Scoreboard Generic Package
User Guide

User Guide for Release 2022.03

By

Jim Lewis

SynthWorks VHDL Training

Jim@SynthWorks.com

http://www.SynthWorks.com

Table of Contents

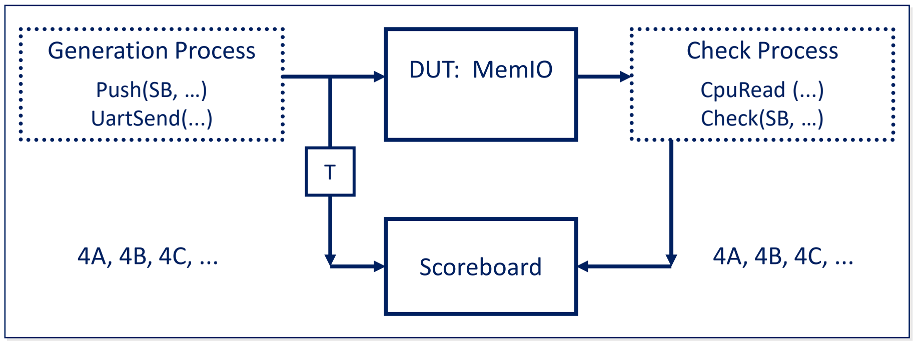
1. ScoreboardGenericPkg Overview

A scoreboard is a data structure used for self-checking in an environment where inputs are closely related to outputs, such as in data transmission (UART, AxiStream, …).

A basic usage of a scoreboard is:

- On the stimulus generation side, we generate the stimulus, push the stimulus into the scoreboard, and then sent the transaction to the interface (UartSend).
- On the stimulus check side, we receive a transaction (CpuRead) and use the scoreboard to check the value.

This is shown in the figure below.



2. Basic Usage

2.1 Package References

The Scoreboard package instances are not in the OSVVM context declaration, so you will need to include them separately.

```
library OSVVM ;
  context osvvm.OsvvmContext ;
  use osvvm.ScoreboardPkg_slv.all ;
```

2.2 Create a Scoreboard ID

A Scoreboard ID is a handle to the scoreboard object that is created internal to an instance of ScoreboardGenericPkg. A Scoreboard ID has the type ScoreboardIDType. ScoreboardIDType is a regular type and can be held in a signal or variable.

```
architecture Uart_Test1 of TestCtrl is
  signal SB : ScoreboardIDType ;
```

If more than one scoreboard package instance is referenced, the ScoreboardIDType must be disambiguated using a fully selected name.

```
architecture Uart_Test1 of TestCtrl is
  use osvvm.ScoreboardPkg_slv.all ;
  use osvvm.ScoreboardPkg_int.all ;

  signal SB_slv  : osvvm.ScoreboardPkg_slv.ScoreboardIDType ;
  signal SB_int  : osvvm.ScoreboardPkg_int.ScoreBoardIDType;
```

2.3 Construct the Scoreboard

NewID allocates and initializes (sizes) the memory data structure. It must be called before any read or write operations are used.

```
SB <= NewID("UART_SB") ;
```

2.4 Push Before Send

In the stimulus generation process, push the expected value into the scoreboard before sending the transaction.

```
 Push(SB, X"10") ;
 UartSend(UartTxRec, X"10") ;
```

2.5 Get before Check

In the stimulus checking process, receive an actual transaction value (UartGet) and then check the value using the scoreboard.

```
UartGet(UartRxRec, RcvD) ;
Check(SB, RcvD) ;
```

2.6 Putting the basic pieces together

The following is a basic use model of the scoreboard. For more information on how OSVVM structures test cases and test processes, see the OSVVM_test_writers_user_guide.pdf.

```
architecture Uart_Test1 of TestCtrl is
  signal SB : ScoreboardIDType ;
  . . .
begin
  ControlProc : process
  begin
    SetAlertLogName("UART_Test1") ;
    SB <= NewID("UART_SB") ;
    wait until nReset = '1' ;
    WaitForBarrier(TestStart) ;
    WaitForBarrier(TestDone, 5 ms) ;
```

```
      ReportAlerts ;
      std.env.stop(GetAlertCount) ;
      wait ;
    end process ControlProc ;

    GenerateProc : process
    Begin
      WaitForBarrier(TestStart) ;
      Push(SB, X"10") ;
      UartSend(UartTxRec, X"10") ;
      . . .
      Push(SB, X"FF") ;
      UartSend(UartTxRec, X"FF") ;
      TestDone <= TRUE ;
      WaitForBarrier(TestDone) ;
      wait ;
    end process GenerateProc ;

    CheckProc : process
      variable RcvD : std_logic_vector(7 downto 0);
    begin
      WaitForBarrier(TestStart) ;
      while not TestDone loop
        UartGet(UartRxRec, RcvD) ;
        Check(SB, RcvD) ;
        wait for UART_BAUD_PERIOD ;
      end loop ;
      WaitForBarrier(TestDone) ;
      wait ;
    end process CheckProc ;
  end architecture UART_Test1 ;
```

Note that the test generation process generates numerous values using either directed (shown) or random methods, the checking side is a simple loop.   Hence, the big advantage of using a scoreboard is that the checking side remains simple and has no need to know what the test generation side is doing.

3. Using ScoreboardGenericPkg Package Generics

The scoreboard package uses package generics to allow the ExpectedType (transaction value sent) and ActualType (transaction value received) to be different for different applications.   The function Match is used to determine if the Actual value matches the Expected Value.   This supports checking that is more complicated than a simple "=" check.   In addition there are functions expected_to_string and actual_to_string to help printing values of ExpectedType and ActualType respectively. Hence, the interface to the package is:

```
  package ScoreBoardGenericPkg is
  generic(
    type ExpectedType ;
```

```
      type ActualType ;
      function Match( Actual : ActualType ;
                      Expected : ExpectedType) return boolean ;
      function expected_to_string(A: ExpectedType) return string;
      function actual_to_string  (A: ActualType)   return string
    ) ;
```

To use the scoreboard for your tests, you will need to create a package instance. The following example creates a scoreboard for integer. Note often you will need to reference the package that defines your types and functions that correspond to ExpectedType, ActualType, match, expected_to_string, and actual_to_string, however, here for integer all types and functions are available in package std.standard which is implicitly visible.

```
    -- reference the package that define types and functions.
    -- use std.standard.all ; -- implicitly included
    package ScoreBoardPkg_int is new osvvm.ScoreBoardGenericPkg
      generic map (
        ExpectedType        =>  integer,
        ActualType          =>  integer,
        match               =>  std.standard."=",
        expected_to_string  =>  to_string,
        actual_to_string    =>  to_string
      ) ;
```

Note the OSVVM library defines ScoreBoardPkg_int and ScoreboardPkg_slv (std_logic_vector).

4. My Simulator Does Not Support Package Generics

All the simulators we test with support package generics. However, we still have you covered. Copy ScoreboardGenericPkg.vhd to ScoreboardPkg_integer_c.vhd. Comment out the generics. Uncomment the subtype and alias declarations that correspond to the generics (they have the same name). Configure these aliases exactly as described in the section titled "Using Scoreboard Package Generics".

The top of your resulting package should look as follows:

```
    -- reference the package that define types and functions.
    -- use std.standard.all ; -- implicitly included
    package ScoreBoardPkg_integer is
      subtype ExpectedType is integer ;
      subtype ActualType   is integer ;
      alias match is std.standard."=" [integer, integer
                                    return boolean] ;
      alias expected_to_string is
        to_string [integer return string];
      alias actual_to_string is
        to_string [integer return string] ;
```

5. Scoreboard API Reference

The intent of the API is to make the creation of a scoreboard easy. The OSVVM scoreboard goes beyond simple though in that it supports out of order execution (via tags) and arrays of scoreboards (via effective use of abstractions). Tags are implemented as an additional string parameter on push, pop, and check.

Behind the simple API are some rather interesting data structures. The important thing to note is that you do not need to understand the data structures to be able use ScoreboardGenericPkg.

Caution, features not documented here are either considered experimental or have been deprecated (and may be removed in a future version). If you decide to use it and want it to stay around, be sure to let us know.

5.1 NewID and Supporting Types

5.1.1 NewID: Data Structure Constructor

NewID allocates and initializes the scoreboard data structure. It must be called before any other scoreboard operations are used.

```
impure function NewID (
  Name          : String ;
  ParentID      : AlertLogIDType          := OSVVM_SCOREBOARD_ALERTLOG_ID ;
  ReportMode    : AlertLogReportModeType  := ENABLED ;
  Search        : NameSearchType          := NAME_AND_PARENT_ELSE_PRIVATE ;
  PrintParent   : AlertLogPrintParentType := PRINT_NAME_AND_PARENT
) return ScoreboardIDType ;
```

NewID also creates an AlertLogID for the data structure using the value of the name parameter and the value of ParentAlertLogID.

5.1.2 ScoreboardIDType

ScoreboardIDType is defined as follows.

```
type ScoreboardIDType is record
  ID : integer_max ;
end record ScoreboardIDType ;
```

5.1.3 AlertLogReportModeType

```
type AlertLogReportModeType is (DISABLED, ENABLED, NONZERO) ;
```

When DISABLED, an AlertLogID does not print in ReportAlerts (text reports) or WriteAlertYaml (Yaml/HTML reports). When NONZERO, an AlertLogID does not print in ReportAlerts, but prints in WriteAlertYaml. Hence, NONZERO shortens the text reports to what is essential. When ENABLED, an AlertLogID prints in ReportAlerts and WriteAlertYaml.

None of these settings impact printing in Alert or Log.

DISABLED is intended to be used for data structures that are not used for self-checking and may produce errors on parameter checks. These errors are not lost as they are still accumulated in the parent ID.

5.1.4 AlertLogPrintParentType

```
type AlertLogPrintParentType is (PRINT_NAME, PRINT_NAME_AND_PARENT) ;
```

When PRINT_NAME, Alert and Log print the name of the AlertLogID. When PRINT_NAME_AND_PARENT, Alert and Log print the parent ID name, the ID separator, and then the ID name.

5.1.5 NameSearchType

```
type NameSearchType is (
    PRIVATE, NAME, NAME_AND_PARENT, NAME_AND_PARENT_ELSE_PRIVATE) ;
```

When NewID is called, first a search of existing IDs is done. If there is a match, that ID is returned, if there is not, a new one is created. Matching is determined by the ID, ParentID, and Search parameters. If the Search parameter is PRIVATE, no searching is done and a new ID is created. If the Search parameter is NAME, then any ID whose name matches is returned. If the Search parameter is NAME_AND_PARENT, then any ID whose name and ParentID match or NAME and stored Search parameter is NAME is returned.

5.2 Push

Push adds the expected value (ExpectedType) to the scoreboard.

```
-- Simple Scoreboard, no tag
procedure Push (
  constant ID     : in  ScoreboardIDType ;
  constant Item   : in  ExpectedType
) ;
```

```
-- Simple Tagged Scoreboard
procedure Push (
  constant ID     : in  ScoreboardIDType ;
  constant Tag    : in  string ;
  constant Item   : in  ExpectedType
) ;
```

5.3 Check

Check a received value (ActualType) with value in scoreboard. The Match function is used to determine if the received and expected values match. Checking is handled by AffirmIf. As a result, if they match a log PASSED is generated, otherwise, an alert ERROR is generated.

```
-- Simple Scoreboard, no tag
procedure Check (
  constant ID           : in  ScoreboardIDType ;
  constant ActualData   : in ActualType
) ;

-- Simple Tagged Scoreboard
procedure Check (
  constant ID           : in  ScoreboardIDType ;
  constant Tag          : in  string ;
  constant ActualData   : in  ActualType
) ;
```

5.4 Pop

The scoreboard can also be used as a FIFO. Pop removes the expected value from the top of the FIFO (oldest value) and returns it. Pop is supported as either a procedure or function.

```
-- Simple Scoreboard, no tag
procedure Pop (
  constant ID     : in  ScoreboardIDType ;
  variable Item   : out  ExpectedType
) ;

-- Simple Tagged Scoreboard
procedure Pop (
  constant ID     : in  ScoreboardIDType ;
  constant Tag    : in  string ;
  variable Item   : out  ExpectedType
) ;


-- Simple Scoreboard, no tag
impure function Pop (
  constant ID     : in  ScoreboardIDType
) return ExpectedType ;
```

```
-- Simple Tagged Scoreboard
impure function Pop (
  constant ID     : in  ScoreboardIDType ;
  constant Tag    : in  string
) return ExpectedType ;
```

5.5 Peek

Peek returns the expected value from the top of the FIFO (oldest value) without removing it.   Peek is supported as either a procedure or function.

```
-- Simple Tagged Scoreboard
procedure Peek (
  constant ID     : in  ScoreboardIDType ;
  constant Tag    : in  string ;
  variable Item   : out ExpectedType
) ;

-- Simple Scoreboard, no tag
procedure Peek (
  constant ID     : in  ScoreboardIDType ;
  variable Item   : out  ExpectedType
) ;

-- Tagged Scoreboards
impure function Peek (
  constant ID     : in  ScoreboardIDType ;
  constant Tag    : in  string
) return ExpectedType ;

-- Simple Scoreboard
impure function Peek (
  constant ID     : in  ScoreboardIDType
) return ExpectedType ;
```

5.6 Empty

Check if the Scoreboard is empty.

```
impure function Empty (
  constant ID     : in  ScoreboardIDType
) return boolean ;

-- Tagged
impure function Empty (
  constant ID     : in  ScoreboardIDType ;
  constant Tag    : in  string
) return boolean ;
```

5.7 GetAlertLogID

GetAlertLogID returns the AlertLogID associated with a scoreboard.

```
impure function GetAlertLogID (
  constant ID     : in  ScoreboardIDType
) return AlertLogIDType ;
```

5.8 Find

Return the ItemNumber of the oldest expected value that matches the received value. Find + Flush support systems that drop items before they are synchronized.

```
-- Simple Scoreboard
impure function Find (
  constant ID          : in  ScoreboardIDType ;
  constant ActualData  :  in  ActualType
) return integer ;

-- Tagged Scoreboard
impure function Find (
  constant ID          : in  ScoreboardIDType ;
  constant Tag         :  in  string;
  constant ActualData  :  in  ActualType
) return integer ;
```

5.9 Flush

Quietly drop all values whose item number is less than the specified item number. Find + Flush support systems that drop items before they are synchronized.

```
-- Simple Scoreboards
procedure Flush (
  constant ID          : in  ScoreboardIDType ;
  constant ItemNumber  :  in  integer
) ;

-- Tagged Scoreboards - only removes items that also match the tag
procedure Flush (
  constant ID          : in  ScoreboardIDType ;
  constant Tag         :  in  string ;
  constant ItemNumber  :  in  integer
) ;
```

5.10 GetPushCount

Get number of items pushed into the scoreboard.

```
impure function GetPushCount (
  constant ID     : in  ScoreboardIDType
) return integer ;
```

5.11 GetPopCount

Get number of items poped from the FIFO in that is part of the scoreboard.

```
impure function GetPopCount (
  constant ID     : in  ScoreboardIDType
) return integer ;
```

5.12 GetCheckCount

Get number of items checked by the scoreboard.

```
impure function GetCheckCount (
  constant ID     : in  ScoreboardIDType
) return integer ;
```

5.13 GetDropCount

Get number of items dropped by the scoreboard.

```
impure function GetDropCount (
  constant ID     : in  ScoreboardIDType
) return integer ;
```

5.14 GetFifoCount

Get number of currently stored in the FIFO that is part of the scoreboard.

```
impure function GetFifoCount (
  constant ID     : in  ScoreboardIDType
) return integer ;
```

5.15 Getting the Scoreboard Error Count

The scoreboard reports errors to the AlertLog data structure. Hence, the error count will be reported with ReportAlerts.

```
ScoreboardErrorCount := GetAlertCount(GetAlertLogID(SB)) ;
```

6. Tagged Scoreboards

Tagged Scoreboards are used for systems that allow transactions to execute out of order.

Tags are represented as string values (since most types convert to string using to_string). A tag value is specified as the first value in the calls to push, check, and pop, such as shown below. In all examples, ExpectedVal has the type ExpectedType, and ReceiveVal has the type ActualType.

```
Push(SB, "WriteOp", ExpectedVal) ;
Check(SB, "WriteOp", ReceiveVal) ;
```

```
    Pop(SB, "WriteOp", ExpectedVal) ;

    if Empty(SB, "MyTag")  then   …
```

For Check (and Pop), the item checked (or returned) is the oldest item with the matching tag.

```
    ItemNum := Find(SB, "ReadOp", ReceiveVal);
    Flush(SB, "ReadOp", ItemNum) ;
```

For Flush, only items matching the tag are removed. In some systems, it may be appropriate to do the Find with the tag and the flush without the tag.

7. Indexed Scoreboards

Indexed scoreboards are for systems, such as a network switch that have multiple scoreboards that are most conveniently represented as an array.

ScoreboardIDType is an ordinary type, so normal VHDL methods can be used to create arrays of it.

7.1 1D and 2D Array Types

ScoreboardGenericPkg defines the following types:

```
    -- 1D array type
    type ScoreboardIdArrayType  is array (integer range <>) of ScoreboardIdType ;

    -- 2D array type
    type ScoreboardIdMatrixType is array (integer range <>, integer range <>)
        of ScoreboardIdType ;
```

7.2 NewID for ScoreboardIdArrayType

NewID is defined for ScoreboardIdArrayType with the following parameters:

```
    -- Vector: 1 to Size
    impure function NewID (
      Name          : String ;
      Size          : positive ;
      ParentID      : AlertLogIDType          := OSVVM_SCOREBOARD_ALERTLOG_ID ;
      ReportMode    : AlertLogReportModeType  := ENABLED ;
      Search        : NameSearchType          := NAME_AND_PARENT_ELSE_PRIVATE ;
      PrintParent   : AlertLogPrintParentType := PRINT_NAME_AND_PARENT
    ) return ScoreboardIDArrayType ;
```

```
-- Vector: X(X'Left) to X(X'Right)
impure function NewID (
  Name          : String ;
  X             : integer_vector ;
  ParentID      : AlertLogIDType          := OSVVM_SCOREBOARD_ALERTLOG_ID ;
  ReportMode    : AlertLogReportModeType  := ENABLED ;
  Search        : NameSearchType          := NAME_AND_PARENT_ELSE_PRIVATE ;
  PrintParent   : AlertLogPrintParentType := PRINT_NAME_AND_PARENT
) return ScoreboardIDArrayType ;
```

7.3 NewID for ScoreboardIdMatrixType

NewID is defined for ScoreboardIdMatrixType with the following parameters:

```
-- Matrix: 1 to X, 1 to Y
impure function NewID (
  Name          : String ;
  X, Y          : positive ;
  ParentID      : AlertLogIDType          := OSVVM_SCOREBOARD_ALERTLOG_ID ;
  ReportMode    : AlertLogReportModeType  := ENABLED ;
  Search        : NameSearchType          := NAME_AND_PARENT_ELSE_PRIVATE ;
  PrintParent   : AlertLogPrintParentType := PRINT_NAME_AND_PARENT
) return ScoreboardIdMatrixType ;

-- Matrix: X(X'Left) to X(X'Right), Y(Y'Left) to Y(Y'Right)
impure function NewID (
  Name          : String ;
  X, Y          : integer_vector ;
  ParentID      : AlertLogIDType          := OSVVM_SCOREBOARD_ALERTLOG_ID ;
  ReportMode    : AlertLogReportModeType  := ENABLED ;
  Search        : NameSearchType          := NAME_AND_PARENT_ELSE_PRIVATE ;
  PrintParent   : AlertLogPrintParentType := PRINT_NAME_AND_PARENT
) return ScoreboardIdMatrixType ;
```

7.4 Arrays of Scoreboards

The following operations are appropriate for any array of scoreboards. Note that each scoreboard is independent of each other.   Also note that each can support different std_logic_vector sizes.

```
signal SB : ScoreboardIDArrayType(1 to 3) ;
. . .

-- Create 3 indexed scoreboards
SB <= NewID("SB", (1, 3));
wait for 0 ns ;

-- Push values into scoreboards
Push(SB(1), X"11") ;
Push(SB(2), X"2222") ;
Push(SB(3), X"3") ;
```

```
    -- Check values using scoreboards
    Check(SB(3), X"3") ;
    Check(SB(2), X"2222") ;
    Check(SB(1), X"01") ;

    -- test completion all scoreboard errors report via AlertLogPkg
    ReportAlerts ;
```

8. Deprecated NewID

First, if you were not using the DoNotReport parameter, there is no difference in calling the deprecated NewID functions and the new functions. The deprecated functions call the new NewID with ReportMode set to ENABLED if DoNotReport is FALSE and DISABLED otherwise.

8.1 Deprecated: NewID for ScoreboardIdType

```
    impure function NewID (
      Name             : String;
      ParentAlertLogID : AlertLogIDType ;
      DoNotReport      : Boolean
    ) return ScoreboardIDType ;
```

8.2 Deprecated: NewID for ScoreboardIdArrayType

NewID is defined for ScoreboardIdArrayType with the following parameters:

```
    impure function NewID (
      Name             : String ;
      Size             : positive ;   -- Array range: (1 to Size)
      ParentAlertLogID : AlertLogIDType ;
      DoNotReport      : Boolean
    ) return ScoreboardIDArrayType ;

    impure function NewID (
      Name             : String ;
      X                : integer_vector ;   -- Array range: (X(X'Left) to X(X'Right))
      ParentAlertLogID : AlertLogIDType ;
      DoNotReport      : Boolean
    ) return ScoreboardIDArrayType ;
```

8.3 Deprecated: NewID for ScoreboardIdMatrixType

NewID is defined for ScoreboardIdMatrixType with the following parameters:

```
    -- Matrix range: (1 to X, 1 to Y)
    impure function NewID (
      Name             : String ;
      X, Y             : positive ;
      ParentAlertLogID : AlertLogIDType;
      DoNotReport      : Boolean
    ) return ScoreboardIdMatrixType;

    -- Matrix range: (X(X'Left) to X(X'Right), Y(Y'Left) to Y(Y'Right))
    impure function NewID (
      Name             : String ;
      X, Y             : integer_vector ;
      ParentAlertLogID : AlertLogIDType ;
      DoNotReport      : Boolean
    ) return ScoreboardIdMatrixType ;
```

9. A Better Way of Calling NewID? Not yet.

As an ordinary type, ScoreboardID could be a constant and NewID called in the constant declaration as follows:

```
    constant ScoreboardID : ScoreboardIDType := NewID("SB") ;
```

One big benefit of this is that NewID is called during elaboration of the design before any other scoreboard API subprogram can run.

While the tools we test with allow this, unfortunately the language does not. We plan to address this in the next revision of the language. See: https://gitlab.com/IEEE-P1076/VHDL-Issues/-/issues/75

10. What about the older revisions of ScoreboardGenericPkg?

Versions of ScoreboardGenericPkg before 2021.06 were solely protected type based. That use model is still supported, however, it is deprecated and not talked about here. If you need the documentation for it, see the subdirectory ProtectedTypes_the_old_way in the Documentation repository.

11. Compiling ScoreboardGenericPkg

See Script_user_guide.pdf for the current compilation directions.

12. About ScoreboardGenericPkg

ScoreboardGernericPkg was developed and is maintained by Jim Lewis of SynthWorks. It has been used for years in our Advanced VHDL Testbenches and Verification class.

We held it back from the OSVVM library waiting for current tool releases to support VHDL-2008 generic packages and resolve some of the bugs. At this point, main stream VHDL simulators seem to be ready.

Please support our effort in supporting ScoreboardGernericPkg and OSVVM by purchasing your VHDL training from SynthWorks.

ScoreboardGernericPkg is released under the Apache open source license. It is free (both to download and use - there are no license fees). Currently the OSVVM family of packages can be downloaded from either osvvm.org or from our GitHub site: https://github.com/OSVVM/OSVVM.

If you add features to the package, please donate them back under the same license as candidates to be added to the standard version of the package. If you need features, be sure to contact us. I blog about the packages at http://www.synthworks.com/blog. We also support the OSVVM user community and blogs through http://www.osvvm.org.

Find any innovative usage for the package? Let us know, you can blog about it at osvvm.org.

13. Future Work

ScoreboardGernericPkg is a work in progress and will be updated from time to time.

Caution, undocumented items are experimental and may be removed in a future version.

14. About the Author - Jim Lewis

Jim Lewis, the founder of SynthWorks, has thirty plus years of design, teaching, and problem solving experience. In addition to working as a Principal Trainer for SynthWorks, Mr Lewis has done ASIC and FPGA design, custom model development, and consulting.

Mr. Lewis is chair of the IEEE 1076 VHDL Working Group (VASG) and is the primary developer of the Open Source VHDL Verification Methodology (OSVVM.org) packages. Neither of these activities generate revenue. Please support our volunteer efforts by buying your VHDL training from SynthWorks.

If you find bugs these packages or would like to request enhancements, you can reach me at jim@synthworks.com.