

1. Syntax defs - composite

```

1: data-type:
2:   stil-type - STIL block references:
3:     Signals
4:     SignalGroups
5:     PatternBurst
6:     PatternExec
7:     Timing
8:     Category
9:     Spec
10:    Selector
11:    ScanStructures
12:    Pattern
13:    Procedures
14:    MacroDefs
15:    DCLevels
16:    DCSequence
17:    Environment
18:  var-type - variables and expressions:
19:    Signal (allow: signals, groups, signal variables, sigref-expr)
20:    Integer (allow: integers, integer constants, integer-expr)
21:    SignalVariable (allow: wfc strings, \W, \r, \h, etc. )
22:    Real (allow: spec variables, eng-units, real-expr)
23:    String (text striing in double quotes)

24: TestMethod TEST_METHOD_NAME (TEST_METHOD_INSTANCE_NAME) {
25:   ( Inherit TEST_METHOD_NAME; )
26:   ( <In | Out > (Once) data-type <NAME | NAME[SIZE]> (= expr ); ) *
27: } // end of test method

28: BinDefs (BIN_DEF_NAME) { // soft bin definitions
29:   ( Bin < Pass | Fail > SOFT_BIN_NAME (SOFT_BIN_NUMBER); ) * // Incr pass-bin if TestResult == Pass
30:   ( Category BIN_CATEGORY_NAME {
31:     ( Bin < Pass | Fail > SOFT_BIN_NAME (SOFT_BIN_NUMBER); ) *
32:   }
33: }

Q1: How/where is TestResult defined? Does it need to be defined in every test-name?

34: BinMap (BIN_MAP_NAME) { // soft to hard bin mapping
35:   ( SOFT_BIN_NAME -> HARD_BIN_NUMBER; ) *
36:   ( [ (BIN_CATEGORY_NAME.SOFT_BIN_NAME) * ] -> HARD_BIN_NUMBER; ) *
37: }

Q2: What is the allowed syntax for "bin" arithmetic? or bin functions? i.e., ((speed.mhz200 && volts.v1) || (speed.mhz300 && volts.v2)) && func1 => 1;

38: ( TestNode TEST_NODE_NAME {
39:   // see TestFlow for definition of allowed statements
40: } ) * // end TestNode

Q3: A stand-alone test node can be created without any pre-actions, post-actions. Does this suffice for what was previously called a "test object"?

41: TestFlow (TEST_FLOW_NAME) {
42:   ( <In | Out > (Once) data-type <NAME | NAME[SIZE]> (= expr ); ) *
43:   ( TestNode (TEST_NODE_NAME) {
44:     ( Title "title string"; )
45:     ( Override < Off | IgnoreFail | Continue >; ) // Override fails from called test-names
46:     ( Position X Y; ) // GUI position

```

```

47:  ( PreActions {
48:    ( < If boolean_expr | Else > {
49:      ( Execute {
50:        (<NAME | NAME[INDEX]> = expr; )*
51:      } // end Execute Actions
52:      ( Bin = (CATEGORY.)SOFT_BIN_NAME; )
53:      ( <
54:        | Skip (TEST_NODE_NAME); // skip this and go to next or named node in this test_flow
55:        | Exit (RESULT); // exit this test_flow, return integer
56:        | Stop; // stop this test_program (i.e., the current On* operation)
57:      > )
58:    })* // end If/ Else
59:  }) // end PreActions
60:  ( Execute {
61:    ( TEST_NODE_NAME; )*
62:    ( TEST_METHOD_NAME; )*
63:    ( TEST_METHOD_NAME {
64:      (<NAME | NAME[INDEX]> = expr; )*
65:    })* // end test_method_name
66:    ( TEST_FLOW_NAME; )*
67:    ( TEST_FLOW_NAME {
68:      (<NAME | NAME[INDEX]> = expr; )*
69:    })* // end of test_flow_name
70:  }) // end Execute
71:  ( PostActions {
72:    ( < If boolean_expr | Else > {
73:      ( Execute {
74:        (<NAME | NAME[INDEX]> = expr; )*
75:      } // end Execute Actions
76:      ( Bin = (CATEGORY.)SOFT_BIN_NAME; )
77:      ( <
78:        | Next (TEST_NODE_NAME); // go to next or named node in this test_flow
79:        | Exit (RESULT); // exit this test_flow, return integer
80:        | Stop; // stop this test_program (i.e., the current On* operation)
81:      > )
82:    })* // end If/ Else
83:  }) // end PostActions
84:  })* // end TestNode
85: } // end TestFlow

```

Q4: Use of modifier "Mutable, Const, Private" on variable definitions? CONST implies parameter value cannot be changed during execution. MUTABLE implies that parameter value can be changed during execution.

*Q5: What is allowed syntax for an initialization expression? How to reference the current element - e.g., In Integer INT[5] = '2 * #'; In Integer SIN[128] = 'sine(#)'*

Q6: Use instance_name.var_name to access test-nodes or test-methods that are created as instances. We need a section that defines allowed expression syntax.

Q7: Need to incorporate Jim O's syntax for PreActions with a Skip (or Bypass) operator?

```

86: TestProgram (TEST_PROGRAM_NAME) {
87:  ( Device "DEVICE IDENTIFYING INFORMATION"; )
88:  ( Position INTEGER INTEGER; ) // x/y position from top left of window
89:  ( OnLoad < TEST_FLOW_NAME | TEST_NODE_NAME | TEST_METHOD_NAME >; )
90:  ( OnInitFlow < TEST_FLOW_NAME | TEST_NODE_NAME | TEST_METHOD_NAME >; )
91:  ( OnPatternLoad < TEST_FLOW_NAME | TEST_NODE_NAME | TEST_METHOD_NAME >; )
92:  ( OnStart < TEST_FLOW_NAME | TEST_NODE_NAME | TEST_METHOD_NAME >; )
93:  ( OnReset < TEST_FLOW_NAME | TEST_NODE_NAME | TEST_METHOD_NAME >; )
94:  ( OnPowerDown < TEST_FLOW_NAME | TEST_NODE_NAME | TEST_METHOD_NAME >; )
95:  ( OnFinish < TEST_FLOW_NAME | TEST_NODE_NAME | TEST_METHOD_NAME >; )
96:  ( OnLotStart < TEST_FLOW_NAME | TEST_NODE_NAME | TEST_METHOD_NAME >; )
97:  ( OnLotEnd < TEST_FLOW_NAME | TEST_NODE_NAME | TEST_METHOD_NAME >; )
98:  ( OnSiteStart < TEST_FLOW_NAME | TEST_NODE_NAME | TEST_METHOD_NAME >; )
99:  ( OnSiteEnd < TEST_FLOW_NAME | TEST_NODE_NAME | TEST_METHOD_NAME >; )
100: ( OnMultisiteEnable < TEST_FLOW_NAME | TEST_NODE_NAME | TEST_METHOD_NAME >; )
101: ( OnMultisiteDisable < TEST_FLOW_NAME | TEST_NODE_NAME | TEST_METHOD_NAME >; )

```

Q8: Need to define the exact semantics of each of the On statements; possibly with a flow chart.*

Q9: Need definition of multi-site operation.

```
102: ( ProgramOption <In | Out > data-type <NAME | NAME[SIZE]> (= expr) ;)* // operator information
103: ( GlobalVariable (Const) data-type <NAME | NAME[SIZE]> (= expr) ;)* // hidden from operator
104: ( BinDefs BIN_DEF_NAME; )
105: ( BinMap BIN_MAP_NAME; )
Q10: Do BinDefs and BinMap need to be unique per-site?
106: } // end TestProgram
```