**Real-Time Pass/Review Tool User Guide**

**1.** **Formatting Requirement Tables**

*1.1.* *HRS vs. RealTime Tables:*

*1.2.* *Making the Requirements Machine Readable (XML):*

**2.** **Expanded Definitions of Requirement Sections**

*2.1.* *ReqID:*

*2.2.* *Parameter Type:*

*2.3.* *UNITS:*

*2.4.* *Test Parameter:*

*2.5.* *CONDITION:*

*2.6.* *MAX/MIN:*

*2.7.* *NOM:*

**3.** **Vector to Vector Comparisons**

*3.1.* *Value\_Cache:*

**4.** **Output File Update**

*4.1.* *XML Output Results:*

*4.2.* *EXCEL:*

**5.** **LabVIEW Implementation**

~~1.~~~~Formatting Requirement Tables 2~~

~~1.1.~~~~HRS vs. RealTime Tables: 2~~

~~1.2.~~~~Making the Requirements Machine Readable (XML): 3~~

~~2.~~~~Expanded Definitions of Requirement Sections 3~~

~~2.1.~~~~ID: 3~~

~~2.2.~~~~Parameter Type: 4~~

~~2.3.~~~~UNITS: 4~~

~~2.4.~~~~Test Parameter: 4~~

~~2.5.~~~~CONDITION: 6~~

~~2.6.~~~~MAX/MIN: 8~~

~~2.7.~~~~NOM: 8~~

~~3.~~~~Vector to Vector Comparisons 8~~

~~3.1.~~~~Value\_Cache: 8~~

~~4.~~~~Output File Update 9~~

~~4.1.~~~~XML Output Results: 9~~

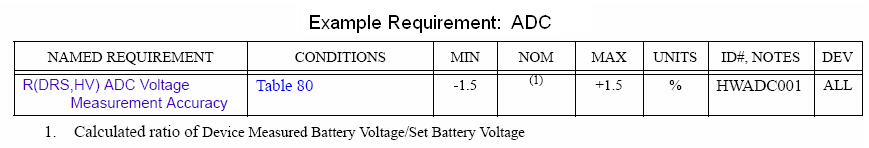
~~4.2.~~~~EXCEL: 10~~

~~5.~~~~LabView Implementation 11~~

**1. Formatting Requirement Tables**

1.1. HRS vs. RealTime Tables:

This tool has been designed to assess data against requirements at runtime. In order for this to occur, the Hardware or Test requirements must be placed in a machine readable format. ~~The~~ An example requirement~~current~~ format comes from the HRS and is as follows:



**Figure 1. Example~~Current~~ HRS Table Format**

Where:

NAMED REQUIREMENT = A name or definition for the given ID#, NOTES selection.

CONDITION = typically refers to a test conditions table.

MIN = minimum value the test parameter can equal and still be considered to be running within the requirement specification.

MAX = maximum value the test parameter can equal and still be considered to be running within the requirement specification.

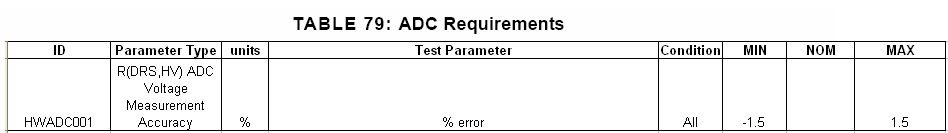
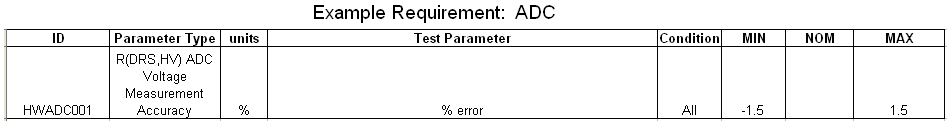
NOM = the nominal value of the test parameter for the given ~~device or~~ vector and/or device configuration.

UNITS = Measured unit or type for the given test parameter

ID#, NOTES = the requirement identification number

DEV = device type the requirement ID# relates to

This format needs to be changed in order to provide enough information for software to decode and process during the execution of test. The re-format is as follows:



**Figure 2. Real Time Table Format**

Where:

Parameter Type = NAMED REQUIREMENT; name or definition for the given ID#, NOTES selection

Condition = CONDITION; will include vector control names and settings specific to the requirement under test

MIN = MIN; minimum value the test parameter can equal and still be considered to be running within the requirement specification.

MAX = MAX; maximum value the test parameter can equal and still be considered to be running within the requirement specification.

NOM = NOM; the nominal value of the test parameter for the given ~~device or~~ vector and/or device configuration.

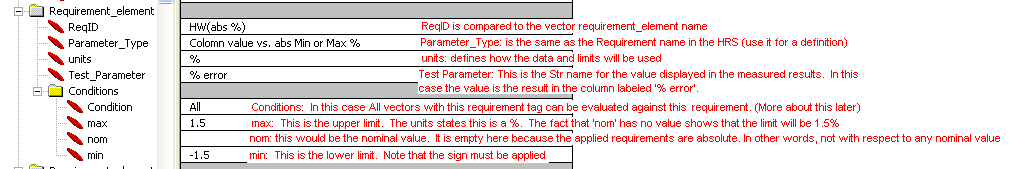
~~UNITS~~ units = UNITS; measured unit or type for the given test parameter

ID = ID#, NOTES; the requirement identification number

Test ~~Parameter =~~Parameter =  ~~This addition~~ tells the software exactly what measured value(s) will be used to be compared to the limits.

1.2. Making the Requirements Machine Readable (XML):

The previous format can be considered “readable” but must be converted in to a machine readable format by converting it into XML as follows:



**Figure 3. Real Time XML Format**

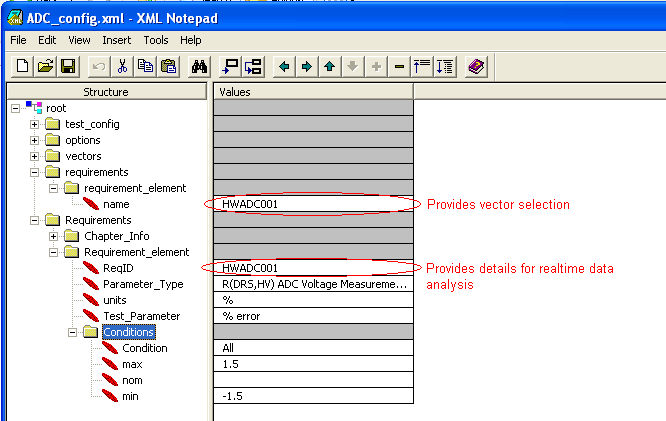
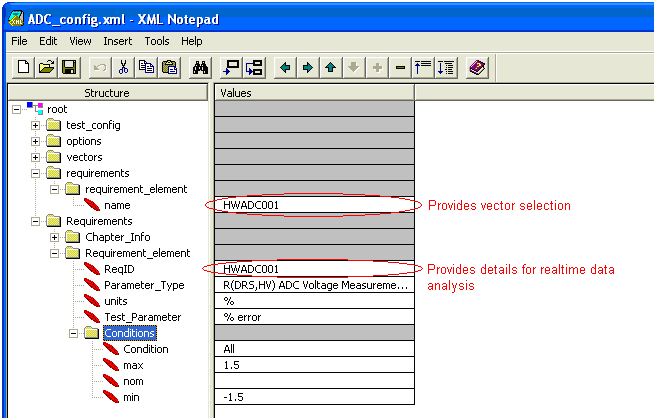
**~~1.~~**

**~~1. In most cases it would be desirable to include this section in the test configuration file.~~**

**2. Expanded Definitions of Requirement Sections**

2.1. ReqID:

The ReqID parameter is used to determine the requirement that is under test. This parameter will be set by the user when they are setting up the vectors they are testing. The vector requirement ID parameter will be looked up at runtime and compared against the requirement table provided by the configuration table. If a match is found, the data will be processed and reported upon. Although this parameter’s intent is to relate to actual HRS ID#, it is a string comparison and would allow for any name the user wishes to use. This parameter must be placed in two locations of the XML configuration file as such:



**Figure 4. ReqID XML Locations**

If the requirement\_element name is removed, the test will execute normally without appending any requirements analysis to the results.

2.2. Parameter Type:

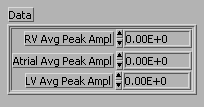
This parameter is maintained for the purpose of readability only. The software does not use this parameter for any decoding or for any calculations. The purpose of the item is for use when the tables are being reviewed. It should provide specific indication to the reviewer the purpose of the measurement and/or requirement.

2.3. UNITS:

The measured unit or type for the given test parameter has two purposes. Its first purpose is to notify the person reading the report of the type or scale of the measurement. Its second is to tell the code how the limits will be calculated. ~~In the case of the code, t~~The ‘%’ will result in the MIN and MAX calculations being represented as a percentage. If a NOM value is provided, the MIN and MAX will be with respect to the nominal. If no NOM is provided, then the MIN and MAX are absolute percentage values. Only one value is allowed under the Units selection.

2.4. Test Parameter:

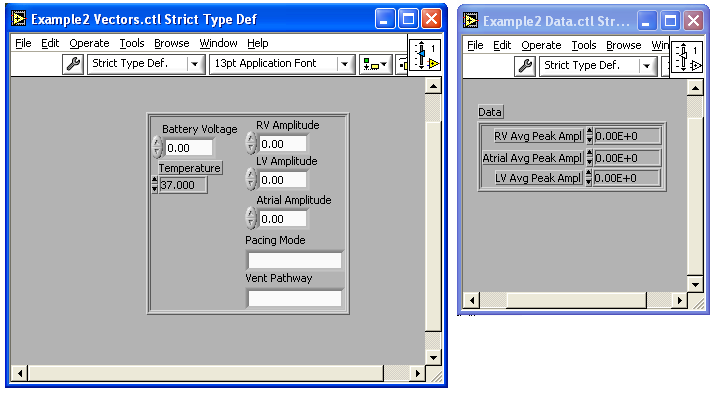
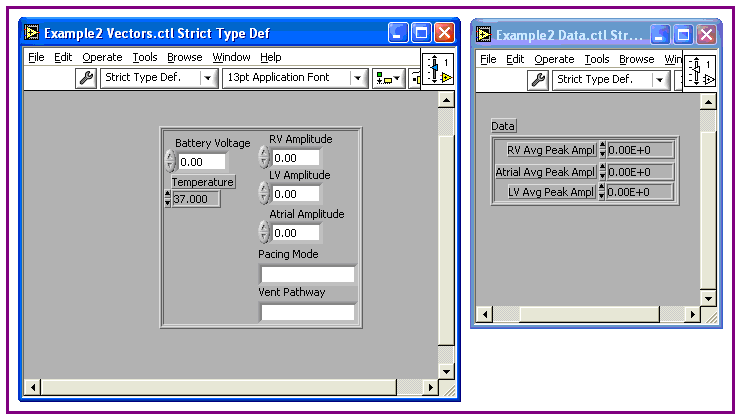
This ~~addition~~ tells the software exactly what measured value will be used to be compared to the limits. The value here would typically contain the name of the data indicator name that is used by the test. The value under test must match the data indicator name. For example:



**Figure 5. Data Indicator Example**

The Test Parameter for the right ventricular peak amplitude would be: RV Avg Peak Ampl. An array of test parameters could also be checked by separating them by a semi-colon. Such as: RV Avg Peak Ampl; Atrial Avg Peak Ampl; LV Avg Peak Ampl; This would allow for multiple data points to be tested against the given requirement.

This parameter can also contain a calculation that is performed on the combination of data and vector values. If the following vector control and data indicators exist and it is desirable for the data indicator to be compared to a percentage of the programmed value the following calculation could be done:



**Figure 6. Example Using Vector and Data**

Test Parameter = (RV Avg Peak Ampl - RV Amplitude)/RV Amplitude \* 100.0 ; An array of measurements could also be done by way of separating them with the use of a semi-colon: (RV Avg Peak Ampl - RV Amplitude)/RV Amplitude \* 100.0 ; (Atrial Avg Peak Ampl - Atrial Amplitude)/Atrial Amplitude\*100.0; Each Test Parameter requested will produce a separate result summary.

‘Simple math’ can be used on the given ‘Test\_Parameter’. Embedded parentheses such as: ((a+b)/c) \* d) will work but will also increase processing time. The Math functions supported in this version are:

/ = division

\* = multiplication

+ = addition

- = subtraction

^x = exponent with x equal to any number

( x ) = parentheses

| x | =absolute value

PEMDAS[[1]](#footnote-1) applies so the calculation of 5-3+2 will be performed correctly. The normal concerns about order of operations should be upheld. The column header names must match the names located in the calculations or a math error will be reported.

2.5. CONDITION:

Include vector and data control names and settings specific to the requirement under test. If this parameter is set to ‘All’, any vector with the associated ReqID will be tested against the requirement. However, if the <Condition> section is not empty or set to ‘All’, multiple ReqID tags can be used by changing the condition string. The format is:

**<Vector Heading>CT<Vector Heading Value>; ……….**

Where CT = the comparison type or how the comparison will be done.

The Comparison Types are defined as follows:

**:** will do a string comparison (stricmp) ~~comparison~~

**=** the value will be typed to a floating point and will be compared

**<** typed to float, Less Than or ‘ lt ‘

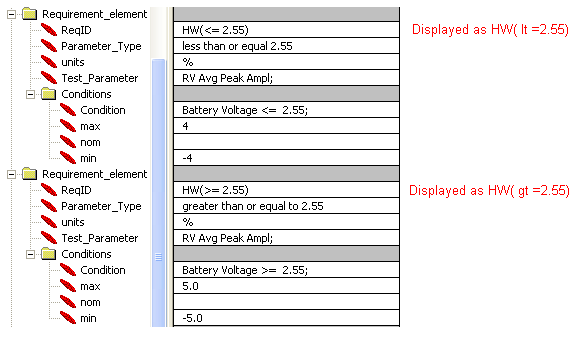
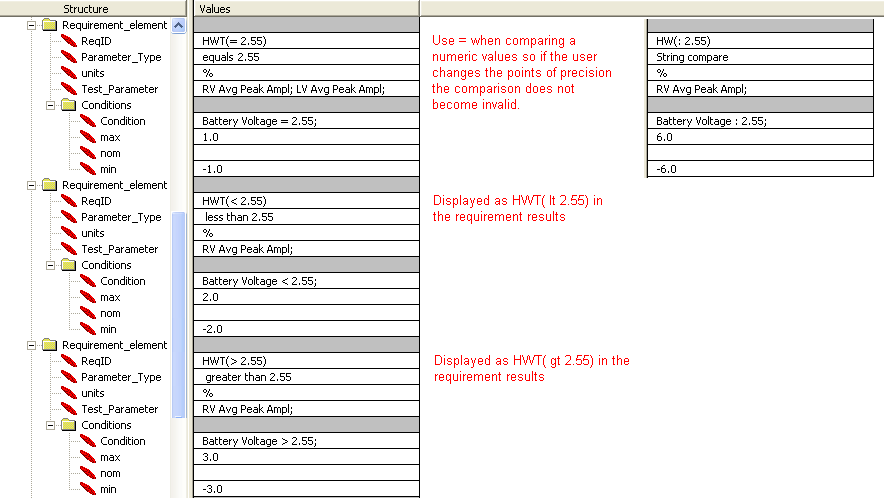
**>** typed to float, Greater Than or ‘ gt ‘

**<=** typed to float, Less Than or equal or ‘ lt =‘

**>=** typed to float, Greater Than or equal or ‘ gt =‘

Because < and > are wide characters that will be formatted to &lt; and &gt; respectively by the test system they needed to be reformatted by the data analysis code into an easier readable form. They will be displayed as ‘ lt ‘ for < and ‘ gt ‘ for >.

When a numeric value is being compared it is recommended that instead of the ‘: ‘or string comparison being used, that a ‘=’ or floating point comparison be~~is not~~ used. This is because the user often has the ability to change the points of precision of the value. If the user chooses less or more points of precision, the string comparison will be detected ~~it~~ as a Mismatch. This will not be a problem during a floating point comparison. The table below shows how the formatting will be displayed.



**Figure 7. Comparison Type Formatting**

It is possible to have multiple conditions verified so ranges can be selected around input vector parameters. In the above example, if you wished for ReqID HW(>=2.55) to be applied to vectors containing battery amplitudes from 2.55 to 3.0 volts the condition statement would be changed to:

Battery Voltage >= 2.55; Battery Voltage <= 3.0;

In this example, pacing amplitude ranges may also be added as a condition check. If the limits of +/- 5% only applied to RV pacing amplitudes from 5 to 8 volts at a battery setting from 2.55 to 3.0 volts, the condition would be formatted as such:

Battery Voltage >= 2.55; Battery Voltage <= 3.0;

RV Amplitude >= 5; RV Amplitude <= 8.0;

2.6. MAX/MIN:

These are values that contain the upper and lower limits. Only one value or mathematical expression is allowed for each. The value can be with respect to the NOMinal. For instance, if MAX is 1.0, Min is -2.0 and NOM is 4. The value of MAX would actually be 5.0 and the MIN would be 2.0. ‘TBD’ can be used during development if the limit of the test parameter is unknown. The result status will be set to review. The MAX value may be left empty in the case of a limit that simply has to be greater than or equal to a MIN value. The MIN value may be left empty in the case where the limit must be less than or equal to a MAX value.

2.7. NOM:

This is the nominal value assigned to the test parameter. Only one value or mathematical expression is allowed. In most cases, NOM maybe left empty. This would mean that the MAX and MIN values would be absolute. If a value is placed in NOM then the MAX and MIN values would be with respect to the NOM value as described in section 2.6~~2.6~~ MAX/MIN. ‘TBD’ can be used during development if the limit of the test parameter is unknown. The result status will be set to review.

**3. Vector to Vector Comparisons**

3.1. Value\_Cache:

A value cache parameter can be added to the XML configuration file to allow for limited vector to vector comparisons. The most typical use of this would be when a ‘nominal’ value needs to be captured and a series of vectors need to be compared with respect to that nominal. The Value\_Cache contains Cache\_Elements comprised of a name and a condition. The name contains the name used within the Requirement configurations section followed by the column name that relates to the value you want to store. The format is:

<name>CP<column value name>;

Where

‘name’ is the~~:~~ name as used in formulas

‘CP’ is ‘:’ (colon) ~~( : )~~ because it will be doing a string comparison~~stricmp~~ for the data element

‘column value name’ is the ~~:~~ data indicator name

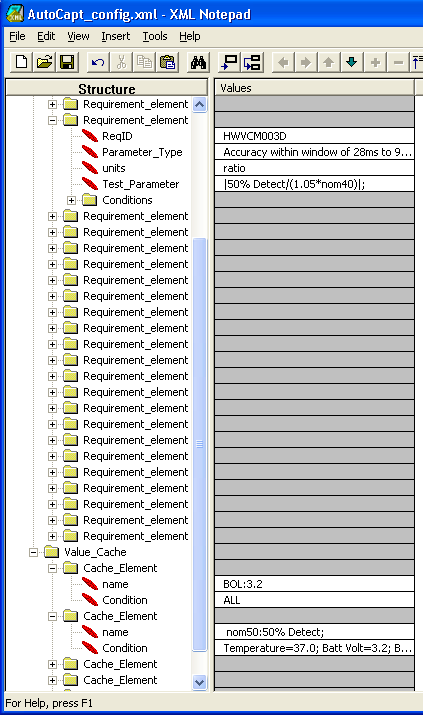
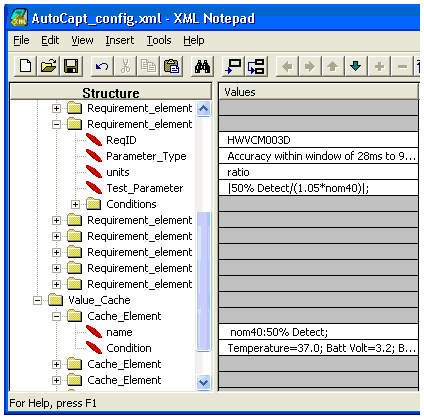
The ‘n~~N~~ame’ can contain an array of values separated by a semi colon.

The C~~c~~ondition is a list of vector parameters and values as has been described before. The format is as follows:

**<Vector Heading>CT<Vector Heading Value>; ……….**

Where ‘CT’ is ~~=~~ the comparison type or how the comparison will be done.

When doing a comparison within the Value\_Cache section the typical CT values would be ‘:’ for a ~~stricmp~~ string comparison or ‘=’ for a floating point comparison. It is also recommended that all vector setting parameters be listed. The following example shows how this is done in the AutoCapt\_config.xml as well as how the defined parameter of ‘nom40’ can be used as a variable in the test parameter calculation:



**Figure 8. Value Cache Example**

Because ‘nom40’ needs to be captured in order to calculate HWVCM003D the ‘nom40 Cache\_Element’ c~~C~~ondition must be run first. It is recommended that the Value\_Cache vectors be run first when testing.

**4. Output File Update**

4.1. XML Output Results:

The real-time pass/~~-~~review tool will add a req\_element to the output result of the data. The req\_element will contain a req\_name, result\_name, req\_upperlim, req\_value, req\_lowerlim, and req\_status for each test parameter requested.

req\_name: This is the ID as provided by the vector

result\_name: Test Parameter Name

req\_upperlim: Requirement upper limit

req\_value: Value of the Test Parameter

req\_lowerlim: Requirement l~~L~~ower limit

req\_status: Pass/Review status. This parameter may also state if a condition or requested ID was not found.

The following is an XML output example showing how two test parameters would be added to the normal output of a given test.

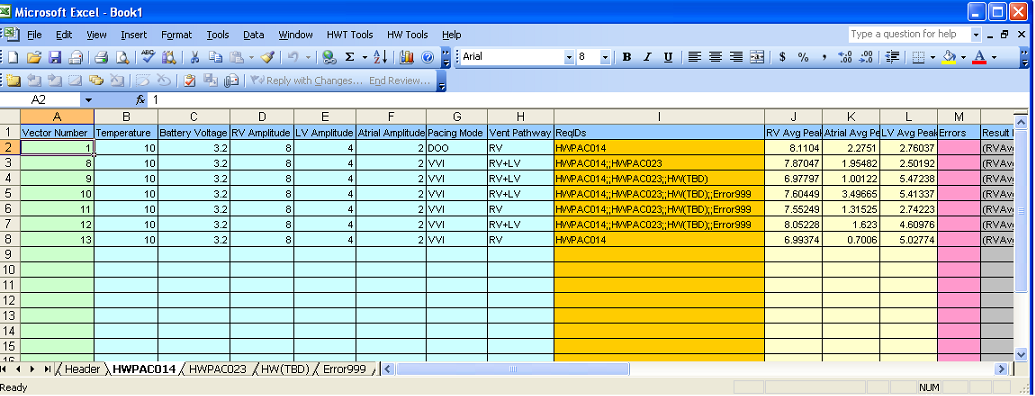


**Figure 9. XML Output**

If you do not wish to capture the requirement analysis in the results, simply remove the requirement\_element name section from the test configuration file.

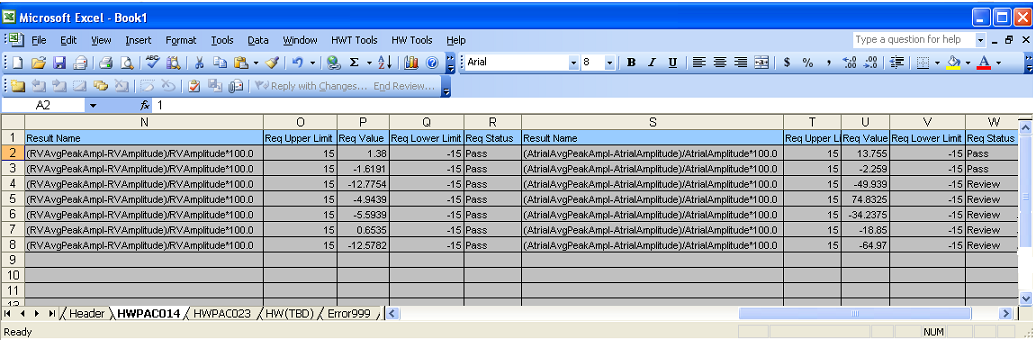
4.2. EXCEL:

A Microsoft Excel Add-In has been developed to assist in viewing the new requirement information data in Excel. This Add-In sorts the vectors and corresponding results displaying each requirement ID on its own tab as shown in ~~f~~Figures 1~~2~~0 and 1~~3~~1.



**Figure 10~~2~~. Addition of ReqIDs**

The real time requirement data addition (shown in F~~f~~igure 11~~3~~) displays requirement data elements as they relate to the requirement tab that they are displayed on. Each test parameter analyzed will contain a section displaying the Result Name, Req Upper Limit, Req Value, Req Lower Limit, and the Pass/Review Req Status.

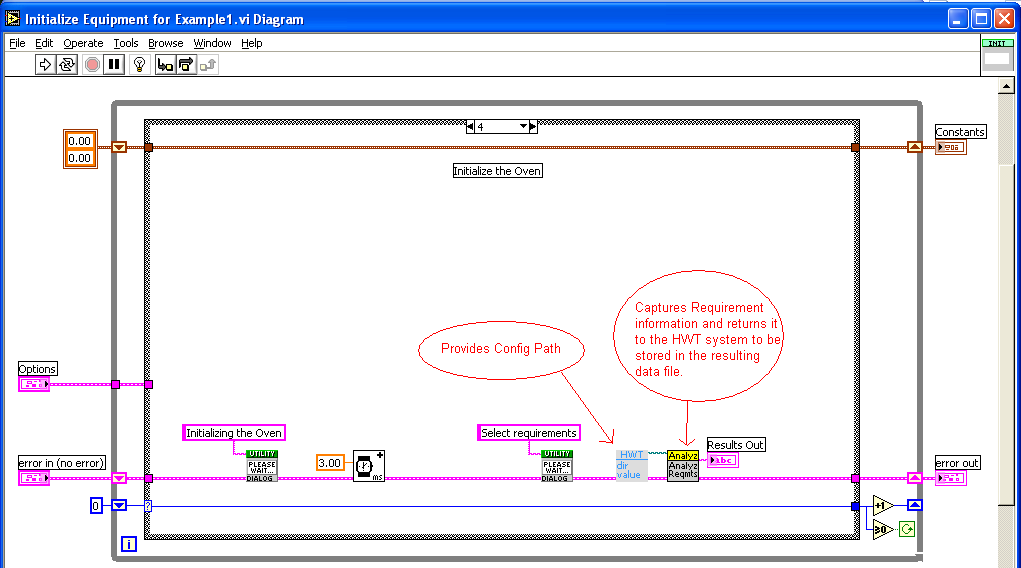
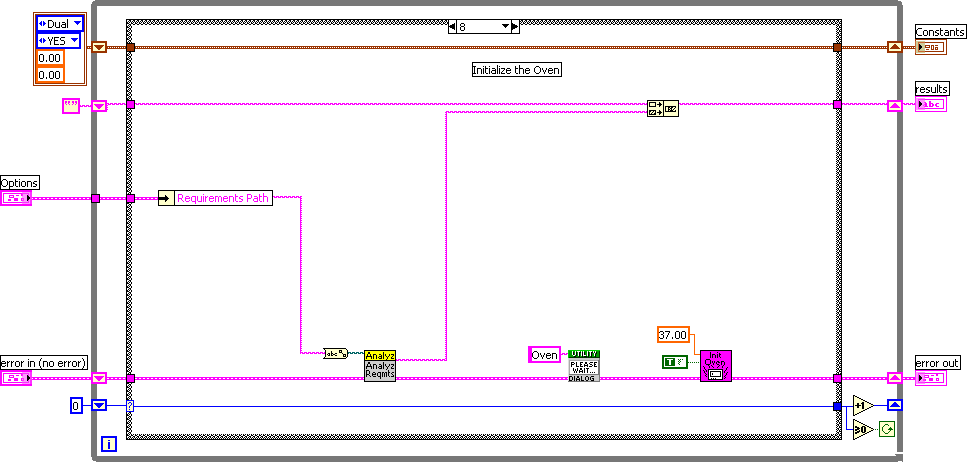


**Figure 11~~3~~. Real-Time Requirement Data Addition**

The Header Tab contains the standard hardware test information such as the result file path, project and version information, test options and equipment list. It will also contain a list of the requirements as they were executed by the Requirements XML configuration.

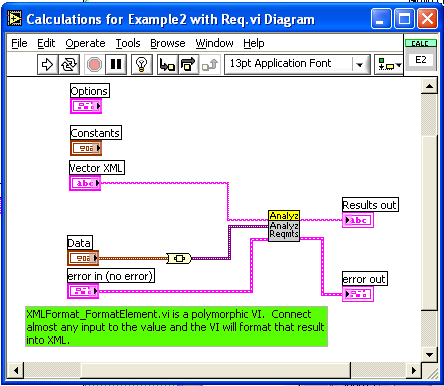
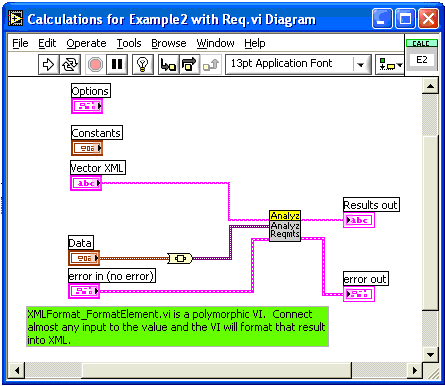
**5. LabVIEW~~iew~~ Implementation**

One VI (Analyze Requirements.vi) will need to be added in two locations within an existing test. The first location is within the initialize equipment sub-vi. In this location, the path to the requirement configuration file is provided. The VI will open this file one time in order to capture the requirements and Value\_Cache. The output of the VI will contain this information and is handed back to the hardware test system to be recorded in the result data. See Figure 12~~4~~.



**Figure 12~~4~~. Analyze Requirements.vi in Initialize Equipment**

The second location is in the calculations VI. Here the data will be analyzed against the requirements and the data will be appended to the existing results and handed back to the hardware test system. See Figure 13~~5~~.



**Figure 13. Analyze Requirements.vi in Calculations**

**~~Figure 5.~~**

This code will simply report the results back to the hardware test system if no requirement information is found within the configuration file.

1. PEMDAS is a mnemonic specifying the order of operations: parentheses, exponents, multiplication, division, addition and subtraction. [↑](#footnote-ref-1)