Reducing Setup Time for Printed Circuit Assembly

In 1994, HP's Man-Link recipe-generation system was enhanced to reduce the time required for setting up pick-and-place machines. This was done by ordering the products to exploit the commonality of parts among them and by creating sequences of setups that differ as little as possible from one another. This paper documents the issues and trade-offs and discusses the potential benefits.

by Richard C. Palm, Jr.

Early in 1993, we began an investigation into ways to decrease the cost of setup time for HP surface mount manufacturing centers. Our initial investigation covered a range of options, including:

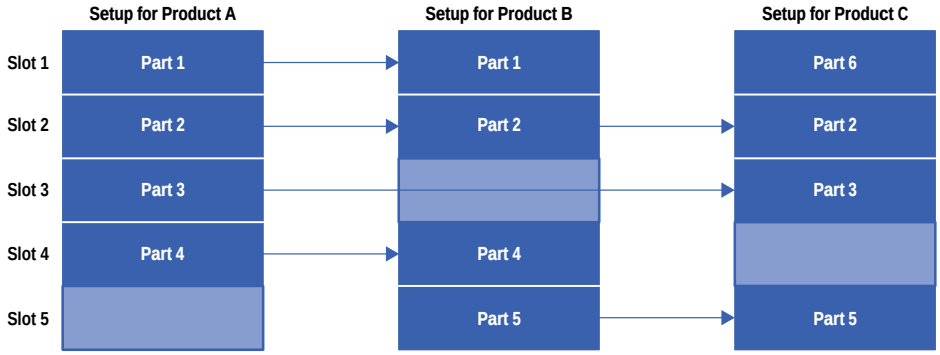
- Common setups for both sides of a printed circuit assembly. All of the parts needed for both sides of the printed circuit assembly are placed in a single setup.
- Partially fixed setups. Commonly used parts are assigned fixed locations on the machines.
- Family setups. All of the parts required to build a group of printed circuit assemblies are placed in a single setup. The printed circuit assemblies in the group are chosen so that all of the required parts will fit into one setup.
- Feeder bank exchange. Some machines offer the ability to change a large number of parts in the setup quickly by means of removable feeder banks. The operator can set up the parts for a product in an offline feeder bank while the machine is building a different product. The operator then trades the offline feeder bank for the online feeder bank, and can start building the new product immediately.
- Optimization by schedule, described below.

We soon narrowed the investigation to two options. We considered these to offer a good return on investment and to be a good fit with the architecture of HP's internal Man-Link system, which creates recipes for pick-and-place machines. The two options were family setups and optimization by schedule. We asked our customers to estimate the benefit to their sites, using mathematical models of these options. Based on their inputs, we chose to proceed with optimization by schedule. This was implemented in 1994.

Setup optimization by schedule takes advantage of the fact that many printed circuit assemblies use common components (Fig. 1). If the machine setup for each new printed circuit assembly is based on the setup of the previous printed circuit assembly, parts that are used in both don't have to be moved, and the total number of parts that need to be set up (called "feeder changes") is reduced. For example, in Fig. 1, four parts each are used on three products. Since parts that are common to multiple products are reused, the operator would only have to do six feeder changes (one each for parts 1 to 6) rather than the twelve that could be required if parts changed slots between products.

Further reduction in feeder changes can be achieved by changing the order in which printed circuit assemblies are built. Printed circuit assemblies with a large number of common parts should be built sequentially.

Fig. 1. *Setup optimization by schedule.*



The advantages of this approach include:

- It works well if feeder space is limited. For example, it has been used effectively in a line containing only one Fuji CP pick-and-place machine and one Fuji IP pick-and-place machine. In contrast, partially fixed setups and family setups require a larger amount of excess feeder capacity to be effective.
- It does not require any additional equipment or stock, as is required to use feeder bank exchange.
- It works well for very small lot sizes, even single panels.

Disadvantages include:

- It requires machine downtime to change feeders. The user can prepare feeders for the next product, but cannot set up offline and use feeder bank exchange or the Fuji CP split-bank mode.
- It has limited effectiveness if the schedule cannot be set by the setup optimization tool. Estimates of the lost effectiveness vary from 10% to 60%.
- Recipe generation must be dynamic, that is, recipes must be generated after identifying the printed circuit assemblies to be built in a particular time period.
- Late schedule changes may reduce the tool's effectiveness, or require that it be rerun.

SOPT

A software tool to optimize schedules by setup, called SOPT (for *s*etup *opt*imization), was created by HP Laboratories and later adapted to an HP proprietary recipe generation system. This system generates programs, or recipes, for pick-and-place machines. Any references to SOPT in this paper refer to this system.

The SOPT tool works on a list of printed circuit assemblies. It considers only those parts assigned to a particular machine for each printed circuit assembly. First, the printed circuit assemblies are ordered using a traveling salesman heuristic with the number of common parts between printed circuit assemblies as the cost function. Next, a setup is generated for each printed circuit assembly, and the number of feeder changes is calculated. Finally, the schedule is perturbed, and the feeder changes are recalculated to determine if a slightly different schedule would be an improvement. When no further improvements can be found, the program stops. The user can alter the order (for example, if one printed circuit assembly must be built first). The program will then recalculate the setups.

Outputs of SOPT include setup files for the printed circuit assemblies, and an operator instruction file giving the ordered list of printed circuit assemblies and the feeder changes required between each pair of printed circuit assemblies.

Man-Link

Man-Link is a newer HP-proprietary tool that generates recipes for pick-and-place machines. The assembly process is modeled as an ordered list of steps. In each step, a machine or person installs parts on one side of the printed circuit assembly. To generate recipes for all of the steps in a process, Man-Link executes the following sequence of tasks:

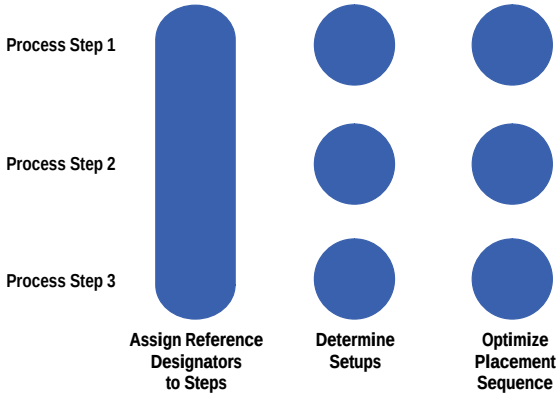
- Assign Reference Designators. Responsibility for the placement of each reference designator is assigned to a step in the process. In making assignments, Man-Link considers other setups, user-input responsibilities, user step preferences, the number of parts each machine can handle, balance among steps, and other factors. This task must be done once for the entire process, to ensure that each placement is assigned to exactly one step.
- Generate Setups. For each step, Man-Link determines where the parts assigned to the step should be loaded on the pick-and-place machine, considering other setups, machine constraints, placement speed, and other factors. This task is normally done separately for each step.
- Optimize Sequence. For each step, Man-Link determines the order in which parts should be placed. This task is always done separately for each step.

Fig. 2 shows these tasks. In Figs. 2, 3, and 4, each shape identifies a part of the problem space that is solved as a unit, either by a single program invocation or by a series of program invocations sharing data. In Fig. 2, assignment of reference designators to steps is done by a single program invocation to ensure that each reference designator is assigned to exactly one step. The other tasks are done by individual program invocations for each step because there is no need to share information among steps.

These tasks are performed in two phases. The first, called *strategic recipe generation*, performs an initial subsequence of these tasks and stores its output in a database. The second phase, called *tactical recipe generation*, performs the rest of the tasks, producing the final recipes that are used to build the printed circuit assembly.

It did not seem prudent to integrate the SOPT tool into Man-Link for a number of reasons. These included the SOPT implementation language (Pascal) and the data structures for setups (SOPT uses files, while Man-Link uses database tables). We decided to create a Man-Link solution using SOPT ideas and algorithms. In the following sections, some of the design issues are discussed.

Fig. 2. *Man-Link recipe generation model—single product, multiple steps.*



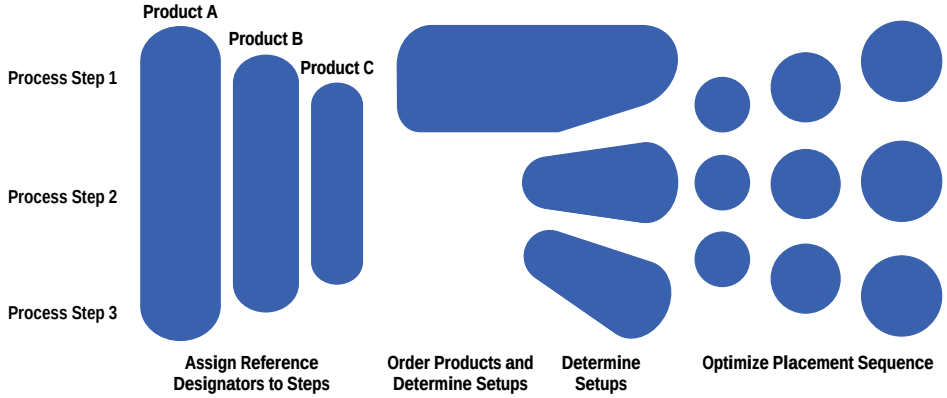
Build Lists

To handle multiple printed circuit assemblies in Man-Link, we introduced *build lists*, which are lists of printed circuit assemblies. We modified the programs to generate recipes for all of the printed circuit assemblies in a build list. In the process, Man-Link must order the build list and create setups with minimal feeder changes.

Ordering and Responsibility

An early issue in the design was how to integrate build list ordering into the list of tasks given above. The SOPT implementation has a single program that orders the list and generates setups for one step (Fig. 3). For each of the other steps, all of the setups are determined together but the list order determined for the first step cannot be altered. The advantage of this approach is that the ordering algorithm has detailed knowledge of what parts will or will not fit on a machine, and can therefore calculate (and optimize) the exact number of feeder changes required. The disadvantages are that the ordering program must include all of the setup code for the target machine (and therefore be machine-specific), and that the ordering considers only one step. An order that is very good for one step in the process may be very bad for another.

Fig. 3. *SOPT recipe generation model.*



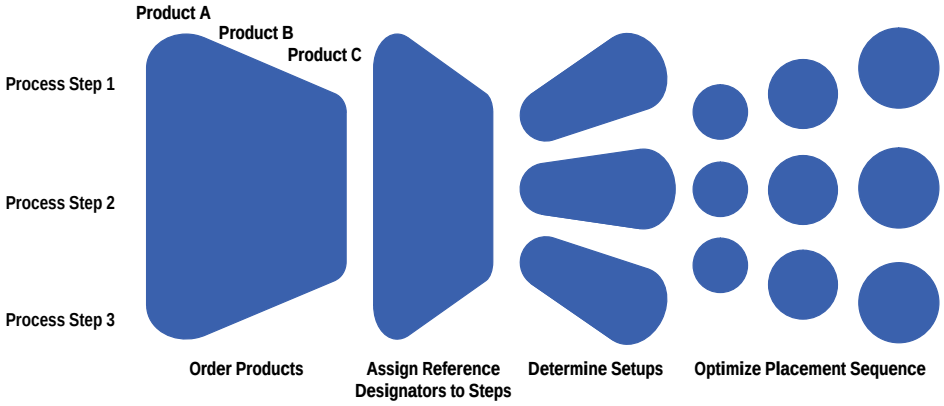
For the Man-Link implementation, we wanted to keep our code modular and to consider all steps when ordering. For these reasons, we decided to separate ordering and setup generation.

As a separate task, build list ordering must precede the generation of setups, since the setups are dependent on the printed circuit assembly sequence. The next question is whether ordering should precede or follow the assignment of reference designators. If ordering is done first, then assignment can use that information. On the other hand, if assignment is done first, ordering can focus accurately on the parts assigned to selected machines.

Perhaps the most compelling consideration was that our users wanted to be able to alter the order after the ordering program had run, but before the setups had been generated. To do this within the Man-Link architecture, we had only two options:

- Make ordering the last task in strategic recipe generation. This means that setup generation must be tactical, and therefore, tactical recipe generation would always have to be run for the entire build list.
- Make ordering a separate phase preceding strategic recipe generation. The strategic phase could then include set-up generation, and the tactical phase could be run for a single printed circuit assembly on a single step.

Fig. 4. *Man-Link recipe generation model—multiple products with setup optimization.*

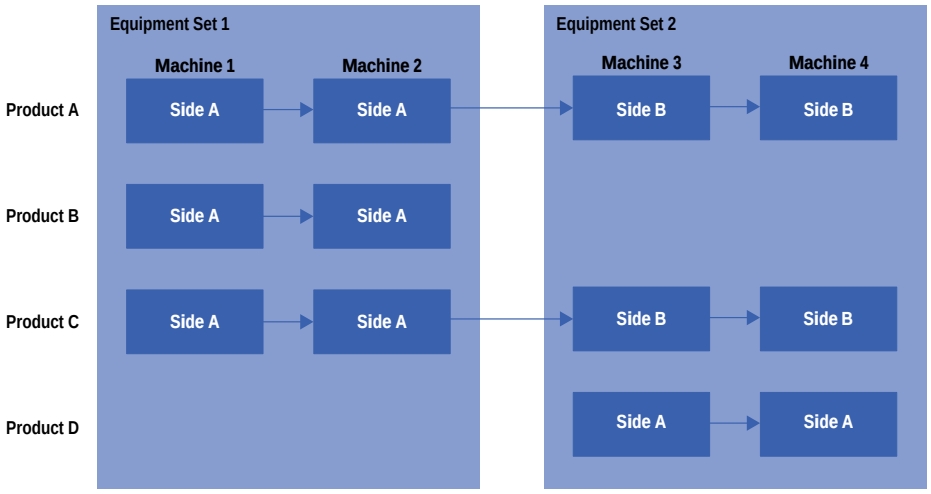


After discussing these issues with our users, we decided to implement the second alternative. Fig. 4 shows that ordering is done for all products, independent of steps, before any other task. Assignment of reference designators to steps is then done by a series of program invocations sharing data and using ordering information.

Equipment Sets

Since build list ordering precedes reference designator assignment in Man-Link, the ordering cannot consider the set of parts assigned to a particular step. We therefore thought we could just order printed circuit assemblies based on all of the parts on the printed circuit assemblies. The problem with this approach is that the parts on the two sides of a printed circuit assembly are often quite different. This could lead to poor results for processes consisting of one set of machines for top-side placement and a second set of machines for bottom-side placement. For this reason, we decided to order top-side parts separately from bottom-side parts. To do this requires the concept of an *equipment set*, which is defined as the set of machines used to place all of the parts on one side of a printed circuit assembly. For example, in Fig. 5, machines 1 and 2 form one equipment set, and 3 and 4 form a second equipment set. The ordering program will consider separately the parts assigned to each of the two equipment sets.

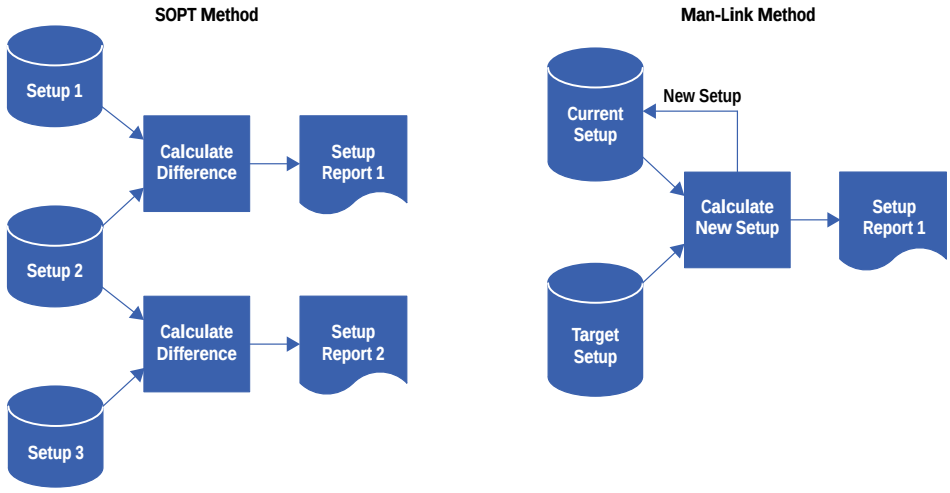
Fig. 5. *An equipment set is the set of machines used to place all of the parts on one side of a printed circuit assembly.*



Schedule Dependence

In the SOPT implementation, any change in schedule normally requires that the SOPT program be rerun. We hoped to alleviate this in Man-Link by separating setup generation from operator instruction generation. The idea is to keep track of the parts loaded on a machine. When the operator selects the next printed circuit assembly to be run, Man-Link runs a machine dependent program that compares the current setup with the setup for the new printed circuit assembly (Fig. 6). This program lists the minimum number of changes required to get the required parts for the new printed circuit assembly in the right slots. If the operator follows the determined schedule, changes are minimized. If the operator deviates from the schedule, there may be more changes, but the setups do not have to be regenerated. This should be particularly effective if the only change is to insert a prototype into the schedule.

Fig. 6. *Calculating operator instructions.*



The difficulty with this approach is that Man-Link must always know the current setup. We have a means in Man-Link to ensure this, but it is not robust and has been a source of frustration to our users.

Steps and Machines

A Man-Link process is made up of a sequence of steps, each of which refers to a machine. A particular machine may be used for more than one step in a process. In this case, Man-Link could do one of two things. First, Man-Link could treat each step independently, creating a separate sequence of setups for each step. This assumes that the user runs all printed circuit assemblies through a given step before going on to the next step, which is not realistic. Alternatively, Man-Link could generate all setups for a given printed circuit assembly before going on to the next printed circuit assembly, and create a sequence of setups for each machine. We chose to implement the second approach.

One result of this decision is that the setups for each printed circuit assembly must be sequential in the setup list. For example, in a two-sided surface mount process, the setup for each printed circuit assembly's bottom side will immediately precede the setup for its top side, assuming the same machine is used for both sides. (See the next section for a way around this constraint.)

This is a change from the SOPT implementation, which treats different sides of a printed circuit assembly as separate printed circuit assemblies. This turns out to be both good and bad—good because the sides can be ordered with other printed circuit assemblies to require fewer feeder changes, but bad because SOPT may put the sides in the wrong order, requiring manual shuffling of the schedule.

One related note—Man-Link can be configured to create a single setup for a pair of steps using the same machine. For example, if a two-sided process uses the same CP3 machine to place both the top and the bottom sides of a printed circuit assembly, then the user can configure Man-Link to create one CP3 setup containing all the parts required for both sides. In this case, the sequence of setups for the CP3 would have only one setup for the two-sided printed circuit assembly.

Separating Sides

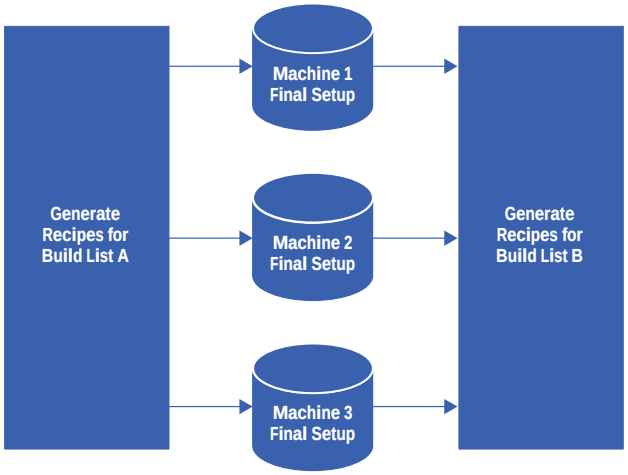
For certain applications, it may be advantageous to treat the different sides of a printed circuit assembly as separate printed circuit assemblies. This would allow the top and bottom sides of the printed circuit assembly to be separated in the build list, or even to be in different build lists. This feature would be useful for processes that use different machines to place the top and bottom sides.

Man-Link does not support this capability directly, but does provide a way to achieve it. The user must define a dummy machine that does not cause setups or recipes to be generated. To satisfy the reference designator assignment module, steps using this machine can take responsibility for parts. The user must then define a top-side process with a dummy step for the bottom side, and a bottom-side process with a dummy step for the top side. To generate recipes for a double-sided printed circuit assembly, the user must run recipe generation using both processes. The two printed circuit assembly/process pairs can be included at different points in a build list, or even in different build lists.

Starting Setups

To get the most out of optimization by schedule, it is best to take advantage of any parts left on the machines from previous builds. In other words, the last setups for yesterday's build list should be an input for recipe generation for today's build list (Fig. 7). In generating the setups for today's first printed circuit assembly, we can use any parts left over from yesterday's last printed circuit assembly setups.

Fig. 7. *Using starting setups.*



Man-Link provides this capability by saving the last setup generated for each machine. This works well as long as setups are generated by only one person at a time. If two people try to generate setups for the same machines at the same time, they will overwrite each other's starting setups, causing all manner of confusion. Man-Link therefore has the constraint that strategic recipe generation for build lists may not be done for the same machines by more than one user simultaneously.

A suggested way around this constraint is to have a directory for starting setups for each user. This would keep multiple users from interfering with each other, but does not address the problem of determining the correct starting setup for each machine, that is, how each machine will be set up before the first printed circuit assembly in the build list is started.

Estimating Benefits

To assist users in projecting the benefits of optimization by schedule, we constructed this model of the setup process:

- Let N = the number of printed circuit assemblies to be built in a schedule.
- Let C = the average number of parts in the printed circuit assemblies.
- Let P = the probability that any given part used by a particular printed circuit assembly is also used by a second given printed circuit assembly. In other words, P is the average fraction of parts shared by any two printed circuit assemblies in the schedule. For typical build lists, this varies from around 0.13 to 0.37, with an average of about 0.25.
- Let F = the number of parts the machines can hold. For a CP3, this is around 112, assuming that three times as many one-slot parts are used as two-slot parts.
- Let U = the total number of unique parts for all of the printed circuit assemblies in the schedule.
- Let X = the total number of feeder changes required by the setups.

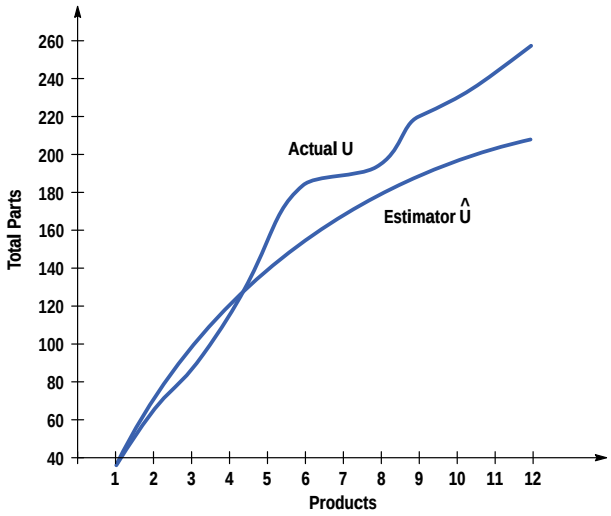
The average number of parts that are common to two printed circuit assemblies is CP. The total number of parts in two printed circuit assemblies is therefore $C + (C - CP)$, or $2C - CP$. Assuming that commonality is uniformly distributed, the average number of parts common to three printed circuit assemblies is CP^2, so the total number of parts in three printed circuit assemblies is $C + (C - CP) + (C - 2CP + CP^2)$. Continuing this reasoning, we arrive at the following estimator of U:

$$\hat{U} = \frac{C}{P}\left(1 - (1 - P)^N\right). \qquad (1)$$

This should immediately raise some alarms, because it predicts that the total number of unique parts converges to C/P as the number of printed circuit assemblies (N) gets large. In fact, this turns out to be a reasonable first-order estimator if the number of printed circuit assemblies is not too big. If N is between 2 and 5, it is pretty good, but as N approaches 10, it is consistently low. Fig. 8 shows estimator values and actual values based on a random collection of products from one surface mount center.

The number of feeder changes required, X, depends on F. If $F < C$, the parts for a printed circuit assembly will not fit on the machines, so C must be a lower bound for F. If $F \geq U$, all of the parts may be mounted at once, so the number of changes depends on the commonality with the starting setups. If the starting setups do not contain any parts in the printed circuit assemblies, then $X = U$. If the starting setups have all of the parts, $X = 0$. In the real world, X is generally between these two extremes, and in fact, $\hat{U}$ turns out to be a reasonable estimator. In an analysis of 25 build lists used at surface mount centers, $\hat{U}$ predicted a 61% decrease in feeder changes, compared to an actual decrease of 65%.

Fig. 8. *Accuracy of the U estimator based on data from one surface mount center.*



Ordering the build list for maximum commonality can raise the effective value of P from an average of 0.26 to an average of 0.28 in the 25 build lists mentioned above.

Setup vs. Cycle Time

Most strategies that reduce setup time do so at the expense of cycle time. In other words, you can save time setting up the machine if you are willing to live with setups that do not build printed circuit assemblies as quickly. There are several reasons why this occurs:

- A setup that can be used for multiple printed circuit assemblies is less compact. It requires greater machine movement (either of the part feeder carriage or of the placement heads) to access the parts for each printed circuit assembly.
- On the Fuji CP machines, a movement of one slot on the part feeder carriage between placements does not slow the machine down. The Man-Link sequence generators take advantage of this by optimizing placement sequences for pairs of adjacent parts. Careful selection of parts to put together in the setup can therefore have a big impact on the cycle time.
- The Fuji CP3 must place all "tall" parts after all "short" parts. An optimized setup will have all of the tall parts at one end of the part feeder carriage, so each printed circuit assembly can be built with a single pass through the carriage.
- Fuji IP2 optimization is compromised. When Man-Link generates a setup for a single printed circuit assembly, it will balance the loads on the two heads. The user can request that a high-use part be loaded on both banks to further improve the balance. Also, high-use parts are assigned to slots with the lowest access times.

The combined impact of these effects has been measured to be an increase of cycle time of between 5% and 10%. For small lot sizes, this is a good trade-off, but for large lot sizes (greater than 100 panels) overall throughput will suffer.

Parallel versus Serial Lines

As noted above, the benefit derivable from this setup strategy is dependent on the amount of excess feeder capacity available. The immediate temptation is to put machines in series to increase the effective feeder capacity. This could be a mistake, however, because the utilization of machines in series decreases as a result of the increased overhead of conveying the panels and reading marks more than once. This cost can be modeled as follows:

- Let V be the overhead per panel. This is the sum of the conveying time, the time to read fiducials, and the time to read image-reject marks.
- Let S be the average number of seconds required to place a single component. For the CP2 and CP3 machines, a figure of 0.3 second works well.
- Let M be the number of placements to be done by the CP machines.

For one CP machine, the time required to build a panel is
V + MS seconds. If two CPs in series build the panel, each placing half of the parts, the total time is:

$$\text{Total machine time} = 2 \times \left(V + \frac{M}{2} \times S\right) = 2V + MS. \quad (2)$$

The increase in cycle time caused by having two machines in series rather than in parallel is:

$$\text{Change in total machine time} = \frac{2V + MS}{V + MS}. \quad (3)$$

Assume that the overhead is 17 seconds. This is enough time for conveying the panel and reading four panel fiducials and two image-reject marks. For panels with 100 placements, the cycle time is increased by 36% by the serial configuration. Again, for some shops this will be a reasonable trade-off.

The Bottom Line

To estimate the potential value of this optimization method, the user must:

1. Estimate the reduction in setup time. If the surface mount center is currently performing a complete teardown and setup for each printed circuit assembly, the number of feeder changes should be reduced from $N \times C$ to approximately $\hat{U}$ from equation 1. If it takes T seconds to change one feeder, the savings is $(NC - \hat{U})T$.

For example, for five products, with an average of 50 parts per product, and part commonality of 0.20, the expected number of feeder changes $\hat{U}$ will be 168. This means that the reduction in feeder changes will be $250 - 168 = 82$. If each feeder change takes one minute, the total reduction in feeder changes will be 82 minutes, or about 16 minutes per product.

2. Estimate the increase in cycle time. This will be about 10% of the total run time. If machines that are currently running in parallel need to be put in series to get sufficient feeder space, the user should also estimate the resulting increase in total machine time, using equation 3 above.

In the above example, if the average number of placements per product is 400, and the average placement time is 0.3 second, the increase in cycle time will be about $400 \times 0.3 \text{ second} \times 10\% = 12 \text{ seconds} = 0.2$ minute/panel.

3. The break-even lot size can then be calculated by setting the decrease in setup time equal to the increase in cycle time. In the above example:

$$\frac{16 \text{ minutes/product}}{0.2 \text{ minute/panel}} = 80 \text{ panels/product}.$$

4. Alternatively, the user can calculate the time savings per lot. In the above example, assuming 10 panels per lot:

- Without optimization: 50 minutes setup + 20 minutes build = 70 minutes
- With optimization: 34 minutes setup + 22 minutes build = 56 minutes
- Savings per lot = 14 minutes.

This represents a 20% decrease in the time to build a lot.

We expect an improvement for lines that are dominated by setup time, such as lines for prototypes or high-mix, low-volume products. If the lot size is consistently less than 100 panels, optimization by schedule may be a good fit.

Conclusion

Life is a series of trade-offs. We believe that the choices we made in this work will provide the best benefit to our customers. We have received encouraging statistics from the surface mount centers that are using the product. The estimation methods given above should allow other surface mount centers to evaluate this strategy for their shops.

Acknowledgments

Many thanks to Shailendra Jain for dreaming this up, creating a prototype, and providing crucial input on our efforts. Thanks to Fereydoon Safai and Ed Katz for their work in the design and implementation of both the SOPT and Man-Link solutions. And thanks to our customers, who funded the project, and to Eiko Johnson, who managed it.

▶ Go to Article 10
▶ Go to Table of Contents
▶ Go to HP Journal Home Page