1

The Gemini Computer: First Machine in Space

Project Mercury was America's first man-in-space effort. The McDonnell-Douglas Corporation developed the Mercury spacecraft in the familiar bell shape. It was barely large enough for its single occupant and had no independent maneuvering capability save attitude control jets. Its orbital path was completely dependent on the accuracy of the guidance of the Atlas booster rocket. Re-entry was calculated by a realtime computing center on the ground, with retrofire times and firing attitude transmitted to the spacecraft while in flight. Therefore, it was unnecessary for the Mercury spacecraft to have a computer, as all functions required for its limited flight objectives were handled by other systems.

Gemini both continued the objectives of the Mercury program and served as a test bed for the development of rendezvous techniques critical to lunar missions[1]. At first glance, the Mercury and Gemini spacecraft are quite similar. They share the bell shape and other characteristics, partially because Gemini was designed as an enlarged Mercury and because the prime contractor was the same for both craft. The obvious difference is the presence of a second crew member and an orbital maneuvering system attached to the rear of the main cabin. The presence of a second crewman meant that more instrumentation could be placed in Gemini and that more experiments could be performed, as an extra set of eyes and hands would be available. Gemini's maneuvering capability made it possible to practice rendezvous techniques. The main rendezvous target was planned to be the Agena, an upper stage rocket with a restartable liquid propellant engine that could be launched by an Atlas booster. After rendezvous with an Agena, the Gemini would have greatly increased maneuvering capability because it could use the rocket on the Agena to raise its orbit.

Successful rendezvous required accurate orbital insertion, complex catch-up maneuvering, finely tuned movements while making the final approach to the target, and guidance during maneuvers with the Agena. Safety during the critical powered ascent phase demanded some backup to the ascent guidance system on the Titan II booster vehicle. The Gemini designers also wanted to add accuracy to re-entry and to automate some of the preflight checkout functions. These varied requirements dictated that the spacecraft carry some sort of active, on-board computing capability. The resulting device was the Gemini digital computer.

The Gemini computer functioned in six mission phases: prelaunch, ascent backup, insertion, catch-up, rendezvous, and re-entry. These requirements demanded a very reliable, fairly sophisticated digital computer with simple crew interfaces. IBM built such a machine for the Gemini spacecraft.

By the early 1960s, engineers were searching for ways to automate checkout procedures and reduce the number of discrete test lines connected to launch vehicles and spacecraft. Gemini's computer

Figure 1–1. First orbital rendezvous: Gemini VI keeps station after using its on-board computer to maneuver to position near Gemini VII. (NASA photo S-65-63175)

did its own self checks under software control during the prelaunch phase. It also accepted parameters needed for the flight during the last 150 minutes before launch[2]. During ascent, the computer received information about the velocity and course of the booster so that it would be ready to take over from the Titan's computers if they failed. Switch-over could either be automatic or manual. The computer could then issue steering and booster cutoff commands to the Titan[3]. Even if the updated parameters were not necessary to boost guidance, they were useful in the calculation of additional velocity needed after the Titan's second-stage cutoff to achieve the proper orbit. That velocity difference was displayed to the crew so that they could use the spacecraft's own propulsion system to reach insertion velocity[4].

Rendezvous operations required an on-board computer because the ground tracking network did not cover all parts of the Gemini orbital paths. Thus, it would be impossible to provide the sort of continuous updates needed for rendezvous maneuvers. For example, Gemini XI was planned as a first-orbit Agena rendezvous, with some of the critical maneuvers conducted outside of telemetry range[5]. That same mission also featured a fully computer-controlled re-entry, which resulted in a splashdown 4.6 kilometers from the target[6]. In computer-controlled descents, the roll attitude and rate are handled by the computer to affect the point of touchdown and re-entry heating. The Gemini spacecraft had sufficient lift capability to adjust the landing point up to 500 miles on the line of flight and 40 miles laterally respective to the line of flight. Five minutes before retrofire, the computer was placed in re-entry mode and began to collect data. It displayed velocity changes during and after the retrofire. During the time the spacecraft traveled from an altitude of 400,000 feet to when it reached 90,000 feet, the computer controlled actual attitude[7].

HARDWARE

IBM Corporation received the contract for the Gemini digital computer on April 19, 1962, amounting to $26.6 million. It provided for the construction of the on-board computer and for integration with other spacecraft systems[8]. The first machine was in its final testing phase by August 31, 1963, and IBM delivered the last of 20 such machines by December 1965[9]. Engineers at IBM believe that the main reason why their company received the contract was the successful development of a core memory used on the Orbiting Astronautical Observatory[10]. One of them, John J. Lenz, said that the contract for Gemini came just at the right time. The best of the engineering teams of the IBM Federal Systems Division plant in Owego, New York were between assignments and were put on the project, increasing its chance for success.

Restrictions on size, power, and weight influenced the final form of the computer in terms of its components, speed, and type of memory. The shape and size of the computer was dictated by the design of the spacecraft. It was contained in a box measuring 18.9 inches high by 14.5 inches wide by 12.75 inches deep, weighing 58.98 pounds[11]. An unpressurized equipment bay to the left of the Gemini commander s seat held the computer, as well as the inertial guidance system power supply and the computer auxiliary power supply. The machine consisted of discrete components, not integrated circuits[12]. However, circuit modules that held the components were somewhat interchangeable. They were plugged into one of five interconnection

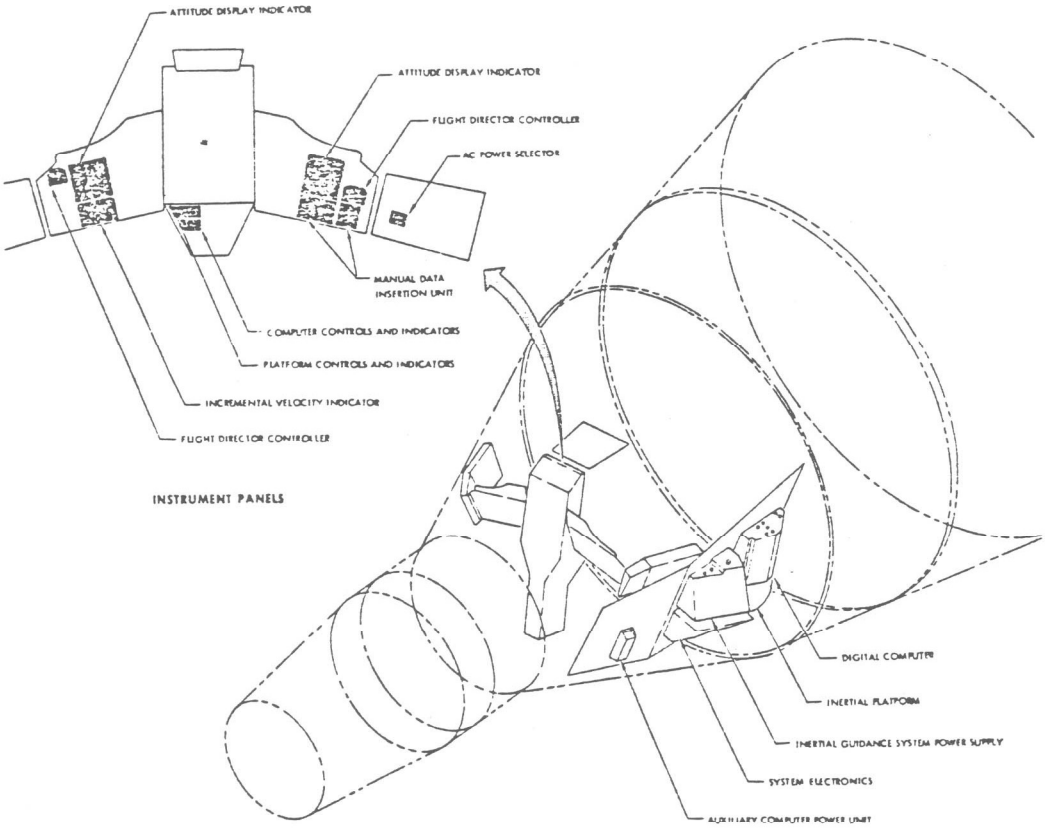


Figure 1–2. Locations of key components of the Gemini Guidance System. (From McDonnell Corp., *Gemini Familiarization Manual*)

boards, and it took 510 of the modules to build the logic section alone[13]. The computer had no redundant circuits, which meant that failures in the computer canceled whatever activity needed to be controlled by it. For example, a failure in the power switch three quarters of the way through the Gemini IV mission caused cancellation of the planned computer-controlled re-entry. It was possible to fly the Gemini computer without a backup because whatever the computer did erroneously could be either abandoned (such as rendezvous) or handled, albeit more crudely, in other ways (such as re-entry using Mercury procedures).

The machine had an instruction cycle of 140 milliseconds, the time it required for an addition. Multiplication took three cycles, or 420 milliseconds, with division requiring double that time[14]. The

arithmetic bit rate was 500 kilocycles, and the memory cycle rate half that[15]. The computer was serial in operation, passing bits one at a time, which explains the relatively slow processing speeds, slower than some vacuum tube computers such as the Whirlwind. Also, its fixed decimal point arithmetic unit design limited the precision of the calculations but greatly reduced complexity. The Gemini digital computer used ferrite cores for its primary memory. Core memories store one bit in each ferrite ring by magnetizing the ring in either a clockwise or counterclockwise direction. One direction means a one is stored and the opposite direction is a zero. Each core is mounted at a perpendicular crossing of two wires. Thousands of such crossings are in each core plane, consisting of rows of wires running up and down (the x wires) and others running left and right (the y wires). Therefore, to change the value of a bit at a specific location, half the voltage required for the change is sent on each of two wires, one in the x direction and one in the y direction. This way only the core at the intersection of the two wires is selected for change. All the others on the same wires would have received only half the required voltage. By the use of a third wire it is possible to "sense" whether a selected core is a one or a zero. In this way, each individual core can be read.

The ferrite core memory in the Gemini computer had a unique design. It consisted of 39 planes of 64 by 64-bit arrays, resulting in 4,096 addresses, each containing 39 bits. A word was considered to be 39 bits in length, but it was divided into three syllables of 13 bits. The memory itself divided into 18 sectors. Therefore, it was necessary to specify sector and syllable to make a complete address. Instructions used 13 bits of the word with data representations of 26 bits. Data words were always stored in syllables 0 and 1 of a full word, but instructions could be in any syllable. This means that up to three instructions could be placed in any full word, but only one data item could be in a full word[16].

The arithmetic and logic circuit boards and the core memory made up the main part of the Gemini computer. These components interfaced to a plethora of spacecraft systems, most of which were concerned with guidance and navigation functions. This system was the Gemini digital computer through the Gemini VII mission. Beginning with Gemini VIII, the computer included a secondary storage system, which had impact on the spacecraft computer systems built by IBM and flown on the Skylab and Shuttle.

During the 1950s and well into the 1960s, the most ubiquitous method of providing large secondary storage for computers was the use of high-speed, high-density magnetic tape. By 1980, tape was used mainly to store large blocks of data unneeded on a regular basis or to mail programs and data between sites. Disk systems have considerably faster access times and have rapidly increased in storage capacity,

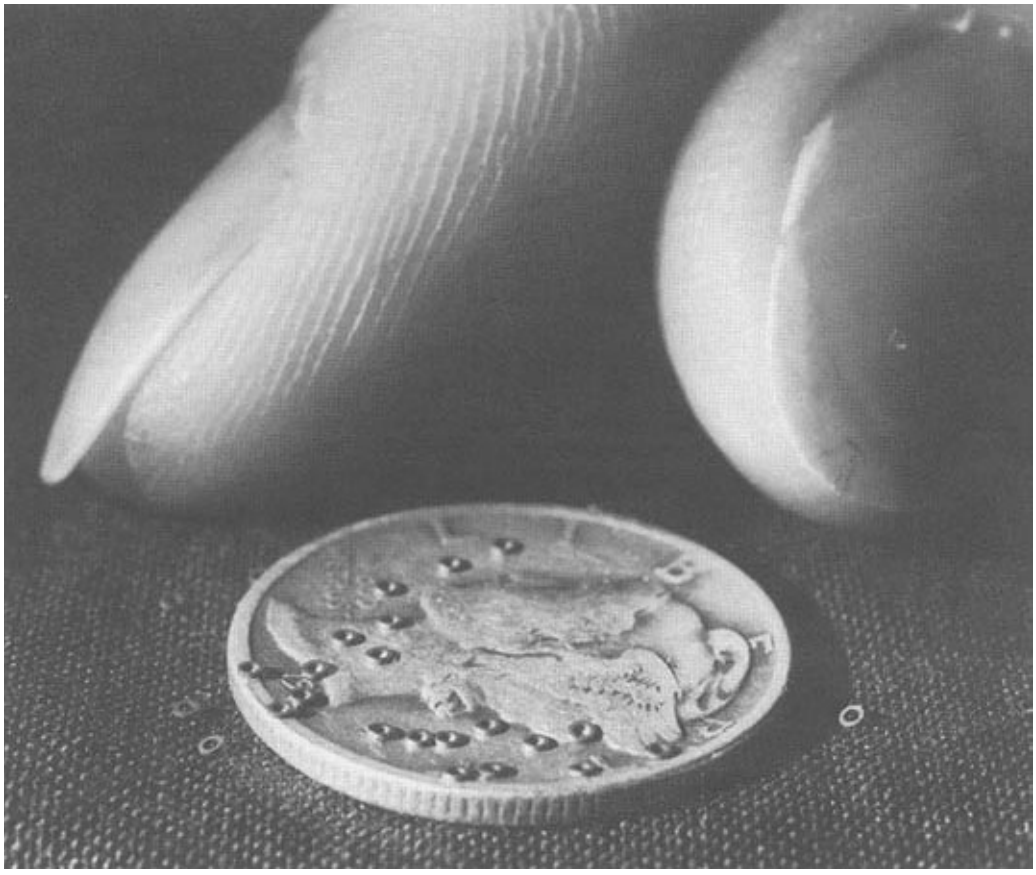


Figure 1–3. Cores like these were used in Gemini's memory. (IBM photo)

rivaling or even exceeding tape, and thus supplanting it in common use. In 1962, disk systems were large, expensive, and far from fully reliable. When the software for the Gemini computer threatened to exceed the storage capacity of the core memory, IBM proposed an Auxiliary Tape Memory to store software modules that did not need to be in the computer at lift-off. For example, programs that provided backup booster guidance and insertion assistance would be in the core memory for the early part of the flight. The re-entry program could be loaded into the core shortly before it was needed, thus writing over the programs already there. This concept, fairly common in earth-bound computer usage, was a first for aerospace computing.

IBM chose the Raymond Engineering Laboratory of Middletown, Connecticut to build the device[17]. The unit weighed 26 pounds and filled about 700 cubic inches of space in the adapter module of the Gemini spacecraft[18]. The tape memory increased the available storage of the Gemini computer by seven and one-half times with its capacity of 1,170,000 bits. Programs loaded from the tape would fill syllables 0 and 1 of the core memory locations[19]. It took 6 minutes to load a

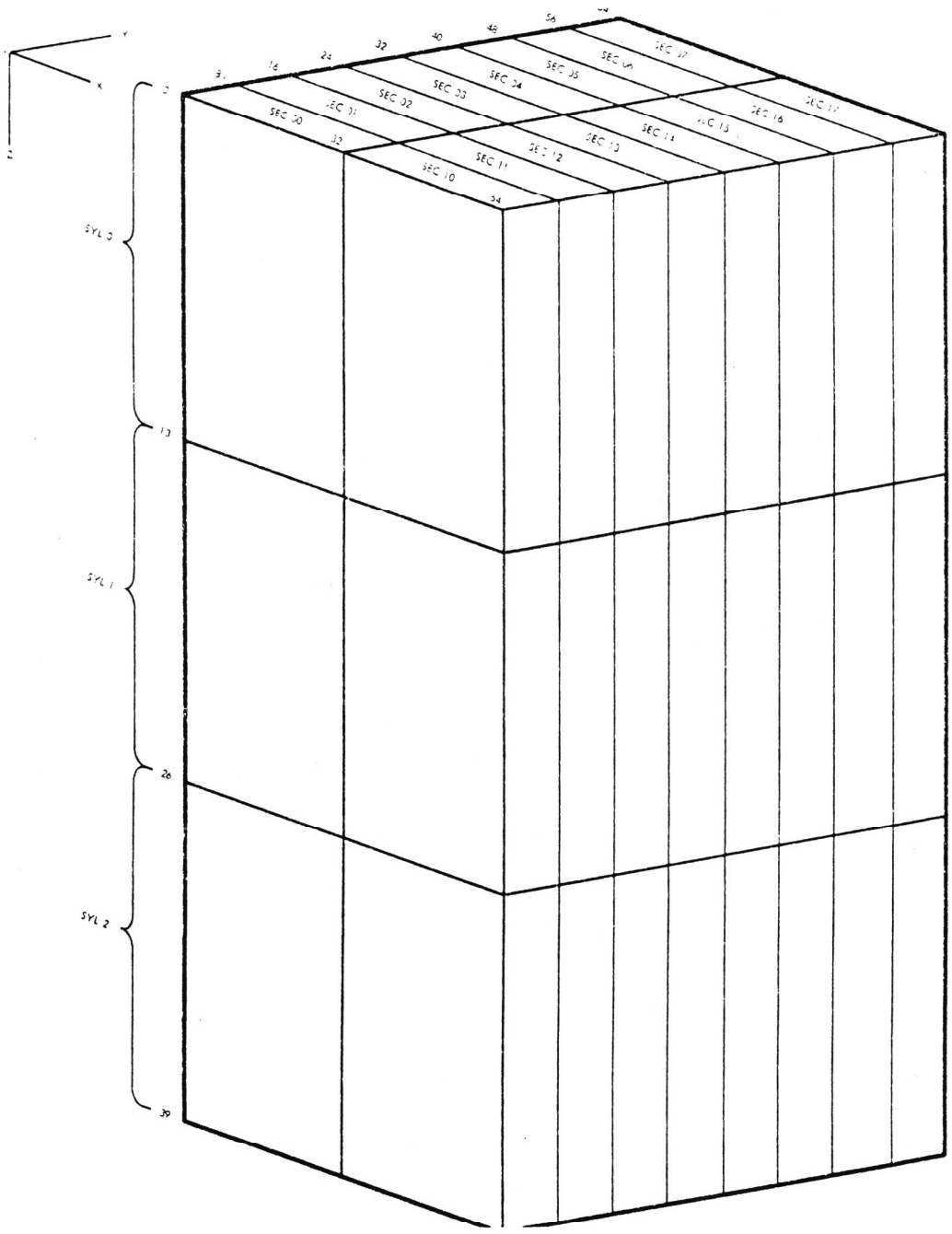


Figure 1–4. Layout of the Gemini Digital Computer core memory. (From McDonnell Corp., *Gemini Familiarization Manual*)

program from the tape drive into core[20].

NASA's natural insistence on high reliability in manned spaceflight operations challenged the computer industry of the early 1960s. Tape error rates were 1 bit in 100,000 and IBM wanted to raise this rate to 1 bit in 1,000,000,000[21]. The method used was to triple record each program on the tape, pass each set of three corresponding bits through a voter circuit, and send the result of the vote to the core memory[22]. This scheme was later used on the Shuttle.

Gemini VIII was the first mission with the Auxiliary Tape Memory on board. Shortly after a successful rendezvous with an Agena, the combined spacecraft began to spin out of control. Mission Control decided to disengage the Agena and bring the Gemini down, as large amounts of attitude control thruster fuel had been wasted trying to regain control of the spacecraft. Thus, the first attempt to load a program from the tape was made while the spacecraft was spinning. Even though the Auxiliary Tape Memory design parameters specified low vibration levels,[23] the re-entry program was successfully loaded and used in the subsequent descent.

IBM obtained this sort of reliability beyond the original specifications as a result of an extensive testing program. For example, the Auxiliary Tape Memory had failed prequalification vibration tests, so IBM added a brass flywheel and weights on the tape reels to increase stabilization[24]. This ensured a successful program load under adverse conditions. There were also problems with transistors shorting out due to loose particles too small to be seen on x-rays but which shook loose during acceleration[25]. Increased cleanliness in manufacturing was one solution to this problem.

The only in-flight failure of a computer component was on the 48th revolution of the Gemini IV mission, when astronaut James McDivitt tried to update the computer in preparation for re-entry. The machine would not turn off, and it could not be used for the planned "lifting bank" re-entry[26]. IBM could not duplicate the failure on the ground, but the manufacturers did install a manual switch that bypassed the failure for Gemini V[27].

SOFTWARE

In 1962, hardware was still the pacing factor in computer applications. Everything associated with computers—processors, memories and input/output (I/O) peripherals—was expensive. Many considered software development an incidental part of the overall applications of computing. Specialists wrote most of the software, usually in arcane assembly languages. FORTRAN, a high-level language, had only been

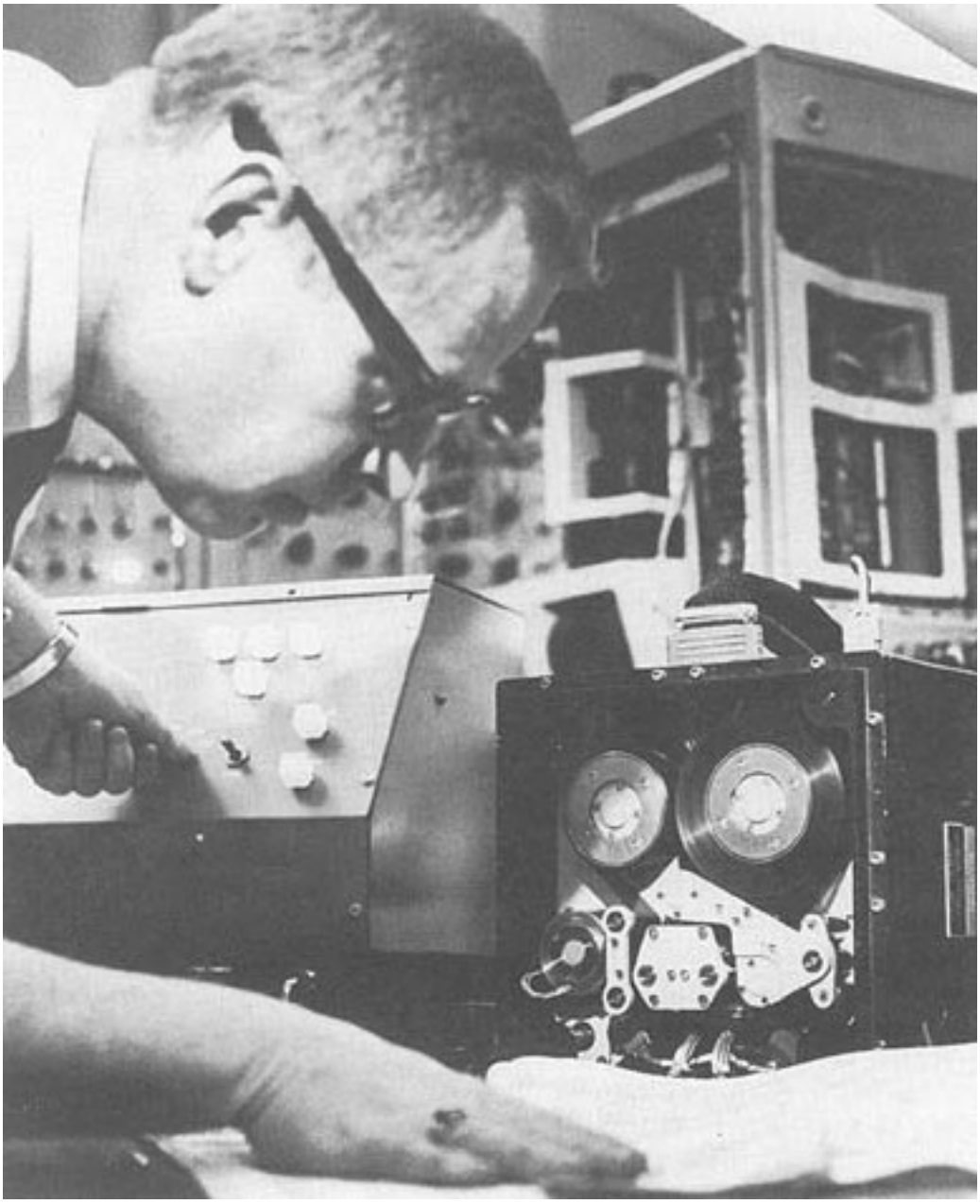


Figure 1–5. Auxiliary Tape Memory in test. (IBM photo)

available for a few years. Although its use in technical applications was rapidly spreading, it was still considered too inefficient for use on computers like the Gemini digital computer. Many thought its compiler-produced machine code to be less effective in utilizing machine resources than machine language programs written by humans. Experts therefore developed applications programs for Gemini using the tiny set of 16 instructions that the computer could execute[28]. This sort of programming was considered to be more of an art than a science. Whereas the design and construction of computer hardware followed conventional engineering principles, software development was largely haphazard, undocumented, and highly idiosyncratic. Many managers considered software developers to be a different breed and best left alone. This concept of software is a myth, and although it persists in some companies and with some people today, by and large software is now considered as an engineered product, little different from a rocket engine or computer.

Although the term "software engineering" did not come into common use until 1968, programmers had applied its basic tenets to both large and small software projects for at least 15 years. Software engineering has evolved as programmers learned which techniques worked, which did not, and what actually occurred in the development of software products. The SAGE (Semi-Automatic Ground Environment) air defense system[29], the IBM 360 operating system[30], and NASA's requirements for both spacecraft software and ground-based software were instances of major software projects that directly contributed to the evolution of software engineering.

Software engineers recognize that software follows a specific development cycle, from formal specification of the product, through the design and coding of the actual program, and then to testing of the product and postdelivery maintenance. This cycle lasts for many years in the case of programs such as operating systems, or a short period of time in the case of specialized, single-use programs. During this development process, strict standards of documentation, configuration control, and managing changes and the correction of errors must be maintained. Also, breaking down the application into smaller, potentially interchangeable parts, or modules, is a primary technique. Communication between programming teams working on different but interconnected modules must be kept clear and unambiguous. It is in these areas that NASA has had the greatest impact on software engineering.

Development of the Gemini software was a learning experience for both NASA and IBM. It was, of course, the first on-board software for a manned spacecraft and was certainly a more sophisticated system than any that had flown on unmanned spacecraft to that point. When the time came to write the software for Gemini, programmers envisioned a single software load containing all the code for the flight, with

new unique programs to be developed for each mission[31]. Soon it became obvious that certain parts of the program were relatively unchanged from mission to mission, such as the ascent guidance backup. Designers then introduced modularization, with some modules becoming parts of several software loads.

Another reason for modularization is the fact that the programs developed for Gemini quickly exceeded the memory available for them. Some were stored on the Auxiliary Tape Memory until needed. The problem of poor estimation of total memory requirements has plagued each manned spacecraft computer system. In the case of Gemini, changed requirements and attempts to squeeze the programs into the allotted space resulted in nine different versions of the software[32]. The different versions were referred to by the name "Gemini Math Flow."

Tracing the development of the math flows shows how identifying new functions caused initial memory estimates to be wrong and how the project handled changes. Math Flow One consisted of just four modules: Ascent, Catch-up, Rendezvous, and Re-entry. Math Flow Two was proposed to add orbital navigation and re-entry initialization, but it caused the overall load to exceed the memory size and the Gemini program office canceled the additions[33]. This version of the software flew on spacecraft II in January 1965. By Math Flow Four, the re-entry initialization program had been successfully added, but the load took up 12,150 of 12,288 available words. The plan had been to use this program on spacecraft III and others, but a NASA directive of February, 1964 changed the guidance logic of the re-entry mode to a constant bank angle rather than a proportional bank angle and constant roll rate. Math Flow Five incorporated this change, but it filled the memory and was scrubbed in favor of a modified Math Flow Three on spacecraft III and IV, followed by Math Flow Six containing some changes on spacecraft V through VII[34]. The final version, Math Flow Seven, was used on spacecraft VIII through XII, all of which incorporated the Auxiliary Tape Memory. It had six program modules with nine operational modes. The six program modules of Math Flow Seven were Executor, Prelaunch, Ascent, Catch-Up, Rendezvous, and Re-Entry[35]. The Executor routine selected other routines depending upon mission phases.

The specification procedure for the software required McDonnell-Douglas to prepare the Specification Control Document (SCD). This was forwarded to the IBM Space Guidance Center in Owego, which developed a FORTRAN program to validate the guidance equations. The use of simulations such as the FORTRAN program was endemic to the Gemini software effort and was later applied to software development for other spacecraft computers.

Gemini used three levels of simulations, beginning with the equation-validation system. The second was a man-in-the-loop simula-

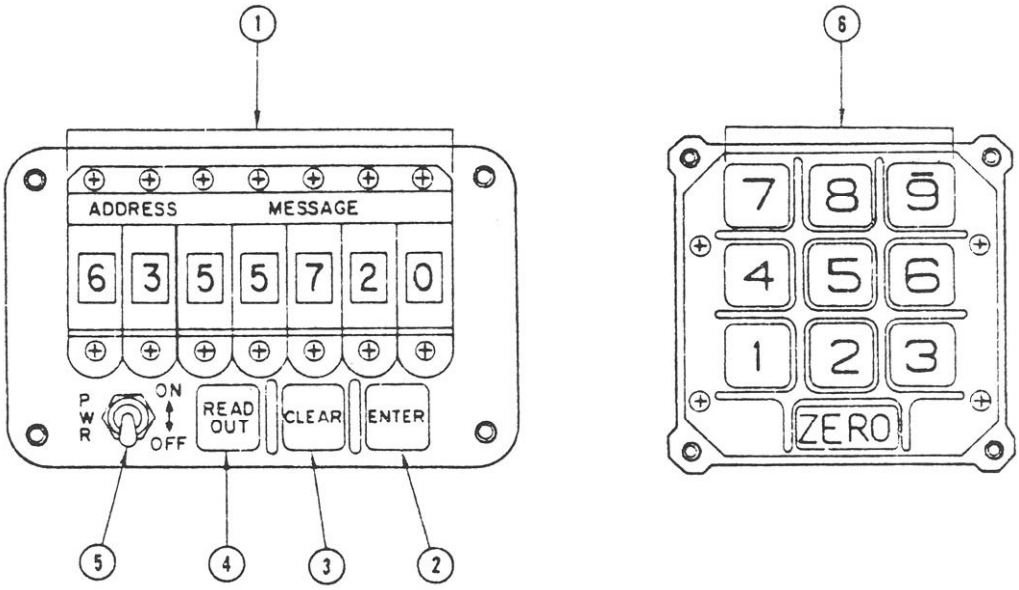
tion to help define I/O requirements, procedures, and displays. The third level was a refined digital simulation to determine the performance characteristics of the software, useful in error analysis. This third level was carried out in the Configuration Control Test System (CCTS) laboratory, which contained a Gemini computer and crew interfaces. This Mission Verification Simulation (MVS) ensured that the guidance system worked with the operational mission program. Further tests of the software were done at McDonnell-Douglas and at the pad[36]. NASA and IBM emphasized program verification because there was no backup computer or backup software. The verification process and the tools developed for it were later applied to military projects in which IBM became involved[37].

Even if the software is perfect, errors may occur because of transient hardware or software failures during operation due to power fluctuations or unforeseen demands on real-time programs. Some of these can be spotted by diagnostic subroutines interleaved in the software and used for fault detection[38]. Such routines were put in the Gemini software and are now a part of all IBM computer systems.

The software produced during the Gemini program was highly reliable and successful. Techniques of specification development, verification, and simulations developed for Gemini were later applied to other IBM and NASA projects. NASA was certainly better prepared to monitor software development for the much more difficult Apollo program.

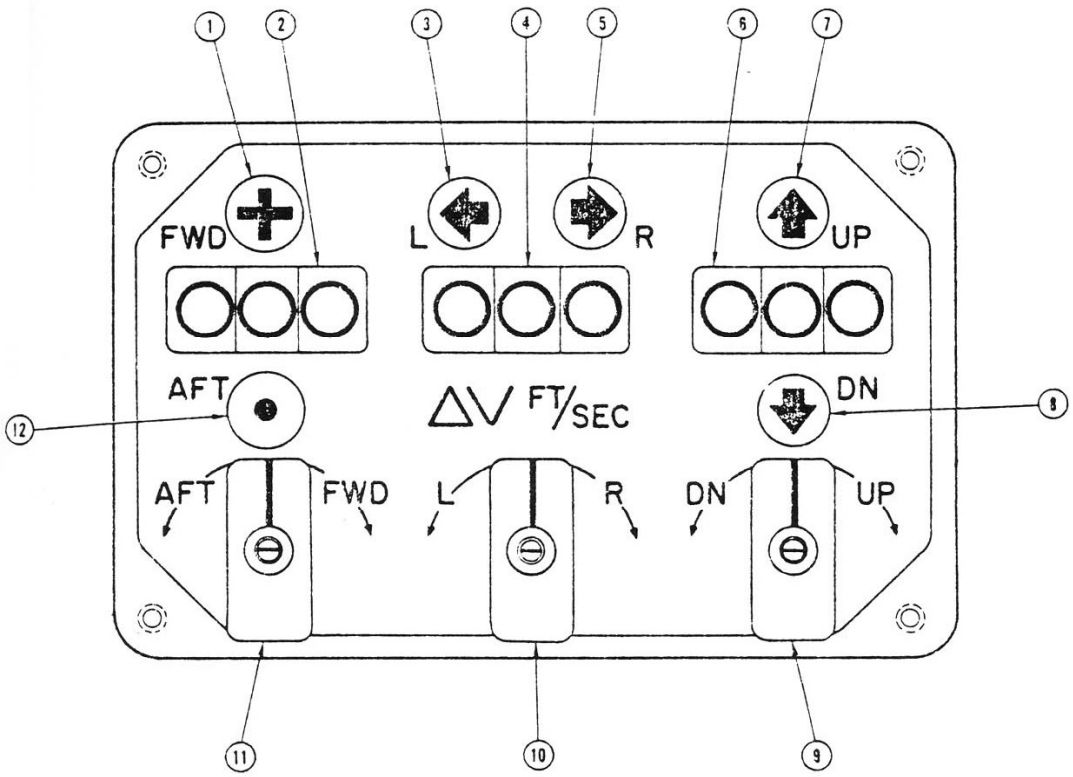
CREW INTERFACES TO THE GEMINI DIGITAL COMPUTER

Gemini's digital computer had three sets of interfaces: the computer's controls, the Manual Data Insertion Unit (MDIU), and the Incremental Velocity Indicator (IVI). The controls consisted of a mode switch, a start button, a malfunction light, a computation light, and a reset switch. The mode switch had seven positions for selection of one of the measurement or computation programs. The start button caused the computer to run the selected program loaded in its memory. The reset switch caused the computer to execute its start-up diagnostics and prepare itself for action. The MDIU consisted of two parts: a 10-digit keyboard and a 7-digit register. The first two digits of the register, a simple odometerlike rotary display, were used to indicate a memory address. Up to 99 such logical addresses could be accessed. The remaining five digits displayed data. Errors caused all zeroes to appear. Negative numbers were inserted by making the first digit a nine; the other digits contained the value. The IVI displayed velocity increments required for, or as a result of, a powered maneuver. It had three-digit feet-per-second displays for each of forward-and-back, up-and-down, and left-to-right axes[39].



LEGEND		
ITEM	NOMENCLATURE	PURPOSE
1	ADDRESS AND MESSAGE DISPLAY DEVICES	DISPLAY ADDRESS AND MESSAGE SENT TO COMPUTER DURING ENTER OPERATION; DISPLAY ADDRESS SENT TO, AND MESSAGE RECEIVED FROM, COMPUTER DURING READOUT OPERATION.
2	ENTER PUSH-BUTTON SWITCH	PROVIDES MEANS FOR CAUSING MESSAGE SENT TO COMPUTER DURING ENTER OPERATION TO BE STORED IN MEMORY.
3	CLEAR PUSH-BUTTON SWITCH	PROVIDES MEANS FOR CAUSING ADDRESS AND MESSAGE SET UP BY MDK TO BE CLEARED OR CANCELED.
4	READ OUT PUSH-BUTTON SWITCH	PROVIDES MEANS FOR CAUSING MESSAGE TO BE READ OUT OF COMPUTER AND DISPLAYED BY MESSAGE DISPLAY DEVICES.
5	PWR (POWER) TOGGLE SWITCH	PROVIDES MEANS FOR CONTROLLING APPLICATION OF POWER TO MDK AND MDR.
6	DATA INSERT PUSH-BUTTON SWITCHES	PROVIDE MEANS FOR CAUSING ADDRESS AND MESSAGE TO BE SENT TO COMPUTER AND TO BE DISPLAYED BY ADDRESS AND MESSAGE DISPLAY DEVICES.

Figure 1–6. Manual Data Insertion Unit. (From McDonnell Corp., *Gemini Familiarization Manual*)



LEGEND		
ITEM	NOMENCLATURE	PURPOSE
1	FWD (FORWARD) DIRECTION INDICATION LAMP	INDICATES THAT PLUS X AXIS VELOCITY IS INSUFFICIENT.
2	FORWARD-AFT DISPLAY DEVICE	INDICATES AMOUNT OF INSUFFICIENT VELOCITY FOR PLUS OR MINUS X AXIS
3	L (LEFT) DIRECTION INDICATION LAMP	INDICATES THAT MINUS Y AXIS VELOCITY IS INSUFFICIENT.
4	LEFT-RIGHT DISPLAY DEVICE	INDICATES AMOUNT OF INSUFFICIENT VELOCITY FOR PLUS OR MINUS Y AXIS.
5	R (RIGHT) DIRECTION INDICATION LAMP	INDICATES THAT PLUS Y AXIS VELOCITY IS INSUFFICIENT
6	UP-DOWN DISPLAY DEVICE	INDICATES AMOUNT OF INSUFFICIENT VELOCITY FOR PLUS OR MINUS Z AXIS.
7	UP DIRECTION INDICATION LAMP	INDICATES THAT MINUS Z AXIS VELOCITY IS INSUFFICIENT.
8	DN (DOWN) DIRECTION INDICATION LAMP	INDICATES THAT PLUS Z AXIS VELOCITY IS INSUFFICIENT.
9	DN-UP ROTARY SWITCH	PROVIDES MEANS FOR MANUALLY SETTING UP Z AXIS VELOCITY ERROR ON UP-DOWN DISPLAY DEVICE.
10	L-R ROTARY SWITCH	PROVIDES MEANS FOR MANUALLY SETTING UP Y AXIS VELOCITY ERROR ON LEFT-RIGHT DISPLAY DEVICE.
11	AFT-FWD ROTARY SWITCH	PROVIDES MEANS FOR MANUALLY SETTING UP X AXIS VELOCITY ERROR ON FORWARD-AFT DISPLAY DEVICE
12	AFT DIRECTION INDICATION LAMP	INDICATES THAT MINUS X AXIS VELOCITY IS INSUFFICIENT

Figure 1–7. Incremental Velocity Indicator. (From McDonnell Corp., *Gemini Familiarization Manual*)

On a typical mission the computer would be in operation during ascent as the backup to the booster. On orbit, if no powered maneuvers were imminent, it could be shut down to save electrical power. Due to the nature of core memory, programs and data stored magnetically in the cores would not disappear when the power was off, as in present day semiconductor memories. This made it possible to load the next set of modules, if necessary, from the Auxiliary Tape Memory, enter any needed parameters, and then shut down the machine until shortly before its next use. It took 20 seconds for the machine to run its start-up diagnostics upon restoration of power. After the diagnostics ran successfully, the current program load was ready for use, all parameters intact.

GT-IV was following such a procedure in preparing for re-entry on June 7, 1965. The computer was placed in the RNTY mode, and the crew received and entered updated parameters given to them when they were in contact with the ground stations. But when they tried to turn the machine off, the manual on/off switch did not function. The power had to be cut off by another means, and the re-entry handled manually[40].

Using the computer for catch-up and rendezvous was a relatively simple task. The difference between catch-up and rendezvous is that catch-up maneuvers are executed to put the spacecraft into position to make an orbit-change maneuver. After the orbit change the spacecraft is prepared to rendezvous with the target[41]. Crews began the catch-up by entering the ground-calculated rendezvous angle desired into address 83. The rendezvous angle indicated how much farther along in a 360-degree orbit the rendezvous was to take place. For example, if the crew desired rendezvous one-third orbit ahead, 12000 was entered into address 83 using the MDIU. The interval at which the pilot wanted to see updates was then entered in address 93. For example, if 04000 was entered, the computer would display on the IVI any required velocity changes at 120 degrees from the rendezvous point (the start), 80 degrees to go, and 40 degrees to go. If the IVI indicated that the computer had calculated that such a rendezvous was possible within the designated fuel limits, the astronauts pressed the START button and the IVI displayed the first set of velocity differentials. The pilot then fired the thrusters until the displays were all at zero (Astronaut John Young reported that there was a tendency to "overshoot" in trying to burn to zero[42].). After that, nothing was done unless the next update indicated a need for more velocity adjustments[43]. The astronauts also did paper-and-pencil calculations of the velocity changes as a backup by using special nomographs based on time and angles to the target[44]. These backup calculations were compared with the ground-calculated solution as well as the computer solution. However, the figures computed on-board were considered the primary solution for the terminal-phase intercept maneuver[45]. In the rendezvous mode,

the radar would feed information to the computer, which used it to calculate the velocity adjustments needed for final approach[46].

These examples of the use of the computer on a typical flight demonstrate that it was a relatively straightforward assistant in guidance and navigation. It permitted the Gemini astronauts to be independent of the ground in accomplishing rendezvous from the terminal-phase intercept maneuver to station keeping, a valuable rehearsal for the lunar orbit rendezvous required for the Apollo program. The astronauts participated in both the hardware and software design of the computer and its interfaces, and they were able to go to Owego and be put in the man-in-the-loop simulations. By flight time, like everything else in the cockpit, use of the computer was second nature.

THE IMPACT OF THE GEMINI DIGITAL COMPUTER

The Gemini Digital Computer was a transitional machine. Dale F. Bachman of IBM characterized it as the "last of a dying breed. It was an airborne computer, ruggedized, special purpose, and slow"[47]. Nonetheless, its designers claim an impressive list of firsts:

- The first digital computer on a manned spacecraft.
- The first use of core memory with nondestructive readout. The machine was designed in an era of rotating drum memories, its designers considered it a step forward[48].
- IBM's first completely silicon semiconductor computer[49].
- The first to use glass delay lines as registers[50].
- Technologically advanced in the area of packaging density[51].
- The first airborne or spaceborne computer to use an auxiliary memory[52].

Development of the Gemini computer helped IBM in significant ways. It contributed more than anything else to the hardware and software of the 4Pi series of computers[53]. This series eventually produced the computer used on Skylab and the AP101 used in the Shuttle. It also helped to develop IBM's reputation for delivering reliable and durable spaceborne hardware and software[54]. One Gemini computer restarted successfully after being soaked in salt water for 2 weeks. Another used system went on to NASA's Electronics Research

Laboratory in Boston for use on vertical and short takeoff and landing projects[55]. Coupled with IBM's involvement in the real-time computing centers used to monitor Mercury and Gemini missions, the company established itself as a major contributor to America's space program as it had been to the military research and development effort. Out of early military work came computer systems such as the Harvard Mark I, the 701, and SAGE computers used in air defense. However, even though identification with the space program has been maintained through several high-visibility projects, no significant commercial hardware products resulted as spinoffs.

For NASA, Gemini and its on-board computer proved that a reliable guidance and navigation system could be based on digital computers. It was a valuable test bed for Apollo techniques, especially in rendezvous. However, the Gemini digital computer itself was totally unlike the machines used in Apollo. With its Auxiliary Tape Memory and core memory, the Gemini computer was more like the Skylab and Shuttle general purpose computers. It is in those systems where its impact is most apparent.