

19. Connectivity Fault Management Protocol(s)

Figure 19-1

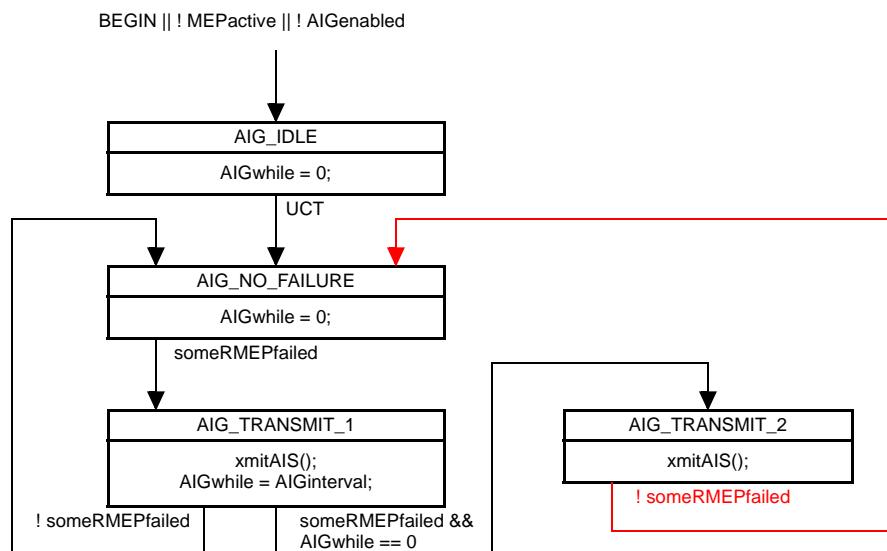
**Figure 19-1—EFF Alarm Indication Generator State Machine REVISED**

Figure 19-2

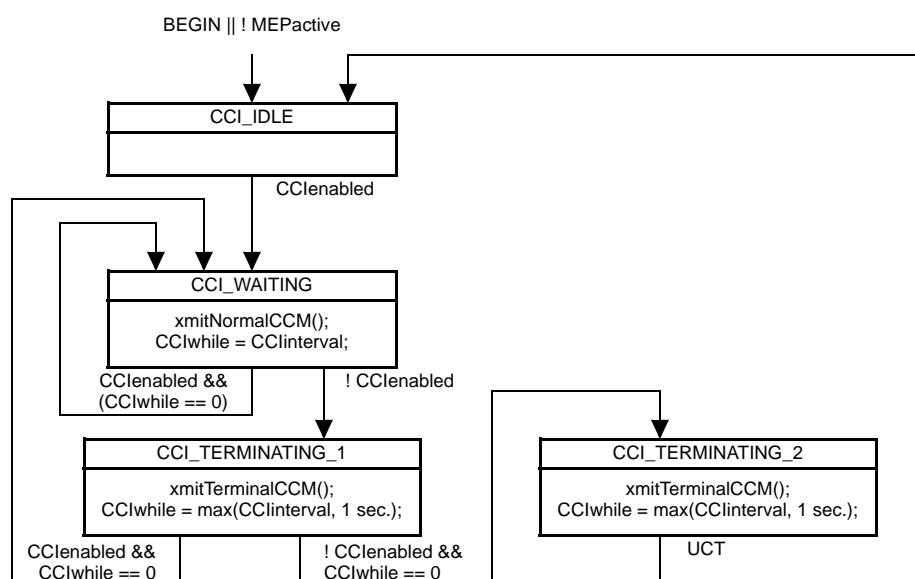
**Figure 19-2—IFF Continuity Check Initiator State Machine**

Figure 19-3

Figure 19-4

Figure 19-5

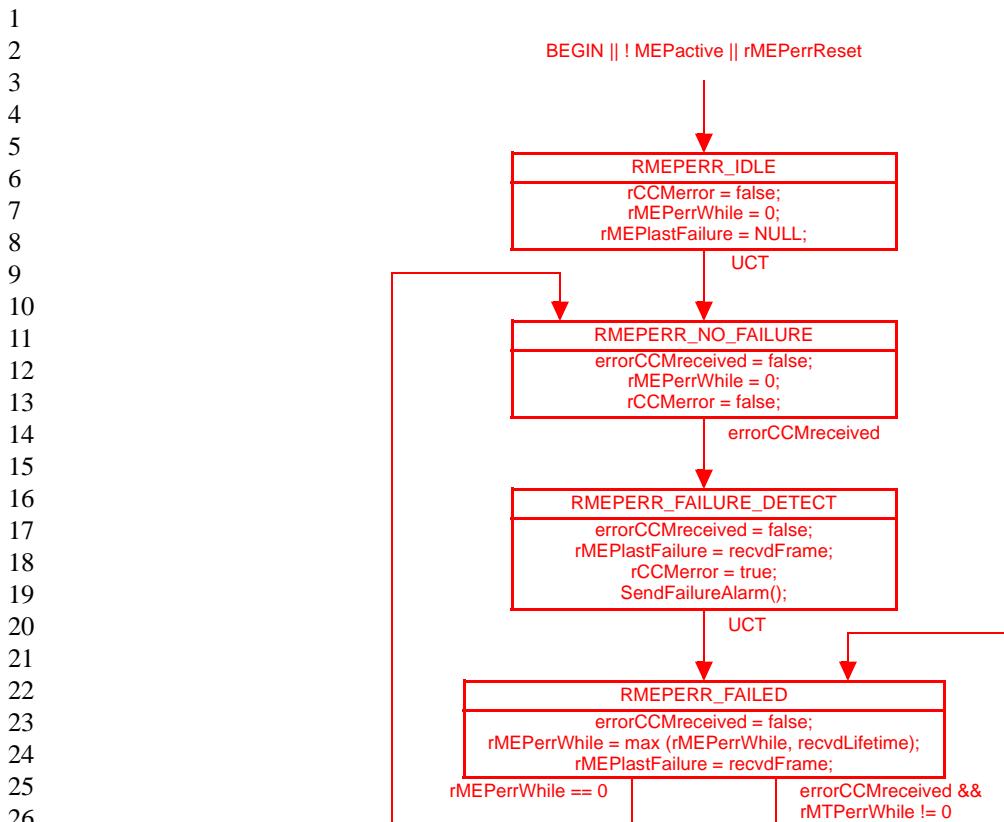


Figure 19-3—IFF Remote MEP Error State Machine NEW

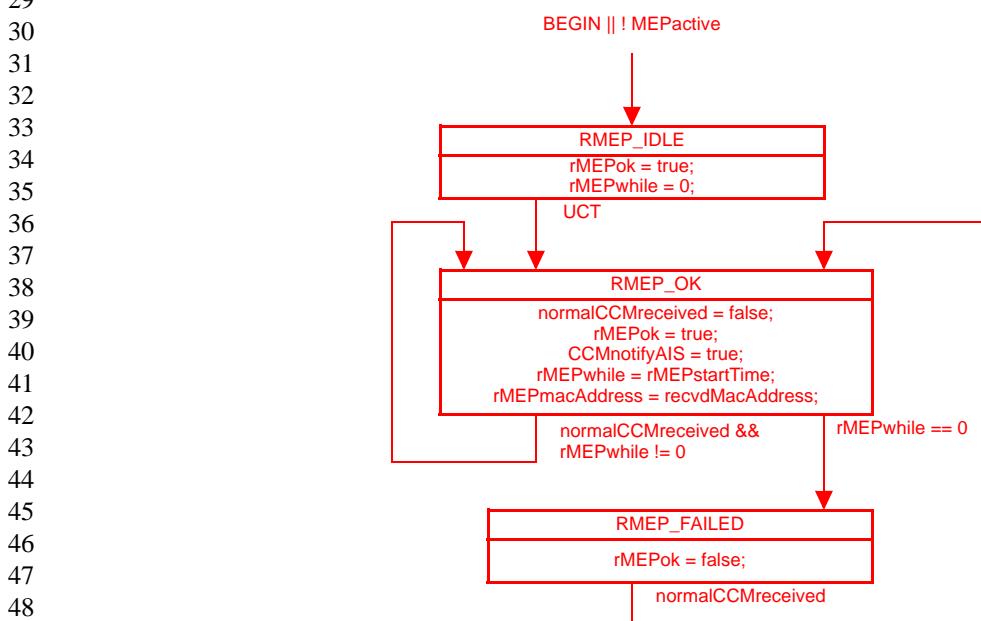


Figure 19-4—IFF Remote MEP State Machine REVISED

Figure 19-6

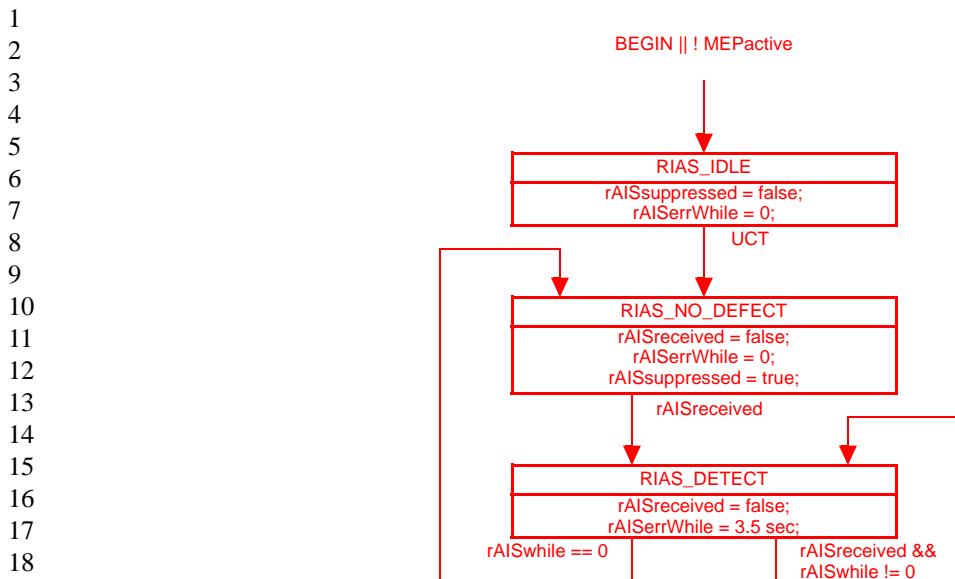
**Figure 19-5—IFF Alarm Indication Signal Receiver State Machine REVISED**

Figure 19-7

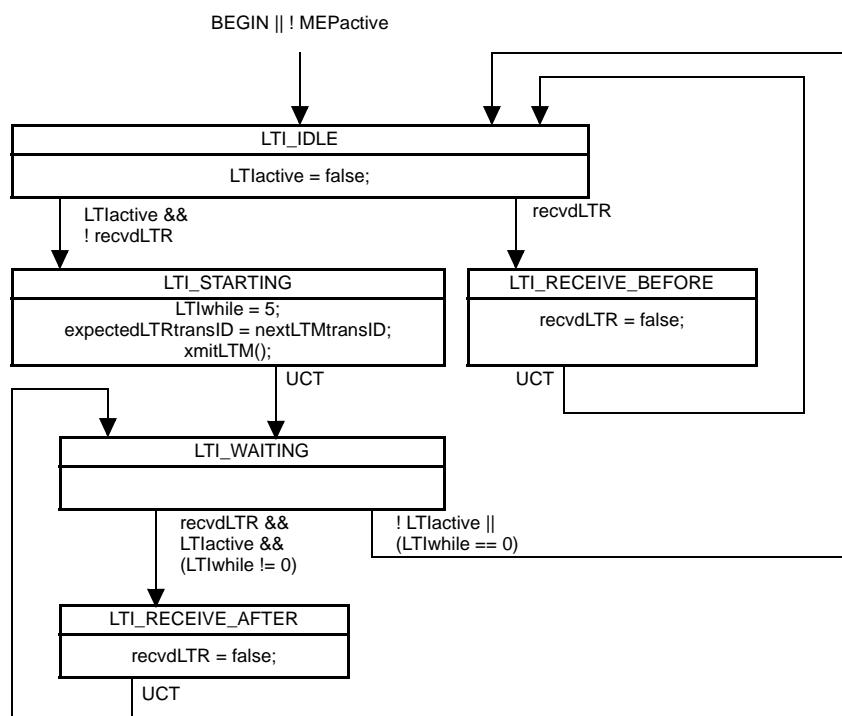
**Figure 19-7—IFF Linktrace Initiator State Machine**

Figure 19-8

Figure 19-9

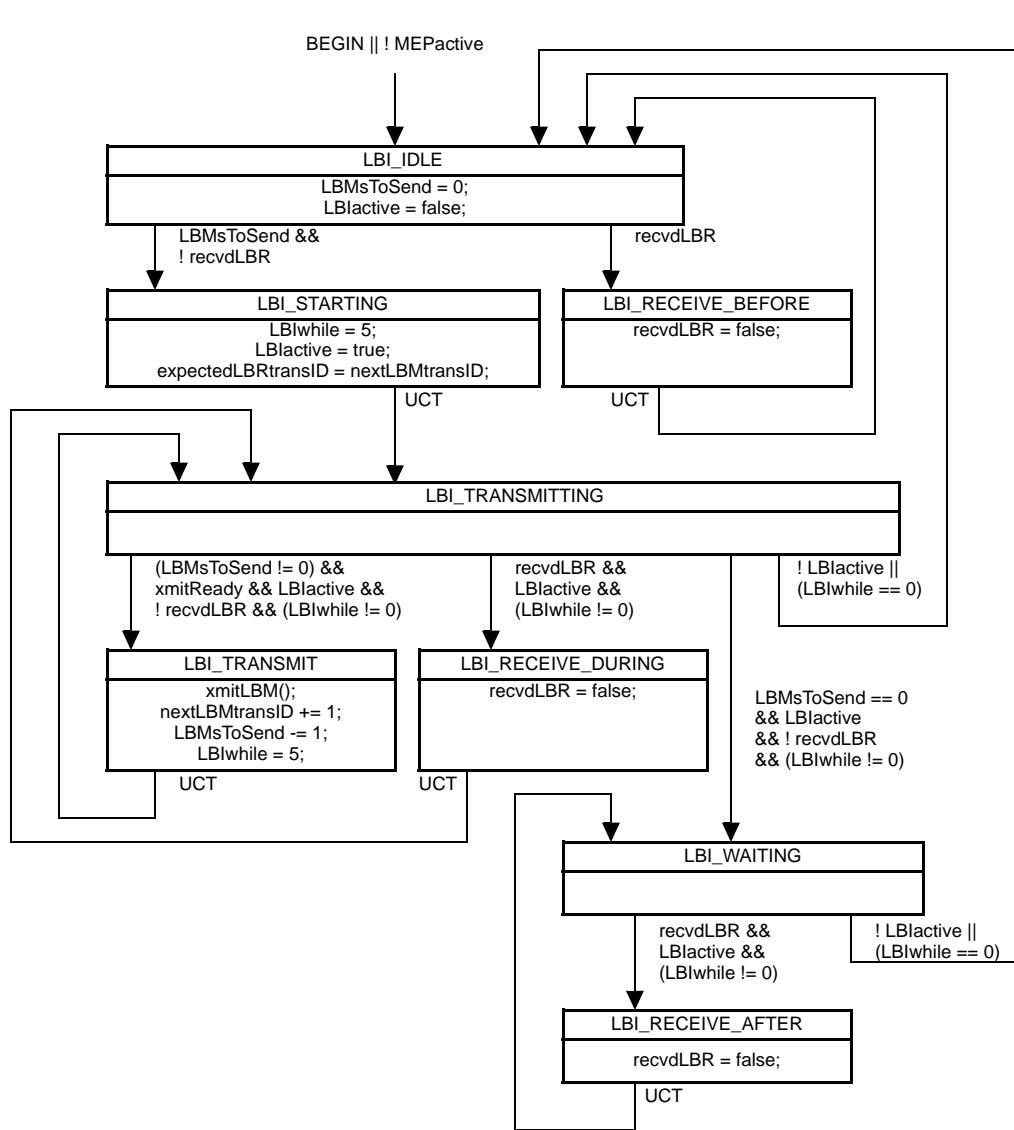


Figure 19-6—IFF Loopback Initiator State Machine

```

    graph TD
        Start(( )) --> LTF_IDLE[LTF_IDLE]
        LTF_IDLE -- "clearPendingLTRs();\n nPendingLTRs = 0;" --> LTF_WAITING[LTF_WAITING]
        LTF_WAITING -- "UCT" --> LTF_WAITING
        LTF_WAITING -- "LTFwhile = 1;" --> LTF_TRANSMITTING[LTF_TRANSMITTING]
        LTF_WAITING -- "LTFwhile == 0" --> End(( ))
        LTF_TRANSMITTING -- "xmitOldestLTR();\n nPendingLTRs -= 1;" --> Decision{nPendingLTRs}
        Decision -- "nPendingLTRs != 0" --> LTF_WAITING
        Decision -- "nPendingLTRs == 0" --> End
    
```

The diagram illustrates the sequence of states and actions for the LTF process in a MEP active state. It starts at the top with a label 'BEGIN || ! MEPactive *'. An arrow points down to a box labeled 'LTF_IDLE'. Inside this box, the action 'clearPendingLTRs(); nPendingLTRs = 0;' is performed. From 'LTF_IDLE', an arrow labeled 'UCT' points down to another box labeled 'LTF_WAITING'. Inside 'LTF_WAITING', the variable 'LTFwhile' is set to 1. From 'LTF_WAITING', an arrow labeled 'LTFwhile == 0' points to an exit point labeled 'END'. From 'LTF_WAITING', an arrow labeled 'LTFwhile = 1;' points down to a third box labeled 'LTF_TRANSMITTING'. Inside 'LTF_TRANSMITTING', the action 'xmitOldestLTR(); nPendingLTRs -= 1;' is performed. Finally, an exit point labeled 'nPendingLTRs == 0' leads to the end of the process.

Figure 19-8—IFF Linktrace Forwarder State Machine

```

    graph TD
        BEGIN["BEGIN || ! MEActive"] --> FG_IDLE[FG_IDLE]
        FG_IDLE["clearPendingLTRs(); FGwhile = 0;"] --> FG_NO_REPORT[FG_NO_REPORT]
        FG_NO_REPORT["FGwhile = 0;"] --> FG_DEFECT[FG_DEFECT]
        FG_DEFECT["FGwhile = 2.5 seconds;"]
        FG_DEFECT -- "someDefect" --> FG_REPORT_DEFECT[FG_REPORT_DEFECT]
        FG_REPORT_DEFECT["xmitFaultAlarm();"]
        FG_REPORT_DEFECT -- "someDefect && FGwhile == 0" --> FG_DEFECT
        FG_DEFECT -- "! someDefect" --> FG_NO_REPORT
        FG_NO_REPORT -- "someDefect" --> FG_REPORT_DEFECT
        FG_NO_REPORT -- "! someDefect" --> FG_IDLE
    
```

The diagram illustrates the state transition logic for the FG_IDLE state. It begins with the condition `BEGIN || ! MEActive`. The flow enters the `FG_IDLE` state, which contains the code `clearPendingLTRs(); FGwhile = 0;`. From this state, it transitions to `FG_NO_REPORT`, which contains the code `FGwhile = 0;`. From `FG_NO_REPORT`, it can transition back to `FG_IDLE` if `! someDefect`, or to `FG_DEFECT` if `someDefect`. In the `FG_DEFECT` state, the code `FGwhile = 2.5 seconds;` is executed. If `someDefect && FGwhile == 0`, the flow transitions to `FG_REPORT_DEFECT` to execute `xmitFaultAlarm();`. From `FG_REPORT_DEFECT`, it can transition back to `FG_DEFECT` if `someDefect`, or back to `FG_IDLE` if `! someDefect`.

Figure 19-9—Fault Alarm Generator State Machine NEW

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54