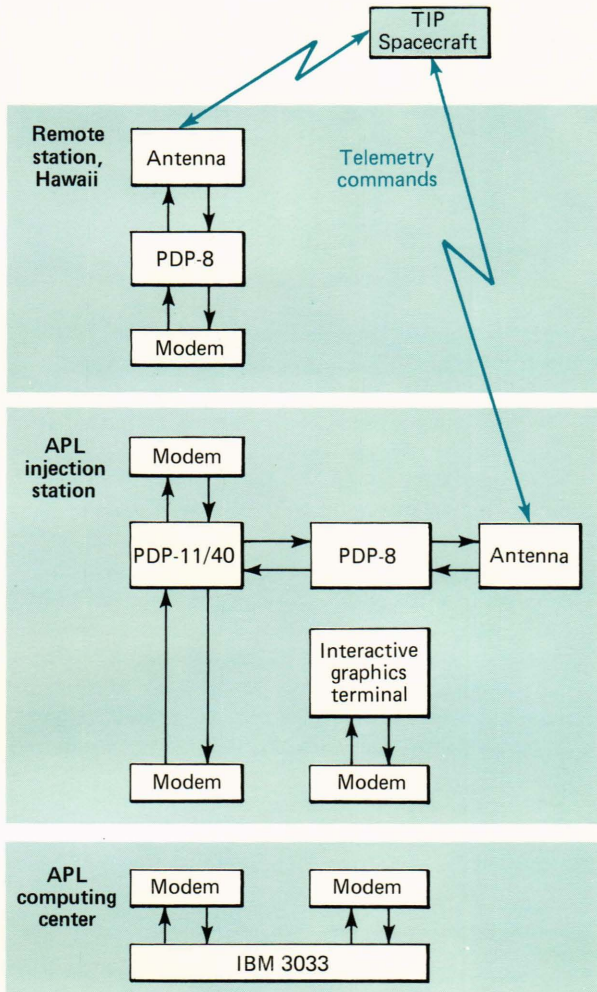


Figure 3—Real-time computing system used for orbit adjust rocket firings.

played on a graphics terminal in real time. A sample of the output display is shown in Fig. 4.

From the program display, the decision was made whether or not to fire. We generally had three or four minutes to run the program and make the decision. The decision was based on many factors, not the least of which was whether or not we believed the answers we were getting from the software.

At times the decision was difficult. For example, in the case where the optimum firing time was nearly one full orbit after set, this meant we had just passed the optimum point. The questions then became: Should we wait for nearly an hour and a half to get back to the optimum point and risk the inevitable nutation buildup? Or should we fire immediately during the pass while we had good stability and accept somewhat less than optimum geometry? This type of decision had to be based on numerous factors such as the current spin rate and nutation angle, how far past the optimum point we were, and how “good” the thrust would be if we waited.

There was one final problem to be solved before the entire scheme could be made to work. This involved

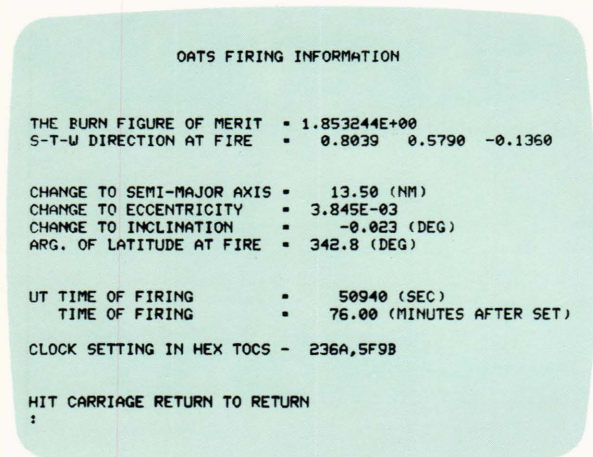


Figure 4—Real-time cathode ray tube display for orbit adjust rocket firing.

item d above—transmitting the firing data back to the satellite computer in real time. As mentioned previously, the time queue and delayed command data are prepared and formatted for transmission by the PDP-11 program called TIPLOAD. The problem was that TIPLOAD was a prepass utility, not designed to be run in real time during a pass. And even if it could be run in real time, the PDP-11 was completely taken up during the pass by the program handling the real-time data links. The TIPLOAD function is a complex task, and there seemed no way to get in the time queue data to control the firing.

The solution was beautifully simple. We made all of the times in the calculation and in the time queue be *relative* to the satellite set time for the pass. Then we simply controlled the firing time by setting the satellite clock to a dummy value to make the rocket fire at the correct amount of time after set. The clock was already designed to be easily set in real time by simple keyboard type-in. The time queue for firing could be formatted as a fixed set of times relative to zero and prepared for injection in a single TIPLOAD run.

The actual rocket firings were quite complicated. A successful operation required that all of the computers involved, as well as the telephone data links, be operable for the pass. We had many operations scrubbed because of computers going down or because of simple human error caused by the time pressures. A synopsis of the entire operation is as follows:

1. On the day prior to the pass, transmit a spin-up scenario into the flight computer, scheduled to begin five hours before the pass.
2. Just before the pass, bring up all computers, establish the telephone data links in full duplex, and initialize interactive sessions on the central computer.
3. As soon as the satellite rises, send commands to begin transmitting telemetry attitude data. At this point, we have less than eight minutes to complete the operations.

FINDING THE FIRING POINT FOR AN ORBIT ADJUST ROCKET, GIVEN AN ATTITUDE

We define the *S*, *T*, and *W* directions in inertial space as follows: *S* is in the direction of the radius vector to the spacecraft, *T* is normal to *S* in the orbit plane in the direction of motion, and *W* is normal to the orbit plane along the angular momentum vector.

The attitude of the spacecraft when spinning about its longitudinal axis is given in terms of the right ascension, α , and declination, δ , of the spin vector in the direction of thrust. α is measured from the equinox, and δ is measured northward from the equator. Hence, the unit vector in the thrust direction is

$$\mathbf{F} = \begin{pmatrix} \cos \delta \cos \alpha \\ \cos \delta \sin \alpha \\ \sin \delta \end{pmatrix} \quad (1)$$

The *S*, *T*, and *W* components of \mathbf{F} are related to \mathbf{F} by

$$\begin{pmatrix} S \\ T \\ W \end{pmatrix} = R_z(\beta) R_x(i) R_z(\Omega) \mathbf{F} \quad , \quad (2)$$

where $R_a(\theta)$ is a rotation of angle θ about the *a* axis, β is the argument of latitude, *i* is the orbit inclination, and Ω is the orbit ascending node.

We can define components *A*, *B*, and *C* as

$$\begin{pmatrix} A \\ B \\ C \end{pmatrix} = R_x(i) R_z(\Omega) \mathbf{F} \quad , \quad (3)$$

so that

$$\begin{pmatrix} S \\ T \\ W \end{pmatrix} = \begin{pmatrix} \cos \beta & \sin \beta & 0 \\ -\sin \beta & \cos \beta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} A \\ B \\ C \end{pmatrix} \quad . \quad (4)$$

Multiplying out the matrix equations 1 and 3 results in

$$\begin{aligned} A &= \cos \delta \cos \alpha \cos \Omega + \cos \delta \sin \alpha \sin \Omega \\ B &= -\cos \delta \cos \alpha \sin \Omega \cos i \\ &\quad + \cos \delta \sin \alpha \cos \Omega \cos i + \sin \delta \sin i \\ C &= \cos \delta \cos \alpha \sin \Omega \sin i \\ &\quad - \cos \delta \sin \alpha \cos \Omega \sin i + \sin \delta \cos i . \end{aligned}$$

The planetary equations to zeroth order in eccentricity can be integrated assuming an impulsive thrust with direction components *S*, *T*, and *W*. Using Eq. 4, the integrated equations can be written in terms of *A*, *B*, and *C* as

$$\begin{aligned} \Delta a &= \frac{2}{n} (-A \sin \beta + B \cos \beta) F \Delta t \\ \Delta e &= \frac{F \Delta t}{na} (A \cos \beta \sin f + B \sin \beta \sin f \\ &\quad + 2\beta \cos \beta \cos f - 2A \sin \beta \cos f) \\ \Delta i &= \frac{F \Delta t}{na} (\cos \beta) C \quad , \quad (5) \end{aligned}$$

where *F* is the the magnitude of the thrust and *f* is the the true anomaly. These give the changes in the orbital elements (*a*, *e*, and *i*) for a firing of duration Δt . The thrust magnitude is known a priori from the fuel tank pressure, and the thrust duration is selected to be of significant length but not so long that the impulsive nature of the thrust is destroyed. Generally, this is four to eight minutes.

The optimum changes in the orbit elements (Δa_T , Δe_T , Δi_T) can be calculated as if the same duration thrust were to be made but with freedom to choose the direction (α , δ) as well as the true anomaly. The equations for the Δa_T , Δe_T , and Δi_T are not given here, but the results are such that the ratios $\Delta a_T/\Delta e_T$ and $\Delta a_T/\Delta i_T$ are maintained constant from firing to firing.

The program to determine the time to fire loops through the true anomaly in 1° increments completely around the orbit. At each position, the changes Δa , Δe , and Δi are computed and a figure of merit (*FM*) constructed:

$$FM = \sqrt{W_1 \left(\frac{\Delta a}{\Delta a_T} - 1 \right)^2 + W_2 \left(\frac{\Delta e}{\Delta e_T} - 1 \right)^2 + W_3 \left(\frac{\Delta i}{\Delta i_T} - 1 \right)^2}$$

where the weights W_1 , W_2 , and W_3 may be input. If the orbit changes were all equal to the optimum changes, *FM* would be zero. The program determines the true anomaly that minimizes the *FM* and repeats the process for a new set of input weights on operator command. Thus the results could be obtained for various cases and compared before a choice was made.

An example of the cases examined was to set $W_3 = 0$ and $W_1 = W_2 = 1$ in order to remove any constraint on changing the inclination and then to do the best we could at raising the orbit and circularizing. The idea here was that if we could get a really favorable along-track thrust today at the expense of a small inclination change in the wrong direction, we could probably make up the inclination change on the next thrust.

Once the optimum true anomaly was selected, the program then calculated from the orbit geometry the length of time after set that would center the firing on that position. This was then converted to a "dummy" setting for the satellite clock and displayed for the spacecraft controller to use if the decision was made to fire.

4. Record attitude data on the IBM 3033 for about three minutes and then begin the attitude and orbit computations.
5. While the orbit computation is being carried out, inject the "relative-time" delayed command and time queue data for firing into the flight computer.
6. Examine the displayed results and make the decisions about the firings.
7. Either fire immediately, set the satellite clock to the correct dummy value for a delayed firing, or abort.
8. On the next pass, set up the computer to begin a two day de-tumble operation using the DAMP program.

The scheme worked well, although it was a trying experience. (On one harrowing pass, we actually fired the rocket backwards, but all other firings were successful.) We were able to average about two 4-minute firings a week for a month or so, using up nearly half the fuel. But things then began to go badly. We had always felt that a half-empty fuel tank would cause worse stability problems than a full one simply because there would be more sloshing around of the hydrazine. Sure enough, this began to happen; worse yet, the spacecraft began to come up spinning, consistently oriented normal to the orbit plane. We attributed this to spin-orbit coupling that became worse with the increased damping of a half-empty tank. This geometry was unfavorable because we were most interested in raising the altitude, which requires an along-track thrust. After trying without success for several weeks, we realized that a new idea was needed. The idea was not long in coming, and once again the on-board computer saved the day.

The Tumble-Thrust Program

The new idea for firing the orbit adjust rocket was a radical departure from our previous method — we abandoned the idea of trying to maintain any attitude control during the firing. Instead, we decided to let the spacecraft tumble, because that is what it wanted to do, and fire the rocket in short bursts when it happened to be pointing in the right direction.

Thus, we could let the flight computer continuously determine the spacecraft attitude and then quickly fire the rocket by delayed command when a good opportunity arose. Attitude determination programs are nontrivial, and this would have been an extremely difficult program to write had it not been for some simplifying circumstances.

At that time, we were getting close to the orbit we wanted; we needed only to raise the perigee altitude. Thus we were willing to do all our firings in the along-track direction in the vicinity of apogee (within 30° or so in true anomaly). That would raise the orbit without increasing the eccentricity and without affecting the inclination. It turns out for a near-polar orbit that there is a simple relationship between the magnetic field in the equatorial regions and the along-track direction. This geometry is illustrated in Fig. 5.

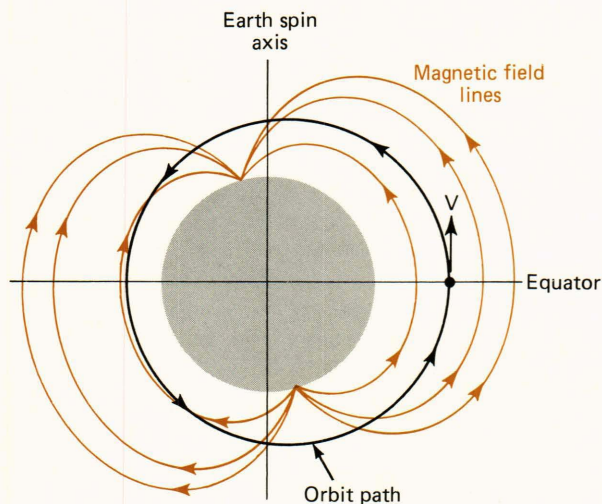


Figure 5—TIP orbit geometry for tumble-thrust program.

It can be seen from Fig. 5 that the field lines are roughly parallel to the flight path in the regions near the equator so that, when the spacecraft is aligned with the field, it is aligned with the velocity vector. Hence the approximate determination of along-track orientation in these regions becomes trivial using the sampled magnetometers.

We had observed during our prior operations that, when tumbling, the spacecraft angular momentum vector tended to align itself normal to the orbit plane. Apparently there was a strong spin-orbit coupling caused by the perturbing torques. In this case, the tumble motion in the orbit plane continuously carries the longitudinal axis through the spacecraft velocity vector, giving ample along-track firing opportunities.

To control the firings, we developed a program called THRUST that was another special version of the telemetry monitoring program. This program continuously monitored all three magnetometer channels and determined when the spacecraft was aligned along-track by the following test:

$$\begin{aligned} |M_x| &< C_1 \\ |M_y| &< C_2 \\ KM_z &< 0, \end{aligned}$$

where M_x , M_y , and M_z are the orthogonal body-fixed magnetic field readings, C_1 and C_2 are selected thresholds, and K equals $+1$ for a north-going geometry or -1 for a south-going geometry. The test on M_z is to establish that the thrust direction will be parallel to the velocity vector, rather than antiparallel. When the tests were satisfied, the program would immediately issue delayed commands to fire the rocket for 2 seconds and then rearm for another firing. The CYCLE program was used to activate the THRUST logic periodically for about 10 minutes near each equator crossing closest to apogee.

The program worked best at a low tumble rate because of the telemetry sampling rate and the 2.3-second time required to issue the firing command. For exam-

ple, if the tumble rate were 1 rpm, the thrust axis would move about 14° in the time needed to issue the command and another 12° during the 2-second burn. This would amount to an effective 20° pointing error for the thrust. Also, at 1 rpm the spacecraft would move about 26° between telemetry samples at the 4.2-second frame rate. Thus we had to open C_1 and C_2 to a reasonably high value (about 20% of full scale) to ensure an angular window large enough to avoid missing opportunities due to the telemetry sampling rate.

We needed only approximate along-track orientation ($\pm 45^\circ$) on each firing to do the job, relying on the off-track components to cancel from firing to firing. The telemetry sampling times tended to fall randomly in the angular window established by C_1 and C_2 ; this helped average the off-track components. However, the command time lag caused biases that were not too serious since the off-track components tended to be radial thrusts that only slightly affected the eccentricity.

Because of the above considerations, about 1 rpm was the practical limit in tumble rate for the program to work effectively. We started the spacecraft tumbling very slowly and found that the firings themselves caused the tumble rate to increase continuously because of the small displacement of the thruster axis with the spacecraft center of mass. The rate increased about 1 rpm for each minute of firing. This meant that the scenario had to switch periodically to the DAMP program to de-tumble and prevent the tumble rate from building up.

Once the scenario was devised, the process worked well. We simply sat back while the flight computer continuously and autonomously pushed the altitude up for a week or so. (The people at the Naval Space Surveillance System, who were determining the orbit, were quite startled when the period began creeping up orbit by orbit. Their software could not handle that case.) From an initial parking orbit of about 180-nautical-mile perigee by 380-nautical-mile apogee, we finally achieved an orbit of about 320 by 450 nautical miles.

At that point, it was important to vent the remaining hydrazine so that the spacecraft could be put into the gravity-gradient mode. The next spacecraft (TIP-III) was scheduled for launch, and we wanted to continue the engineering checkout so that other potential problems would be uncovered before launch. The venting system was designed to release the hydrazine to the side with the spacecraft spinning about the z axis. Since the spacecraft was unstable in spin, a large tumble motion resulted from the venting operation.

The tumble motion was much faster than we had hoped. Two of the four solar panels were wrenched from their stowed position by the centrifugal force, and the spacecraft ended up tumbling at 45 rpm. This was being dissipated at the rate of about 0.2 rpm per day because of the spacecraft magnetic hysteresis. The gravity-gradient boom could not be erected until the spacecraft was stationary, and we needed to do that in less than three months to lead the upcoming TIP-

III launch sufficiently. We were in an obvious time bind and needed to use the z coil to help dissipate the tumble motion. The program developed for this was the most complicated of all the special post-launch programs.

A Digital Phase-Locked Loop for De-Tumble

The DAMP program, which switched the z coil to produce a de-tumble torque, begins to lose effectiveness over about 1.5 rpm. At 2 rpm it does not work very well, and at 45 rpm it is useless. Some simple arithmetic shows the problem. It takes two telemetry frames to sense the derivative in the z magnetometer and then another 2.3 seconds to send the command to switch polarity. The resulting average time delay is 6.6 seconds,⁹ and at 2 rpm this amounts to 79° in the phase of the tumble motion. At 45 rpm, the period of the motion is 1.3 seconds so that even continuous commanding cannot switch the z coil polarity once per cycle.

A program was needed that could determine the phase of the tumble motion well enough to anticipate the peaks and troughs in the z magnetometer reading. Then the command strings could be started with precisely the lead time needed to have the relay switch at the right instant. For motions faster than 25 rpm where the period is less than the 2.3-second command time, the polarity could be switched in phase with the motion every other cycle. Switching every other cycle makes the damping only 50% as effective, but it was still an improvement over the damping provided by magnetic hysteresis.

We decided to implement a digital phase-locked loop in the flight computer to lock onto the phase of the tumble motion and control the z coil switching. There were some nontrivial problems to overcome in this implementation. First, we were performing the computations on a computer that had no floating-point arithmetic capability (either hardware or software) and also had no hardware divide capability.¹⁰ Consequently, we had no programs for generating the trigonometric functions needed to construct (digitally) the local oscillator signal inside the computer. To work around this difficulty, we decided to lock a sawtooth function onto the magnetometer signal rather than a sine wave. The unit amplitude, shown in Fig. 6, can be generated from simple fixed-point operations:

At any time t , the sawtooth value, $f(t)$, is given by

$$f(t) = 4\omega_j(t - T_i) - 1 \quad \text{when } \omega_j(t - T_i) < 0.5,$$

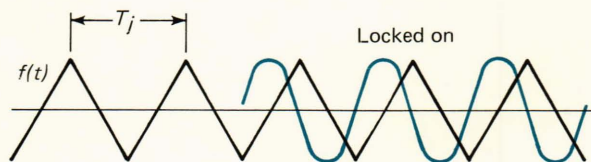
$$f(t) = 3 - 4\omega_j(t - T_i) \quad \text{when } \omega_j(t - T_i) > 0.5,$$


Figure 6—Sawtooth function for phase-locked loop.

where ω_j is the current sawtooth frequency ($1/T_j$) and T_i is the time of the last sawtooth trough.

After estimating the amount of work involved in developing a software floating-point package, we decided that the phase-locked loop could be written in fixed-point arithmetic using the sawtooth function. The calculations had to be properly scaled, and double and triple word precision was used in some places where needed.

The calculations were designed to avoid division wherever possible. In two places it was unavoidable, so we wrote a software division algorithm using an iteration method for the inverse. This consists of iterating the following equation:

$$Y_{j+1} = Y_j (2 - XY_j) \quad \text{from } Y_0 = 1,$$

where X is the number for which we wish to obtain the inverse, and Y is the estimated inverse. The method converges for all positive $X < 2$, and we ensured convergence by proper scaling.

The error signal for the phase-locked loop was the product of the sampled signal (the z magnetometer reading) and the sampled voltage-controlled oscillator output (the internally generated sawtooth function). The filtered error signal then drove the sawtooth in phase, frequency, and frequency drift.

The main problem to be overcome was the aliasing problem caused by the low sampling rate of the telemetry system. Our sampling frequency was 0.236 hertz, and we were trying to stay locked to a signal (the z magnetometer reading) whose frequency was in the range 0 to 0.75 hertz. As the tumble rate decreased, the signal frequency passed through the multiples of the sampling frequency causing singularities where the error signal was driven to a constant value. We avoided this problem by making the loop second order with variable gain and then using the estimate of frequency drift to "flywheel" through the singular points. As the frequency approached a multiple or half multiple of 0.236 hertz, the gains were reduced to zero so that the sawtooth was running open-loop. As the frequency passed through the singular point, the gains were gradually reestablished to their closed-loop values.

We were aided in the implementation by knowing rather accurately the frequency drift (i.e., the tumble rate change) when the z magnet was both off and on. Those rates were determined experimentally using the DAMP program at low tumble rates. It can be shown from simple mechanics that the effect of the z coil on the tumble frequency is independent of the frequency. We had determined the effect to be 0.09 rpm per 10 minutes of z coil operation near the earth's equator. Thus we could switch to the appropriate drift value as the z coil was turned on or off and thereby avoid introducing large transients into the loop. This meant we could duty cycle the z coil as necessary and still maintain accurate phase lock.

We could also determine to within about 0.1 rpm the frequency of the tumble motion by independent analog means. Because the tumble motion rotated the

spacecraft antenna, the ground receiver automatic gain control voltages could be monitored to determine the tumble rate. That method was used to give the program starting values for the frequency when needed.

We provided an automatic gain control feature to compensate for the varying amplitude of the z magnetometer signal caused by the change in the earth field strength with orbital motion. The field amplitude changes by a factor of three within one orbital revolution. That control was accomplished by filtering the absolute value of the signal and using it to normalize the magnetometer readings to unit amplitude before generating the error signal. The telemetry readings also had to be corrected for the effects of the z coil dipole.

The details of the loop are given in the second boxed insert. We determined the appropriate gains and filter time constants by making a computer simulation of the complete process and adjusting experimentally until we could make it work through the range of frequencies from 45 to 2 rpm. We found that the phase error would build up to 20° or so as the frequency passed through the aliasing points, but this was more than enough accuracy to provide effective de-tumble. The simulation showed that the loop could achieve phase lock from a 180° initial phase error with an initial frequency error of 0.2 rpm. The time required to lock on was about 20 minutes.

The system was implemented in the flight computer as two separate programs: (a) a special version of the telemetry monitoring program that processed the magnetometer readings and locked onto the phase of the motion, and (b) a clock-synchronized delayed command program that used the latest phase information from the first program to control the z coil switching commands. The two programs ran independently but worked in concert, passing information back and forth through shared memory locations. The phase-lock program ran continuously, and the z coil switching program was cycled on for about 15 minutes near the equator crossings in each orbit period.

It took less than one month for the computer to de-tumble the spacecraft, allowing us to go on with the spacecraft checkout in time to change the TIP-III spacecraft operation in some important ways.

Generating a Tumble Motion for Centrifugal Force

After the de-tumble was complete, the gravity-gradient boom was erected on TIP-II. Because of an unforeseen problem with scissors booms, the links broke during the motor-driven deployment. It was too late to change the boom on the next spacecraft but the problem was studied and finally understood. The conclusion was that the boom could be successfully deployed if it were kept in exactly the correct amount of tension using centrifugal force, which could be generated by tumbling the spacecraft at the correct rate, an operation that we were beginning to master.

On TIP-III, the boom was successfully deployed in this manner: The spacecraft was tumbled by using the same phase-locked loop program and simply revers-

DIGITAL PHASE-LOCKED LOOP FOR Z COIL CONTROL

Each telemetry frame, the program receives a sample of the raw z magnetometer, M_i , which we refer to as the i th measurement. This measurement is then processed to give an updated phase and period for the sawtooth function to control the z coil commands. The loop is shown schematically in the figure. The computations are given in the following steps.

1. Each measurement, correct for the effect of the z coil:

$$m_i = M_i - K\Delta,$$

where m_i is the corrected measurement, M_i is the raw measurement, Δ is the z coil dipole moment, and K is ± 1 , depending on the z coil state.

2. Normalize measurements:

$$AM_i = (1 - A) AM_{i-1} + A |m_i|,$$

$$FM_i = \frac{m_i}{1.5 AM_i} = \text{normalized measurement},$$

$$A = 1/256.$$

3. Compute the sawtooth value based on the last known trough time and frequency:

$$r_i = [(t_i - T_i) \omega] \text{ modulo } 1 \text{ (fractional part)},$$

$$\begin{cases} f_i = 4r_i - 1, & \text{if } r_i < 0.5 \\ f_i = 3 - 4r_i, & \text{if } r_i \geq 0.5, \end{cases}$$

where r_i is the fractional part of a cycle since the last trough, t_i is the time of the present measurement, T_i is the time of the last trough, ω is the current sawtooth frequency, and f_i is the value of the sawtooth function.

4. Compute and filter the error signal recursively:

$$P_i = FM_i \times f_i \text{ (current product)},$$

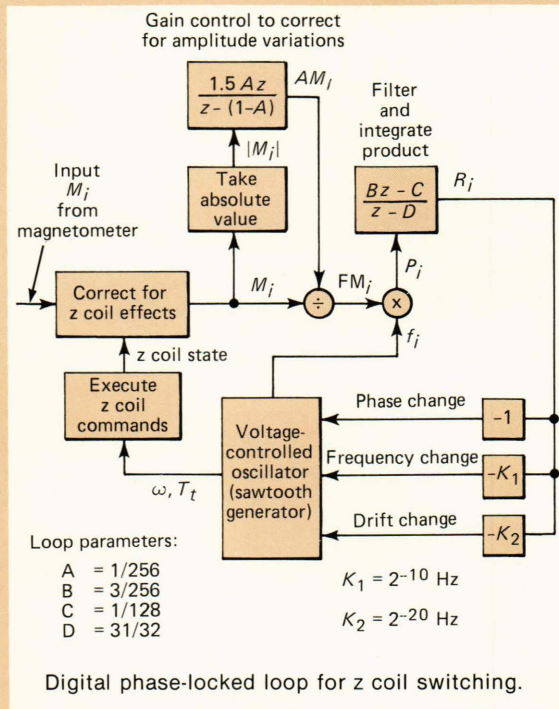
$$R_i = BP_i - CP_{i-1} + DR_{i-1} \text{ (error signal)},$$

$$B = 3/256,$$

$$C = 1/128,$$

$$D = 31/32.$$

5. Check for aliasing points and reduce the error signal if necessary:



Digital phase-locked loop for z coil switching.

$$\text{If } |\omega - \omega_a| < \delta, R_i = R_i \left(\frac{\omega - \omega_a}{\delta} \right)^2,$$

$$\delta = 0.1 \text{ rpm.}$$

6. Apply controls to the sawtooth value:

$$\dot{\omega} = \dot{\omega} - K_2 R_i \text{ (drift)},$$

$$\omega = \omega + \dot{\omega} \Delta t - K_1 R_i \text{ (frequency)},$$

$$T_i = t_i - \frac{r_i - R_i}{\omega} \text{ (time of last trough)},$$

where r_i is the fractional part of a cycle calculated in step 3.

Commands to control the z coil are then issued based on T_i and $1/\omega$. The value of $\dot{\omega}$ is changed to the appropriate theoretical value if the z coil is turned off or on. This results in a small ramp-type transient.

ing the z coil polarity commands. It was extremely important during the tumble-up operation that the rate stay within narrow limits. If the rate became too high, the boom links would have been broken in tension; if it became too low, the links would have broken in compression. The tumble rate had to be kept within 10% of its prescribed value at all times. It proved easy, using the existing special programs, to let the flight computer automatically monitor and increase the tumble momentum bit by bit as the boom was gradually deployed over a three-week period.

FINAL REMARKS

As a result of the flight computer control capability, the TIP-II and TIP-III spacecraft achieved a substantial success. The TIP-III spacecraft was able to achieve three-axis stability with the disturbance compensation subsystem in full operation. This allowed valuable in-orbit testing of the important subsystem, which is difficult (if not impossible) to fully test on the ground. As a result of the testing, several important engineering changes were developed and im-

plemented in the Nova version of the spacecraft. The TIP-II spacecraft became operational as a navigation satellite for limited periods (when the percentage of sunlight was high enough). As the prototype spacecraft in a new series, it also provided a valuable training and test capability to the Navy personnel who operate the navigation satellite system.

Later in the program, after the first Nova became operational, various other changes were introduced into the flight software to reduce operational manpower, expand the spacecraft capability, and correct unanticipated difficulties. These changes will not be discussed here; however, they all served to enhance the mission greatly.

The overall program experience proved to be a dramatic illustration of the power of having a programmable computer on board a spacecraft. Equipped with the powerful capability to modify the on-board logic after launch, we seemed to be limited only by our own ingenuity in finding ways to use the system.

It cannot be emphasized enough that an on-board processor that has its program burned into programmable read-only memories will not provide a similar capability, even if the processor is much more powerful. A very real value lies in the adaptability of a system *after* launch. The approach taken to achieve mission goals, and even the setting of the goals themselves, will be greatly influenced by, for example, a fully reprogrammable fifth generation computer aboard a spacecraft, able to communicate with and control the various other sensors and subsystems on board. One can envision such a system being programmed in high-level language using artificial intel-

ligence techniques to produce a program able to control the spacecraft in intelligent ways, recover from failures, and adapt to changing situations.

REFERENCES and NOTES

- ¹The first TIP spacecraft, also designated the Triad satellite, differs in design and will not be discussed here.
- ²J. A. Perschy and B. M. Elder, *TRIAD Computer*, JHU/APL TG 1212 (Aug 1973).
- ³H. K. Utterback, *A Guide to the TIP Computer*, JHU/APL TG 1263 (Sep 1974).
- ⁴C. Marvin, *NNSS Software System: An Introduction to TIP-II Digital Operations*, JHU/APL SDO 4318 (Jan 1976).
- ⁵J. M. Whisnant and R. E. Jenkins, *NNSS Software System: Post-Launch and Operational Flight Computer Programs*, JHU/APL SDO 4268, Vols. I and II (1977).
- ⁶H. K. Utterback, J. M. Whisnant, and R. E. Jenkins, "A System of Software for the TIP Spacecraft Computer," British Interplanetary Society Symp. on Computer Techniques for Satellite Control and Data Processing, Slough, England (11-12 Oct 1977).
- ⁷We also ran out of spin; the hydrazine-induced nutation continuously robbed energy from the spin.
- ⁸The attitude software was developed chiefly by C. E. Williams and S. E. Ciarrocca.
- ⁹The average delay is based on one frame because the first frame is equally likely to occur just before or just after the derivative changes sign.
- ¹⁰The TIP computer, in keeping with its primary functions, was oriented toward logic and bit string manipulations rather than arithmetic operations.

ACKNOWLEDGMENTS—The pressure of carrying out and debugging the flight computer program changes described here was shared by J. M. Whisnant. As recorded in the author's log of these activities, several people contributed to the ideas used to accomplish these operations, in some cases unknowingly. The idea of developing the phase-locked loop arose from an offhand comment by J. W. Follin. A lunchtime suggestion by C. Marvin led directly to the use of the sawtooth function in the phase-locked loop. A comment by F. F. Mobley in a discussion of another matter suggested the use of the z coil for nutation damping. A jesting remark by R. W. Newman after a softball game created the idea for the tumble-thrust program. Finally, the steadfast insistence by some people that the ideas would not work provided a major impetus to their implementation.