Proposals for syntax and semantics of test program variables

1. Variables statement inside test program block. This alternative requires that the TestProgram block be defined BEFORE any of the flows or tests, and so the flows or tests named in the EntryPoints would have to be forward references. Note that STIL allows forward references in some situations (specifically, the Pattern names called out in a Pattern Burst are forward references, as the actual patterns are the last thing to be defined in a dot0 STIL file).

```
Variables TestProgramVars {
      Integer DeviceCount;
}

. . .

TestProgram WaferSortProgram {
      . . .
      Variables TestProgramVars;
      EntryPoints {
            // MainFlow is defined AFTER TestProgram
            // WaferSortProgram, and any test or flow blocks
            // which follow can use any global variables (in
            // unnamed variables blocks) or any variables defined
            // in variables blocks TestProgramVars
            OnStart MainFlow; // Flow MainFlow is a forward
                              // reference, to be defined AFTER
                              // the TestProgram block
      }
}

. . .

// Definition of Tests and Flows
```

Pros:

- Follows conventional STIL block structure (especially as far as variables are concerned).
- Explicit identification of variables to be used in TestProgram (Variables statement is contained within TestProgram block - variables to be used in the test program (TestProgram block) are unambiguously associated with the test program.)

Cons:

- Forward references needed (but allowed for in STIL).
- TestProgram block must be defined before tests and flows (perhaps counter-intuitive)

2. "Using" statement. This alternative allows any needed Variables blocks to be made available prior to use via the “Using” statement. These variables are available to any subsequent block definitions. The Tests and Flows can then use these variables as needed, and there is no need to have the TestProgram block (with its “Variables” declaration) defined before the tests or flows which need to use the variables.

```
Variables TestProgramVars {
      Integer DeviceCount;
}

Using {
      // All variables mentioned in the "Variables domain_name"
      // statement(s) are available for use by subsequent block
      // definitions
      Variables TestProgramVars;
}

. . .

// Test and flow Definitions
Flow MainFlow {   // No flowtype mentioned, so MainFlow is an
                  // instance of the default STIL flowtype
      . . .
}

TestProgram WaferSortProgram {
      . . .
      EntryPoints {
            // MainFlow is defined BEFORE TestProgram
            // WaferSortProgram, and any test or flow blocks
            // which follow can use any global variables (in
            // unnamed variables blocks) or any variables defined
            // in variables blocks TestProgramVars
            OnStart MainFlow; // Flow MainFlow is already defined
                              // (not a forward reference)
      }
}
```

Pros:

- No forward references needed (can always adhere to “define-before-use”).

Cons:

- Need new keyword/block (Using keyword/block).
- Using block implies namespaces; STIL namespaces (domain names) and C/C++ namespaces aren’t 100% analagous.
- Variables to be used in the test program (TestProgram block) are not unambiguously associated with the test program.

3. Variables statement. This is similar to the "Using" statement, except that it is not required to enclose the "Variables var_domain;" statement inside a Using block.

```
Variables TestProgramVars {
      Integer DeviceCount;
}


// All variables mentioned in the "Variables domain_name"
// statement(s) are available for use by subsequent block
// definitions.  This "Variables var_domain" statement is not
// contained inside any other block, and is at the top
// level.  Similar to the version with the "Using" block, but no
// "Using" block is required.
Variables TestProgramVars;

  . . .

// Test and flow Definitions
Flow MainFlow {   // No flowtype mentioned, so MainFlow is an
                  // instance of the default STIL flowtype
      . . .
}

TestProgram WaferSortProgram {
      . . .
      EntryPoints {
            // MainFlow is defined BEFORE TestProgram
            // WaferSortProgram, and any test or flow blocks
            // which follow can use any global variables (in
            // unnamed variables blocks) or any variables defined
            // in variables block TestProgramVars
            OnStart MainFlow; // Flow MainFlow is already defined
                              // (not a forward reference)
      }
}
```

Pros:

- Fits in with other parts of STIL smoothly.
- No forward references needed (can always adhere to “define-before-use”).

Cons:

- Potential semantic ambiguity with free-standing “Variables VAR_DOMAIN;” statement outside any block context.
- Variables to be used in the test program (TestProgram block) are not unambiguously associated with the test program.