

```
1 // Handle Transition for up to 63-step sequence
2 // Copyright (c) 2011 Dogwood Valley Press, LLC
3
```

```
4 FUNCTION_BLOCK Seq63_Trans // FB91
5 VAR
6   Step_Ctr: CTUD;           // Handles transition to next step
7 END_VAR
8
9 VAR_IN_OUT
10  Sequence: Seq63_Type;
11 END_VAR
12
13 // If counter accumulator is not the same as the step number, step has changed
14 // Reset current step, reconcile step number to accumulator, and then set next step
15 // If next step is zero (counter reset), do not set step-in-progress bit
16 IF Sequence.Step_Num <> Sequence.Ctr_Acc THEN
17   Sequence.Step[Sequence.Step_Num] := FALSE;
18   Sequence.Step_Num := Sequence.Ctr_Acc;
19   IF Sequence.Step_Num <> 0 THEN
20     Sequence.Step[Sequence.Step_Num] := TRUE;
21   END_IF;
22 END_IF;
23
24 // Increment to next step if transition bit is set.
25 // Can also jump to another step.
26 Step_Ctr( CU:=Sequence.Trans[Sequence.Step_Num],
27          R:=Sequence.Ctr_Reset,
28          LOAD:=Sequence.Do_Jump,
29          PV:=Sequence.Jump_Dest);
30 Sequence.Ctr_Acc := Step_Ctr.CV;
31 Sequence.Ctr_Reset := FALSE;
32 Sequence.Do_Jump := FALSE;
33
34 END_FUNCTION_BLOCK
35
36 // Handle Transition for up to 127-step sequence
37
```

```
38 FUNCTION_BLOCK Seq127_Trans // FB92
39 VAR
40     Step_Ctr: CTUD;           // Handles transition to next step
41 END_VAR
42
43 VAR_IN_OUT
44     Sequence: Seq127_Type;
45 END_VAR
46
47 // If counter accumulator is not the same as the step number, step has changed
48 // Reset current step, reconcile step number to accumulator, and then set next step
49 // If next step is zero (counter reset), do not set step-in-progress bit
50 IF Sequence.Step_Num <> Sequence.Ctr_Acc THEN
51     Sequence.Step[Sequence.Step_Num] := FALSE;
52     Sequence.Step_Num := Sequence.Ctr_Acc;
53     IF Sequence.Step_Num <> 0 THEN
54         Sequence.Step[Sequence.Step_Num] := TRUE;
55     END_IF;
56 END_IF;
57
58 // Increment to next step if transition bit is set.
59 // Can also jump to another step.
60 Step_Ctr( CU:=Sequence.Trans[Sequence.Step_Num],
61          R:=Sequence.Ctr_Reset,
62          LOAD:=Sequence.Do_Jump,
63          PV:=Sequence.Jump_Dest);
64 Sequence.Ctr_Acc := Step_Ctr.CV;
65 Sequence.Ctr_Reset := FALSE;
66 Sequence.Do_Jump := FALSE;
67
68 END_FUNCTION_BLOCK
69
```