7

The Evolution of Automated Launch Processing

Rocket technology is both old and new. Since the Chinese first started shooting off fireworks a millennium ago, the sight of a rocket streaking ever faster skyward, a comet's tail of fire behind, has excited even those unimpressed with machines. Fireworks rockets, and, later, military bombardment rockets through the first three decades of this century, shared the same components: casing, fuel, and payload. Construction was complete when the gunpowder fuel was loaded in the casing, warhead affixed, and a fuse planted the base. Such rockets could be stored without maintenance and fired with little preparation, needing only to assure that the fuse was still attached. The difficulty came in the area of guidance. A set of fins or a balancing stick passively guided the early rockets. Frequently they would turn on the men who launched them or shoot horizontally over the heads of fireworks watchers. Thus, the old technology of preparing rockets for flight consisted of keeping them dry, aiming them carefully, and lighting the fuse.

In Germany during the late 1930s the new technology of rockets began to mature. Increased interest in rocketry developed in Europe and the United States after World War I. Rocket societies flourished in England, Germany, and the United States. Robert Goddard flew a liquid propellant rocket, the first of its kind, in Massachusetts in 1926. Liquid fuels, with their higher specific impulse and thrust potential, soon replaced solid fuels as the primary area of propulsion research. Shortly after Hitler came to power, the German army established a rocket development program that led to the liquid-propellant A-4 (popularly known as the V-2). A-4 rockets far exceeded the capabilities of previous ones, terrorizing the populations of London and Antwerp in the latter stages of World War II. Over 14 meters tall and weighing over 12,000 kilograms, an A-4 carried nearly 1,000 kilograms of explosive payload up to 400 kilometers. Its guidance system was a radio beam-rider type with an electronic analog computer controlling vanes in the exhaust and elevons on the fins. If wind deflection caused the rocket to veer horizontally off course, the analog computer would calculate corrections and activate the vanes. Complex plumbing and turbopumps were needed to feed the engine with fuel. Experience gained in nearly 2,000 expensive failures led German technicians working on the A-4 to develop techniques of testing the many components of the rocket during manufacture and before committing it to flight. For example, the guidance system was tested at the factory by an electronic analog computer that simulated the flight of the rocket so that the system's reactions could be observed[1]. On the launching pad, engineers could test various moving parts of the vehicle by activating them using actual physical connections to the firing room.

German rocket scientists who came to the United States after World War II brought this new technology with them. Eventually based in Huntsville, Alabama, at the Army's Redstone Arsenal, they conceived an increasingly sophisticated series of rockets: Redstone,

Jupiter, Juno, and Saturn. Concurrently, the Air Force chartered the Atlas, Titan, and Thor ballistic missiles. During the 1950s, each of these vehicles was developed in programs marred by frequent flight failures. Actual numbers and the complexity of components grew by several factors over the A-4. The new devices and their failures led to more testing, both at the factory and before launch. The concept of a "countdown," during which each flight-critical component of the vehicle is systematically checked, reached a high level of efficiency.

As the 1960s began, most rockets and their payloads were still being checked out by discrete connections between the components and a test panel. When the countdown reached an advanced stage, particularly after fueling, the test engineers were cloistered in a blockhouse. Through cables from the rocket to the blockhouse, the engineers could monitor the status of various components and activate tests. An engineer would flip a switch, and something would happen, either on a dial or a strip chart, that he could actually see and interpret. When the first Saturn I rockets were launched and the Mercury spacecraft made their appearance, both in 1961, it became obvious that the level of complexity of both vehicles and payloads had reached the point where manual test methods were inadequate. Individual NASA engineers and managers on different programs began to evaluate the possibility of automating some of the checkout procedures using digital computers. Eventually, this led to the Shuttle's fully automated Launch Processing System.

The heart of the Shuttle is its computer system. Without it, no component of the spacecraft could be adequately tested or monitored. When a Shuttle is being refurbished after a flight in the Kennedy Space Center's orbiter Processing Facility, a large double hangar near the landing runway, the spacecraft's computers are connected to checkout and launch computers located in a firing room in the Launch Control Center. When moved to the Vehicle Assembly Building for mating with its fuel tank and solid propellant boosters, the Shuttle is reconnected to the firing room. After being transported to the pad, the final preparations are also controlled from the firing room. Finally, countdown and launch are executed from the same firing room. This scenario came after two decades of evolution, during which the role of computers became dominant both on board spacecraft and in launch processing. The integrated techniques exemplified in the Shuttle Launch Processing System developed from separate automated systems devised for vehicle checkout, spacecraft checkout, and telemetry monitoring. Important in the evolution is the part played by on-board computers. The journey toward full automation got great impetus from the Saturn and Apollo programs.

Figure 7–1. Launch processing facilities at the Kennedy Space Center: the Shuttle Orbiter Processing Facility (left), the Vehicle Assembly Building (center), and the Launch Control Center (right). (NASA 116-KSC 377C-82/41)

LAUNCH PROCESSING IN THE SATURN ERA

A Saturn V rocket with an Apollo spacecraft on top presented a magnificent sight, which engineers nonetheless viewed with a mixture of prideful awe and dread. No earthly booster since the Skylab launch has been as large or as powerful. Shuttles being mated in the Vehicle Assembly Building originally designed for the Saturn appeared as dwarfs in houses made for giants. It looked as though there was nearly enough room to stack them two high. The dread came from the fear of failure among the thousands of components, many capable of bringing disaster and killing a crew in flight. Early in the Saturn program automation began to be introduced in the testing of the gargantuan rockets. Marshall Space Flight Center acquired computers for Saturn vehicle checkout. Marshall also had responsibility for the Launch Vehicle Digital Computer (LVDC) housed in the Instrument Unit that was the last stage below the Apollo spacecraft on both Saturn IB and

Saturn V configurations. NASA's Launch Operations Center, later renamed Kennedy Space Center, acquired computers for telemetry data reduction and display and began work on the checkout systems used for the Apollo spacecraft, a project later transferred to the Manned Spacecraft Center in Houston. Each of these computer-controlled systems contributed to the concepts and development of the Shuttle Launch Processing System, now wholly based at the Kennedy Space Center.

Checkout of the Saturn Vehicle

Marshall Space Flight Center in Huntsville had primary responsibility for the design, manufacture, and flight preparation of the Saturn vehicles. In 1951, when Marshall was still the headquarters of the Army Ballistic Missile Agency, Kurt H. Debus formed a launch team that commuted to the Air Force's Eastern Test Range in Cape Canaveral, Florida. Within a short period of time, the frequency of launches made it necessary to establish a permanent group at the Cape, called the Missile Firing Laboratory. When Marshall was established on July 1, 1960, the Laboratory was renamed the Launch Operations Directorate. By 1962, the activities at Cape Canaveral grew to the level that the Launch Operations Center was formed separately from Marshall and given status equal to other NASA centers. However, its charter stated that the centers responsible for a particular vehicle or spacecraft had to perform its checkout and test, so during the Apollo era Marshall prepared Saturns and the Manned Spacecraft Center worked on the Apollos. Personnel at the Launch Operations Center performed facilities management and provided telemetry data reduction.

Computers were used both on-board the Saturn vehicles and in preparing them for flight.* Ten Saturn I vehicles were launched between 1961 and 1965. Each was unmanned, the series being used

*For a complete description of the evolution of the Saturn and its components, see Roger Bilstein, *Stages to Saturn: A Technological History of the Apollo/Saturn Launch Vehicles*, NASA SP-4206, 1980. Chapter 8 centers on the use of computers in checkout and the development of the Instrument Unit and its flight computer. Chapter 16 of Charles D. Benson and William B. Faherty, *Moonport: A History of Apollo Launch Facilities and Operations*, NASA SP-4204, 1978, describes the development of automated launch operations. Due to these prior treatments, my account will concentrate on briefly summarizing the use of the computers to provide the necessary introduction to the section on the Launch Processing System, rather than retelling the whole story. Some new evidence is presented where applicable, but the reader is urged to consult both previous works.

primarily to demonstrate that clustered-engine first stages and high-energy upper stages were feasible. The first five launches did not use a computer for guidance. Each was a suborbital mission utilizing a German-made mechanical time-tilt device for control[2]. On the fifth flight, an ASC-15 computer, built by IBM originally for the Air Force's Titan, flew as a passenger and handled telemetry transmissions[3]. It guided the last five missions, several into earth orbit. When Saturn evolved into the IB and V series, an Instrument Unit containing the LVDC was mounted atop the S-IVB stage on each vehicle. Termed the "integrating element" of Saturn, IBM was not only responsible for its computer but for its construction[4]. Besides the computer, the Instrument Unit contained the Launch Vehicle Data Adapter as an I/O front end, analog control circuits and an ST-124 guidance platform. On lunar missions the LVDC guided the spacecraft until the S-IVB stage separation after the lunar trajectory insertion.

IBM's LVDC was architecturally quite similar to the Gemini guidance computer[5]. It used nearly the same instruction set, 26-bit data words and 13-bit instructions. One difference was that the memory had two-syllable locations instead of Gemini's three. Construction of the LVDC, however, was radically different. For reliability reasons, triple modular redundant (TMR) circuits were adopted. Even though the component count went up just 3.5 times, the reliability increased 35 times[6]! Three logic channels, each with seven functional modules, required 395 voters[7]. Packaging the computer used techniques developed under the Advanced Saturn Technology Program commissioned by Marshall and executed by IBM[8]. First of the "flat pack" integrated circuit series, IBM applied this silicon semiconductor technology in its System 360 commercial machines[9].

Use of a computer in the launch vehicle led directly to using ones for checkout. Marshall bought an RCA 110 to communicate with the IBM ASC-15 used in the Saturn I. Later, RCA upgraded its machine by enlarging the memory to 32K 24-bit words of core and an additional 32K on an associated magnetic drum. When the Saturn IBs began to be launched, discrete circuits for interfaces with the rest of the launch vehicle were added[10]. Renamed RCA 110As, these computers continued to be augmented to handle more communications circuits, so that by the time Saturn Vs appeared, the computers could maintain the status of each of 1,512 signal lines[11]. At first the 110s simply handled communications and switching. Activating test procedures and conducting tests were still done manually. But in 1962, IBM suggested that Chrysler convert the 110s they used for stage checkout of the Saturn I to do the tests automatically[12]. Even though the advantages of automating procedures seem obvious, chief among them the fact that all are done exactly alike, it was difficult to get people responsible for checkout to convert from doing things manually, a

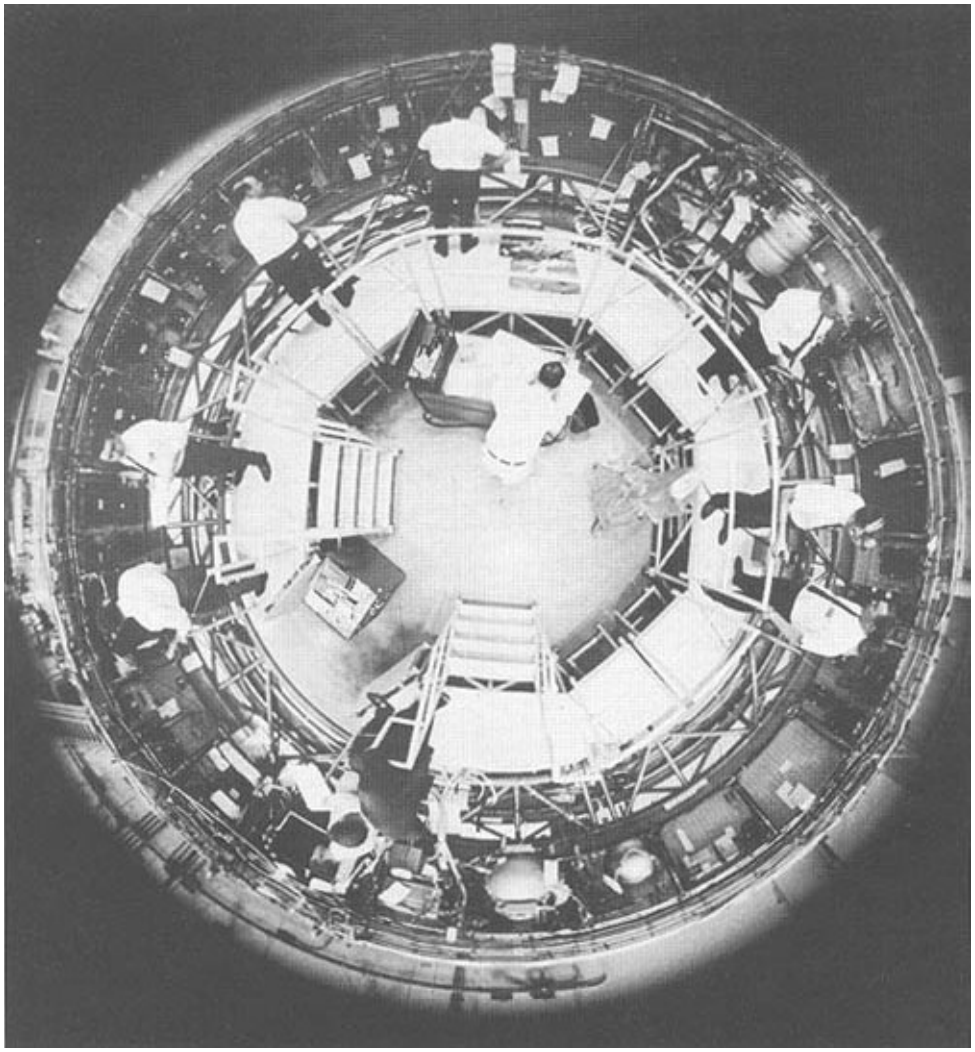

Figure 7–2. IBM engineers work inside the Saturn launch vehicle Instrument Unit. (IBM photo)

theme repeated in other parts of the Apollo program[13]. At that time, computers were seldom used for on-line work, and most engineers were still unfamiliar with them and wary of any more innovations in an already innovative program. However, Chrysler converted some factory tests to automatic, using a special language, "HYLA," to define them. Additionally, several Packard Bell computers connected to a common memory automatically checked out parts of the Saturn I. Use of a common memory as a computer interconnection device reappeared in several later systems and is critical to the success of the Shuttle's Launch Processing System. Engineers wrote the language "SOL," or Systems-Oriented Language, for the Packard Bell machines.

By late 1962, the Saturn V stage contractors accepted the concept of automatic checkout and settled on a common system, the Control Data Corporation CDC-924A computer, as the factory test machine, with 110As assigned to the S-I stage and for the assembled vehicle at the launch site[14].

By this time, it was clear to Ludie Richard, a NASA engineer, and his team at Marshall that preparing a language to help test engineers write automated procedures was the key to continued acceptance of the principle. A custom-designed programming language would leave control over the definition of the tests in the hands of the engineers, avoiding communication problems that might arise with computer programmers inexperienced in checkout techniques[15]. IBM eventually wrote routines for the RCA computers in assembly language, but the majority of the automated tests were ATOLL (Acceptance, Test, or Launch Language) programs stored on tape. Richard acquired the over two dozen RCA 110As that were eventually used. His deputy, Charles Swearingen, was put in charge of managing the flight computer, ground computer, and checkout software[16]. James Lewis and Joseph Medlock were instrumental in developing the checkout systems and defining ATOLL[17]. IBM wrote both the flight programs and the Saturn Operating System that ran on the RCA computers and executed ATOLL procedures.

By mid-1963 the final configuration of the Saturn checkout computers was set by Richard's group. At Launch Complex 34, the Saturn IB launch site, one master RCA 110A was in the blockhouse and a slave underground at the pad. For Saturn Vs at Complex 39, one RCA 110A was located in each of the four firing rooms in the Saturn Launch Control Center, which was attached to the Vehicle Assembly Building in which the Saturns would be stacked. Each of four mobile launchers also contained a computer. In addition to the 110As, the firing rooms also had a DDP-224 minicomputer as a display driver for the CRTs showing output data to the engineers, as well as a controller for slides and other visuals. Computers in the mobile launchers could be used for checkout in the Assembly Building as well as at the pads, a foreshadowing of the later Launch Processing System. Due to reliability problems with the 110As, the launcher computers used a dual memory configuration. Checkout programs filled just half the memory, so the other half acted as a duplicate for redundancy, the same principle as applied to the LVDC memory.

Part of the credit for the perfect success record of the Saturn vehicles (*all* Saturn I, IB and V boosters flew without a failure) must be due to the effectiveness of the checkout procedures. Without automatic testing the confidence in the rockets could not have been attained, since they were too complex for effective manual procedures. In addition to checkout methods specific to the launch vehicle, the launch directors in the firing rooms had access to automated test data from the spacecraft

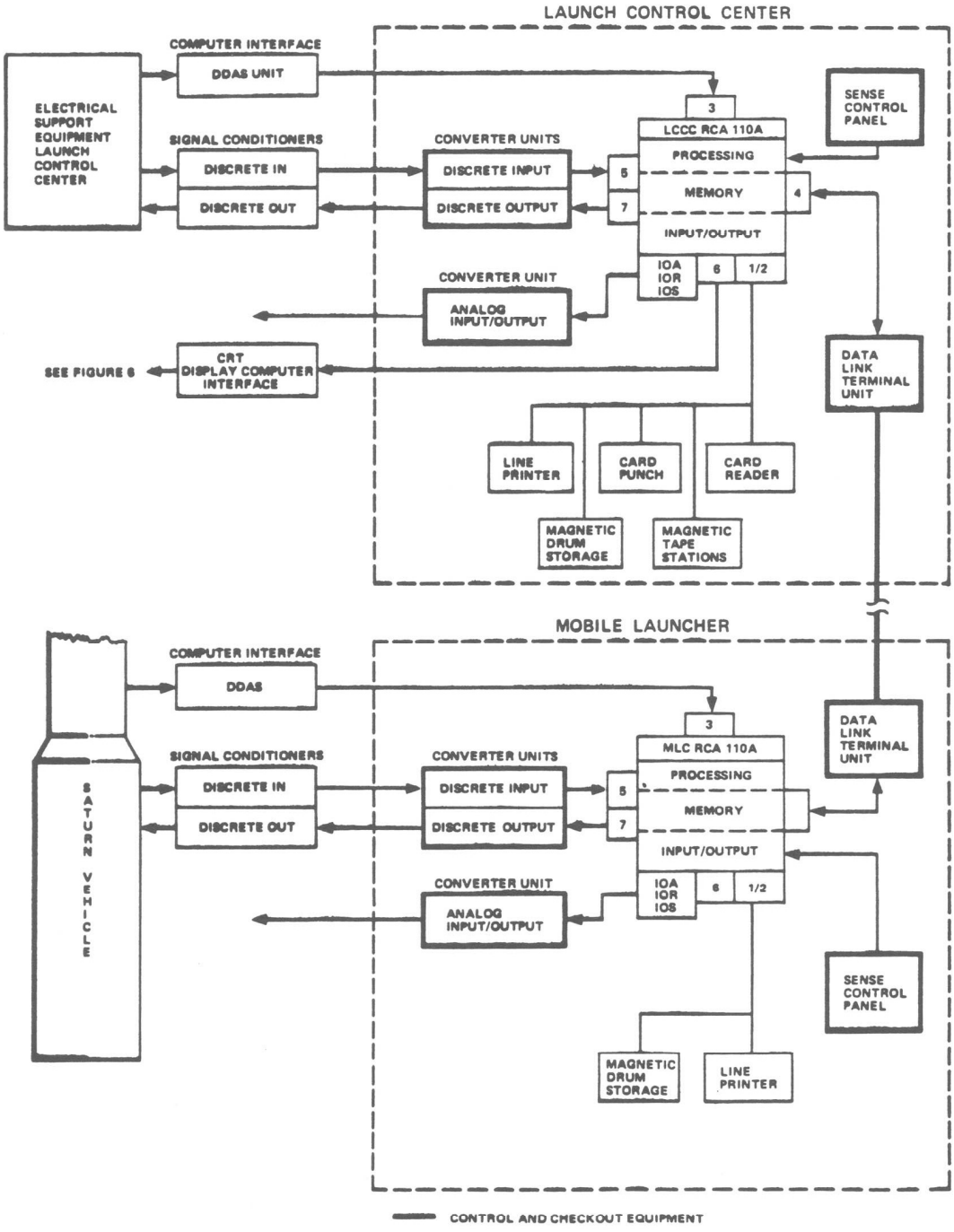

Figure 7–3. A block diagram of the automated preflight checkout hardware for the Saturn launch vehicle. (From IBM, *SLCC Programming System*)

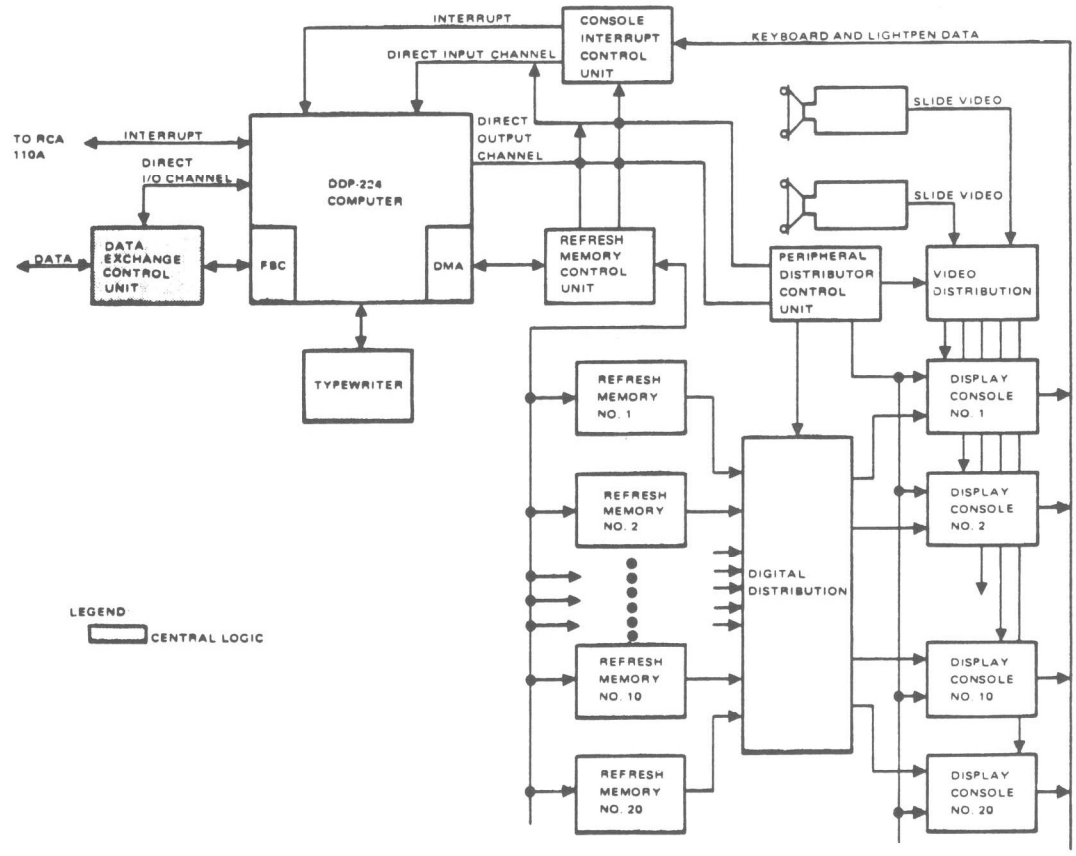

Figure 7–4. A block diagram of the Saturn Operational Display System. (From IBM, *SLCC Programming System*)

preflight test equipment developed by both the Launch Operations Center and Manned Spacecraft Center.

Development of Apollo's Acceptance Checkout Equipment

From the first Apollo earth orbital flights through the lunar missions, Skylab, and the Apollo-Soyuz Test Project, ground testing and countdown support of the spacecraft and its associated systems were the responsibility of the ACE, or Acceptance Checkout Equipment**.

**The acronym ACE evolved from PACE, or Preflight Acceptance Checkout Equipment, which appears in some of the literature. It was discovered that the name conflicted with a commercial product, so the "Preflight" was dropped. Prior to PACE, there was a short period when the equipment was known as SPACE, but apparently not officially.

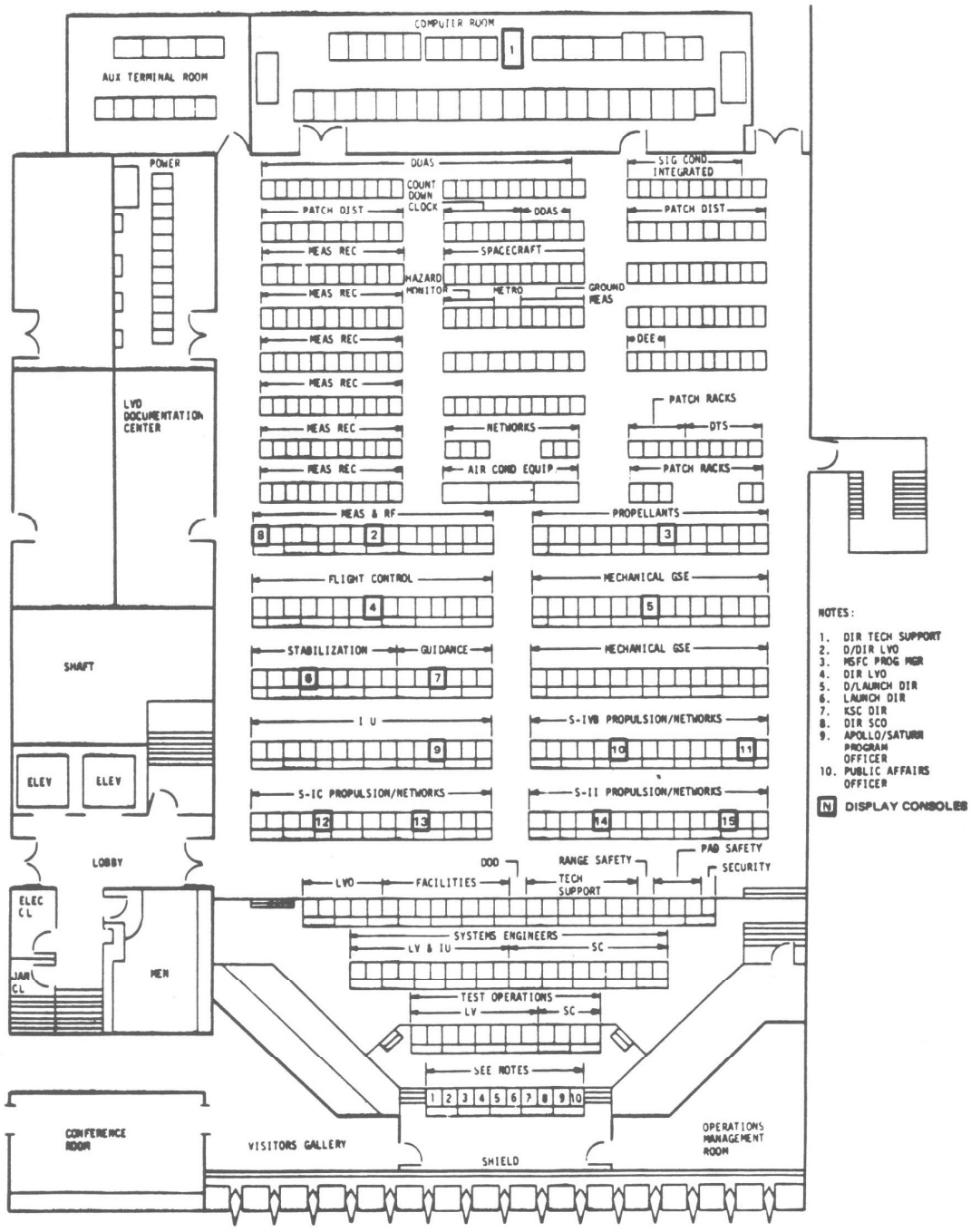

Figure 7–5. Typical firing room layout during the Saturn era. Nearly 250 engineers would crowd this area during a countdown. (From IBM, *SLCC Programming System*)

ACE stations were located in the Apollo Operations and Checkout Building in the Industrial Area of the Kennedy Space Center, at Launch Complexes 34 and 37, at the Johnson Space Flight Center, and at North American Aviation and Grumman Corporation assembly plants. Two of the North American stations were modified from use in assembling the Shuttle in Palmdale, California[18].

ACE resulted when a spacecraft checkout engineer figured that there had to be an alternative to manual methods. Thomas Walton transferred from a job in the computer room at Langley Space Flight Center to the Cape Canaveral Launch Operations Center in early 1961. Assigned to the checkout of Mercury spacecraft, by September he had enough of manual testing and applied his background in computers to devising another way of doing things[19]. Walton convinced his boss, Harold G. Johnson, to let him build a digital ground station for telemetry from the capsule. Using the Mercury missions as prototypes, he proved that digital equipment could display the engineering data previously shown on dials and strip charts. His success led to a search for a computer system to handle the data in real time. With NASA's Gary J. Woods, he traveled to several companies in search of a machine. Walton did not believe that a computer like an IBM mainframe of that era could do the job. Since they were designed for large-scale batch processing, the difficulties of adapting such a computer to the real-time world of telemetry displays and automated checkout would be too great. Instead, he and Woods looked for simpler minicomputers such as the Digital Equipment Corporation PDP-1 and the Control Data Corporation CDC-168[20]. Walton convinced the Gemini Project Office to buy a pair of CDC-168s to be used for checking out their spacecraft. Meanwhile, plans continued to create a system dedicated to the Apollo.

Marshall and Launch Operations personnel met in 1963 to determine whether the checkout equipment for both the Saturn vehicle and the Apollo spacecraft could be combined. Richard's and Walton's teams decided to continue separate paths[21]. The results were the Saturn checkout system and the first ACE unit using General Electric discrete equipment and CDC-168 computers going on line in late 1964.

Although the first ACE stations were under construction, a small political battle was raging over who would have ownership of the program. Joseph Shea, then at NASA Headquarters in Washington, wanted to control it from the Apollo Project Office there. Transferred to Houston in 1963 to take over management of Apollo, he moved the ACE development group of between 50 and 60 persons there instead[22]. This act reinforced the feelings that the Kennedy Space Center was to be strictly an operations center, staying clear of research and development activities.

Figure 7–6. An ACE station with twin Control Data computers. (NASA photo 107-KSC-67C-919)

Each ACE station used two digital computers with a common memory. One was the Digital Command Computer, which processed commands from the control or firing rooms to the spacecraft and was interconnected with the ground support equipment. A second machine was the Data Processing Computer, which drove the displays and controlled peripherals. Memory could be directly accessed by discrete circuits in the ground station, so data for both computers could be placed there[23]. ACE stations could function in manual command mode, semiautomatic, or fully automatic with manual override. Stations in the Kennedy Industrial Area serviced the spacecraft both before and after mating with the Saturn V boosters. When Launch Complex 39 went into use, checkout wires carrying digital formatted data ran over 15 kilometers from the ACE stations to and from the pad and firing rooms in the Saturn Launch Control Center using a video wideband transmission system[24]. Of course ACE had to cooperate with the RCA 110As at various points, so interfaces between the different computers consisted of dedicated I/O registers, sense lines, and priority interrupts[25]. ACE also had to talk to the Apollo guidance computers in the command module (CM) and the lunar excursion module (LEM). On the average, the CM computer operated for 50 hours in support of the countdown. A CRT display controlled by ACE duplicated the data shown on the display and keyboard while the Apollo computer was in operation[26].

Walton judged that the development of ACE did a lot to stimulate the technology of on-line processing. Certainly it helped create techniques of interconnecting multiple different computer systems. Also, this was one of the first times that data transfers in the megabit range were accomplished over distance.

Digital Displays of Telemetry

Telemetry transmissions from the vehicle are one important source of data for rocket engineers. In the early days of rocket flight research, the causes of failures often could only be guessed. When the larger size of later rockets made it possible to carry radios for sending back data, sensors were added to supply engineering data from critical components throughout the flight of the vehicle. If a failure occurred, it was often possible to determine which components contributed most to it by examining the reams of data sent back and originally recorded in analog form on the ground. However, Tom Walton's pioneering digital ground station for Mercury displayed the data in processed digital form. In 1962, the Atlas-Centaur project automated postflight telemetry data reduction[27]. By the mid-1960s, digital telemetry displays were standard at Kennedy Space Center, provided by a pair of mainframe computers in the Central Instrumentation Facility.

Kennedy acquired two General Electric 635 computers for telemetry monitoring and batch processing of institutional programs. GE 635s were 36-bit processors capable of double-precision arithmetic[28]. Programmers prepared separate code for each of the Delta, Atlas-Centaur, and Saturn flight vehicles. Delta and Atlas launch pads, as well as Complexes 34, 37, and 39, could be switch connected to the computers at any one time. Forty different displays were possible and could be transmitted to the appropriate blockhouse or Launch Control Center firing room[29]. NASA's Bruce Miller was in charge of systems programming for the GE computers, with Bradley Hughes as the chief scientific programmer.

These computers had the longest operational life of any installed at the Kennedy Space Center. GE delivered the first machine in late 1965. A second came on line in early 1966. Until May of 1983—18 years later!— one was still in use driving blockhouse displays for Delta and Atlas-Centaur. GE had long been out of the computer business by then, having sold its digital computer division to Honeywell in the early 1970s. Kennedy retained a permanent systems programmer from GE (who later moved to Honeywell) to keep the operating systems going and used a retired blockhouse 635 from Wallops Island as a source of parts[30]. From the beginning the computers had a dual operating system. Batch institutional jobs could be run at the same instant a

real-time telemetry program was running, except when a Saturn was being supported, as its program was so big it pushed out the batch programs. When Kennedy Space Center officials searched for a second machine for the Central Instrumentation Facility, they considered other vendors. IBM's branch manager in Cape Canaveral, W. O. Robeson, sent a letter suggesting a System 360/50 as an administrative computer, pointing out that evidence from prior telemetry computers indicated that they rarely failed[31]. The dual operating systems could then be abandoned. However, Kennedy bought the second 635 to provide a redundant backup anyway, accepting the loss of batch processing during Saturn operations.

Telemetry data reduction computers thus provided yet another source of information to the launch directors in the Apollo/Saturn era. Still, some engineers were convinced that the computer data were never accurate, just as their colleagues in the checkout world had to be dragged into automation[32]. Regardless, telemetry displays became an integral part of the technology of launch processing.

Impacts of the Apollo/Saturn Era on the Shuttle Launch Processing System

Developing the major computer components of the launch processing system for Apollo/Saturn provided software contractors such as IBM and the Kennedy Space Center staff valuable experiences later transferred to the Shuttle Launch Processing System and onboard software for the Shuttle program. Additionally, some techniques known in theory, but never properly applied, found justification during the Apollo/Saturn programs. The areas of impact included the modularization of software, lessons learned by IBM as a key future contractor, and Walton's continued influence on ground computer concepts.

Software written for the LVDC and the GE 635 computers started as single monolithic programs and evolved to modularized programs at just about the same time. Flight software for the ASC-15 computer used on the Saturn I vehicles was necessarily monolithic because it had to be sequentially executed and strictly timed[33]. Any changes impacted on the execution time, and therefore had to be carefully integrated. The computer could not handle waiting for an interrupt to instigate an action. Actions had to be initiated by the program relative to its starting time. When the ASC-15 gave way to the LVDC, a more powerful and flexible machine, programmers continued in the monolithic mode. Finally, IBM staff realized that by preparing the software in essentially free-standing chunks, the impact of changes would be limited to the modules and not spread side effects throughout the software. This discovery came late in the Saturn program but early enough to affect the development of

the Skylab on-board software. Also, IBM separated the modules into groups consisting of the control subsystem and applications subsystem, which is a prototype of the Shuttle on-board software organization[34]. IBM helped transfer this concept to the Shuttle by moving people such as Kyle Rone and Lynn Killingbeck from working on the Saturn computer directly to the Houston office to support the Shuttle software development. NASA also independently moved toward modularization when, in 1973, it broke down the programs used on the GE 635s to support telemetry data reduction. Before then, it took an average of 3 months to implement a simple change in the monolithic version of the program, because of the massive debugging necessary to eliminate side effects[35]. Thus, modularization came to be expected by NASA as part of software design. If modularization was not used on the Shuttle on-board software, preparing new flight loads would have been impossible within the projected time between flights of an individual orbiter.

Besides modularization, Apollo/Saturn significantly influenced IBM's later work on the Shuttle's on-board software, especially the company's design of the system used for Shuttle launch processing. IBM summarized its conclusions in a document released in late 1972, just at the time both Shuttle ground and on-board software contracts were being let[36]. The study recommended that the vehicle's flight software be capable of reloading all programs on board[37]. This was implemented on the Shuttle, as the mass memory units (MMUs) contain all preflight and flight software for the primary avionics computers, the display computers, and the engine control computers. Ground software recommendations required that all checkout functions use a higher order language and that checkout be conducted using one computer system[38].

During Saturn, both ATOLL and machine language programs controlled preflight tests, the machine language routines absorbing a considerable amount of development and maintenance time. This lesson helped spur the creation of an improved checkout language, GOAL. In regard to consolidating all functions in one computer, IBM thought that the difficulties of integrating the two RCA computers, the DDP-224 display computer, and the telemetry reduction computers were excessive. By taking that position, IBM found itself squared off against the distributed concepts envisioned by Tom Walton and his team for the Shuttle system. Walton refused to move to Houston when Shea transferred the ACE team. By staying at Kennedy, he was able to influence the structure of the Shuttle Launch Processing System and help make the Center fully responsible for all checkout and launch operations for the entire vehicle, a significant change from the Apollo/Saturn program.

THE SHUTTLE LAUNCH PROCESSING SYSTEM

When NASA began planning for the Space Transportation System (STS), it espoused ambitious requirements, such as an eventual launch rate of 75 per year. A projected fleet of three orbiters would be limited to a maximum 2-week turnaround between flights and a 2-hour countdown in order to achieve that many firings[39]. Compared to the 5-month checkout of a Saturn V and its 3-day countdowns, this seemed outrageous, especially since the Shuttle would be no simpler than an Apollo/Saturn. NASA put considerable effort into examining commercial aircraft maintenance techniques to see what could be adopted for Shuttle use. One study indicated that only 53% of the tests done on a Saturn V would need be repeated if the spacecraft were reusable[40]. Even with this reduction, nearly 46,000 measurements have to be made and monitored in real time in the process of preparing a Shuttle for launch[41]. Clearly, there was no way NASA could do the Shuttle checkout with Apollo concepts[42]. As Henry Paul, who headed the Launch Processing System development for NASA, said, "Automation. . . becomes a requirement for operations, not an elective"[43]. Still, some engineers needed to be convinced that hardwired testing could be successfully eliminated, even though the last 20 hours of a Saturn countdown was 85% automated[44]. Building the present system, during which almost all preflight testing and preparation is done under control of software, and in which much of the countdown, sometimes even including the calling of "holds," is done by computing machinery, was a remarkable effort[45]. One of the biggest changes from the Apollo/Saturn preflight checkout systems is that Kennedy Space Center became responsible for the development of the Launch Processing System. Given the organization of NASA at the time, this was one of the biggest surprises as well.

Kennedy Space Center Gets the Job

During the late 1960s NASA began studies of the configuration of the eventual STS. Most designs were predicated on a winged booster, which would return to the launch site immediately after separation from an orbiter with internal fuel tanks. Such a design could theoretically be launched from anywhere in the United States isolated enough to handle aborts safely. Project staff examined a number of sites and made projections of the cost of an "ideal launch site" that would have all the

facilities necessary for handling the Shuttle. Chief among these were a hypergolic and cryogenic fuels facility, a hangar for the orbiters and boosters, a mating building, a control center, the launch pads, and a runway with a safing bay for emptying residual fuels after landing. One study placed the cost of a new facility with these characteristics at $1.9 billion. On the other hand, modifying existing Apollo/Saturn facilities at Kennedy and adding new equipment where needed would cost $355 million, a significant savings[46]. In March 1972, NASA selected the solid rocket booster/external tank configuration for the Shuttle. All inland launch sites were thus eliminated, and just Kennedy Space Center, Vandenberg Air Force Base, and a site in Texas remained under consideration[47]. Since existing facilities could be modified at both Vandenberg and Kennedy, the cost-conscious administrators settled on those two launch sites. Vandenberg was expected to handle polar orbit launches and most military payloads. Kennedy would launch eastward, continuing the established situation and giving Kennedy the opportunity to try for the development of the checkout system.

Phase B Shuttle studies conducted by a number of contractors included concepts of the checkout system[48]. Some hinted at the direction the eventual system would take. One pointed out the need for efficient and simple man-machine interfaces, and called for having ATOLL, FORTRAN IV, and COBOL compilers available to the engineers[49]. Kennedy's own early study, based on Rockwell and McDonnell-Douglas Shuttle configurations, called for a central data processing facility connected to every part of the Shuttle handling equipment, including a mission simulator on site and by communications link to Mission Control in Houston[50]. Meanwhile, remnants of the old ACE group at Johnson had started work on a Shuttle checkout system.

A "Checkout Systems Development Lab" at the Johnson Space Center did research on new concepts of preparing manned spacecraft for flight[51]. Individual BIC, for "built-in checkout," cells would be located at test points throughout a spacecraft, each cell with I/O registers. Automated tests would read and write to these cells. Johnson's development team wanted a single central computer to be connected to several sets of Universal Test Equipment consoles and thence to the Shuttle[52]. General Electric built a prototype of a "Universal Control and Display Console" for the Laboratory. Each control console would have two color display tubes and be capable of supporting tests on any specified parts of the spacecraft[53]. The system was similar to earlier Apollo/Saturn concepts, with a big computer in the middle doing all the testing and displays, communicating with the spacecraft, and so on. One improvement was that the Universal Test Equipment meant that units could be mass produced and assigned to different checkout tasks without significant hardware changes. When the time

came for the Shuttle project office at Johnson to make a decision about preflight checkout, the hometown lab made a proposal that was "underdeveloped" and vague[54]. Most likely, the engineers in the Development Lab thought the job was theirs because in all previous programs the center responsible for the spacecraft was responsible for checkout. When Kennedy had tried to do some ACE development, it was moved to the center responsible for the Apollo. Therefore, a full-blown proposal did not seem necessary. They were in for a surprise.

Impetus to make Kennedy the development center for the Launch Processing System came from many levels. The center's director, Kurt Debus, made his support clear to his engineers in 1972[55]. Walton saw a chance to do another ACE, but this time as a fully integrated system for all parts of the spacecraft. The consensus was that by having Kennedy Space Center do the development, much money would be saved and civil servants would be more actively involved[56]. Even though the work originated with Walton's Design Engineering Directorate, talent for developing the Launch Processing System came from across the Center[57]. A study group of about half a dozen engineers, led by Theodore Sasseen and including Henry Paul, Frank Bryne, George Matthews, and others who had key roles in the later implementation of the system, met and began work on a prototype[58].

Making the prototype turned out to be one of the key factors in landing the Launch Processing System development job for Kennedy. The engineers made a small model of a liquid hydrogen loading facility, with real valves and tanks. Using a Digital Equipment Corporation PDP 11/45, they devised software that graphically displayed a skeletal view of the piping and valves, with actual pressures printed next to the appropriate valve. The prototype could transfer fuel to the model spacecraft under software control, with the user able to monitor flows and pressures at the console. Confidence in their ability to create automated procedures encouraged the engineers, and they also now had a physical version of their system to help in selling it. Johnson's Universal Test Equipment had no counterpart in terms of functionality.

The prototype represented a single, and complete, part of the total system, a system quite different in concept from previous ideas. Launch processing and mission control prior to the Kennedy developments were based on using a minimum number of mainframe computers. Frank Byrne had the technical vision to develop a distributed computing system, in which dozens of small computers would do the checkout functions. Walton provided the leadership and tenacity to hold to the concept and see it put into place[59]. Several important advantages result from using distributed computers. First, the tasks more closely fit the power of the machine. Using a mainframe computer for relatively simple procedures such as solid rocket booster checkout would be overkill[60]. Second, a distributed system would free software developers

from worrying about fitting their programs in with others in a big machine's memory. Each discipline, such as engines, cryogenics, and avionics, would have a separate console[61]. Third, parallel testing could be done[62]. A mainframe would have to be inordinately large to contain all the checkout programs. Therefore, they would have to be loaded and run serially, as in the RCA 110As, defeating the short countdown requirement. Finally, Paul was convinced that overall hardware costs would be reduced compared with mainframe configurations[63].

In 1972 Robert F. Thompson was the head of the Shuttle project office at Johnson and in charge of deciding where to place the checkout development. Faced with a choice between a homegrown system similar to tried and true predecessors and a new concept developed at Kennedy that even had opposition there, he ruled in favor of Kennedy's proposal against the opinions of his advisors. The winners are gracious toward Mr. Thompson, calling him an "honest manager" and a "nonterritorial individual"[64]. Thompson judged Kennedy's to be the best proposal, but he also thought it more efficient for NASA to develop the Launch Processing System where it would eventually be used and by the people who would use it.

Getting Started: Contracting For the Launch Processing System

Due to the earlier site studies and the building of the prototype, Kennedy Space Center had a good idea of what it wanted in the Launch Processing System. Reflecting the detailed requirements developed for the Shuttle on-board computers, the Design Engineering Directorate's engineers started in March 1973 to prepare the "Launch Processing System Concept Description Document"[65]. Released in October, the document specified the architecture and concepts of the System in detail, before any major contractor involvement[66]. Kennedy's efforts on the Launch Processing System are reflected by the fact that nearly 100 civil servants were involved in the planning between 1973 and the March 1976 freeze of the design[67].

Plans for the System included extensive remodeling of Saturn facilities. The Processing System itself is largely contained in the Launch Control Center. Hardware is divided into the Checkout, Control, and Monitor Subsystem (CCMS), the Central Data Subsystem (CDS), and the Record and Playback Subsystem (RPS). Small, task-dedicated computers are in the four firing rooms of the Control Center and are the primary component of the CCMS. Large mainframe computers located on the floor below the firing rooms make up the biggest part of the CDS. NASA's Joseph Medlock, Thomas Purer, and Larry Dickison

envisioned test engineers developing their own procedures using an engineer-oriented language like ATOLL in concept but better and easier to use[68]. These procedures would then be developed on the mainframes and tested against simulations stored on the mainframes. When verified, they would be included in the system and stored on disk. When a firing room became active to support a vehicle, the engineer would load his test procedure from the mainframe to the minicomputer attached to his console and execute it from the console.

Depending on which spacecraft subsystem is involved, the tendrils of the Launch Processing System may follow it wherever it goes on the Space Center site. The firing rooms are connected to the Vehicle Assembly Building, the launch pads, the Cargo Integration and Test Equipment, and the new orbiter Processing Facility, a two-bay horizontal hangar. At each location, hardware interface modules make it possible to test and monitor the orbiter and other parts of the spacecraft from the firing rooms. So the System is locally distributed computationally, but physically centralized—especially compared with the RCA 110As and GE 635s of the Saturn era.

One critical side effect of using a mix of mainframe data base machines and minicomputers for individual system checkout, as well as the need to talk to a pervasive on-board computer system, was that for the first time, several different network architectures had to be combined into one[69]. The inherent difficulties involved led NASA to award the software contract before choosing hardware so that the software contractor could help in the computer selection[70]. Further, the minicomputers were chosen apart from the contract for the consoles and other hardware associated with the CCMS. Four source selection boards eventually convened: one each for software, minicomputers, the CDS, and the CCMS[71].

Since test engineers would write the applications software, the software contractor would be primarily responsible for the operating system under which the applications would run, the new test language, GOAL (for Ground Operations Aerospace Language), and any modifications to the microcode for the minicomputers and other equipment needed to successfully connect them. Interfacing largely became a software problem because the changes were to be implemented in microcode. Six contractors tried for the job, with IBM beating out General Electric, TRW, Computer Sciences Corporation, McDonnell-Douglas, and Harris Computer Corporation[72]. The initial $11.5 million contract ran from May 1974 to March 1979[73]. This contract was extended several times due to delays in launching the first Shuttles, but IBM's involvement ceased in the operations era. The company did its usual good job, and users of the eventual system believed it fulfilled the requirements[74]. IBM used a top-down structured approach in designing the software, holding weekly formal reviews during the development stage so that NASA could closely monitor activities[75].

By winning the software contract first, IBM found itself in the unusual position of having to program other people's computers. One IBM employee said that his company was encouraged *not* to bid on the hardware contracts[76]. According to Byrne, IBM was not kept out of the hardware bids so much as they lacked a suitable minicomputer to offer. The System 34 was under development at that time, as was the Series/1, but IBM chose not to make its new minicomputers public[77]. Three companies made the final round of bids on the minicomputers: Prime, Varian Data Machines, and a small new company called Modular Computers, Inc[78]. Design Engineering had built a prototype of a launch processing console set for the solid rocket boosters using Prime computers. (Later shipped to Marshall for awhile, it finished its career in the Vehicle Assembly Building nearly 10 years after construction[79].) Because of this, many thought Prime had the contract won, but it was edged out by Modular Computers, much to the surprise of Byrne and Walton[80]. ModComp initially contracted for 60 machines at a cost of $4.2 million, a number later extended considerably as console sets were placed in all four firing rooms, the cargo integration facility, the Shuttle Avionics Integration Lab (SAIL) at Johnson, and the hypergolic maintenance facility, as well as at Vandenberg. Two months after the computer contract was let in June of 1975, Martin-Marietta defeated Grumman Aerospace Corp., Aeronutronic Ford, and General Electric for the remaining CCMS hardware.

By November 1976, IBM received the first minicomputer for software development, and by February of 1977, the first station for GOAL applications development was delivered[81]. Honeywell won the CDS hardware contract in the fourth quarter of 1975, and John Conway of NASA managed the acquisition of equipment and personnel for that Subsystem during 1976–1977[82]. By 1977, the Launch Processing System began to take physical shape.

The Common Data Buffer: Heart of the System

Most diagrams of the physical components of the Launch Processing System show an inordinately large rectangle at the center of the drawing, with all other components either directly or indirectly connected to it. That rectangle represents the common data buffer, which Thomas Walton called the "cornerstone of the system"[83]. The biggest problem with creating distributed computing systems is devising a method of intercomputer communication that is reliable, fast, and simple. In a system such as the Launch Processing System, which depends on a number of computers "knowing" the same data about the spacecraft, some method of protecting and centralizing the common data is needed.

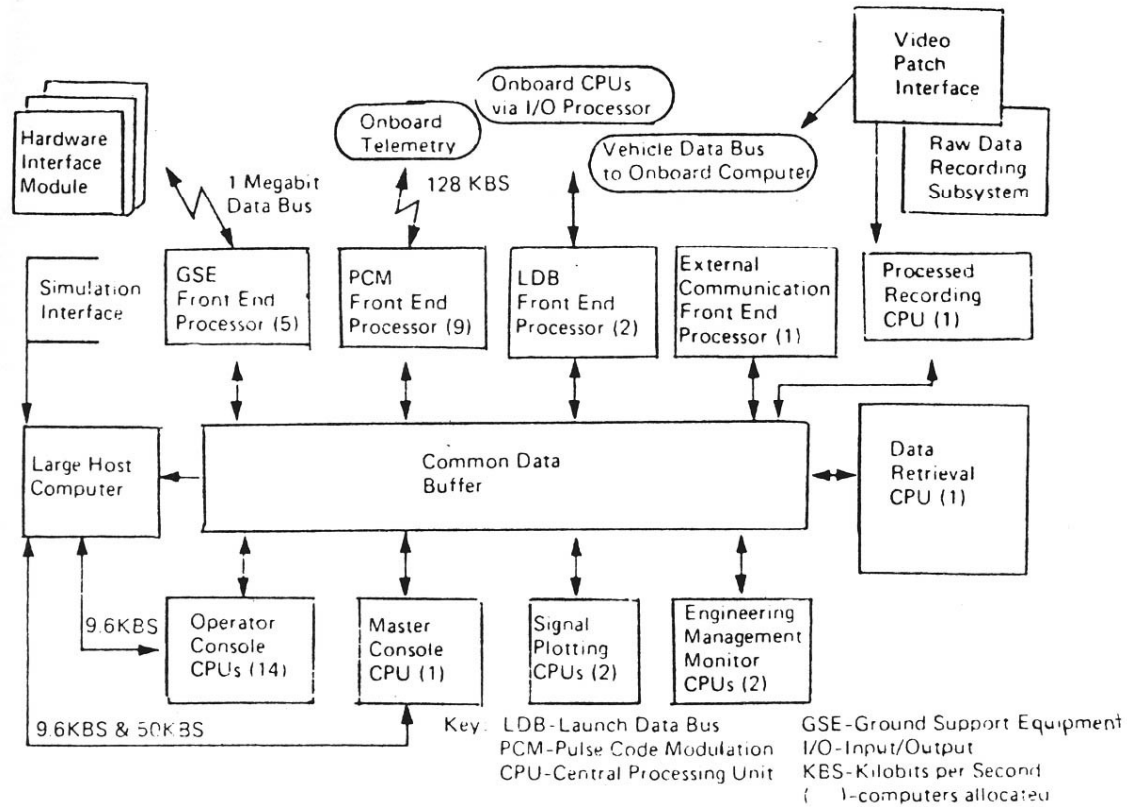

Figure 7–7. Shuttle Launch Processing System hardware structure. (Courtesy IBM)

Frank Byrne, who was involved in the planning for the Processing System from the start, took on the job of designing a device to keep track of commonly needed data that also made it possible for the various computers to communicate with each other. Kennedy followed a plan to use commercially available equipment in as many parts of the Launch Processing System as possible. Minicomputers and mainframe computers are used largely unchanged. However, since no organization had tried to closely connect such large numbers of machines, some of which were quite different in architecture from the others, there was no commercially available solution to the common data problem. Byrne had to design one on his own: the common data buffer.

Byrne noted that as the number of computers in a distributed system increases, the complexity of intercomputer communication increases. He wanted to remove the complexity. By placing the common data at a central location he eliminated the need to update multiple copies of the data in separate machine memories. Possibly he got the idea for a common data area from his work on the GE 635s in the Central Instrumentation Facility. Those machines had a "data core" which could be accessed by both[84]. ACE stations also used common

data areas. Basically the common data buffer provides each machine in the system with a set of "post office boxes." Specific parameters, such as valve pressures, voltages, and fuel levels, are assigned a location in the buffer memory. Each machine can read any location in the memory, but only the machines explicitly assigned to the task of maintaining a certain parameter can write to that parameter's location. That way a secondary machine cannot spuriously change a parameter's value. Programmers do not have to worry about where any particular parameter is kept. As long as it is referred to by its proper name in a GOAL program, the system build process will assign it its correct address as the program is compiled and integrated. In addition to acting as a common storage area for data, the buffer maintains the entire system interrupt stack and flags and status variables. Since it is centrally located, it is also used to temporarily store application programs as they are loaded from the repository in the CDS to the individual minicomputers. As such, it acts as a "way station."

Box 7-1: Inside the Common Data Buffer

Even though the common data buffer is a unique design, it uses standard commercial chips and boards. Nothing was custom built[85]. Memory chips are made of negative metal-oxide semiconductors (N-MOS), with each section consisting of 64K of one bit and 32 sections making up 64K of 32-bit words, matching the word size of the ModComp computers in the firing rooms and the error-correcting code used in transmissions[86]. Memory can be read in 200 nanoseconds, very fast by any current standard. Motorola 6800 microprocessors are used in the buffer as controllers, each with 2K of read-only memory and 1K of read/write memory[87]. The 64K main memory has the first 1K words set aside for interrupts, the next 1K as a common read/write area for flags and other variables, and the remainder as the protected memory area[88]. Data can move through the temporary storage areas and out to the computers at a rate of 8 megabytes per second[89].

A common data buffer can have up to 64 devices (computers or other buffers) hooked to it at one time. Each device is connected to a buffer access card. The cards are scanned by the buffer in rotation, looking for incoming data or requests for data. If a device is needing the buffer and its request is noted on the access card, the device then has a slice of time to do its work, after which the scanner (which has been "looking ahead") goes to the next card indicating a usage request[90]. In this way, when one machine is writing or even reading, all other machines are shut out, preventing both contention for resources or simultaneous attempts to update data[91].

An early criticism of the common data buffer concept was that it would be a single point of failure[92]. Standard protections were built into the buffer, such as dual power supplies and the use of triple modular redundancy in some components[93]. However, the biggest problem in a system of this type is protecting against communication errors. Millions of bits are speeding throughout the network each second, providing considerable opportunity for lost or garbled data. Byrne's answer was to include a powerful set of error-correcting codes, which he created with the help of Robert W. Hockenberger of IBM, who was brought in specifically to work on the problem[94].

The resulting codes enable the data buffer to successfully operate with any 2 bits of the 16-bit words in error! When a word is being transmitted between computers and the buffer or vice versa, it is sent as a 32-bit message. The first 8 bits are data, the second error-correcting code, then the next word's first 8 bits are code, and the last 8 data. Data are alternated in this way to protect against "big" signal losses[95]. Individual bits are checked using the correcting codes at each end of a transmission. One hundred per cent of the 1-bit errors can be corrected, 99%+ of the 2-bit errors can be fixed, 70% of 3-bit errors, and even half of the four bit errors. Since the memory chips are arranged in 64K by 1-bit banks, the loss of an entire sector of memory means the loss of just 1 bit per word, which can then be corrected. The error-correcting codes themselves are generated by software on read-only memories[96]. Even though such extensive protection is provided, in a decade of operation there has been no failure of a common data buffer, and internally never more than 1 bit has been garbled[97].

In terms of the architecture of distributed systems, the common data buffer was a pioneer. Currently, many distributed systems exist partly because of the proliferation of minicomputers and microcomputers. Micros, especially, can be connected to common data bases on shared hard disks. NCR Corporation briefly marketed a system called Modus from 1982 to 1984 that featured the ability to connect with different types of microcomputers, a shared data base, and microprocessor control of communications that effectively locked out other computers from corrupting data being updated by another one on the network. In general, though, no commercial system is as effective as the Launch Processing System in terms of speed, simplicity, and reliability. Most intercomputer communication is clouded by different protocols, nonadherence to declared international standards, and lack of speed. Frank Byrne's work stands as an original and brilliant solution to the key problem in implementing the Launch Processing System. Fittingly, Byrne received proper recognition for his achievement. NASA granted a $10,000 bonus and an award[98]. The buffer itself was a rarity for the government side of the space program[99].

CCMS Hardware

The hardware of the CCMS consists of the common data buffer and everything else in the four firing rooms of the Launch Control Center. Even though the buffer appears on the charts as the largest item, in reality it is one of the smallest, filling two electronics racks in the back of a firing room. Most of the equipment in a room is blue-colored consoles and boxes: the ModComp computers and their consoles. The number and arrangement of consoles are dependent on the function of the particular room. Firing rooms one and three are for flight operations, with three capable of being made secure for DOD launches. Rooms two and four are for software development and testing, number four used for secure operations[100]. Firing room two has three buffers to facilitate multiple parallel software development[101]. Operations firing rooms normally are confided for 12 consoles, plus a master, integration, and a backup console[102].

During a countdown, consoles in the adjacent software development room are kept active as a further backup[103]. Each ModComp has three display terminals, and those three make up one console. These are mounted in a half semicircle, so two computers and their attached consoles located side by side look like a "D" with the rounded part facing the front of the room. Each of the computers contains either a 5-megabyte hard disk or 80-megabyte hard disk for storing applications programs uploaded from the mainframes in the CDS[104]. Early in the program each engineer had his own disk, and could carry his programs to different computers, but when configuration control began to be needed the removable disks were replaced[105]. Loading an entire firing room through the buffer to the ModComps takes a full shift. Each computer can run up to six GOAL programs concurrently.

Individual consoles have marvelous capabilities. NASA commissioned Mitre Corporation to do a human factors study for the Launch Processing System[106]. Some of the resulting concepts make the usability of the system outstanding, and it is superior to many workstations in existence today. Color displays, programmable function keys that make it possible to replace long strings of keystrokes with a single push, full cursor control, and other features make it possible for an engineer to create applications programs that can be run without using the keyboard[107]. This concept antedates by 10 years the now ubiquitous "mouse" found on such machines as the Apple Macintosh. In addition, the consoles can be switched to become a terminal to the CDS for procedure development and to examine data recorded during operations. Special keys on the console enable program execution to be temporarily halted or single-stepped, aiding debugging of GOAL applications[108]. As a further convenience, each console has hard-copy capability; "snapshots" of displays can be made, with all the graphics

intact, but in black and white. Graphics use is assisted by special keys that provide corners, standard symbols for valves, transducers, and other components that can be put at cursor positions on the screen. Thus, the engineers can build pictorial skeletons of the systems they are testing for greater clarity. In general, these consoles are among the best available in any computer installation and are ideally suited to the purpose of the Launch Processing System.

Besides the use of ModComps attached to consoles, other ModComps are used as front end processors to provide interfaces between the spacecraft and ground service systems and the buffer. ModComps used for applications programs have 64K words of memory, but the front end processors have 48K, 64K, or 304K, depending on their connections to other devices[109]. Hardware interface modules at the actual points of entry to the ground support equipment plugged into the spacecraft send and receive data from the front end processors. Those, in turn, examine the data for parameters that are approaching their test limits. If a parameter nears a limit, the processor issues an interrupt and calls in a "control logic" program to handle the matter[110]. Control logic is a subset of GOAL used for making sure things are not done outside their proper order and within specific time constraints. For orbiter communication, a launch data bus front end processor communicates directly with the on-board general-purpose computers. Other "downlink" front end processors only receive pulse code modulated data to be processed for orbiter, main engine, and payload components.

Between the console computers and front end processors, a typical operations firing room contains over 30 minicomputers, each interconnected through the buffer. These computers can control tests and monitor the Shuttle anywhere hardware interface modules are available to connect it to the firing room, whether in the orbiter processing facility, the Vehicle Assembly Building, or the pad. Since each console can do the functions of any other console simply by changing its software load, the system has tremendous flexibility.

CDS Hardware

Supporting the CCMS is the CDS. Two sets of two Honeywell 66/80 mainframe computers are the heart of the CDS. NASA purchased the original pair of these 36-bit machines with half a million words of main memory each and an additional half a million words for sharing. As software for the Launch Processing System grew in size, the memories were upgraded to 1.5 million words each and 1 million words of shared memory[111]. One hundred seventy-two disk drives are connected to the machines as mass storage, with a total capacity of almost 30 billion bytes. Originally, the computers used Honeywell's

Figure 7–8. Typical firing room layout for the Shuttle. Less than 50 engineers are needed for the countdown. (NASA photo 108-KSC-78PC-240)

4JS1 operating system, which is no longer supported by the company. One NASA computer scientist said that "we have taken almost every piece of standard software and modified it" to meet the unique needs of the Launch Processing System[112]. Most often the first pair of Honeywells support an operations firing room, whereas the second set is being used for software development. If one of the pair notices that it has failed self-tests for 10 machine cycles, it automatically switches control to its partner.

The third part of the Launch Processing System is the RPS. Initially implemented with Apollo-era equipment, it was later modernized with new recorders and computers. The RPS records most data telemetered from the spacecraft for later playback and produces records and printouts in real time for immediate analysis by system engineers conducting tests[113]. Firing room engineers can play back tests or other data directly to their firing room consoles for problem resolution or trend analysis[114]. At first, the RPS only had enough equipment to support one Shuttle at a time, so to switch from one to another required rearranging a number of connections[115]. This situation was corrected during the modernization so that the RPS can now handle multiple Shuttle data flows.

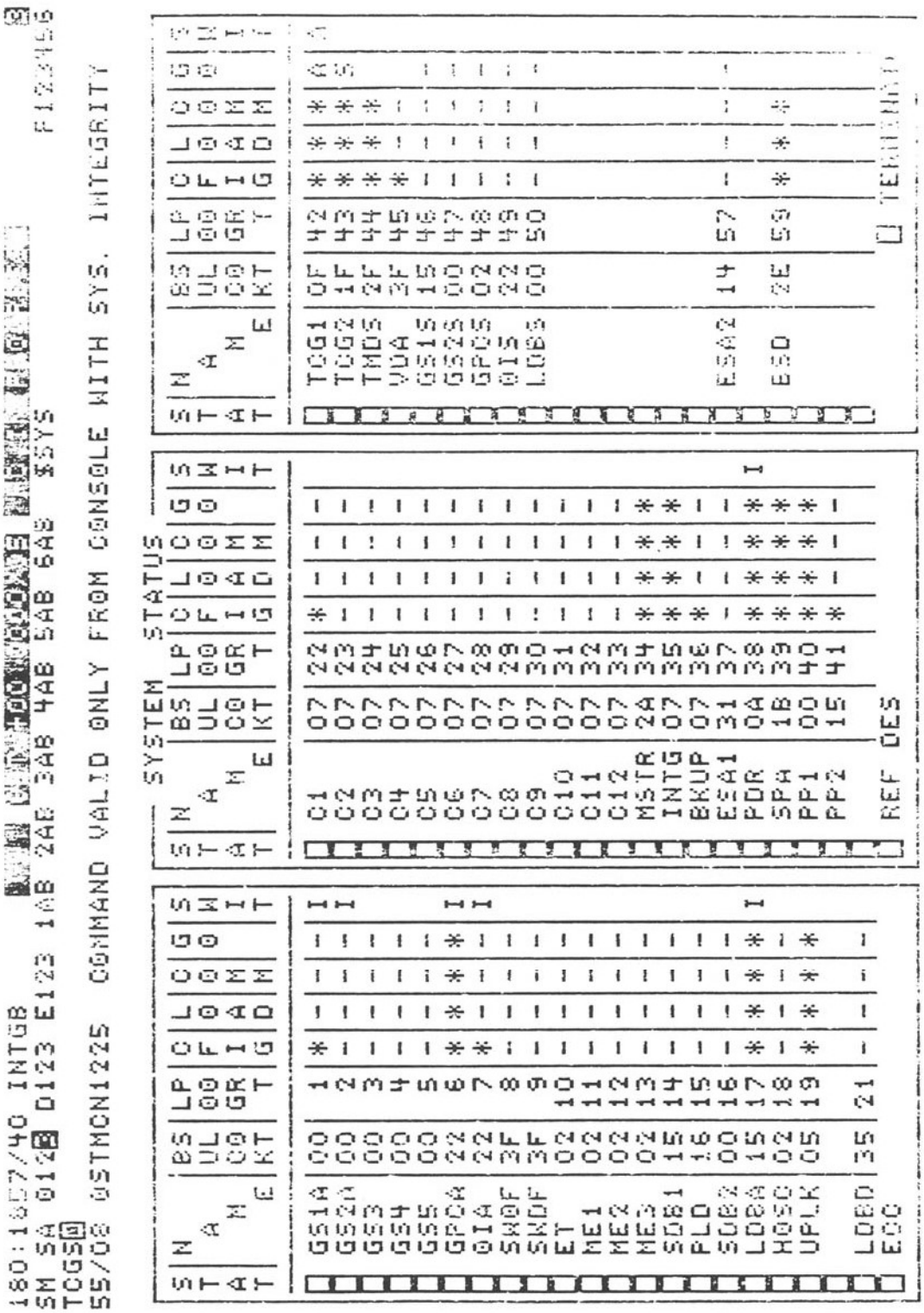

Figure 7–9. Hard copy of a display from one of the firing room consoles.

Launch Processing System Software

Software for the Launch Processing System is of three types: applications software, written in GOAL, performs the test and integration functions; simulations software enables engineers to verify their GOAL programs before using them on the real equipment; and systems software controls the execution of the other types. In the Launch Processing Division at the Kennedy Space Center, two branches support the hardware of the Launch Processing System, one for each major subsystem, whereas the Applications and Simulations Branch supports software by developing simulations and assisting test engineers in GOAL procedure writing. One of the reasons for the utility and success of the System is that civil service operations and maintenance personnel have been included in software planning and design from the beginning in order to make the System better meet their needs[116]. They essentially built their own tools[117]. That policy continues in the Applications and Simulations Branch, which helps the engineers refine their test requirements[118].

GOAL applications programs are the largest part of the Launch Processing System software, totaling 14.2 million words by the early 1980s. As a comparison, the displays, control logic, and test control software added up to less than 700K words[119]. Despite the early resistance of engineers to the automation of testing, they found that they learned more about their assigned part of the spacecraft by writing ATOLL or GOAL programs[120]. In instructing a computer, saying "pressurize the tank until the pressure is high enough" is too vague. Engineers writing programs are forced to think through the proper parameters and values and to account for anomalies ahead of time.

Kennedy engineers abandoned ATOLL because it had deficiencies in ease of use and in comprehensiveness: Too often assembly languages had to be used to do something ATOLL could not. Henry Paul assigned ATOLL veteran Joseph Medlock of Kennedy to head the GOAL development group[121]. Medlock and his team of civil servants received help from Martin-Marietta Corporation in defining the language, and then IBM implemented GOAL. The result was a highly readable, self-documenting procedural language. Just over four dozen statements are available, and training time is short, taking half days for 3 weeks (see Appendix III for an example of a GOAL program)[122]. IBM designed GOAL's compiler to disallow any undefined branches or procedures, making it more strict than FORTRAN compilers[123]. GOAL is highly flexible and permits engineers to decide for themselves the degree of interaction required to do a test[124]. GOAL programs are run within the computers in time slices of 10 milliseconds[125].

When an engineer is developing a GOAL procedure, he writes the procedure on his console using it as a terminal to the CDS. After the procedure is complete, it is tested against a simulation in the Honeywell computers, if a simulation is available. A Shuttle Ground Operations Simulator, consisting of GOAL-like statements, is available for developing models to test programs relating to ground equipment such as fueling systems and external power[126]. However, due to the lack of an AP-101 processor and Shuttle on-board software, procedures for checking out the flight equipment are limited or nonexistent. There is no way at Kennedy to test those procedures except against an actual spacecraft, so they must be sent to SAIL at the Johnson Space Center[127]. In addition to that restriction, simulations programs are limited to 256K words, the largest program a Honeywell 66/80 can run, since it is not a virtual memory machine. As a result, some models have to be run in parts[128].

A subset of GOAL is used to write control logic. Control logic prevents things from being done out of the proper order and within specific time constraints, avoiding disaster. It is necessary because of the parallel nature of testing. For example, before liquid oxygen can be moved through pipes and valves, they must be prechilled to near the temperature of the liquid, or the oxygen will flash evaporate. Control logic of the "prerequisite sequence" type checks to make sure the prechilling has been done. Or, if a valve pressure or voltage is nearing a dangerous level, "reactive sequence" control logic programs are automatically called by the front end processors to eliminate the anomaly[129]. Control logic thus makes parallel operations safe.

GOAL and control logic procedures must be integrated with the rest of the System before use to resolve potential conflicts and assign real addresses in the buffer to logical addresses in the programs. Integration is done in a laboratory containing the "Serial 0" ModComp/console, the first set delivered[130]. Integration requirements led NASA designers to abandon some of the flexibility envisioned in the early stages of the program[131]. Originally, they thought the engineers would have more responsibility for their programs and changes, but the complexity of the system required some measure of configuration control. Some 200 GOAL programs are needed just to load the liquid oxygen tank automatically[132]. With thousands of GOAL procedures to integrate, engineer autonomy had to be limited.

Cargo Integration and Test Equipment

One part of the Kennedy Space Center with an important role in the Shuttle program and also a user of Launch Processing System resources is the Cargo Integration and Test Equipment (CITE). With

hundreds of Shuttle flights planned to carry a variety of payloads from all over the world, the process of properly integrating cargo with the orbiter is a large task. Both electronic and physical interfaces must be checked in order to verify, for example, that a Spacelab module built in Germany will properly fit to a vacuum-proof seal in the cargo bay and be able to "talk" to the Shuttle computers as well.

Soon after beginning work as the prime contractor, Rockwell International and NASA did a study to find out how much and what kind of Interface Verification Equipment (IVE) would be needed for the operations era. By doing things the "traditional" way, in which the payload supplier did the interface verification, an estimated 20 sets of very expensive equipment were required. Robert Thompson favored sense over politics and decided in early 1976 to let Kennedy develop a centralized version of the IVE and do all the final interface testing for all the customers[133].

During June, July, and August of 1976 the formal requirements for the cargo facility were baselined. But when a source selection board met to begin choosing equipment, the members realized that they were about to violate the basic principles of the Launch Processing System by bringing in new equipment and doing things in a unique way instead of using existing contracts and computers[134]. Accordingly, Kennedy stocked the cargo facility with the same physical and electronic interfaces present in the orbiter, permitting the same contractors and maintenance contracts to be used. In 1978, an AP-101 was added to provide a means to test software interfaces. Equipment in the cargo facility can also directly connect with the Launch Processing System so that payloads can be further integrated. CITE is another user of the simulations kept on the CDS[135].

Payloads delivered to Kennedy are checked out and further prepared in either the horizontal facility (Spacelab would be worked on there) or the vertical facility (communications satellites are integrated vertically). After completion of the integration tests, a special transporter with cargo space as large as the Shuttle's bay moves the payloads to the Vehicle Assembly Building for installation in the orbiter.

The Launch Processing System in the Operations Era

Originally, Henry Paul had a goal of reducing the number of technicians in a firing room from the 250 of the Apollo era to about 45. Although he succeeded, in the early 1980s, the Launch Processing System was still labor intensive, with 75 civil servants and 700 contractors involved[136]. However, in late 1983, NASA awarded the shuttle maintenance contract to Lockheed, which is now responsible for physical equipment and software relating to Shuttle launch processing.

That award marks the end of the multicontractor development era and the beginning, for the first time in NASA's history, of an operations era for a manned spacecraft. Before the Shuttle, each flight and the preparations beforehand were idiosyncratic. Now some degree of standardization and routine is possible, largely because of the nature of the Launch Processing System. Carl Delaune, a NASA engineer in the Applications and Simulations Branch, is exploring ways of applying artificial intelligence to checkout procedures, such as creating a program that makes suggestions to test engineers if strange values occur[137]. If his inquiry bears fruit, eventually the amount of human interaction during checkout will shrink even further.

As the development effort at Kennedy matured, the purposely staggered development of a Launch Processing System at Vandenberg Air Force Base began. Plans were to build the military facility after most of the developmental bugs were out of the NASA model. The Air Force saved money at its installation by modifying facilities built for the Gemini-technology Manned Orbiting Laboratory program in 1966[138]. Originally designed as a Titan III launch site, the complex provides for mating orbiter, tank, and boosters at the pad, as no Vehicle Assembly Building exists there. Ground checkout facilities are split between locations at North Vandenberg and South Vandenberg, so the CDS, CCMS, and RPS are physically separated[139]. With the commissioning of the western launch site in the early 1990s, the Shuttle program will have reached its full flowering.

Summary

Distributed computing, connecting different vendors' equipment successfully, good user interfaces, and automation are all topics of continued concern and research in the computer industry. The Launch Processing System solves all those problems in a specific arena. It is difficult to think of a system better suited to its task. A marvel of integration, efficiency, and suitability, it reflects the ingenuity and clear-sightedness of its originators. Lessons learned in the 1960s during the first attempts at automating checkout were applied in toto to the Launch Processing System. Rarely has a second system so completely eliminated the deficiencies of its predecessor.