# Design and Development of a Program-and-Place Surface-Mount-Assembly Machine

C. Greg Bess
Walter W. Jones
Edward E. Lewis

As circuit-pack designs migrate to surface-mount-technology components that have finer and higher lead counts, the manufacturing processes and assembly techniques used must be enhanced. This paper describes the research and development efforts undertaken at AT&T's Little Rock Operations Center to design and introduce an assembly workstation that picks, programs, verifies, and places programmable, surface-mount devices on a printed-wiring board during the circuit-pack manufacturing process. This *program-and-place* machine combines the convenience of a pick-and-place machine with the capability of a device programmer. It gives circuit-pack manufacturers a robust and flexible assembly tool that is capable of responding to the dynamics of a fast and volatile product-introduction environment.

## Background

Several years ago, AT&T's Little Rock Operations Center in Arkansas was allocated the manufacturing responsibility for the AT&T StarServer® E computer—a symmetrical, multiprocessing server based on an Intel 80486 processor with an EISA bus. (EISA stands for *Extended Industry Standard Architecture*. This bus architecture standard for Intel 80386 and higher processors is an extension of IBM's Industry Standard Architecture or ISA bus, and provides upward compatibility with earlier ISA-bus systems. Panel 1 defines acronyms and terms used in this paper.)

During an early design review, it became apparent that the product introduction and process development would be a major challenge. Of particular concern was that the circuit-pack products were heavily laden with surface-mount technology (SMT) components that required programming prior to assembly. From preliminary discussions with the developers, we learned that a circuit-pack assembly could have 100 or more differently programmed parts.

To ensure the shortest product-introduction intervals, the Little Rock Operations Center was also given the responsibility for building all the early development models (i.e., preproduction prototypes). This meant that the *microcode* (i.e., the programming) for these 100 parts per circuit pack could change many times during product development. The challenge that we faced was how to handle large numbers of programmable, surface-mount devices and cope with constantly changing microcode throughout the product's life cycle. (The *life cycle* starts with the initial concept for the product and ends with its extension or replacement by a newer product.)

**Handling Preprogrammed Devices.** The most widely used method of handling programmable devices has been to program the devices individually off-line and mark and store them before the circuit-pack assembly process begins. This method creates a different device for each preprogrammed part. For instance, one blank device type that is programmed with ten different microcodes becomes ten distinct parts at the assembly process.

Preprogramming the parts results in problems in several areas:

- Control and identification of the different parts
- Maintaining adequate inventories of each preprogrammed part
- Loss of assembly efficiency
- Added handling.

**Control and Identification.** To maintain control, all preprogrammed devices must be marked. Each part must have either a part identification (e.g., IC22) and microcode-

**Panel 1. Abbreviations, Acronyms, and Terms**

alignment nest — a physical fixture with several openings of different sizes; enables the manipulator to pick up a part at its center before placing the part in a programming nest or on the circuit pack

COMCODE — the nine-digit part number that is assigned to a component within AT&T and used for planning and ordering material

EISA — Extended Industry Standard Architecture, a bus architecture standard for 80386 and higher processors. This extension of the ISA bus provides upward compatibility with earlier ISA bus systems.

ERC — Engineering Research Center

FDR — the feeder location number (i.e., 1 through 10) on a magazine

FWS — flexible workstation

I/O — input/output

ICxx — part designation for an integrated circuit

IC# — number of the integrated circuit currently being processed

ISA — Industry Standard Architecture, a bus architecture that IBM developed for the AT version of its personal computer

JEDEC — Joint Electron Device Engineering Council; composed of manufacturers and users of solid-state products who are concerned with developing standards, test methods, specifications, and other engineering matters

life cycle — the life of a product or service; starts with the initial concept for the product or service, and ends with extension or replacement by a newer product or service

$M^2L$ — AT&T's modular manufacturing language; an interpretive, multitasking software language used to control industrial machines and processes

MAG — magazine; i.e., a group of feeders located at either the right side, left side, or rear of the machine. Each magazine has a maximum of ten locations for feeder placement.

PNP machine — program-and-place machine; combines the convenience of a pick-and-place machine with the capability of a device programmer

preprogrammed — the component receives additional functionality or capability via information added before its assembly, i.e., while the part is loose (not attached)

programmable — the device can receive additional functionality or capability via some information added to it

programmed — the device has additional functionality or capability via information added to the part, or this information is being added to the part

programming nest — a physical fixture with electrical contacts where parts are positioned to be programmed. Unlike a socket, which is usually part-type dependent, this fixture will accommodate more than one type of device.

SEQ# — number of the current sequence

SUB — sublocation on the feeder. Some feeders have more than one presentation location.

version control number or a distinct part number that changes each time the microcode changes. The challenge in marking small surface-mount parts can only be overcome with expensive part-handling and marking equipment.

**Inventory.** To assure availability for manufacture, the various parts must be programmed and stored before circuit-pack assembly starts. This could result in significant quantities of preprogrammed parts in inventory, depending on the assembly schedule and the lead time of the off-line programming process. If a microcode changes, disposing of these inventories or reworking the devices could be expensive.

**Assembly efficiency.** The manufacturing equipment that assembles the circuit packs must have an input station or feeder for each distinct part.

If the circuit pack has ten different preprogrammed parts, then the machine has to have ten different input stations. Because this equipment may also pick and place other nonprogrammable parts on the same circuit pack, some of the machine's input space usually is not available for preprogrammed parts. (*Pick and place* means to remove a part from the appropriate input station and insert the part in the correct position on the circuit pack.) Depending on the method used to convey the parts to the machine (e.g., by tape and reel or tubes), a

machine's available input space can easily be exceeded. The typical, off-the-shelf workstation will accommodate from 20 to 60 different parts of this size.

Attempts to sequence preprogrammed devices in their order of assembly and feed them in at one input station fail because the machine does not or cannot provide intelligent feedback. Most equipment on the market today works with a simple processing procedure. First, the machine picks a part from an input station that contains many of the same parts. It then centers the part on the placement tooling (using a machine-vision system, physical means, etc.), and places the part on the circuit pack. Should the machine drop or reject a device before placing it on the circuit pack, it will return to the input station to pick again. If the parts in the input station were arranged in an assembly sequence, the machine will get out of sequence and place devices in the wrong locations.

**Handling.** The additional handling steps required for the preprogrammed parts in a product can result in greater defects.

If parts are programmed correctly but have the wrong markings and are placed on the circuit pack, then the pack is defective. If parts are not programmed but are marked and placed on the circuit pack, again the pack is defective. If parts are marked and programmed correctly but are placed in the wrong input station of the machine, then the machine will populate the wrong component location, resulting in a defective circuit pack. Thus, the probability of defects increases dramatically because of the almost limitless permutations for errors.

**Need for a Solution.** A shop that has a high throughput, low product mix—where the program code for preprogrammed devices is stable—may be able to manage these problems and limit their effect on product quality. But for a high mix, low volume assembler whose products are constantly being upgraded, the management of these issues can be costly. (*High throughput, low product mix* means the shop assembles many circuit packs of a few stable designs, while *high mix, low volume* refers to a shop that assembles a few circuit packs of many different designs.)

### The Little Rock Solution

We knew that most of the programmed devices on a circuit pack were obtained by programming only a few types of blank parts. For example, to produce 50 programmed devices may require programming only 10 sets of 5 different types of the blank components. Therefore, our solution was to do the programming at the time the circuit pack was assembled.

The time required to pick and place a nonprogrammed component is 2 to 4 seconds. The time needed to program a device is 3 to 5 seconds for a programmable logic array and slightly longer for a programmable read-only memory. Most of the programmable devices we are using fall into the 3- to 5-second range. By tolerating the additional seconds required to program each device during assembly, we can combine the two operations. Because the devices are programmed at the last possible moment before assembly of a circuit pack, we can easily handle last-minute microcode changes from the ongoing development and for increased product customization. (*Customization* means that we can accommodate many different versions of microcode for a pack, as customers require.)

Through design-for-manufacturability reviews with our product developers, we obtained agreement to eliminate the individual part-marking requirements. Revised part numbers and issue control are still maintained, but we no longer have the requirement to mark the actual component. Issue control at the circuit-pack level provides all the information needed to associate the microcodes of the programmed devices for repairs in the field, so the correct issue or version is used.

To accommodate our needs, we first asked the machine vendors to design a workstation that integrates the pick-and-place operations and the programming. After several discussions, it became apparent that the vendors' interest was minimal or that their development costs or lead times would not accommodate our time requirements for product introduction. We then decided to develop the workstation locally.

### Hardware Components

Most pick-and-place equipment on the market operates by using an inflexible, proprietary, machine-control program. This inflexible controlling program, or *machine-control software* as it is referred to in the industry, was the major problem we encountered in choosing an automation platform. Most vendors had the space and capability to incorporate a device programmer into their machine's workspace but their operating software required extensive rework. (By *device programmer*, we mean the equipment with fixture, cables, and software

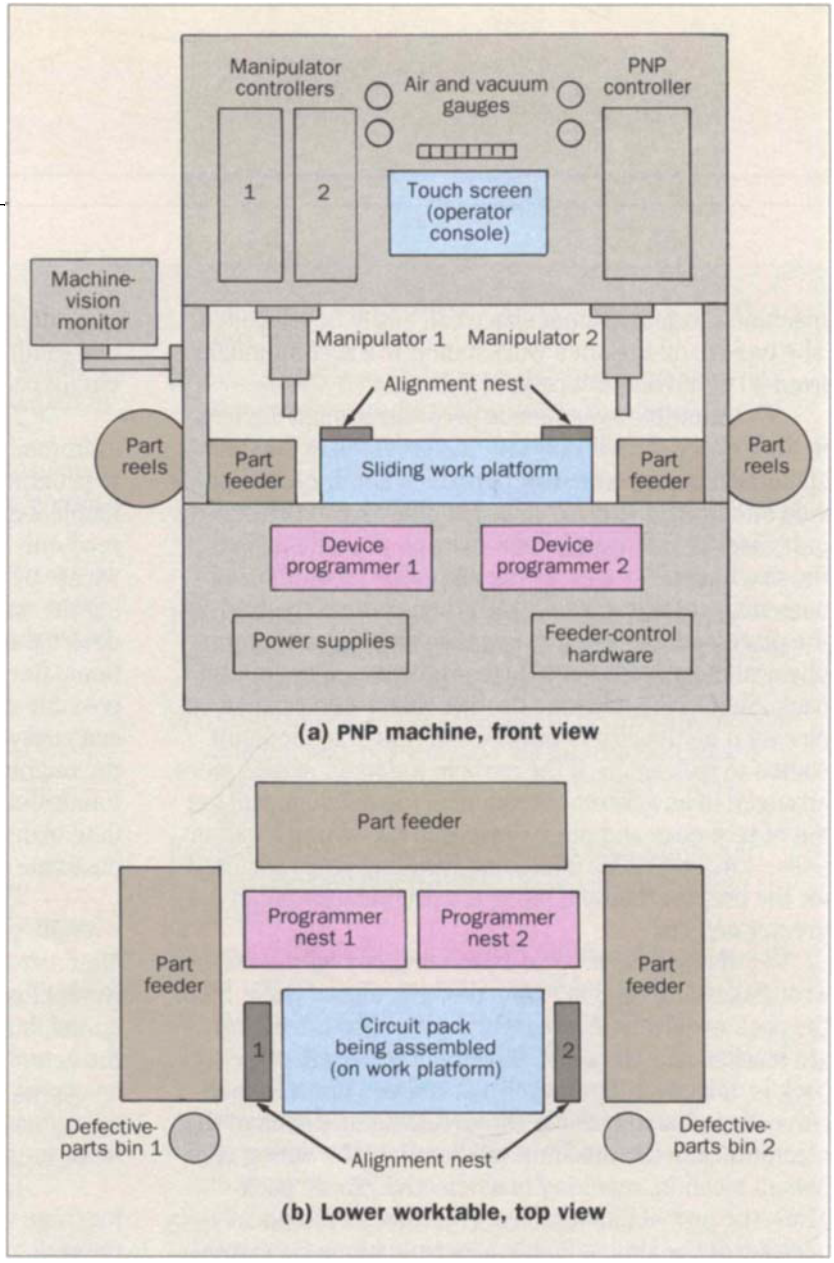


**Figure 1. Machine configuration of the program-and-place (PNP) implementation. The workstation's upper and lower cabinets house the controllers, power supplies, and device programmers. Two manipulators are located on the top ceiling of the workspace. Each manipulator has its own device programmer and part-alignment nest, and will process each part in the input feeders closest to it. The work platform in the center of the lower worktable supports the circuit pack being assembled. Because the platform slides forward, the operator can easily load a blank printed-wiring board for assembly and, later, remove the assembled circuit pack. He or she chooses the assembly program on the touch screen of the operator's console.**

needed to program the devices.)

Therefore, we selected the *flexible workstation* (FWS),[1] developed and manufactured by the AT&T Bell Laboratories Engineering Research Center (ERC) in Princeton, New Jersey. The FWS is a four-axis (*X*, *Y*, *Z*, and θ or tilt angle), gantry-style workstation that can precisely position light- to moderate-weight workpieces. It supports multiple manipulators within a common workspace (Figure 1a) and can function as a standalone unit or as an integrated part of an existing assembly line. The system's *X*–*Y* motion is produced by a linear stepper motor that rides on an air bearing. Our reasons for choosing the FWS include:

- Access to low-level functions in the software allowed the control needed to handle complex parts requirements and handling exceptions.
- The machine controller supports access to our local-area network (i.e, the AT&T StarLAN network), thus providing easier machine-program storage, microcode storage, backup administration, and system support.
- The machine-vision feature is easily programmed for setup and placement-compensation capability.
- The system provides discrete I/O (input/output) for control of peripheral equipment, as well as user-definable control displays for the operator interface.
- It supports individual bit manipulation of the parallel ports, allowing control for part feeders.
- AT&T's modular manufacturing language—i.e., the $M^2L$ machine language—that is provided as part of the FWS supplies machine-control and geometry-control functions. ($M^2L$, a product of the AT&T Engineering Research Center, is an interpretive,

multitasking software language[2] that is used to control industrial machines and processes.)

- The workstation has dual-manipulator capability and a large work-surface area. Both manipulators can access all areas of the workspace.
- Ample *Z*-axis down pressure is provided to position and hold the parts during device programming.
- Because the FWS is an AT&T product, the technical support and integration assistance we may need are easily accessible. This fosters a team development environment between ERC and Little Rock personnel.

We chose a leading device-programmer vendor to supply the two integrated, device-programming stations needed for the application, one for each machine manipulator. Our concerns were focused on obtaining broad device coverage, good reliability, and remote-communications capability. Our experience with this vendor's device programmers in other areas in the Little Rock factory determined our selection.

For the tape-and-reel feeders, we selected a standard module that was already being supplied by one of our equipment manufacturers. This choice was mainly a result of our familiarity with maintenance and floor-support issues. These feeders also required minimal interface support from the placement machine.

Once the major components of the machine had been selected, we turned our attention to the larger task of integrating the system's components and controlling the operating sequence. Figure 1 shows the various components of the program-and-place (PNP) implementation.

The manipulators are located on the top ceiling of the workspace. A sliding work platform is in the center of the lower worktable. The operator slides the platform forward, positions the printed-wiring board to be assembled on this surface, and then slides the platform with the board into the workspace for processing.

Feeder control and the device-programming equipment are housed in the lower frame. The physical interface or nest for the device programmer is located at the rear of the workspace, behind the circuit pack to be assembled. (The *nest* is a physical fixture with electrical contacts where parts are positioned to be programmed. We use the term nest because the fixture will accommodate more than one type of device, as opposed to a socket, which is usually part-type dependent.) Wiring from this interface to the device programmers is kept as short as possible. Parts feeders are positioned around the sides and rear of the machine.

Following delivery of the flexible workstation and the programming stations, we started development using a temporary fixture and skeleton programs. (*Programming station* refers to the main bulk of the device programmer. Each station has a nest on board for programming the devices, but remote nests can also be used.) To determine early feasibility for the project, we put a programming station inside the workspace of the FWS. (Later, we just placed remote nests in the workspace and put the main unit into the lower cabinet.) The manipulator would place and hold a part in the programming nest of the device programmer. Within a month after initial equipment delivery, this temporary system was being used to produce early production models of the StarServer E circuit packs.

Early on, we discovered that the physical connection between the part to be programmed and the programming nest was a constant source of failures. By design, most programming nests are tight to provide good electrical contact with the part being programmed. Mechanical interference problems with these part connections caused mechanical fatigue problems with the programming nests.

To eliminate these problems, we switched to a conductive polymer material for the mechanical interface.[3] This material allows current to flow only in the vertical direction, not in the horizontal direction. The polymer material is laid atop a special circuit pack, supplied by the vendor of the device programmer. This circuit pack provides the connection to the programming station and is etched with a universal pattern that matches the device-lead profile of several device types. Sandwiched between the part and this special circuit pack, the polymer provides a compliant medium that conforms to the device leads and the etched surface of the pack, thus supplying a conduction path for programming. (See Figure 2.)

This change eliminated all interference problems and made the whole program-and-place project feasible. The polymer also allowed us to use a common site to program all parts, regardless of their size. Before we began to use the polymer material, different nest sites were required for each size.

The polymer material—which is used as conductive, 1-inch by 1-inch pads—can be removed and cleaned. During the course of a day's programming, the machine

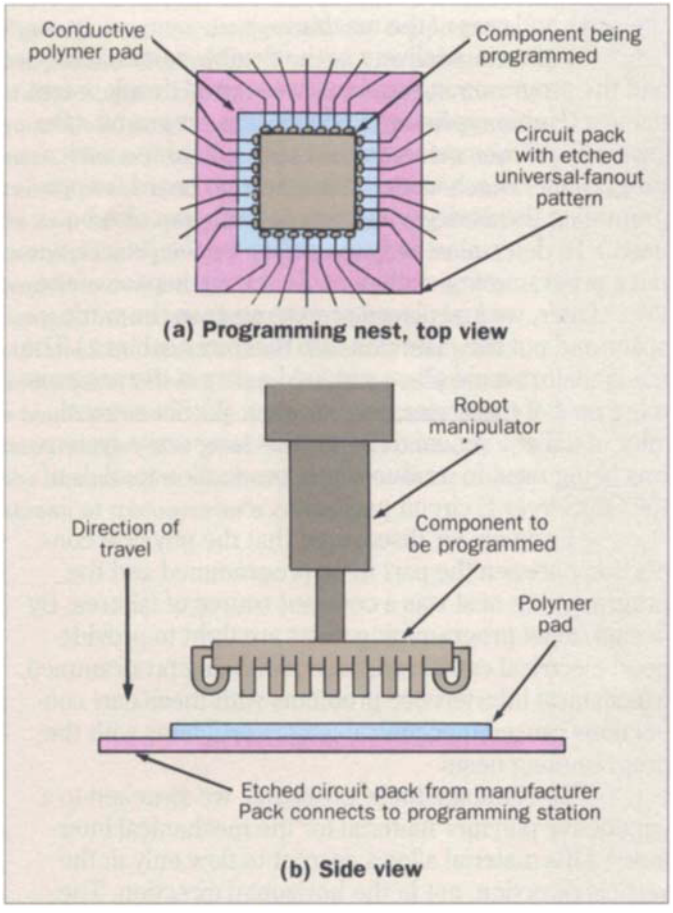


**Figure 2. A polymer pad allowed us to use a common site to program all parts, regardless of their size. The conductive polymer material is sandwiched between the part and a special circuit pack whose surface is etched with a universal pattern that matches the device-lead profile of several device types. With the 1-inch square polymer pad in place, current flows only in the vertical direction. The compliant polymer conforms to the device leads and the circuit pack's etched surface, thus supplying a conduction path for programming.**

automatically changes the pads when it detects frequent programming errors. The machine operator then cleans the old pads and places them back in a stack at the machine for reuse.

The original design called for each device to be aligned by a machine-vision system. But in our early attempts with its use, the machine-vision method proved slow and unreliable.

An alternative approach—and the one we adopted—was to use a part-alignment nest designed by the ERC. The nest (Figure 3b) has several tapered openings to fit the various sizes of components being aligned. Each manipulator has its own nest for alignment. A manipulator places the part in the appropriate tapered opening of the nest and releases the vacuum, so the part settles and is automatically centered. Then, the manipulator picks up the device from the center of the nest opening, acquiring the part at its center. Alignment is required before programming of the device because the feeder is unable to present a centered part and, again, after programming because the part shifts slightly when it is raised from the polymer pad.

The photograph sequence in Figure 3 shows a part being removed from a feeder location (here, a tape-and-reel slot), aligned, programmed, and placed.

### System Operation

The circuit-pack developers provide the microcode for the programmable devices electronically in JEDEC Standard 3–A format. (JEDEC is the Joint Electron Device Engineering Council, composed of manufacturers and users of solid-state products who are concerned with developing standards, test methods, specifications, and other engineering matters.) This format provides for the transfer of fuse, test, identification, and comment information and defines the "intermediate code" between device programmers and data-preparation systems.

The device microcodes for each circuit pack to be assembled are grouped together and delivered as a set. The microcode for each device in the set is identified by the appropriate device-component identification (e.g., IC244, IC002). This grouping of microcode is referenced by an identifier that consists of the:

- Pack name or code — a four-character identification code assigned by the circuit-pack designer
- Art master — a number that refers to the revision level of the artwork used to make the printed-wiring board for the circuit pack
- Series level — a number and letter combination that indicates the revision level of the circuit pack.

Any change in the microcode for one or more devices on the circuit pack will result in a series-level change at the pack level. Regardless of how many device microcodes changed, the developers must deliver a complete group of microcodes for each new series update.

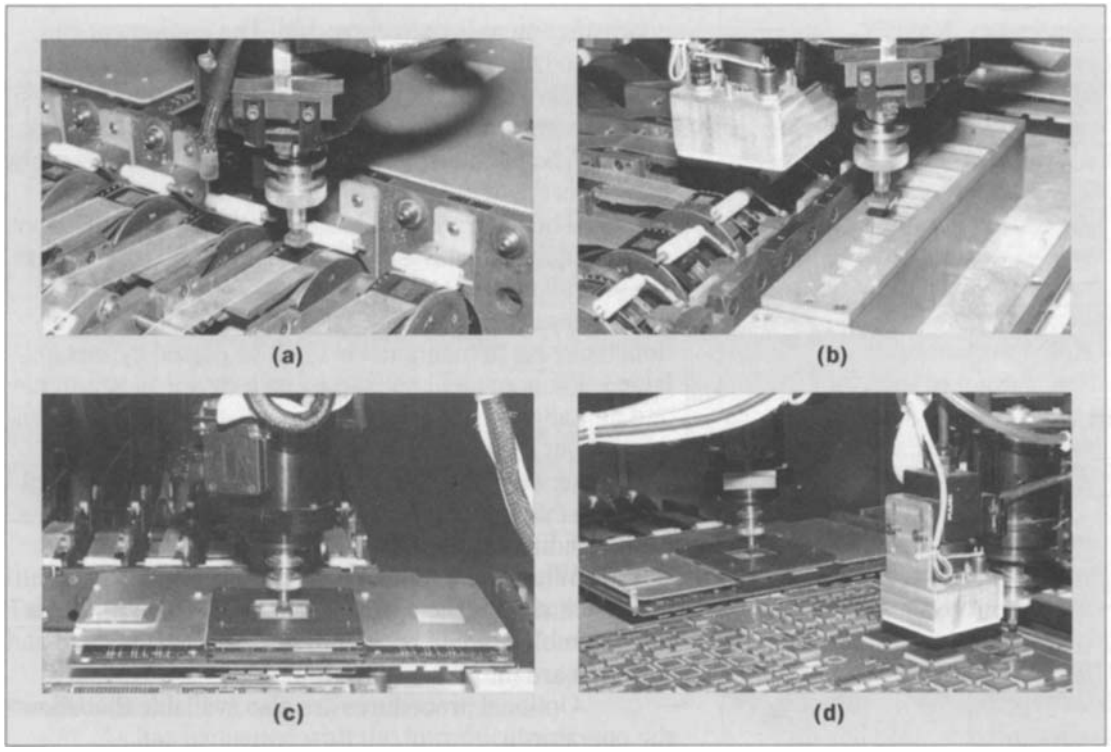


**Figure 3. The PNP machine allows devices to be programmed just before they are placed. (a) The manipulator removes a component, either programmable or nonprogrammed, from the feeder. (b) Because a feeder cannot present a centered part, the component then is centered using the manipulator's alignment nest. The nest's tapered openings accommodate parts of different sizes. (c) If the component is a blank device to be programmed, the manipulator uses its programming nest. (d) Finally, the manipulator places the component in the proper location on the ciruit pack being assembled.**

In addition to the microcode, all circuit-pack assembly information is also transmitted electronically from the developers to engineering minicomputers at Little Rock. Software routines on this computer automatically intercept this information and translate it into a file structure that the PNP machine understands. The placement data is correlated with the microcode information via the component identification for the device that is being programmed and placed. To optimize the device-programming process, parts that use the same blank-device types are grouped together, which minimizes the number of times the device programmers must switch between part types.

All preprocessing of data and generation of machine-processing routines are done at the engineering-computer level, where a complete picture of the circuit-pack processing environment is maintained. Once generated, this data is deposited in a common storage area of the minicomputer for access by the PNP machine, using the local-area network.

When the microcode transmittal from the designers is received, local routines generate unique test vectors for each device. The PNP machine uses these vectors to verify that the devices have been programmed correctly and are functional before the devices are placed on the circuit pack. By checking at this point, we not only guarantee that a part is functional but also ensure that the device matches the characteristics needed at the component location on the circuit pack. With other placement equipment, such a test usually is not possible and a visual check of device markings must suffice.

To initiate assembly, the operator manually loads the printed-wiring boards onto the work platform, chooses the applicable assembly program on the touch

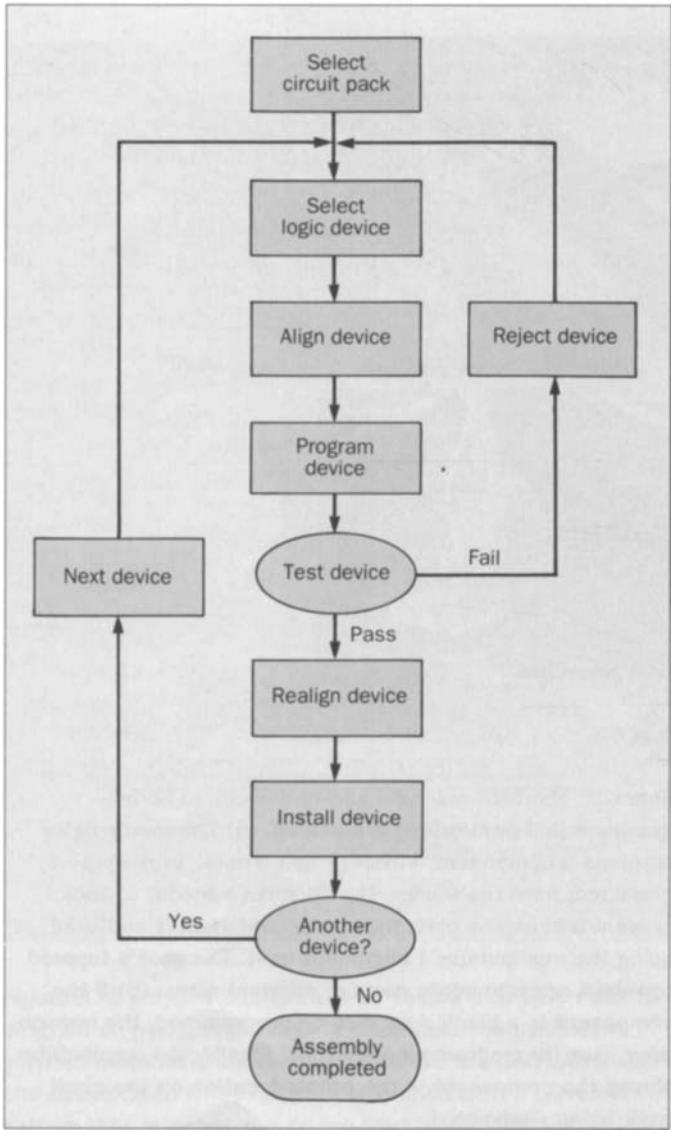


**Figure 4. Process flow for the PNP machine. Each manipulator processes parts in the feeders closest to it. Both manipulators work independently over the entire area of the work surface, except when their paths physically cross. Dynamic path-control software keeps the manipulators from colliding.**

screen of the operator's console, and pushes the start button. Figure 4 shows the process flow of the PNP machine.

Before the placement cycle begins, fiducials (i.e., reference marks) on the printed-wiring board are read optically to determine if a board is misaligned. If needed, board alignment is then corrected. The equipment can accommodate coupons of multiple circuit packs. (A coupon allows multiple circuit packs to be grouped physically into one larger, more easily assembled circuit-pack outline. The packs are separated later, after the assembly process is completed.)

The placement task is divided between two robot manipulators. The physical placement of the part feeders around the machine dictates which of the manipulators will process each part. For example, parts that are positioned closest to manipulator 1 will be placed by manipulator 1. Each manipulator has its own device programmer and part-alignment nest. The manipulators work independently over the entire area of the work surface, except when their paths physically cross. Dynamic path-control software that the ERC provided keeps the manipulators from colliding.

When the machine has finished placing all the devices, it signals the operator. He or she then removes the assembled circuit pack and loads the next printed-wiring board into the workstation.

Optional procedures are also available that allow the operator to:

- Skip placement of a part, if the part is not available or if engineering requires that it be omitted from the circuit pack.
- Program selected loose components for rework or repair.
- Dry cycle the machine. (That is, do all the mechanical motions and software transfers, without placing or programming a part. Usually, dry cycling is used to test or repair the machine.)

Also provided are touch-screen menus that help the operator configure the part-feeder locations and change vendor information for each feeder. For easy access, touch screens are located at the front and rear of the machine. The rear screen is a portable unit that is tethered by a coiled cable. Figure 5 gives a general view of the runtime screen during circuit-pack processing.

#### Operational Metrics

During each machine cycle, the machine collects process data such as part-cycle time, part-programming time, device-failure modes, and complete circuit-pack cycle times. This information proved invaluable during the machine-development process. Currently, the machine operators and engineers use the data to tune

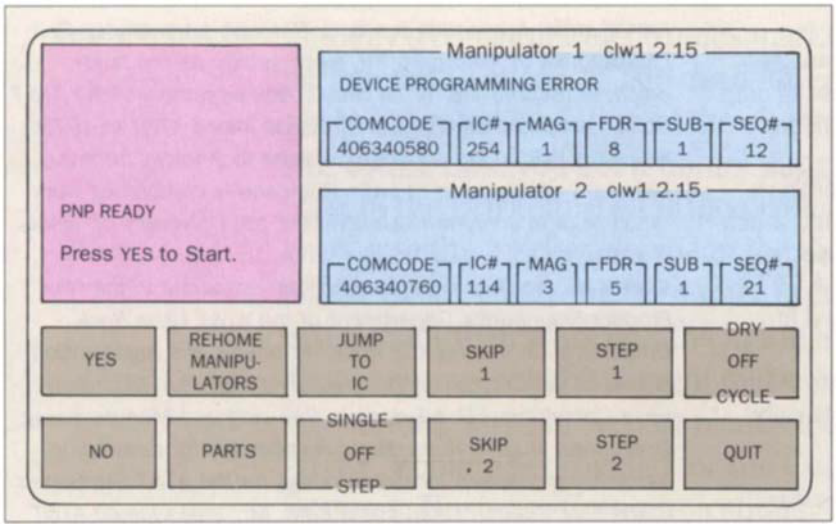


**Figure 5. Touch-screen menus help the operator configure the part-feeder locations and change vendor information for each feeder. The machine needs to know each part's magazine number, feeder-slot number, and sublocation on the feeder. Here, we show a typical PNP runtime screen during circuit-pack processing. The two areas in the upper right display information about the part a manipulator is currently processing, while the area on the upper left contains machine-status information and instructions to the operator. For easy access, touch screens are located at the front and rear of the machine.**

the process and schedule work in the manufacturing cell. Table I provides some general operating information that has been acquired to date.

**Conclusion**

As has happened with the introduction of any leading-edge product, the StarServer E computer provided many manufacturing challenges. The manufacture of a highly customized, quickly evolving product posed daunting problems and the handling of numerous programmed SMT devices was certainly among them. The development efforts discussed here were undertaken because of our urgent need for something better. Originally, the only goal was to obtain a machine that could automatically pick, program, and place SMT devices quickly and accurately. We realized quickly that there were no off-the-shelf solutions. The project's goal was then redefined to include developing the equipment locally to provide that capability. The PNP machine, which resulted from this effort, did this and more.

**Table I. PNP Operational Metrics**

| Metric | Value | Comment |
|---|---|---|
| Total blank-part types | 25 | Machine capacity = 30 |
| Total programmed parts | 490 | Current conditions |
| Average part-cycle time | 7.5 sec | Two manipulators |
| Maximum part-cycle time | 16 sec | Includes time to reconfigure the device programmer |
| Minimum part-cycle time | 5 sec | — |
| Different parts programmed | 2700 | 1990 to present |

Our basic calculations show that, before the PNP equipment existed, the cost of attaching a programmed device to the circuit pack was over a dollar. By using the program-and-place machine, we can now do it for less than 15 cents. In addition, this development accomplished the following:

- Moved the actual time of programming a device to the last possible moment in the manufacturing process. This gave the manufacturing process greater flexibility and eliminated the need to maintain costly preprogrammed devices in stock.
- Made programming of devices invisible to the operating personnel, yet highly reliable.
- Gave circuit-pack designers greater design flexibility and longer development time before they had to "freeze" a design. (Now, changes can be made up to five minutes before assembly starts.)
- Made the process of programming a high mix of SMT devices cost-effective, reliable, and feasible.

As with many manufacturing scenarios, tracking all the costs can be difficult. Some of the accomplishments listed surely fall outside the normal capabilities of cost tracking.

Although the solution discussed here is not appropriate for all manufacturing environments, we feel it will give us the greatest flexibility, while maintaining the quality standards necessary for AT&T to compete successfully in the 1990s.

**Acknowledgments**

We wish to express our thanks to the management of AT&T's Little Rock Operations Center for giving us the opportunity and for supporting the creative environment.

We also thank the personnel of the Enterprise Manufacturing Cell for their constructive input and much needed support during development. (They are part of the Sourcing and Manufacturing Operations for AT&T Business Communications Systems and are based in Little Rock.)

Special thanks go to James Ellis and Larry Atkinson who brought the whole project together physically.

**References**

1. P. F. Lilienthal II et al., "A Flexible Manufacturing Workstation," *AT&T Technical Journal*, Vol. 67, No. 2, April/May 1988, pp. 5–14.
2. J. P. Flemming and G. VanOrden, "M²L—AT&T Modular Manufacturing Language," *Conference Notes of the 1989 Symposium on Advanced Manufacturing*, Lexington, Kentucky, September 25–28, 1989, Center for Robotics and Manufacturing Systems, University of Kentucky, Lexington, Kentucky, 1989, pp. 69–88.
3. J. A. Fulton et al., "Use of Anisotropically Conductive Polymers in Electronic Applications," *Third International SAMPE Electronic Material and Process Conference*, Los Angeles, California, June 20–22, 1989, The Society for the Advancement of Material and Process Engineering, Covina, California, 1989, pp. 578–589.

*(Manuscript received January 7, 1992)*

***C. Greg Bess*** *is a member of technical staff in the VLSI Process Technology Department at AT&T Bell Laboratories in Allentown, Pennsylvania. A member of the Information Systems Group of the VLSI Process Technology Laboratory, he is responsible for developing advanced computer automation for the wafer fabrication process. Before his transfer to Bell Laboratories in Allentown, Mr. Bess served as the lead engineer responsible for all circuit-pack assembly of the AT&T StarServer client computers. Mr. Bess joined AT&T in 1979. He has a B.S. in electrical engineering technology from the University of Arkansas at Little Rock and is completing work on an M.S. in computer science from the University of Illinois at Urbana.*

***Walter W. Jones*** *is a senior technical associate in the New Product Engineering Department of the AT&T Little Rock Operations Center in Little Rock, Arkansas. His organization is part of the Enterprise Manufacturing Cell of AT&T's Business Communications Systems, Sourcing and Manufacturing Operations in Little Rock. He is responsible for developing assembly processes for circuit packs for the AT&T StarServer E and StarStation™ client computers. Mr. Jones joined AT&T in 1976. He has an A.S. in electronics from United Electronics Institute in Little Rock, and is completing work on a B.S. degree in electrical engineering technology from the University of Arkansas at Little Rock.*

***Edward E. Lewis*** *is a member of technical staff in the New Product Engineering Department of the AT&T Little Rock Operations Center in Little Rock, Arkansas. His organization is part of the Enterprise Manufacturing Cell of AT&T's Business Communications Systems, Sourcing and Manufacturing Operations in Little Rock. He is responsible for developing assembly and test processes for the AT&T Smart Card. Mr. Lewis joined AT&T in 1977. He has an A.S. in electronics from Southwest Technical Institute in Camden, Arkansas, and a B.S. in computer science from the University of Arkansas at Little Rock.*