

Section 35

Subroutine Event

Topics Covered in this Section:

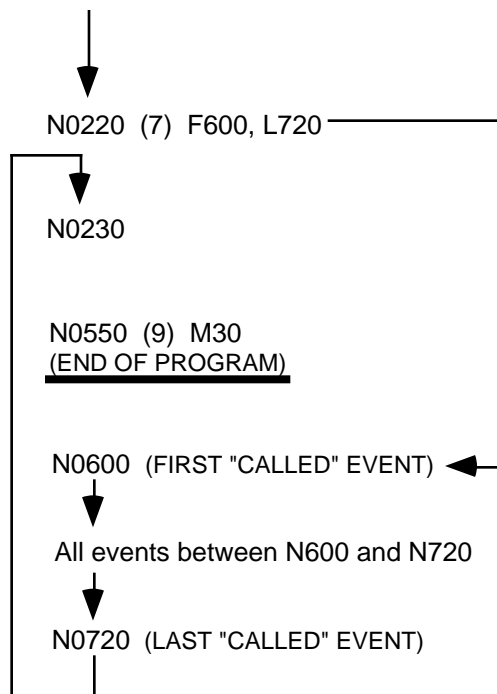
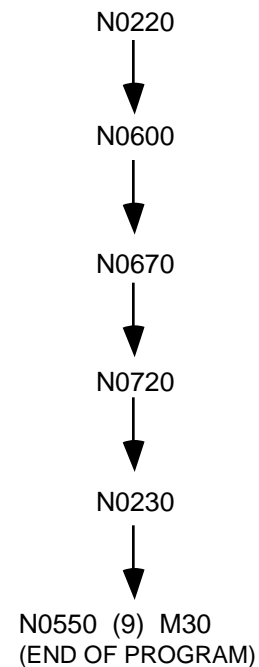
- Introduction
- Subroutine Event Data
- Programming the Subroutine Event
- Illustrations and Program Examples

F - FIRST EVENT

F is the N number of the first event that will execute. This event may be located anywhere within the part program, even after an M02 or M30 - End of Program.

L - LAST EVENT

L is the sequence number of the last event that will execute. If L is greater than F, the events starting with F and continuing through L will execute. The L sequence number may not be smaller than F. If L equals F, only the single "called" event will execute.

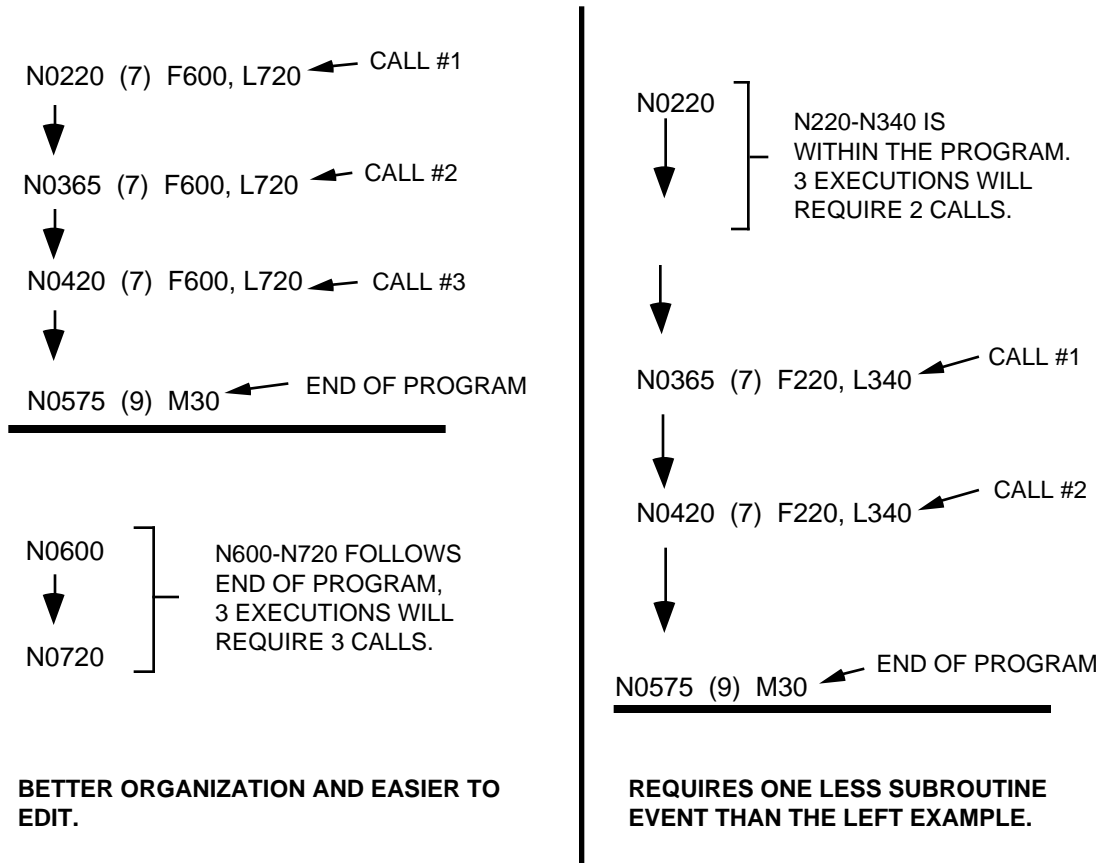
SUBROUTINE CALLS THE SERIES OF EVENTS FROM N600 - N720**ORDER OF EXECUTION****PROGRAMMING THE SUBROUTINE EVENT**

The program sequence numbers designated by the Subroutine event's F and L entries must identify "stored" events. A series of events called by the Subroutine event may precede or follow the Subroutine event. A run time error will occur if the Subroutine event's N number does not fall within the programmed F - L range.

In typical part program arrangement, the series of events called by one or more Subroutine events appears after the M02 or M30 - End of Program command (**following page, left**). Since the "called" series of events are outside of the part program's range, they will be performed only when "called", and not as a result of the normal sequential execution of the program.

In another type of arrangement, the "called" series of events is located at the first logical point of execution in the part program (**following page, right**). Subsequent repetitions of the "series" are called by later Subroutine events. In comparison, one less Subroutine event is required by this approach at the possible expense of some part program clarity.

TWO STRUCTURAL APPROACHES FOR SUBROUTINES

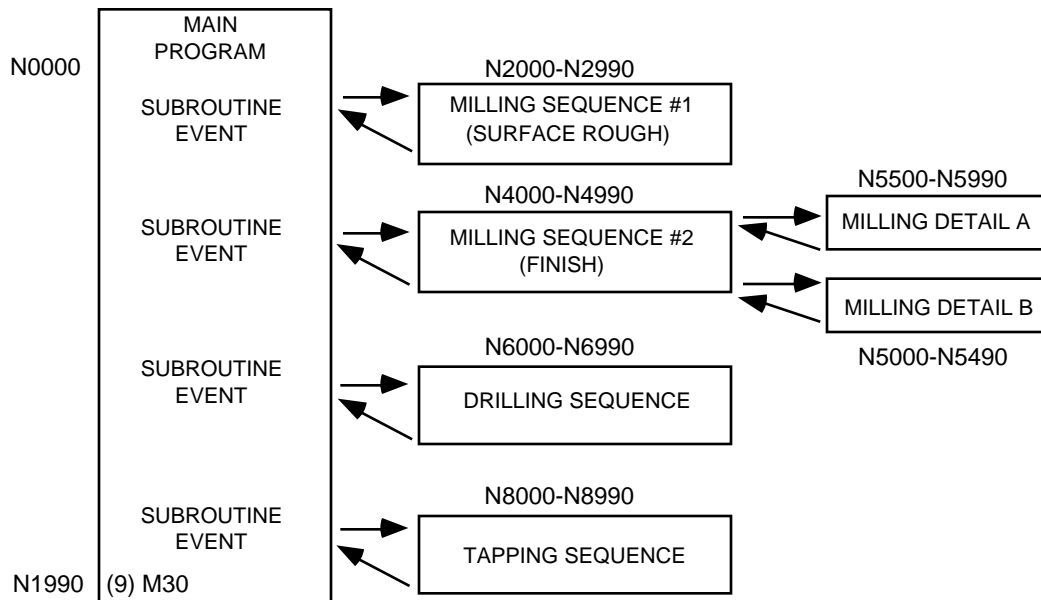


ILLUSTRATIONS and PROGRAM EXAMPLES

A Subroutine event can considerably reduce programming effort and memory storage for repetitive sequences by initially programming the sequence end then addressing it as often as required via the Subroutine event. Often the called routine will be constructed as a series of incremental feeds fixed to a relocatable starting point.

Subroutine events can also permit modular construction of a part program. This approach groups machining tasks into a series of modules which may be individually programmed and tested. The modules are later called in logical order by a series of Subroutine events within a main “calling” program. Modular construction adds clarity and organization to the part program while avoiding the tedium of moving and rearranging program data. Refer to the **following** illustration.

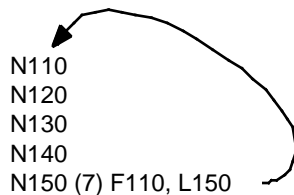
SIMPLIFIED LAYOUT FOR A SERIES OF OPERATIONS



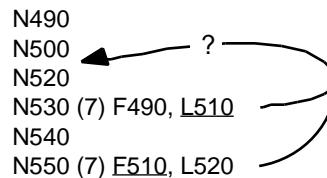
SUBROUTINE ERRORS

Due to its nature, the Subroutine event is susceptible to several types of programming errors. The most common errors are illustrated **below** and will result in run time errors.

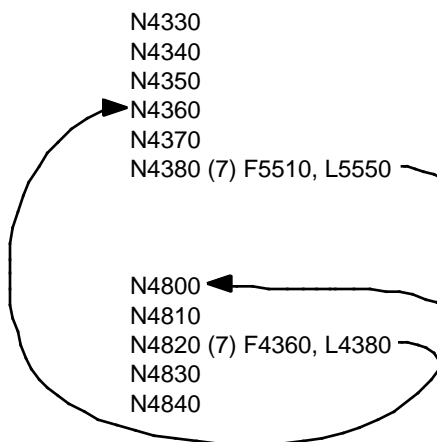
RECURSIVE CALL



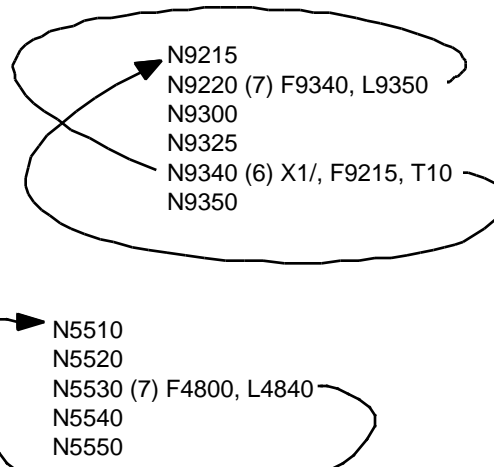
FIRST OR LAST NOT PROGRAMMED



INFINITE LOOP



INFINITE LOOP



NESTING

A series of events called by a Subroutine event may contain another Subroutine event. A Subroutine event contained in a called series of events is said to be “nested”. When executed, the nested Subroutine event will delay execution to call a second series of events. The control can follow and retrace its path through as many as four calling Subroutine events, nested as shown **below**.



SUBROUTINE and REPEAT EVENTS

Subroutine and Repeat events may be programmed in various combinations to perform extremely complex machining sequences. A Subroutine event may call one or more Repeat events, or a Repeat event may include one or more Subroutine events in its range of events. Nesting may continue as long as each event type does not exceed its nesting limit, and the construction avoids infinite loops.

As a practical consideration, the use of a Subroutine event is often preferred to the Repeat event. While it is simpler to repeat a single event 99 times than enter 99 Subroutine events to achieve the same effect, the Subroutine event is often able to avoid many of the Repeat event's restrictions.

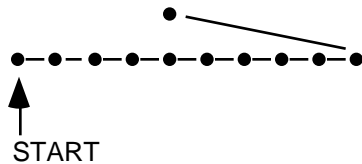
A Repeat event usually relies on preparatory activity such as an incremental step, rotation or scaling between each repeat sequence. Each preparatory activity, like the milling or drilling sequence that follows it, must mimic the last, and the series of repetitions must occur without interruption.

Some parameters may not allow multiple repetitions within the Repeat event's structure. Examples include programmed part offsets, X and Y symmetry, travel limits changes, inch/metric changes and nonincremental changes in scale or angle of rotation. The Subroutine event can save considerable programming effort and program storage by recalling a machining sequence after one of these types of preparatory activities.

EXAMPLES

The following part program segment demonstrates an interruption of a series with repetitions. One way to program the hole pattern would be to treat the 10 hole linear array as a drill cycle with nine repeats, then position and drill the remaining hole, this example is shown, **lower left**.

REPEAT

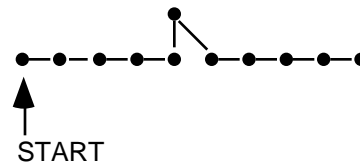


```
N 10 (0) X0/,G1,W.1,F5
N 20 (6) X5/,F10,T9
N 30 (0) X-25/,Y5/
```

ALTHOUGH PART PROGRAM IS SHORTER, EXECUTION TIME IS LONGER

**60.61" XY TOOL MOTION
95.7 SECONDS EXECUTION**

REPEAT AND SUBROUTINE



```
N 10 (0) X0/,G1,W.1,F5
N 20 (6) X5/,F10,T4
N 30 (0) Y5/
N 40 (0) X5/,Y-5/,G0
N 50 (7) F10,L20
```

NOW PART PROGRAM IS LONGER, BUT EXECUTION TIME IS SHORTENED

**52.07 XY TOOL MOTION
91.7 SECONDS EXECUTION**

The **upper right** example treats the first five holes as a drill and repeat, drills the odd hole, then positions and calls the earlier drill and repeat sequence using a Subroutine event. The two additional events resulted in a negligible increase in program storage while eliminating 8.54 inches of XY tool travel and 4 seconds of cycling time.¹

¹ Notice that 5 holes are drilled when the Subroutine event at N50 calls events N10 and N20. Only 4 holes will

WHEEL STRUT - EXAMPLE

The following part program segment is used to mill the finish profile for a wheel's strut. The left recess is profiled with a series of linear and arc milling motions. The milling direction is counterclockwise with cutter radius compensation to the left. The plunge occurs midway along the linear path. Events N40 - N140 mill the left profile then return the cutter to the part's centerline (X0).

The right recess is then milled by enabling X axis symmetry, and then recalling the events N10 - N140 with a Subroutine event. Notice that cutter compensation is turned on and off within the range of events that is called by the Subroutine event. Also notice that X axis symmetry is turned off in the block (N180) the precedes the M30 End of Program command. Refer to the illustration on the **following page**.

WHEEL STRUT PROGRAM

N00	(9)	M03 T1 H1 D1 S1000	- tool 1, turn on spindle
N10	(0)	X-.375 Y.839 Z1	- position to start point
N20	(1)	Y.939 F20 C0	- dummy move to prepare cutter comp
N30	(1)	Y1.039 C1	- comp on left. Tool is still above work ²
N40	(1)	Z-1.625	- feed to Z depth
N50	(1)	Y1.4416	- begin profile with comp on left
N60	(2)	X-.9241 Y1.7742 I-.75 J1.4416 D1	-
N70	(2)	X-1.9610 Y-.3928 I0 J0 D1	-
N80	(2)	X-1.2393 Y-.4427 I-1.5933 J-.3192 D1	-
N90	(2)	X-.5833 Y.3007 I0 J-.875 D0	-
N100	(2)	X-.375 Y.6367 I-.75 J.6367 D1	-
N110	(1)	Y1.045	- end profile
N120	(1)	Z1	- retract in Z
N130	(1)	Y1.050	- continue linear move
N140	(1)	Y1.055 C0	- turn cutter compensation off ²
N150	(0)	X0	- position to X0
N160	(S)	I1	- turn on X symmetry
N170	(7)	F10 L140	- Subroutine N10-N140
N180	(S)	I0	- turn X symmetry off
N190	(9)	M30	- end of program

2. The Z infeed and retract moves do not cause the control to lose "look ahead" and gouge the part because they occur in the middle of linear moves. The infeed sequence begins at Y.939. N30 then makes a short Y axis move and turns cutter compensation on while the tool is above the work. N40 now plunges the tool to a depth of Z -1.625". To ensure that the tool does not gouge the part, the following move (N50) must continue to feed

