1450.4 meeting minutes – 11/12/09

Attendees: Jim O'Reilly, Ernie Wahl, Bruce Parnas, Ajay Khoche
Not present: Markus Seuring
Agenda:

- IEEE Meeting Preamble (No discussion of proprietary information)
- Minutes from last week's meeting (10/29) now posted on the web.
- No meeting on 10/8 (cancelled by prior notice).
- I can only stay on until 9:00 am PST. Hard stop then.
- Met with STARC representatives at ITC (Bruce, Ajay, Jim, Paul Roddy of CAST). Summary to follow shortly.
- **For next week, can we move the meeting to Friday (same time?).**
    - **Agreed. Next week's meeting to be held on Friday, 11/20/09.**
- Discussion items (same as 10/22/09).
    - SignalRefVariable type (let's do this first, it might be quicker!).
        - To allow passing Signal lists (signals, lists of signals, SignalGroups) into Tests or Flows.
        - Discussion issues: syntax, semantics, capabilities
            - Proposed syntax: New data type called **SignalRefVariable.** **SignalRefVariable** mySigRefVar = *sigref_expr*;
            - Issues:
                - What about signal groups, individual signals (DataBus0), or bussed signals (DataBus[0])? How are these handled?
                - How do you access individual elements of a *sigref_expr*?
                - How do we figure out how many individual signals are present in a *sigref_expr*? How do we walk through a *sigref_expr*, signal by signal?
            - Resolution:
                - Use keyword **SigGroup** for new data type.
                    - **SigGroup** mySigGroup = *sigref_expr;*
                    - Name of data type changed for clarity. Proposed type has is essentially a single member of an individual SignalGroup as defined in the SignalGroups block. It holds any valid *sigref_expr*. Chose keyword **SigGroup** rather than SignalGroup (as singular of existing keyword **SignalGroups**) to distinguish it from the existing keyword for both human eyes and computer search patterns.
                - mySigGroup.Size gives number of signals in a *sigref_expr*
                - Can index through a SigGroup variable using array indexing notation – i.e., mySigGroup[<index>].
                - Should we allow multiple indices instead of just a single index (for bussed signals)?
                    - Will talk with Greg to clarify properties, behavior, and treatment of bussed signals. To be discussed next week.
            - Added to GoogleDocs issues (link below) as issue #45

- SpecVariable type.
  - **Did not get to in-depth discussion of this topic – simply (re)enumerated the issues. Will pick up with this topic next week**.
  - To allow passing spec variables into Tests or Flows
  - Discussion issues: syntax, semantics, capabilities
    - Questions: (to be clarified with Greg).
      - If only Typ is specified, no selector is needed. Do Min and Max exist or not in this case? If yes, what value? If no, then they can't be used in expressions.
      - Does Meas exist (it must . . .). What value does it have if not initialized?
      - Are only options to define Typ only, or Min, Typ, Max? i.e., you can't define Min and Typ only, or Typ and Max only, or Min and Max only? (Syntax shows that when using block format - specifying Min, Typ, Max - each element is optional . . .
      - Do we need an both an initial and a current value for Meas? (Meas.original and Meas.current, as is done for bin counters).
    - If an initial value is not specified for Min, Typ, Max, or Meas, what value is assigned? (0 with appropriate units? NaN - a new value type?). If NaN, and used in an expression, what is value of expression? NaN also, or just ignore that component of expression (as if it was 0)?
    - What happens if .Meas is not specified? Is it available for use or not? Same as unspecified .Min or .Max
    - Suggestions: <spec_var_name>.Name returns spec name as a string. <spec_var_name>.Units returns units (unscaled) as a string. Each component (.Min, .Typ, .Max) must have the same units, but can be scaled differently (i.e., .Min = 100 pS, .Typ = 0.2 nS, etc.). Thoughts?
- Other issues:
  - Conversions from one data type to another – what are the rules? These need to be explicitly defined.
  - Limits as a specific data type?
    - How to specify an open limit on one end of a comparison?
    - Specify >=, >, <=, < - what does the syntax look like?
    - Limits x 0nS >= n <= 3nS
  - List of reserved words by tester type as an integral part of dot4? (should this be a dot3 addition?). Does NameChecks in dot3 cover this, or could it cover it with some extensions.
  - Added issue #44 (extensions to signal types In, Out, InOut, Pseudo) are needed – for instance, to permit specification of unused package pins.
    - Perhaps a signal type called NC (no-connect) would be useful in these cases.
- Open issues - are there other open issues that should be considered? A review of the open issues list can guide us here.
  - http://spreadsheets.google.com/ccc?key=pEI1-gPUmt2ZTw_kcCTgnKw&inv=jim_oreilly@ieee.org&t=9330484534885518 71&guest).
  - If logged into your google account, can edit. If not, can only view.
- Next Meeting 11/20/09.

For reference STIL .4 information can be found at the IEEE STIL website:
http://grouper.ieee.org/groups/1450/ (select the P1450.4 link from the table) or use the direct link
http://grouper.ieee.org/groups/1450/dot4/index.html