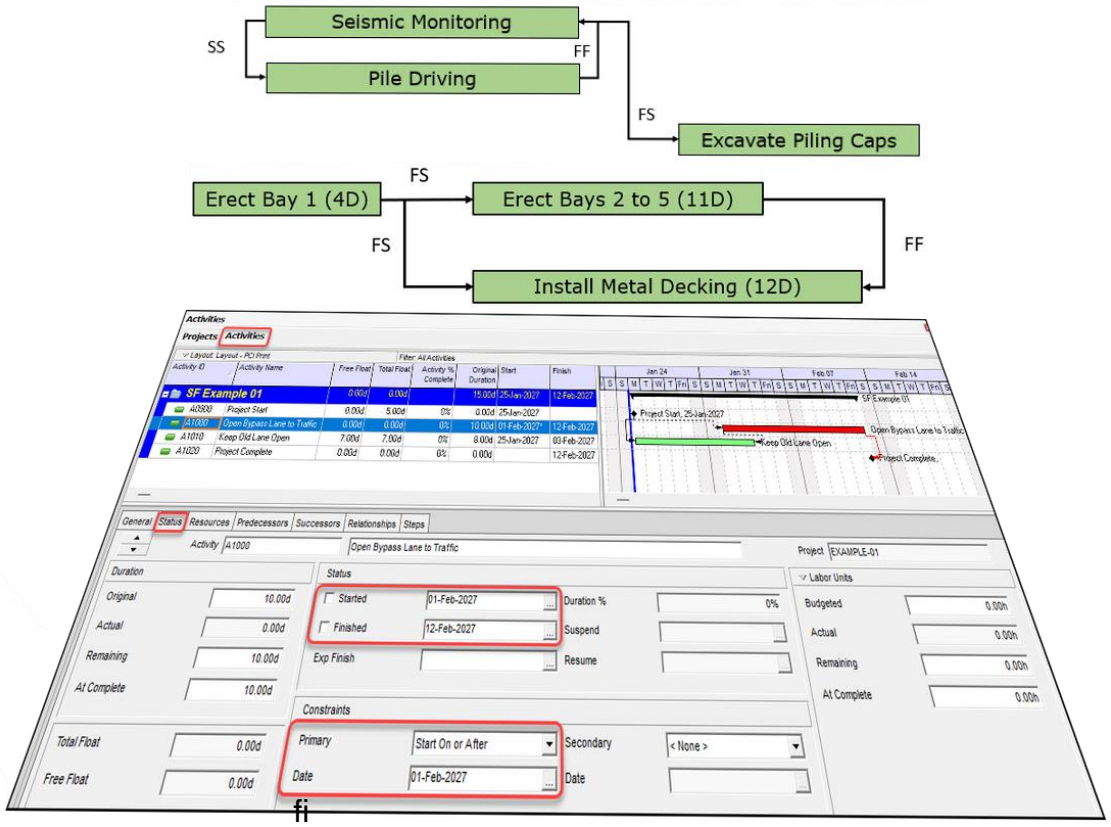
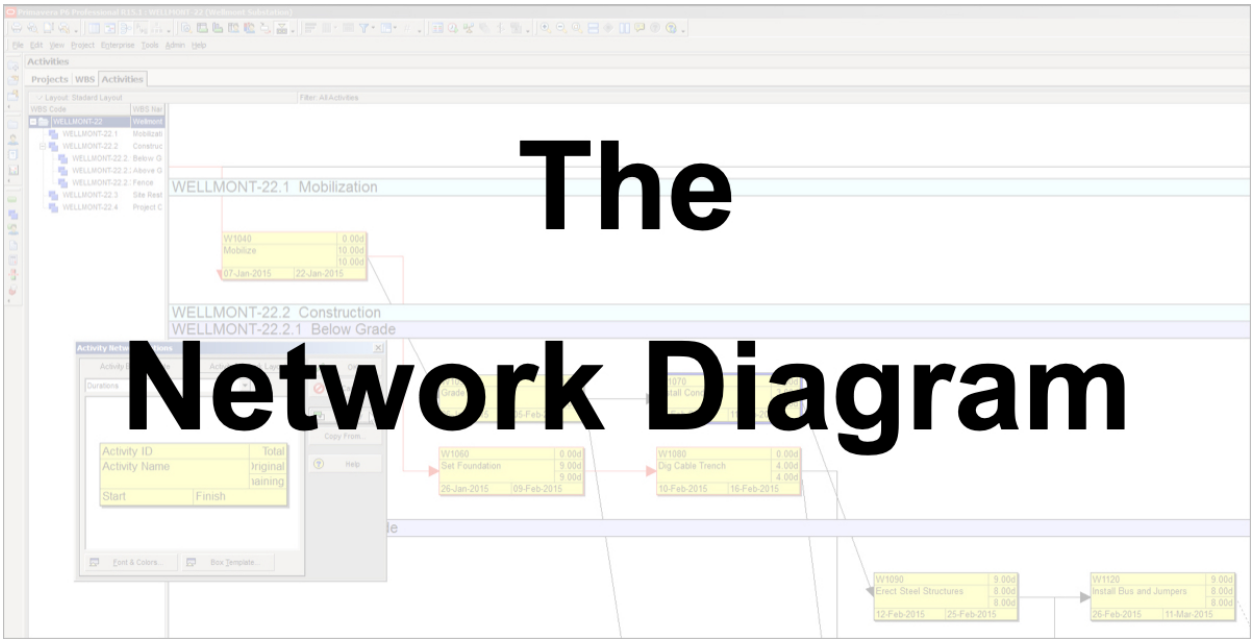
The Definitive Guide to Primavera P6 Relationships



fi



We start with the Network Diagram to provide some context before we go into the Primavera P6 Relationship Types.

While most people recognize a P6 schedule by its Gantt Chart, the chronological roadmap of a project, the Network Diagram is the true engine behind the dates. It is the primary tool for visualizing project logic, focusing purely on how tasks interconnect rather than where they sit on a calendar.

For further details on the network diagram in P6 Professional, we recommend this Ten Six insight:

The Primavera P6 Professional Network Diagram

The Network Diagram uses the same data found in the Gantt chart (tasks and Work Breakdown Structure (WBS) elements, i.e., deliverables), but it strips away the calendar timescale to highlight the project's "web" of dependencies. It consists of two fundamental elements:

- Nodes (Activity Boxes) – These represent individual tasks. They display key data, such as Activity ID and Start/Finish dates, which the system uses to calculate the project's total duration.
- Arrows (Logical Relationships) – These lines connect the nodes, defining the specific dependencies that dictate the sequence of work.

Nodes are connected by four types of Arrows or Relationship lines that help the scheduler describe the true scheduling situation.

This article covers the essential elements of a network diagram, providing the practical foundation for building an effective project schedule.

Dynamic Scheduling

A proper Network Diagram is necessary for dynamic scheduling. A scheduler should not include manually assigning the start and finish dates of every task in the schedule. Manually assigning dates negates the purpose of the scheduling software, turning a dynamic tool into a static document.

Instead, the goal is to build a dynamic schedule in which P6 automatically calculates dates from core, high-quality inputs. A dynamic P6 schedule calculates each task's start and finish time based on its duration and relationships with predecessor and successor activities. Robust schedules leverage these

logical relationships across the network to ensure that every task is dynamically tied to the project as a whole.

Elements of a Network Diagram

The network diagram consists of Activity Boxes connected by Arrows. Like the Gantt chart, it displays critical date information; however, these dates are not determined by the box's physical position on a timeline. Instead, the dates are populated directly inside each Activity Box as the software's output data fields. This allows the scheduler to focus on the project's logical flow rather than its placement on a calendar grid.

Activity Box's Pertinent Information

The minimum information a scheduler provides for an activity is a descriptive task name and its duration. Once these are entered – along with the relationships between them (see the arrows section below) – and the schedule is calculated, P6 populates the remaining fields. Figure 1 illustrates the standard layout of an Activity Box:

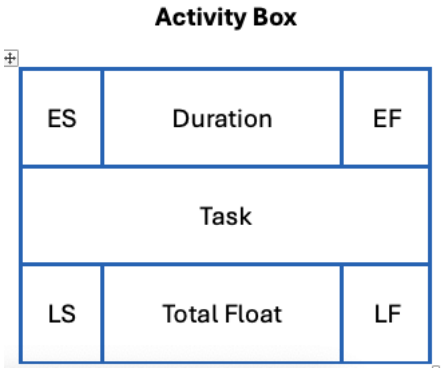


Figure 1

Breakdown of the Activity Box

1. Task (Center Middle) – Descriptive name of the task effort.
2. Duration (Center Top) – The duration in days to complete the task.
3. Early Start (ES, Upper Left) – The earliest a task can begin.
4. Early Finish (EF, Upper Right) – The earliest a task can conclude, calculated as ES + Duration.
5. Late Start (LS, Lower Left) – The latest date the task can start, calculated as LF – Duration.
6. Late Finish (LF, Lower Right) – The latest date the task can finish without delaying the project's completion date.
7. Total Float – The time in days a task can delay without delaying the project's finish date.

Arrows

The arrows graphically show how one Activity Box relates to another, always pointing in the direction of workflow (left to right). These connections serve three critical roles:

- Defining Dependency: They establish the sequence of events by designating predecessors and successors.

- Driving the Dates: They act as the “sinew” of the schedule, transmitting timing data from one task to the next.
- Path Generation: Linking these nodes establishes the logical network necessary for P6 to identify the Critical Path.

Requirements for a Practical Working Network

Good logic is the backbone of a healthy schedule. The following requirements are essential for maintaining a network with mathematical and operational integrity.

8. Eliminate Dangling Activities: Every activity must have at least one predecessor and one successor. Without these links, a task becomes unresponsive to network changes, masking the ripple effect of delays and preventing an accurate completion forecast.

9. Prioritize Dynamic Logic: Avoid Hard Constraints (like Mandatory Finish) which override network relationships. These constraints “freeze” activities, making them unresponsive to changes in predecessors and preventing automatic date recalculations.

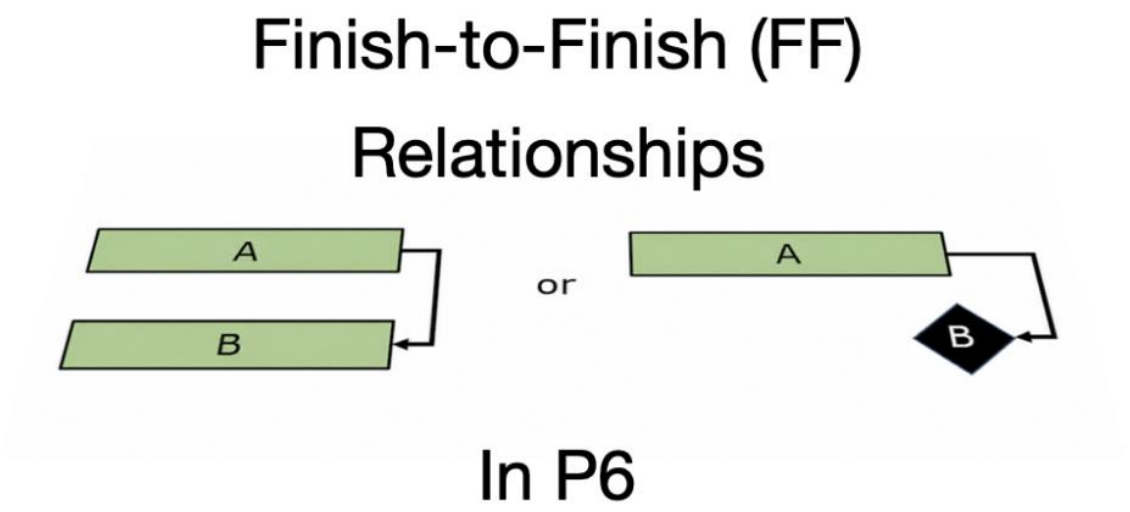
10. Confirm Driving Logic: Every task must belong to a network where driving predecessors are identifiable. This ensures the software accurately calculates and communicates which specific activities are controlling the project timeline.

Summary

The network diagram is the engine of the schedule; it determines the start and finish dates of tasks. The network diagram is composed of Activity Boxes and connecting Arrows. The minimum required information to calculate start/finish dates is the task duration and logic connections.

But the Activity Box also includes ES, EF, LS, LF and Total Float data. An arrow between boxes displays how one task relates to another. A working, practical network requires eliminating dangling tasks, prioritizing dynamic logic and confirming the driving logic.

In P6 Professional, the network diagram includes WBS swim lanes, which organize the network by deliverables and offer deep visibility into how tasks across different WBS elements interconnect.



Finish-to-Finish (FF) relationships define how two tasks conclude in relation to each other. They are most helpful when a task cannot officially finish until another concurrent task crosses the finish line. Let's take a look.

The FF relationship dictates that a successor task cannot finish until its predecessor task concludes. A classic example is writing and editing a report. There is no logical way you can conclude editing a report before it is completely written. And after the writing is finished, the editor will need a distinct window of time, a positive lag, for a thorough review of the report.

If the Finish-to-Start (FS) relationship is the tool for fostering transparency and the Start-to-Start (SS) is the tool for acceleration, the FF is the tool for coordination. The running analogy is relevant: SS manages the send-off at the starting gun, but the FF governs the handoff at the finish line, which requires significantly more coordination than starting simultaneously. And it is always easier to begin an effort than to conclude one, and to ensure that the conclusion is well orchestrated.

This article explains the fundamentals of the P6 FF relationship and its utility in coordinating the conclusion of two tasks.

The Network Diagram

Schedulers map out their projects using a Network Diagram. This graph creates a clear visual representation of the schedule, showing the chronological flow of work and the dependencies between activities. We introduce the basic elements of this visual tool in our following Ten Six article:

The Network Diagram

Much like a Start-to-Start link, Finish-to-Finish (FF) relationships allow two activities to run concurrently. The successor must wait for the predecessor to cross the finish line before it can cross the finish line itself. This can result in a simultaneous "photo finish" or a longer separation, giving the successor time to wrap things up. Yet, while the FF relationship is essential for describing a concluding handoff, it is used only when necessary, favoring standard FS relationships whenever feasible.

Understanding the FF relationship

An FF relationship means Activity B cannot finish until Activity A finishes. Both activities can run in parallel because the logic constraint applies only to their completion dates. In Figure 1, the successor task-dependent activity or milestone cannot finish until the predecessor concludes.

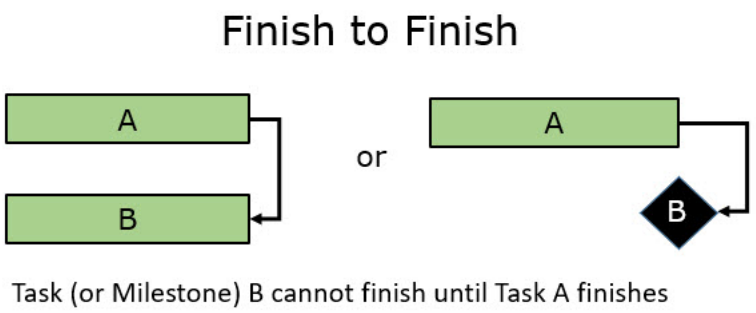


Figure 1

Like the SS, the FF can overlap in whole or in part. Add a lag (FF + 2D), and you're saying B must finish at least 2 working days after A finishes, as displayed in Figure 2.

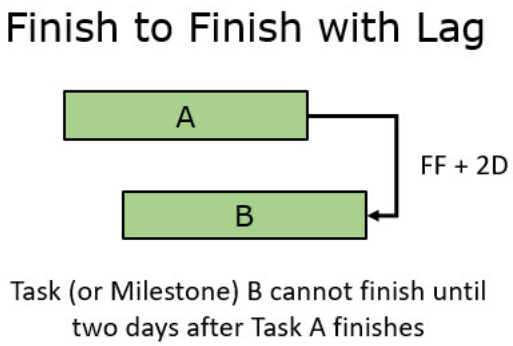


Figure 2

A negative lag reverses this logic, allowing the successor to finish before the predecessor. This is generally bad practice and highly confusing in a Finish-to-Finish (FF) relationship. The DCMA forbids negative lag on any relationship type: FS, SS, FF, and Start-to-Finish (SF).

Of the relationship types, the FS is favored because it makes schedules more transparent. When the SS, FF, and SF are summed together, their presence in the schedule should be 10% or less of the total number of relationships. While the use of FF is not prohibited, you should limit it to scenarios where coordinated completion is critical.

Calculating FF Relationships

Important Note on Paper vs. Software
On paper, we use +1 and -1 steps to jump across calendar day boundaries. When you type these relationships into P6, the software handles these adjustments automatically in the background.

Forward Pass — Finding the Earliest Dates

The forward pass works left-to-right, computing the earliest each activity can start and finish. With FF, P6 calculates the successor's Early Finish first, then back-calculates its Early Start.

The Formula

Early Finish: EF(B) = EF(A) + Lag

Early Start: ES(B) = EF(B) – Duration(B) + 1

The key difference from FS is that here P6 is constraining B's finish date, not its start date. ES(B) is derived from the constrained EF(B). If B has multiple predecessors, P6 evaluates all of them and takes the most constraining (latest) result.

Heads Up On Calendars: *FF lag is applied on the predecessor's calendar by default; lag days are counted against B's working days, not A's. This matters when the predecessor and successor are on different shift patterns. Verify in Tools | Schedule | Options > Calendar for scheduling Relationship Lag.*

Example:

Activity A (Predecessor): Starts on Day 1, with a duration of 10 days.

Activity B (Successor): Linked via FF + 2, with a duration of 6 days.

Activity	Early Start	Calculation Logic	Early Finish	Calculation Logic
A	Day 1	Project start	Day 10	ES(A) + Duration(A) – 1 = 1 + 10 – 1 = 10
B	Day 7	EF(B) – Duration(B) + 1 = 12 − 6 + 1 = Day 7	Day 12	EF(A) + Lag = 10 + 2 = Day 12

B runs Days 7–12, overlapping with A for 4 days (Days 7–10), but it can't wrap up until 2 days after A finishes.

Backward Pass — Finding the Latest Dates

The backward pass works right-to-left from the project finish date. With FF, the backward pass constrains the predecessor's Late Finish.

The Formula

LF(A) = LF(B) – Lag

LS(A) = LF(A) – Duration(A) + 1

Subtract the lag from B's Late Finish to get the latest A can finish. A's Late Start is then back-calculated from there. If A feeds multiple successors, P6 takes the minimum (most constraining) Late Finish.

Example (Project Finish Deadline: Day 14):

Activity B (Successor): Duration 6 days. LF = Day 14.

Activity A (Predecessor): Duration 10 days. Linked to B via FF+2.

Activity	Late Start	Calculation Logic	Late Finish	Calculation Logic
B	Day 9	LF(B) – Duration(B) + 1 = 14 – 6 + 1 = 9	Day 14	Constrained by the project deadline
A	Day 3	LF(A) – Duration(A) + 1 = 12 – 10 + 1 = 3	Day 12	LF(B) – Lag = 14 – 2 = 12

⚠ **FF Late Start Is Not Directly Constrained:** ***With FF, the backward pass constrains A's Late Finish, not its Late Start. A's LS is just derived from LF(A) − Duration. This means A has flexibility on when it starts that isn't immediately obvious from the FF link alone. Always check float and any other incoming relationships to get the full picture.***

Total Float and the Critical Path

Once both the forward and backward passes are complete, P6 computes Total Float (TF):

TF = LS – ES (or LF – EF)

From our example with deadline Day 14:

Activity	ES	EF	LS	LF	Float	Critical?
A	Day 1	Day 10	Day 3	Day 12	2 days	No
B	Day 7	Day 12	Day 9	Day 14	2 days	No

Tighten the deadline to Day 12 (B's early finish) and both go critical.

Negative Lag (Leads)

A negative FF lag means B can finish before A. P6 will compute dates for it, but it's flagged as a quality deficiency by the DCMA 14-Point Assessment. If you need that kind of overlap, restructure with a positive lag or an intermediate milestone instead.

Now that we have established how standard Finish-to-Finish (FF) Relationships drive forward- and backward-pass calculations, we must examine how our choice of relationship and lags directly impacts schedule mechanics.

FF and the Dangling Start

We want Pour Concrete and Test Concrete to be simultaneous efforts, with Pour Concrete finishing before Test Concrete. In Figure 3, we model these simultaneous efforts with Pour Concrete concluding first, followed by Test Concrete, using an FF between them.

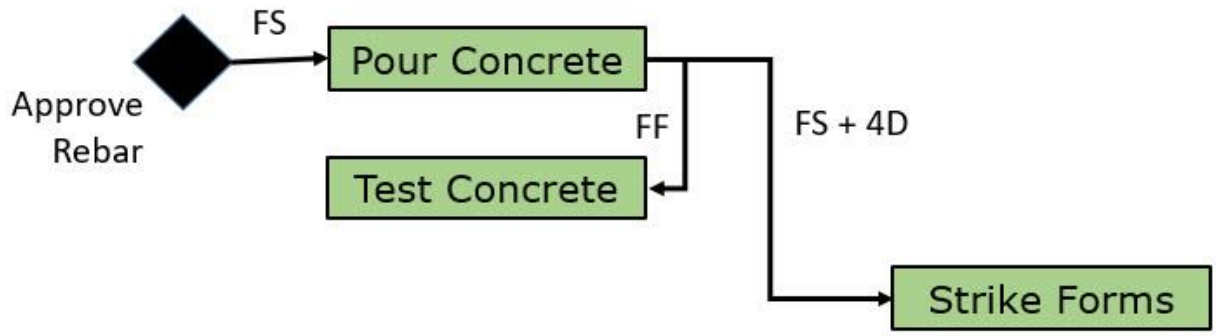


Figure 3

The problem with this approach is that Test Concrete is left with an open start. For both open start and open finish activities, Primavera P6 looks forward when calculating the Total Float, anchoring it either to the final project completion date or to an existing activity constraint (whether on that specific task or a downstream successor).

Because the activity lacks a proper logic anchor on its open end, this forward-looking calculation frequently results in excessive, unrealistic Total Float. This artificially inflates the task's flexibility and hides its true impact on the schedule."

So, while a windowed view cannot determine whether the entire project exceeds the allowable 5% threshold for missing logic in the DCMA 14-Point Assessment, this specific open-start logic still introduces a significant vulnerability in total float.

Though the FF link may technically satisfy the predecessor requirement and the activity may pass the full schedule Missing Logic check, the resulting float inflation directly risks driving the schedule over the 5% threshold for the High Float assessment.

Furthermore, if the schedule is cost-loaded, the software will attempt to distribute budgeted costs across this artificial, elongated window. This directly distorts the project's monthly cash flow projections and resource forecasting.

FF and SS Redundant Logic

In Figure 4, we address the Test Concrete open start by including an additional SS link from Pour Concrete to Test Concrete.

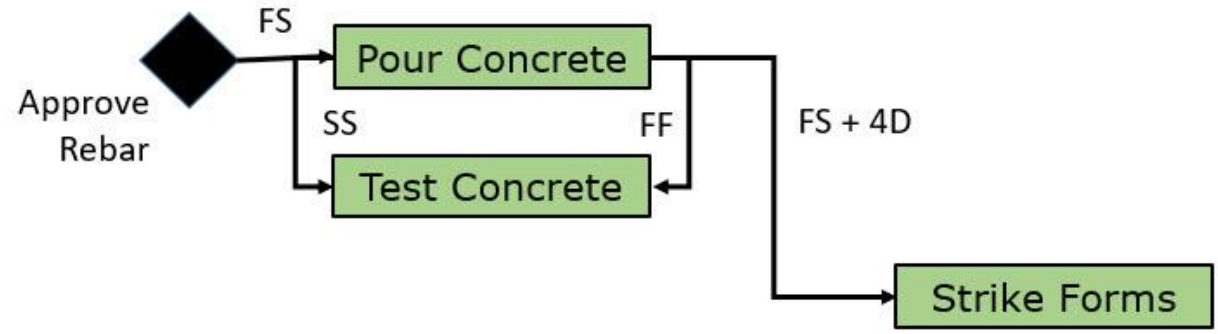


Figure 4

While this eliminates the open start, it introduces a closed logic loop with three distinct problems:

1. **Causes out-of-sequence progress errors** – Real-world field progress rarely follows the exact plan. If the pour is interrupted or crews record progress differently, the double-locked logic triggers out-of-sequence warnings, loop errors or negative float during monthly updates.

2. **Artificially Locks the Activity Duration** – A standard discrete activity must have its own independent duration based on its assigned resources. When you apply both an SS and an FF link, you create a conflicting dual constraint. Standard scheduling software is not designed to calculate discrete durations this way. It will often indiscriminately prioritize one link, ignore the other, and leave you with a rigid timeline that does not reflect the site's real-world flexibility.

3. **Fails Professional Schedule Audits** – Major auditing bodies frown upon double-linking because it masks the true critical path. This redundant logic will immediately trigger red flags in automated health checks.

FF and FS Tied Concurrent Paths

In Figure 5, both the open start on Test Concrete and the closed logic loop are resolved by removing the SS link and inserting an FS successor relationship from the Approve Rebar milestone to Test Concrete.

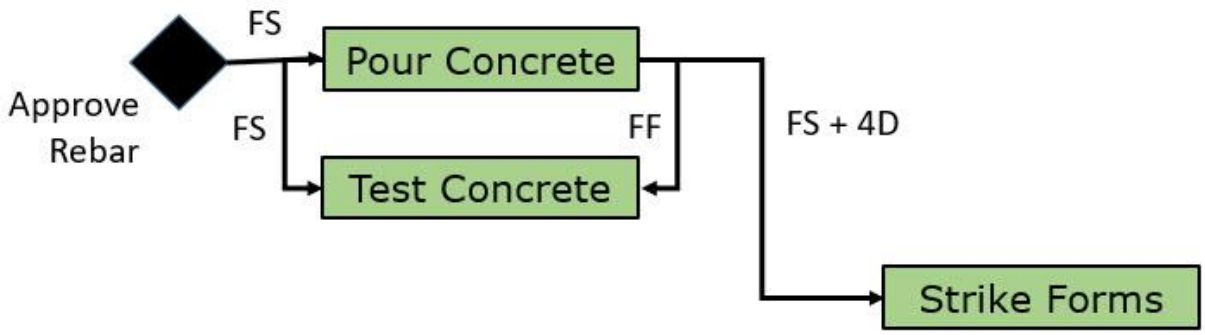


Figure 5

This logic structure is used when a single milestone triggers two separate activities that must run simultaneously, but one activity cannot officially conclude until the other is 100% complete.

- The Driver (FS): A Finish Milestone pushes out an FS link to both the primary task (Pour Concrete) and the support task (Test Concrete).
- The Tie (FF): The primary task anchors the support task's completion with an FF link.

There is some operational leeway at the start of these two simultaneous tasks. While it is theoretically possible, though not practical, for Test Concrete to begin before Pour Concrete, its start date cannot be earlier than the finish of the Approve Rebar milestone. This specific framework successfully eliminates out-of-sequence progress errors in the closed network loop while simultaneously removing the excessive total float because of the open start.

As an additional scheduling improvement, the four-day lag between Pour Concrete and Strike Forms should be modeled as a discrete task rather than a positive lag.

For a deeper analysis of FS relationships, refer to the Ten Six article on FS relationships at the following link:

Finish-to-Start (FS) Relationships in P6

Summary

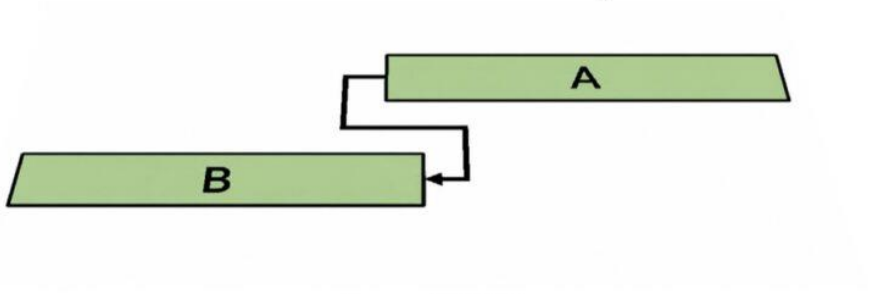
Finish-to-Finish (FF) Relationships in P6 anchor the finish of one task directly to the conclusion of another. It works exceptionally well for modeling simultaneous tasks that must be completed in a specific order. When using them, be careful to watch for open starts. These are highly problematic for scheduling software and will directly distort the accuracy of your schedule risk analysis.

While you can technically avoid an open start by using a dual-relationship solution (combining one SS and one FF link to connect the tasks), this approach generally does not work well in practice. A notable exception to this rule is the final scenario we used to demonstrate the proper utility of the SS relationship in our Ten Six article:

Start-to-Start (SS) Relationships in P6

Instead, using FF and FS to tie concurrent paths provides a much more workable, meaningful solution. Under this framework, both paths are cleanly restricted by a predecessor milestone's finish. They then branch out into simultaneous efforts using two separate FS relationships, ensuring the project's conclusion remains well-coordinated, with one task's finish tightly anchored to the other.

Start-to-Finish (SF) Relationships



In P6

Start-to-Finish (SF) relationships say one task cannot conclude until the former task starts. Its application is limited, but it may fit well in the right scenario. Let's take a look.

The SF relationship specifies that a successor task cannot finish until its predecessor begins. It is difficult to find a scenario where the SF relationship does not violate physical laws and is useful.

This rare relationship type, nevertheless, finds its place in the general scenario of maintaining an old system until the new one is in place. A good example comes from the construction of a traffic bypass lane. Workers must maintain the old lane until the bypass traffic lane opens (or starts). The key advantages of the SF relationship when properly applied:

- **Guarantees Zero Downtime**: It ensures a critical system (like traffic flow) never experiences a gap during handoff.
- **Protects Fail-Safes**: The old system remains fully operational as a backup until the new system is proven ready.
- **Automates Resource Retention**: If the start of the new system is delayed, the old system automatically extends, preventing premature demobilization.

This article outlines the fundamentals of the Primavera P6 Start-to-Finish (SF) relationship and its usefulness for modeling the continued operation of an old system until a new one begins.

The Network Diagram

Schedulers map out their projects using a Network Diagram. This graph creates a clear visual representation of the schedule, showing the chronological flow of work and the dependencies between activities. We introduce the basic elements of this visual tool in our following Ten Six article:

The Network Diagram

While Finish-to-Finish (FF) coordinates a joint ending, only SF guarantees a seamless handover by linking the old system's retirement directly to the new one's opening.

Understanding the SF relationship?

An SF relationship means the successor (Task B) can't finish until the predecessor (Task A) starts. This allows a brief window where both tasks operate in parallel.

Essentially, the predecessor's start triggers the release for the successor to complete the task. As shown in Figure 1, the dependent successor task is held up until the predecessor begins.

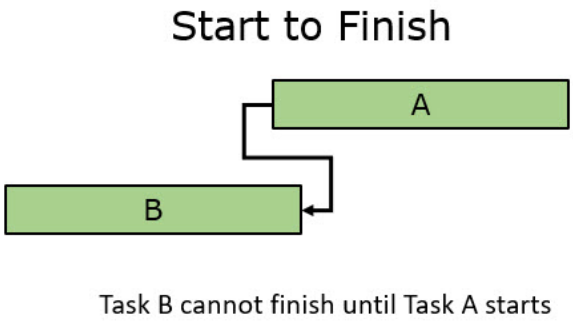


Figure 1

Adding a lag slightly modifies this dynamic. For example, an SF+2 lag means Task B cannot finish until at least two working days after Task A starts, as illustrated in Figure 2.

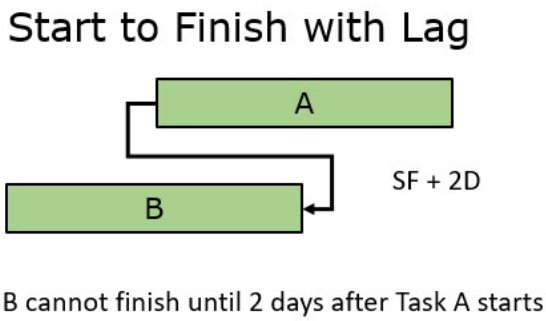


Figure 2

Some of the overarching applications of the SF relationship include.

- Just-in-time handovers: A night-shift team (B) cannot stand down until the day-shift team (A) arrives and starts. Task B's finish is gated by Task A's start.
- Phased resource transitions: An old system (B) must keep running until the replacement system (A) comes online. Task B cannot be decommissioned until Task A begins.

⚠ **SF is rarely the right answer:** *Before using SF, ask whether a restructured FS or SS relationship would make the same logic clearer. SF can obscure the driving constraint, making schedule analysis significantly harder. If you do use it, document the business reason, as auditors will ask.*

Calculating SF Relationships

Important Note on Paper vs. Software
On paper, we use +1 and -1 steps to jump across calendar day boundaries. When you type these relationships into P6, the software handles these adjustments automatically in the background.

Forward Pass – Finding the Earliest Dates

In the SF relationship, the predecessor's start date drives the successor's finish date — not its start date. Because the successor must finish the workday before the predecessor can start, we must subtract 1 day on paper to find the correct finish boundary.

The Formula

EF(B) = ES(A) + Lag – 1

ES(B) = EF(B) – Duration(B) + 1

P6 constrains B's Early Finish first, then back-calculates B's Early Start from there. This is the opposite of how you'd normally think about an activity's dates.

Calendars and Lag: *SF lag defaults to the predecessor's calendar in P6. Verify in Tools | Schedule > Options, particularly when predecessor and successor are on different shift calendars.*

Example:

Activity A: 10-day duration, starts Day 1 (EF = Day 10).

Activity B: 8-day duration, SF+2 from A.

Activity	Early Start	Calculation Logic	Early Finish	Calculation Logic
A	Day 1	Project start	Day 10	ES(A) + Duration(A) – 1 = 1 + 10 -1 = Day 10
B	Day(−5)*	EF(B) – Duration(B) + 1 = 2 − 8 + 1 = Day (−5)	Day 2	ES(A) + Lag – 1 = 1 + 2 – 1 = Day 2

* Yes, Day (-5). In practice, P6 will constrain this to the data date (Day 1), forcing B's scheduled finish to Day 8. Because logic demands it finish on Day 2, this creates -5 days of negative float right away. This is one of the clearest signals that SF relationships need to be calibrated very carefully, or reconsidered entirely!

Data Date Floor: *If the back-calculated ES(B) falls before the project data date, P6 uses the data date as the floor. This can mask the real logic issue. Always check the Driving Relationship column to see what's actually governing the activity.*

Backward Pass — Finding the Latest Dates

In the backward pass, SF constrains the predecessor's Late Start (not its finish).

The Formula

LS(A) = LF(B) – Lag + 1

LF(A) = LS(A) + Duration(A) – 1

Subtract the lag from B's Late Finish to get the latest A can start. A's Late Finish is then calculated forward from that Late Start. If A feeds multiple SF successors, P6 takes the minimum (most constraining) Late Start.

Example (continuing from above)

Project deadline Day 12. LF(B) = Day 12

LS(B) = LF(B) – Duration(B) + 1 = Day 12 − 8 + 1 = Day 5.

Activity	Late Start	Calculation Logic	Late Finish	Calculation Logic
B	Day 5	LF(B) – Duration(B) + 1 = 12 – 8 + 1 = Day 5	Day 12	Constrained by the project deadline
A (via SF)	Day 11	LF(B) – Lag + 1 = 12 – 2 + 1 = Day 11	Day 20	LS(A) + Duration(A) – 1 = 11 +10 – 1 = Day 20

Total Float and the Critical Path

TF = LS – ES (same as LF – EF)

From our example with deadline Day 12:

Activity	ES	EF	LS	LF	Float	Critical?
A	Day 1	Day 10	Day 11	Day 20	10 days	No
B	Day (−5)*	Day 2	Day 5	Day 12	10 days*	No

* Constrained to data date in practice. The large float values are a red flag. This SF configuration is poorly calibrated for the given durations and deadline. That's typical when an SF relationship is applied without careful planning.

Negative Lag (Leads)

Negative lag, where B could finish before A starts, is almost never valid and should be avoided entirely.

Now that we have established how standard FF relationships drive forward and backward-pass calculations, we must examine how our choice of relationship and lags directly impacts schedule mechanics.

SF and no Lag

In our scenario, we are constructing a new bypass traffic lane. We want to maintain the old lane operations and maintenance until the new bypass traffic lane opens. In Figure 3, we model this handoff with start traffic in the bypass lane, followed by the wrap-up of operations on the old lane.

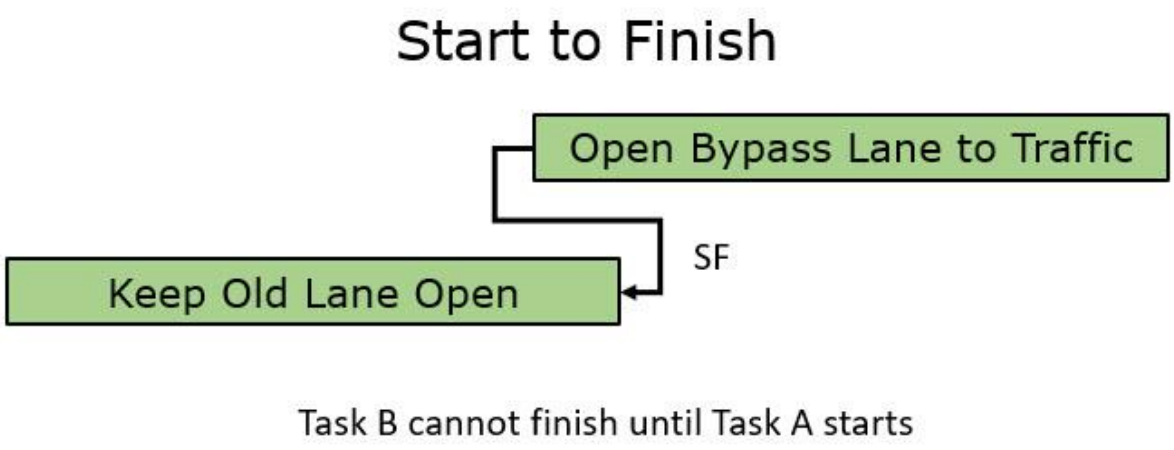


Figure 3

SF and Lag

An SF with a negative lag is problematic, but a positive lag is useful for the SF relationship. In Figure 4, we have our traffic system construction project, including a two-day lag.

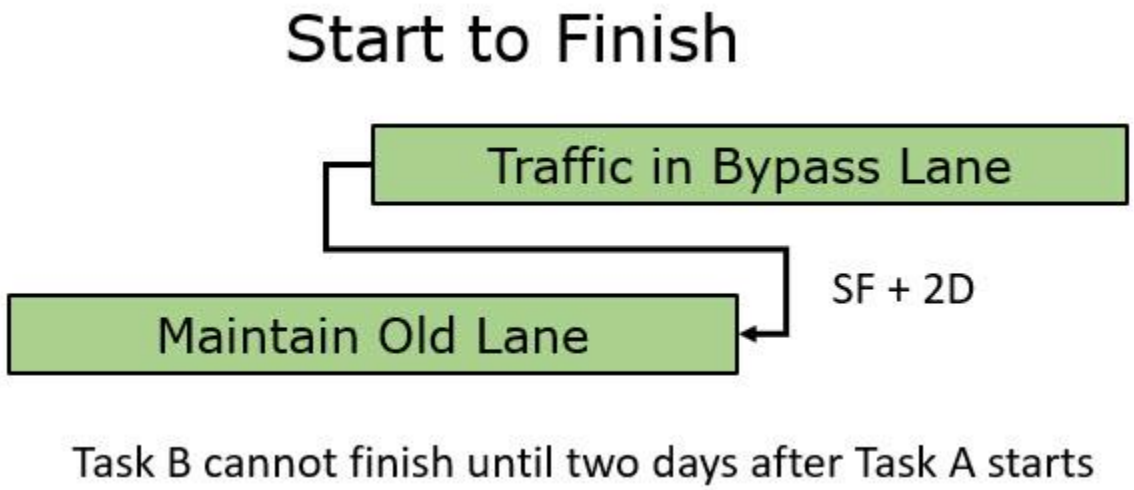


Figure 4

The two-day lag could mean it will take two days to ramp up and stabilize traffic on the new bypass lane (ensure the bypass handles the volume safely and effectively) before you can conclude the successor keep old lane open.

Start-to-Finish (SF) in P6

Figure 5 displays the SF relationship implemented in a P6 Professional schedule.

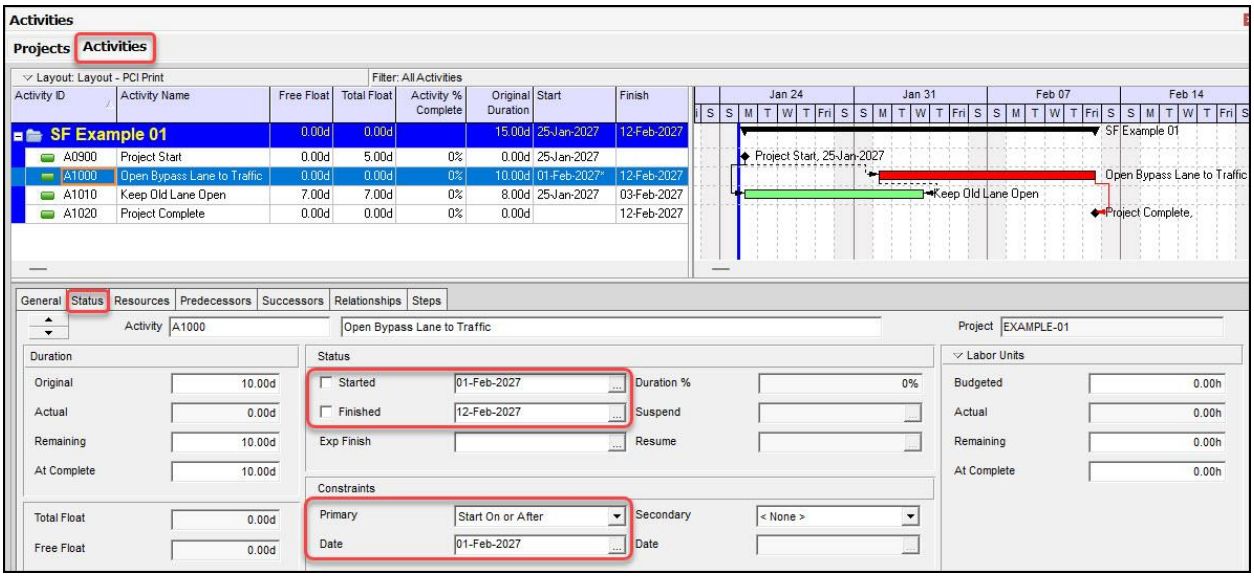


Figure 5

To model the Keep Old Lane Open task as an ongoing activity from day one, a Project Start Milestone was introduced. This milestone is mapped as a Start-to-Start (SS) predecessor to the Keep Old Lane Open task, anchoring its start date. The same Start Milestone is connected to the Open Bypass Lane to Traffic task.

Additionally, a Start On or After (SOOA) activity constraint is assigned to the Open Bypass Lane to Traffic task. This constraint manually enforces the desired delay from the project start, which in turn drives the Keep Old Lane Open task's finish date via the SF logic.

In Figure 6, we have progressed to the beginning of week three, a two-week update.

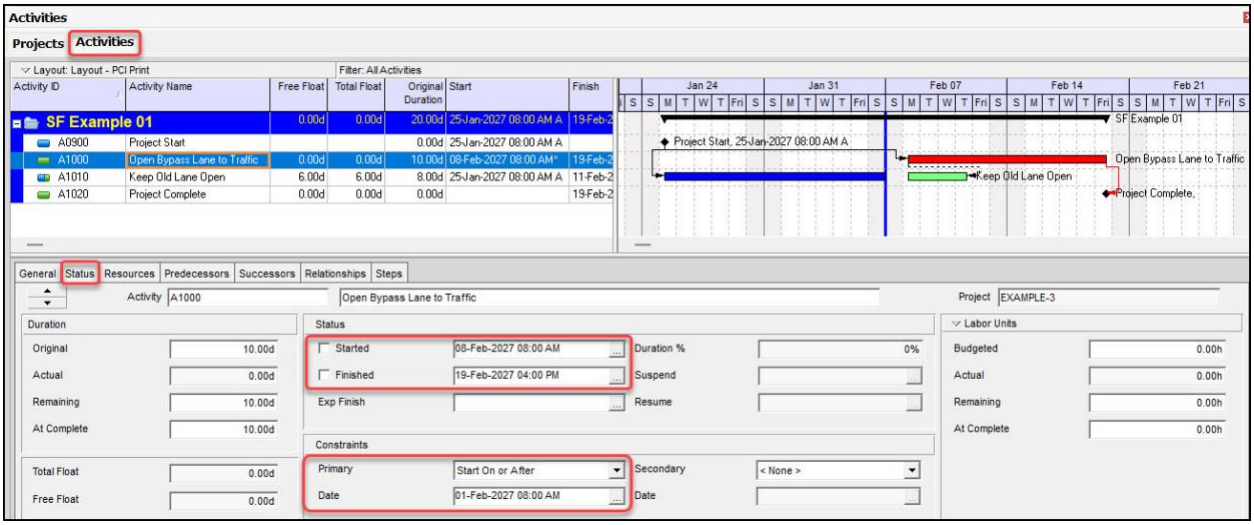


Figure 6

Observe in Figures 5 and 6 that the start of Open Bypass Lane Traffic delay from February 1 (SOOB constraint date) to February 8, and pushes the finish of Keep Old Lane Open from February 12 to February 19. This is the intended effect we want a delay in Open Bypass Lane Traffic to have on Keep Old Lane Open.

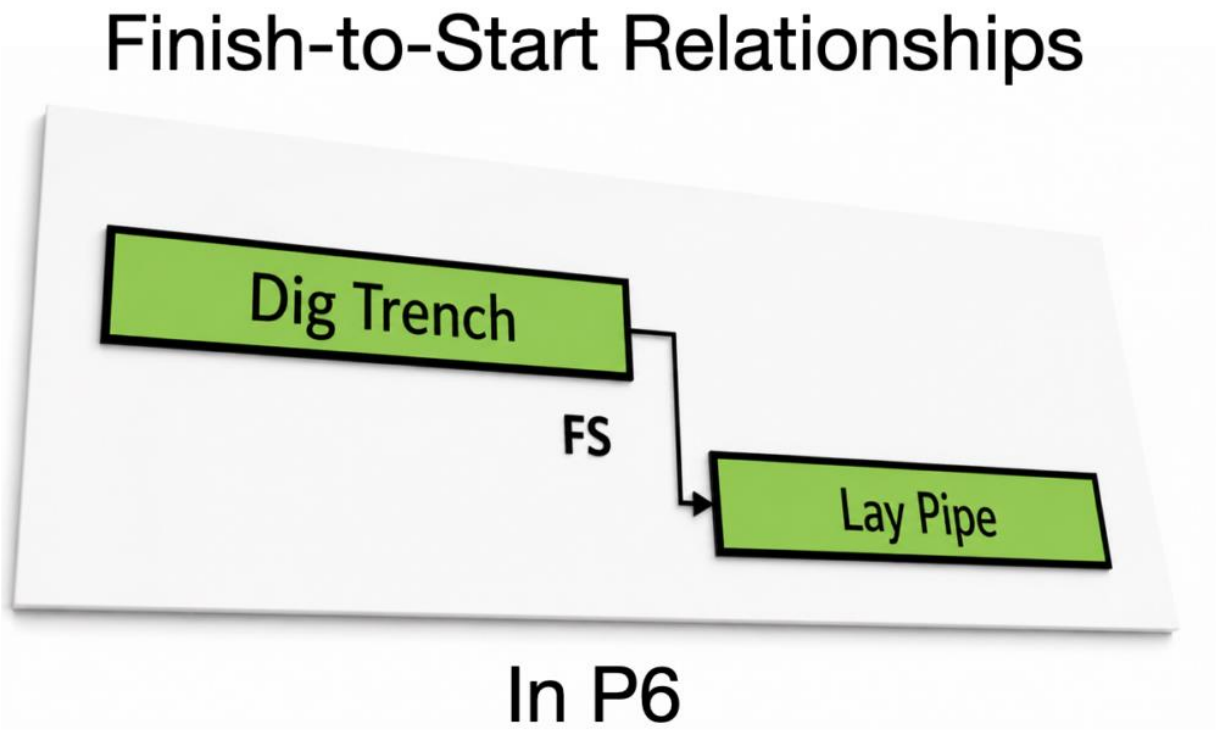
Summary

A Start-to-Finish (SF) relationship ties the finish of a successor task directly to the start of a predecessor. It prevents operational suspension by forcing a parallel work interval where the old system remains active while the new platform is initialized.

Schedulers can deliberately extend this concurrent runtime overlap by adding positive lag. This logic provides a fail-safe mechanism against critical service gaps, guaranteeing that any unexpected delay in launching the new system automatically prolongs the operation of the old one.

In P6 Professional, the predecessor task must be held in place with an activity constraint (such as SOOA) to prevent the scheduling engine from immediately pulling it to the data date and collapsing the duration of the ongoing successor.

Because SF utility is limited to these specialized handover scenarios, most schedulers rarely use it. Always evaluate whether standard FS or SS links can express the project logic more cleanly before implementing a Start-to-Finish (SF) relationship.



Project success relies on schedule transparency. Finish-to-Start relationships (FS) drive schedule clarity. Let's take a look.

Quality schedules have a much higher probability of success. Of particular importance in schedule quality assessments are the relationships between activities. Activity relationships make schedules dynamic, so that the effects of schedule changes propagate throughout the schedule. This means that activity start and finish dates automatically respond to and update to changes in the schedule or to the schedule's progression.

So, it's important for schedules to be responsive to input. But it is not enough for a schedule to be dynamic; it also must be understood by stakeholders. Much of the clarity of a schedule is determined by the types of relationships defined between activities. Relationship types are, therefore, an important measure of a schedule's quality.

FS relationships make schedules more transparent. For this reason, the Defense Contract Management Agency's (DCMA) 14-point assessment prefers the FS relationship for defining the interface between activities.

This article discusses the fundamentals of the P6 FS relationship in scheduling and demonstrates why it is preferable.

The Network Diagram

Schedulers map out their projects using a Network Diagram. This graph creates a clear visual representation of the schedule, showing the chronological flow of work and the dependencies between activities. We introduce the basic elements of this visual tool in our following Ten Six article:

> The Primavera P6 Professional Network Diagram

While P6 supports various ways to connect tasks on the network diagram, the FS relationship remains the most common choice.

Understanding the FS relationship

An FS relationship requires one task to finish before the next can begin. As the most intuitive dependency type, it forms the backbone of the Critical Path Method (CPM). In Figure 1, the FS relationship precedence diagram, this sequential logic ensures a clean flow in which Activity B follows the completion of Activity A.

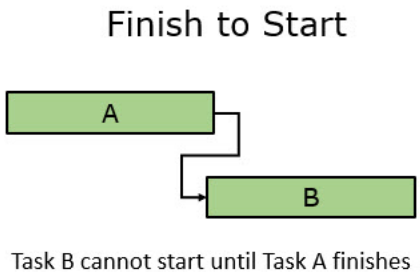


Figure 1

You can build on this relationship by adding lag to introduce a waiting period— for example, concrete curing time, a regulatory review window or a procurement lead time. Figure 2 shows the precedence diagram for an FS relationship with a lag.

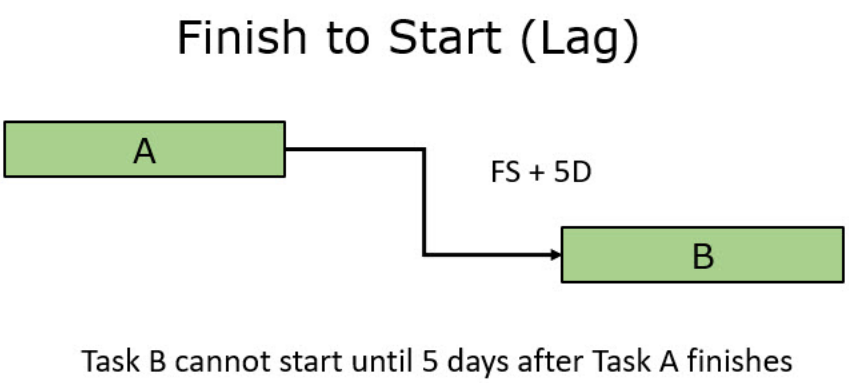


Figure 2

While Figure 1 shows a clean handoff, the lag in Figure 2 introduces a gap between tasks. This gap doesn't just affect how the schedule looks; it directly impacts the underlying calculations. To understand how the FS relationship and its lag influence start and finish dates, we need to examine how predecessor and successor dates are calculated during the forward and backward pass.

Forward Pass — Finding the Earliest Dates

The forward pass moves left-to-right through your network logic, calculating the earliest possible date each activity can start and finish. The math evaluates the path moving from the predecessor to the successor. When calculating these early dates using the working days in an assigned calendar, the math accounts for task durations, any added lag and the day-boundary step.

The Formulas (FS with Lag)
Schedulers apply three sequential steps to move from Predecessor A to Successor B across inclusive calendar days:

1. **Calculate Successor Start from Predecessor Finish**

ES(B) = EF(A) + Lag + 1 (Note: In a zero-lag environment, Lag = 0)

(Where +1 is the step between tasks, end-of-day A to next start-of-day B, and Lag is the additional waiting time.)

- **Calculate Successor Finish from Successor Start**

EF(B) = ES(B) + Duration(B) – 1

- **Verify the Successor's Duration Span**

Duration(B) = EF(B) – ES(B) + 1

Applying these sequential steps drives B's Early Start directly from A's Early Finish and ensures the mandatory time gap is strictly honored.

How P6 Counts Lag Days: *Lag is measured in working days on the predecessor's calendar by default in P6. A 5-day lag with a 5-day working week will span 7 calendar days if it straddles a two-day weekend. Check Admin Preferences to confirm whether P6 is using the predecessor or successor calendar for lag counting, especially when they're on different shift patterns.*

Example 1: Forward Pass with No Lag

Setup: Activity A (8 days, starts Day 1). Activity B (5 days, FS+0 dependency from A).

Activity	Duration	Early Start	Early Finish	Calculation Logic
A	8 days	Day 1	Day 8	Project start
B	5 days	Day 9	Day 13	ES(B) = EF(A) + 1 step = Day 9

The +1 Day Step: *When lag = 0, B starts the working day after A finishes. P6 handles this as an implicit +1 calendar step, and it's not a separate addition you need to make. When you add lag, P6 adds those days in addition to that step.*

Example 2: Forward Pass with Positive Lag

Setup: Same setup as Example 1, but with an FS+3 relationship (e.g., a 3-day review period).

Activity	Duration	Early Start	Early Finish	Calculation Logic
A	8 days	Day 1	Day 8	Project start
B	5 days	Day 12	Day 16	ES(B) = EF(A) + 3 working days + 1 step = Day 12

Example 3: Forward Pass with Multiple Predecessors:

Setup: Activity C (Duration irrelevant for ES) follows both Activity A (EF = Day 8, FS+0) and Activity B (EF = Day 6, FS+2). Primavera P6 evaluates all paths and enforces the maximum (latest) calculated date.

Predecessor	Relationship	Their EF	Resulting ES(C)	Governs?

A	FS+0	Day 8	Day 9	Yes (Tied)
B	FS+2	Day 6	Day 9	Yes (Tied)

Analysis: Because both paths result in Day 9, both Activity A and Activity B are currently driving the start of Activity C. If Activity A's finish slipped to Day 9, its path would result in Day 10, making it the sole driving predecessor.

Backward Pass — Finding the Latest Dates

The backward pass works in reverse from the project finish deadline to the project start. It computes the latest possible date each activity can start and finish without delaying the overall project completion date.

The Formulas (FS with Lag)

Schedulers apply three sequential steps to move backward from Successor B to Predecessor A across inclusive calendar days:

2. **Calculate Predecessor Finish from Successor Start**

LF(A) = LS(B) – Lag – 1 (Note: In a zero-lag environment, Lag = 0)

(Where -1 is the step between tasks from the start-of-day B back to the previous end-of-day A, and Lag is the additional waiting time.)

- **Calculate Predecessor Start from Predecessor Finish**

LS(A) = LF(A) – Duration (A) + 1

- **Verify the Predecessor's duration span**

Duration(A) = LF(A) – LS(A) + 1

Applying these sequential steps drives A's Late Finish directly from B's Late Start and ensures the mandatory time gap is strictly honored.

If an activity feeds multiple successors, Primavera P6 evaluates the backward path for each one and enforces the earliest, minimum (most constraining) Late Finish date. The tightest downstream deadline always governs.

Example 4: Backward Pass with No Lag:

Setup: Continuing from the previous FS+0 Example 1 scenario. The project's current finish is on Day 16. Activity B has a duration of 5 days, and Activity A has a duration of 8 days.

Activity	Late Start	Late Finish	Calculation Logic
B	Day 9	Day 13	Based on project completion
A(via FS+0)	Day 1	Day 8	LF(A) = LS(B) − Lag − 1 step = Day 11; LF(A) = 12 − 0 – 1 = Day 11

Example 5: Backward Pass with Positive Lag:

Setup: Continuing from the previous FS+3 Example 2 scenario. The project's current finish is on Day 16. Activity B has a duration of 5 days, and Activity A has a duration of 8 days.

Activity	Late Start	Late Finish	Calculation Logic
B	Day 12	Day 16	Based on project completion
A(via FS+3)	Day 1	Day 8	LF(A) = LS(B) − Lag − 1 step = Day 8; LF(A) = 12 − 3 – 1 = Day 8

The -1 Step In The Backward Pass: ***Just like the forward pass has an implicit +1 step, the backward pass has an implicit -1 step. LF(A) is the working day before LS(B) when lag = 0. If lag exists, it is also subtracted (LF(A) = LS(B) – 1 – Lag). Because the +1 and +Lag of the forward pass are perfectly countered by the -1 and -Lag of the backward pass, the mathematical loop closes seamlessly, and this is why critical activities show ES = LS and EF = LF.***

Example 6: Backward Pass with Multiple Predecessors:

Setup: Tracing the Example 3 scenario backward, Successor Activity C has a Late Start of Day 9. Primavera P6 evaluates the backward path from Activity C to determine the Late Finish for both Predecessor A (FS+0) and Predecessor B (FS+2). The tightest downstream deadline governs each path.

Successor	Relationship	Their LS	Resulting LF(A)	Governs?
A	FS+0	Day 9	Day 8	Yes (Tied)
B	FS+2	Day 9	Day 6	Yes (Tied)

Total Float and the Critical Path

Total Float (TF) is the amount of time an activity can be delayed without delaying the project finish date. It is calculated by comparing your forward and backward pass results:

TF = LS – ES (or LF – EF)

Zero float means the activity is on the critical path. In our FS+3 example, with a deadline on Day 16:

Activity	ES	EF	LS	LF	Float	Critical?
A	Day 1	Day 8	Day 1	Day 8	0	Yes
B	Day 12	Day 16	Day 12	Day 16	0	Yes

The chain exactly fills the project window; both activities are critical. Total float is one of the most useful measures in schedule analysis because it shows which activities can slip and which ones cannot.

Free Float

While Total Float looks at the project deadline, Free Float (FF) measures how much an activity can slip without delaying the Early Start of its immediate successor.

When accounting for lag, the mathematical formula is:

FF(A) = ES(B) – EF(A) – Lag – 1 (Note: In a zero-lag environment, Lag = 0)

In a tightly scheduled FS chain, Free Float on a task is typically zero because its successor begins the moment the mandatory step and lag time are honored. Identifying Free Float across parallel paths is highly beneficial for resource-leveling, as it isolates which activities can absorb minor delays without cascading to downstream tasks.

Now that we have established how standard FS relationships drive forward and backward pass calculations, we must examine how our choice of relationship types and lags directly impacts both schedule mechanics and network transparency.

Negative Lag (Leads)

Schedulers often model overlapping activities to compress construction timelines. A Finish-to-Start (FS) relationship with a negative lag—commonly referred to as a lead—allows a successor activity to begin before its predecessor finishes. In structural steel-frame construction, for example, a scheduler might determine that after four days of erecting steel frames, sufficient progress has been made to begin installing metal decking. Figure 3 illustrates the precedence diagram for these partially overlapping efforts.

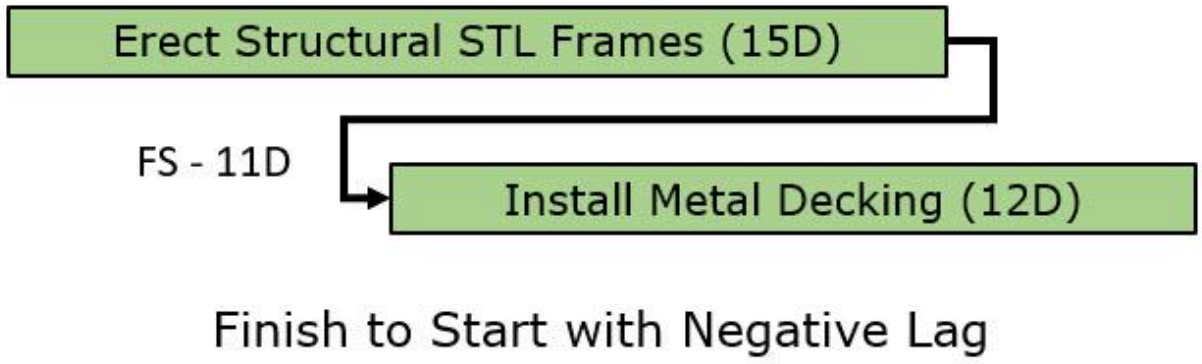


Figure 3

While utilizing negative lags (leads) appears to be an efficient acceleration method, project controls standards strongly discourage the practice for two primary reasons:

3. The logic is counterintuitive to understand within the network.
4. The DCMA 14-Point Assessment flags it as a quality deficiency.

Because negative lags are hard to decode and are often forbidden in government contracting, they should be avoided in favor of more transparent logic.

Positive Lag

Rather than “pulling” a successor backward with a negative lag, a more workable and easier-to-follow alternative is to join work efforts with a Start-to-Start (SS) relationship. In our structural steel and metal decking scenario, one could schedule a positive offset lag between the two activities, Figure 4.

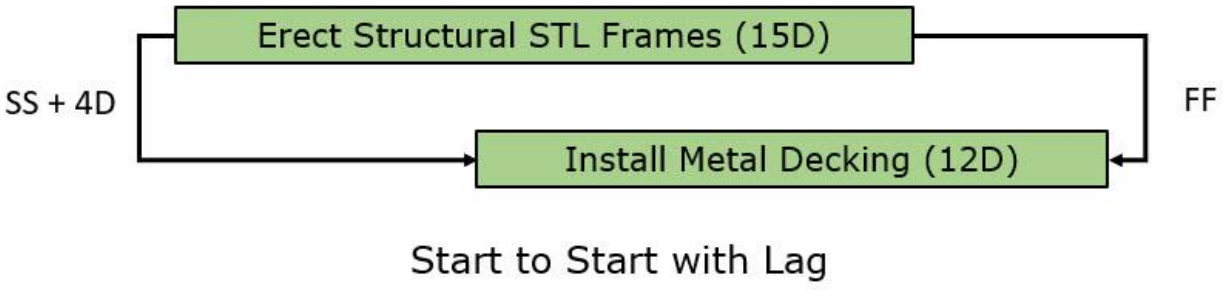


Figure 4

(Note the addition of the FF relationship to ensure the decking installation cannot finish before the structural steel frames are complete.) This approach allows the tasks to overlap while maintaining clear, forward-moving logic.

FS Logic in Lieu of SS and Positive Lag

But even positive lags are discouraged. The main reason is that, like the negative lag, they decrease the schedule transparency. A hallmark of a quality schedule is that it is readily apparent to all stakeholders. Schedules must be understood, and FS relationships drive schedule clarity. Replacing the four-day lag with explicit tasks defines the exact effort driving the interval.

In Figure 4, the lag delay is the time required to first erect Bay 1, which serves as the building’s structural anchor. During this period, workers must plumb, square and guy this bay to ensure precision, then install permanent cross-bracing to prevent ‘racking’. Because Bay 1 is distinct in both effort and structural significance, it warrants a standalone task.

Replacing the vague SS relationship and positive lag with explicit tasks for ‘Erect Bay 1’ and ‘Erect Bays 2 to 5’ eliminates ambiguity. Connecting these with Finish-to-Start (FS) relationships transforms a hidden delay into a transparent, logic-driven path that accurately reflects field operations. Figure 5 displays the building construction schedule, isolating the erection of Bay 1 (the anchor bay) as a distinct task.

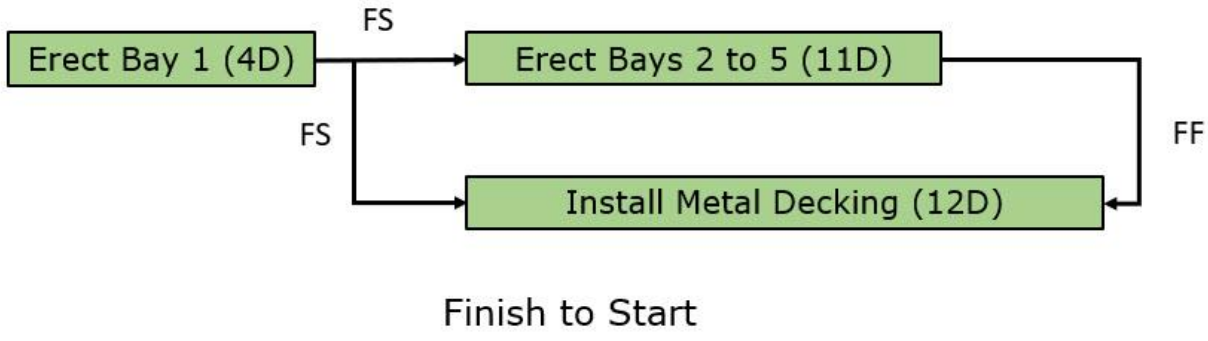


Figure 5

We can clearly see that this anchor bay is now the driver for both the remaining steel erection and the metal decking installation.

Summary

The early dates calculated during the forward pass are inclusive, factoring in both the lag duration and the one-day step required to move to the next workday. Conversely, the late dates calculated during the backward pass are inclusive and subtract the lag and day step to revert to the prior workday.

These passes establish the early and late dates for each task, governed strictly by the defined logical relationships.

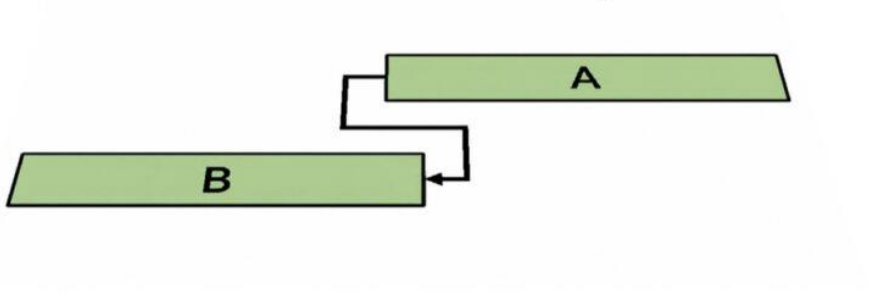
While a Finish-to-Start (FS) relationship with a negative lag is efficient to model overlapping efforts, its logic can be confusing to stakeholders.

Consequently, the DCMA 14-Point Assessment strictly forbids negative lag. An SS relationship with a positive lag is more intuitive and acceptable, though scheduling successors based on a fully completed, verified scope remains ideal.

In addition, lag appears as an unlabeled line on the Gantt chart, obscuring the work being done. Replacing a positive lag with a dedicated, named task increases transparency for stakeholders. FS relationships are best suited to connecting these discrete tasks into the schedule logic.

This transparency is the primary reason the DCMA 14-Point Assessment states that at least 90% of the relationships in a schedule should be FS.

Start-to-Finish (SF) Relationships



In P6

Start-to-Finish (SF) relationships say one task cannot conclude until the former task starts. Its application is limited, but it may fit well in the right scenario. Let's take a look.

The SF relationship specifies that a successor task cannot finish until its predecessor begins. It is difficult to find a scenario where the SF relationship does not violate physical laws and is useful.

This rare relationship type, nevertheless, finds its place in the general scenario of maintaining an old system until the new one is in place. A good example comes from the construction of a traffic bypass lane. Workers must maintain the old lane until the bypass traffic lane opens (or starts). The key advantages of the SF relationship when properly applied:

- **Guarantees Zero Downtime**: It ensures a critical system (like traffic flow) never experiences a gap during handoff.
- **Protects Fail-Safes**: The old system remains fully operational as a backup until the new system is proven ready.
- **Automates Resource Retention**: If the start of the new system is delayed, the old system automatically extends, preventing premature demobilization.

This article outlines the fundamentals of the Primavera P6 Start-to-Finish (SF) relationship and its usefulness for modeling the continued operation of an old system until a new one begins.

The Network Diagram

Schedulers map out their projects using a Network Diagram. This graph creates a clear visual representation of the schedule, showing the chronological flow of work and the dependencies between activities. We introduce the basic elements of this visual tool in our following Ten Six article:

The Network Diagram

While Finish-to-Finish (FF) coordinates a joint ending, only SF guarantees a seamless handover by linking the old system's retirement directly to the new one's opening.

Understanding the SF relationship?

An SF relationship means the successor (Task B) can't finish until the predecessor (Task A) starts. This allows a brief window where both tasks operate in parallel.

Essentially, the predecessor's start triggers the release for the successor to complete the task. As shown in Figure 1, the dependent successor task is held up until the predecessor begins.

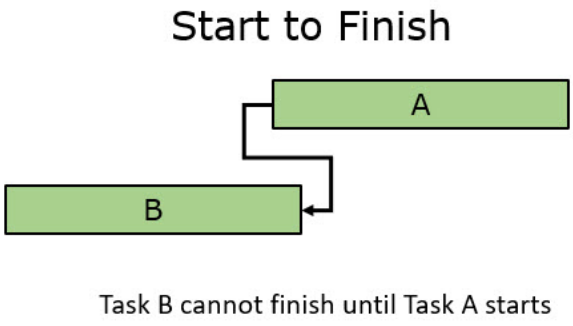


Figure 1

Adding a lag slightly modifies this dynamic. For example, an SF+2 lag means Task B cannot finish until at least two working days after Task A starts, as illustrated in Figure 2.

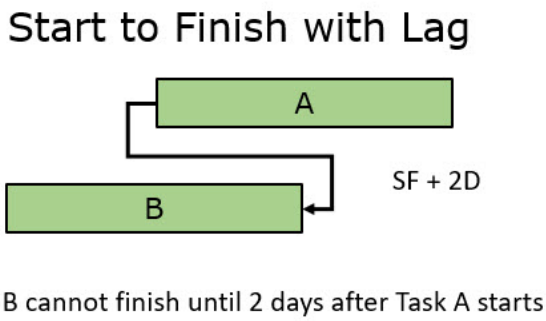


Figure 2

Some of the overarching applications of the SF relationship include.

- Just-in-time handovers: A night-shift team (B) cannot stand down until the day-shift team (A) arrives and starts. Task B's finish is gated by Task A's start.
- Phased resource transitions: An old system (B) must keep running until the replacement system (A) comes online. Task B cannot be decommissioned until Task A begins.

⚠ **SF is rarely the right answer:** *Before using SF, ask whether a restructured FS or SS relationship would make the same logic clearer. SF can obscure the driving constraint, making schedule analysis significantly harder. If you do use it, document the business reason, as auditors will ask.*

Calculating SF Relationships

Important Note on Paper vs. Software
On paper, we use +1 and -1 steps to jump across calendar day boundaries. When you type these relationships into P6, the software handles these adjustments automatically in the background.

Forward Pass – Finding the Earliest Dates

In the SF relationship, the predecessor's start date drives the successor's finish date — not its start date. Because the successor must finish the workday before the predecessor can start, we must subtract 1 day on paper to find the correct finish boundary.

The Formula

EF(B) = ES(A) + Lag – 1

ES(B) = EF(B) – Duration(B) + 1

P6 constrains B's Early Finish first, then back-calculates B's Early Start from there. This is the opposite of how you'd normally think about an activity's dates.

Calendars and Lag: *SF lag defaults to the predecessor's calendar in P6. Verify in Tools | Schedule > Options, particularly when predecessor and successor are on different shift calendars.*

Example:

Activity A: 10-day duration, starts Day 1 (EF = Day 10).

Activity B: 8-day duration, SF+2 from A.

Activity	Early Start	Calculation Logic	Early Finish	Calculation Logic
A	Day 1	Project start	Day 10	ES(A) + Duration(A) – 1 = 1 + 10 -1 = Day 10
B	Day(−5)*	EF(B) – Duration(B) + 1 = 2 − 8 + 1 = Day (−5)	Day 2	ES(A) + Lag – 1 = 1 + 2 – 1 = Day 2

* Yes, Day (-5). In practice, P6 will constrain this to the data date (Day 1), forcing B's scheduled finish to Day 8. Because logic demands it finish on Day 2, this creates -5 days of negative float right away. This is one of the clearest signals that SF relationships need to be calibrated very carefully, or reconsidered entirely!

Data Date Floor: *If the back-calculated ES(B) falls before the project data date, P6 uses the data date as the floor. This can mask the real logic issue. Always check the Driving Relationship column to see what's actually governing the activity.*

Backward Pass — Finding the Latest Dates

In the backward pass, SF constrains the predecessor's Late Start (not its finish).

The Formula

LS(A) = LF(B) – Lag + 1

LF(A) = LS(A) + Duration(A) – 1

Subtract the lag from B's Late Finish to get the latest A can start. A's Late Finish is then calculated forward from that Late Start. If A feeds multiple SF successors, P6 takes the minimum (most constraining) Late Start.

Example (continuing from above)

Project deadline Day 12. LF(B) = Day 12

LS(B) = LF(B) – Duration(B) + 1 = Day 12 − 8 + 1 = Day 5.

Activity	Late Start	Calculation Logic	Late Finish	Calculation Logic
B	Day 5	LF(B) – Duration(B) + 1 = 12 – 8 + 1 = Day 5	Day 12	Constrained by the project deadline
A (via SF)	Day 11	LF(B) – Lag + 1 = 12 – 2 + 1 = Day 11	Day 20	LS(A) + Duration(A) – 1 = 11 +10 – 1 = Day 20

Total Float and the Critical Path

TF = LS – ES (same as LF – EF)

From our example with deadline Day 12:

Activity	ES	EF	LS	LF	Float	Critical?
A	Day 1	Day 10	Day 11	Day 20	10 days	No
B	Day (−5)*	Day 2	Day 5	Day 12	10 days*	No

* Constrained to data date in practice. The large float values are a red flag. This SF configuration is poorly calibrated for the given durations and deadline. That's typical when an SF relationship is applied without careful planning.

Negative Lag (Leads)

Negative lag, where B could finish before A starts, is almost never valid and should be avoided entirely.

Now that we have established how standard FF relationships drive forward and backward-pass calculations, we must examine how our choice of relationship and lags directly impacts schedule mechanics.

SF and no Lag

In our scenario, we are constructing a new bypass traffic lane. We want to maintain the old lane operations and maintenance until the new bypass traffic lane opens. In Figure 3, we model this handoff with start traffic in the bypass lane, followed by the wrap-up of operations on the old lane.

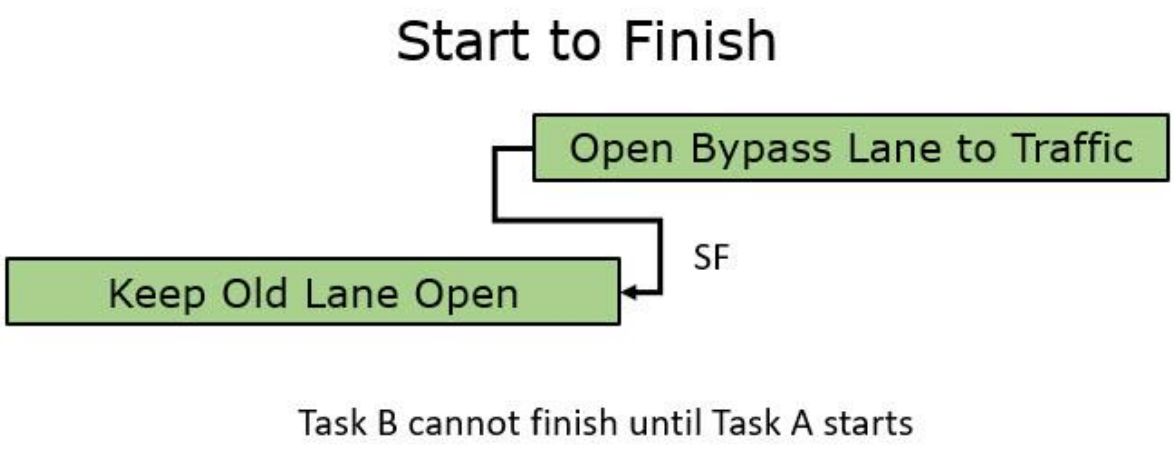


Figure 3

SF and Lag

An SF with a negative lag is problematic, but a positive lag is useful for the SF relationship. In Figure 4, we have our traffic system construction project, including a two-day lag.

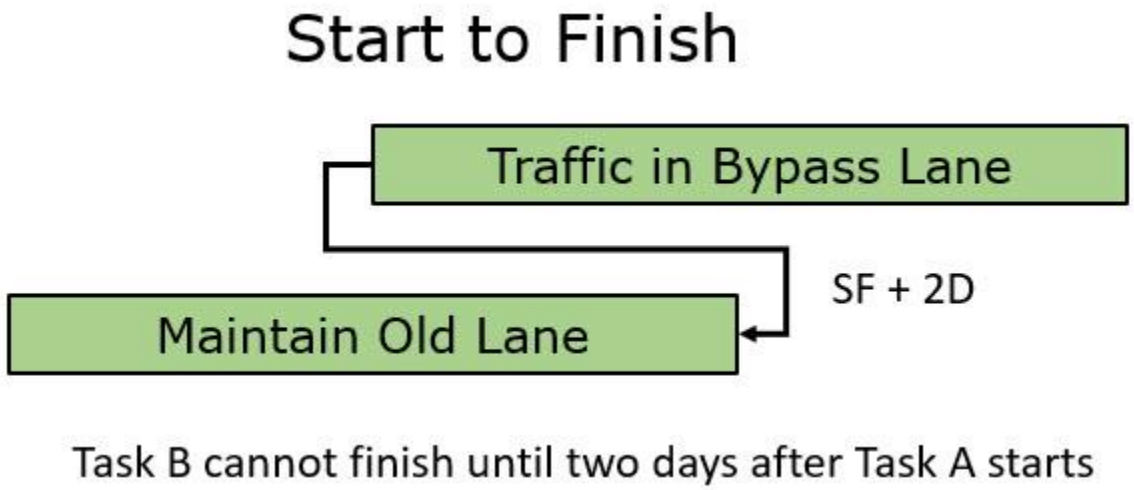


Figure 4

The two-day lag could mean it will take two days to ramp up and stabilize traffic on the new bypass lane (ensure the bypass handles the volume safely and effectively) before you can conclude the successor keep old lane open.

Start-to-Finish (SF) in P6

Figure 5 displays the SF relationship implemented in a P6 Professional schedule.

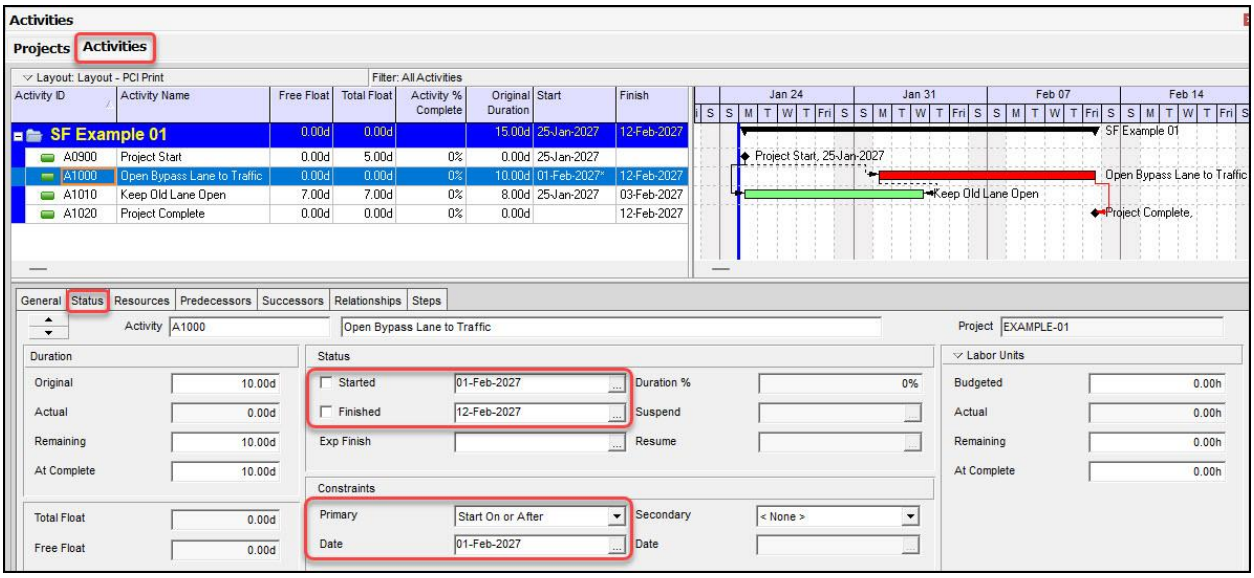


Figure 5

To model the Keep Old Lane Open task as an ongoing activity from day one, a Project Start Milestone was introduced. This milestone is mapped as a Start-to-Start (SS) predecessor to the Keep Old Lane Open task, anchoring its start date. The same Start Milestone is connected to the Open Bypass Lane to Traffic task.

Additionally, a Start On or After (SOOA) activity constraint is assigned to the Open Bypass Lane to Traffic task. This constraint manually enforces the desired delay from the project start, which in turn drives the Keep Old Lane Open task's finish date via the SF logic.

In Figure 6, we have progressed to the beginning of week three, a two-week update.

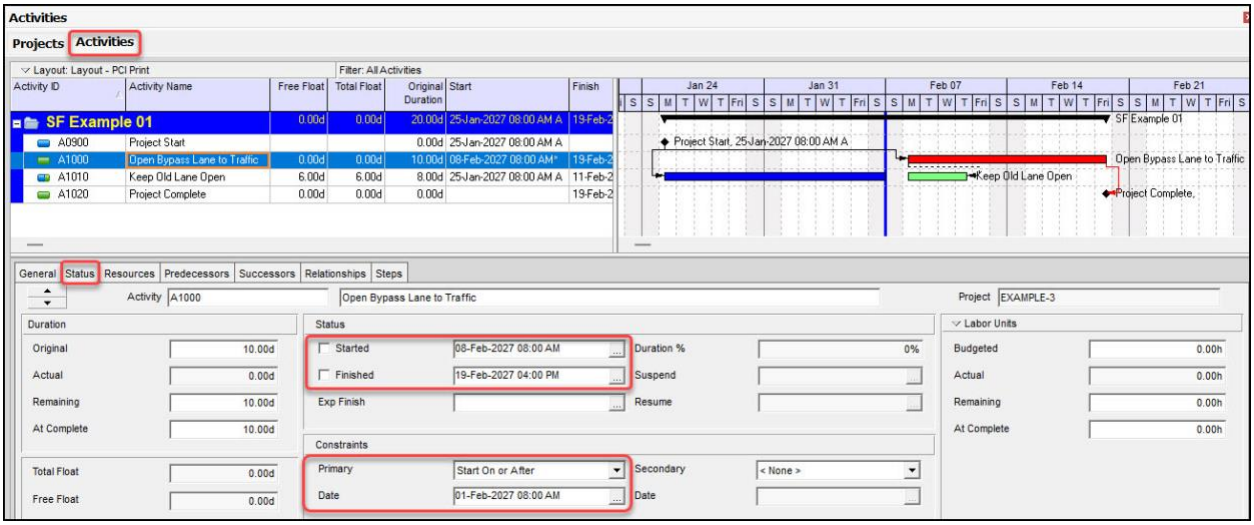


Figure 6

Observe in Figures 5 and 6 that the start of Open Bypass Lane Traffic delay from February 1 (SOOB constraint date) to February 8, and pushes the finish of Keep Old Lane Open from February 12 to February 19. This is the intended effect we want a delay in Open Bypass Lane Traffic to have on Keep Old Lane Open.

Summary

A Start-to-Finish (SF) relationship ties the finish of a successor task directly to the start of a predecessor. It prevents operational suspension by forcing a parallel work interval where the old system remains active while the new platform is initialized.

Schedulers can deliberately extend this concurrent runtime overlap by adding positive lag. This logic provides a fail-safe mechanism against critical service gaps, guaranteeing that any unexpected delay in launching the new system automatically prolongs the operation of the old one.

In P6 Professional, the predecessor task must be held in place with an activity constraint (such as SOOA) to prevent the scheduling engine from immediately pulling it to the data date and collapsing the duration of the ongoing successor.

Because SF utility is limited to these specialized handover scenarios, most schedulers rarely use it. Always evaluate whether standard FS or SS links can express the project logic more cleanly before implementing a Start-to-Finish (SF) relationship.