Guardbanding Using MET/CAL V7

MET/CAL V7.10 (and later) supports guardbanding. The simplest way to enable guardbanding is to add the necessary *VSET* statements at the top of the main procedure. The *VSET* parameters that allow guardbanding to be enabled and configured are described in detail below.

Specifying Guardbanding Parameters

In general, guardbanding-related parameters may be specified in several ways:

- You may specify a parameter using a *VSET* statement at any point in a procedure. The specification takes effect when the *VSET* statement is executed, and remains in effect until the parameter value is changed or reset. To make the specification to apply to the entire procedure, place the *VSET* statement before the first evaluation step in the main procedure.
- You may specify a parameter value using a procedure *TSET* statement. A *TSET* specification applies only to the individual test in which the *TSET* statement occurs.
- A parameter value may be specified in the MET/CAL initialization file (*metcal.ini*). An initialization file specification applies to all procedures executed on the workstation, unless overridden by a *VSET* or *TSET* statement.
- A parameter may be specified in a guardbanding initialization file. By placing the guardbanding initialization file on a network drive, you can configure a system where multiple workstations share the same guardbanding parameters. The guardbanding initialization file is described in more detail later in this document.

Some parameters do not support all of the specification methods listed above. Refer to the documentation below which describes each parameter in detail.

Precedence of Specification Methods

As described above, there are four distinct ways to specify a value for a guardbanding parameter. The four methods have the following precedence:

1. A *TSET* specification has the highest priority. *TSET* takes precedence over *VSET*, the MET/CAL initialization file, and the guardbanding initialization file.
2. A *VSET* specification has the second highest priority. *VSET* takes precedence over the MET/CAL initialization file and over the guardbanding initialization file.
3. Specifying the value of a guardbanding parameter in the MET/CAL initialization file has the third highest priority. An initialization file specification has precedence over a guardbanding initialization file specification, but not over a *TSET* or *VSET* specification.
4. The guardbanding initialization file has the lowest priority.

Guardbanding Parameters

- *gb*

  The *gb* parameter is used to enable or disable guardbanding, and to specify the guardband method.

  Legal values for *gb* are:

  - *mu*

    Setting *gb* to *mu* selects the *measurement uncertainty* method. When this method is used, the guardband limits are determined by tightening the specification limits by the expanded measurement uncertainty. The guardband factor, *gbf*, may be used to tighten the limits by a specified fraction of the measurement uncertainty.

    It is important to realize that when the guardband method is *mu* you must configure MET/CAL to calculate measurement uncertainty. If the measurement uncertainty calculation is not enabled setting *gb* to *mu* has no effect.

  - *tsr*

    Setting *gb* to *tsr* selects the *test specification ratio* method. *tsr* is a table-based method where the calculated test specification

ratio is used as the lookup parameter when the guardband table is accessed.

The terminology *tsr* may be a bit confusing. The *tsr* is the value that MET/CAL has traditionally called the TUR (Test Uncertainty Ratio). The test specification ratio is the ratio of the specification of the UUT to the reference accuracy.

See also the description of the *guardband_table* parameter below. A *guardband_table* specification must be given whenever a table-based guardband method is specified.

- *tur*

  Setting *gb* to *tur* selects the *test uncertainty ratio* method. *tur* is a table-based method where the calculated test uncertainty ratio is used as the lookup parameter when the guardband table is accessed.

  The *tur* used for this guardband method is the ratio of the specification of the UUT to the expanded measurement uncertainty. (This is not the same as the value MET/CAL has traditionally called the TUR. The TUR is really the test specification ratio (TSR), also commonly referred to as the test accuracy ratio (TAR).)

  See also the description of the *guardband_table* parameter below. A *guardband_table* specification must be given whenever a table-based guardband method is specified.

- *ntur*

  Setting *gb* to *ntur* selects the *normalized test uncertainty ratio* method. *ntur* is a table-based method where the calculated normalized test uncertainty ratio is used as the lookup parameter when the guardband table is accessed.

  The *ntur* used for this guardband method is the ratio of the normalized specification of the UUT to the standard measurement uncertainty. See also the description of the *uconf*

parameter. It may be necessary to specify a *uconf* value to correctly normalize the UUT specification.

See also the description of the *guardband_table* parameter below. A *guardband_table* specification must be given whenever a table-based guardband method is specified.

- *rds*

  Setting *gb* to *rds* selects the *root difference square* method. The *rds* method recalculates the test tolerance as:

  $$\sqrt{tol^2 - emu^2}$$

  where *tol* is the UUT specification and *emu* is the expanded measurement uncertainty.

  As with the *mu* method, it is important to realize that when the guardband method is *rds* you must configure MET/CAL to calculate measurement uncertainty. If the measurement uncertainty calculation is not enabled setting *gb* to *rds* has no effect.

  When the guardband method is *rds*, guardbanding is automatically disabled for any test in which the expanded measurement uncertainty exceeds the UUT specification. (In other words, if *emu* is greater than *tol*, the *rds* method cannot be used.)

- *direct*

  Setting *gb* to *direct* selects the *direct guardband factor* specification method. When the method is *direct*, the procedure writer may directly specify the guardband factor to use for a particular test or set of tests. See the description of the *gbf* parameter used to set the guardband factor.

- *off*

Setting *gb* to *off* unconditionally disables guardbanding.

If you have not specified the guardbanding method in the MET/CAL initialization file or in the guardbanding initialization file then setting *gb* to *off* is the same as setting *gb* to *'*'*.

However, if you have specified the guardbanding method in the MET/CAL initialization file or in the guardbanding initialization file, then assigning the value *'*'* to *gb* in a *VSET* or *TSET* statements causes the parameter value to revert to the value specified in the initialization file. In this case setting *gb* to *off* is required to disable guardbanding. Use a *TSET* statement to disable guardbanding for individual tests. Use a *VSET* statement to disable guardbanding for an entire procedure, or portion of a procedure.

The value *off* is not supported in V7.10 or V7.11. V7.11b or later is required to set *gb* to *off*.

- *gbf*

The parameter *gbf* is used to specify the guardband factor. The guardband factor is used in different ways, depending on the guardband method.

If the guardband method is *mu*, the guardband factor is applied to the expanded measurement uncertainty.

For example, if you set *gbf* to 0.8, the guardband limits are determined by tightening the specification limits by 80% of the expanded measurement uncertainty.

When the guardband method is *mu,* the default guardband factor is 1.0.

If the guardband method is *direct*, the guardband limits are determined by directly applying the guardband factor to the specification limits.

For example, suppose a test is being done at 100 V, with a UUT specification of ± 1%. In this example, the lower specification limit is 99 V and the upper specification limit is 101 V. If you set the guardband factor to 0.75, the lower guardband limit will be 99.25 V and the upper guardband limit will be 100.75 V.

The guardband factor must be specified when the guardband method is *direct*. If *gbf* is not specified no guardbanding is performed.

- *gb_init*

  The parameter *gb_init* is used to specify the name of the guardbanding initialization file. *gb_init* is an optional initialization file parameter. *gb_init* may not be specified in a *TSET* or *VSET* procedure statement.

  If specified, the value of the *gb_init* parameter is the name of a file containing values of one or more guardband parameters.

  Parameters that may be specified in the guardbanding initialization file are:

  - Guardband Method (*gb*)
  - Guardband Factor (*gbf*)
  - Guardband Mode (*gb_mode*)
  - Guardband Result Mode (*gb_result*)
  - Guardband Table (*gb_table*)
  - UUT Confidence (*uconf*)

  The purpose of the *gb_init* parameter is to allow multiple workstations at a site to share the same guardband parameter values. By placing the guardbanding initialization file on a network drive in a location accessible to all MET /CAL workstations, you can configure your system so that the only local guardband parameter data on each workstation is the *gb_init* specification in the MET/CAL initialization file (*metcal.ini*).

  The guardbanding parameters in the guardbanding initialization file must be preceded by the following section header:

  [MET/CAL Guardbanding Initialization]

Guardbanding Initialization File Example

The following example illustrates how to make use of a guardbanding initialization file. Suppose you have three MET/CAL workstations and you wish to share the same guardbanding parameter specifications among the workstations.

Suppose also that your system contains a network drive mapped as "H:" and that "H:" is accessible from each of the three workstations.

To set up a guardbanding initialization file you might create a *metcal* subdirectory on "H:", and in that directory place a file named *gbinit.ini* containing the following lines:

```
[ MET/CAL Guardbanding Initialization ]
gb = rds
```

This example assumes that you choose to specify only the guardbanding method in the guardbanding initialization file. However, you are free to specify one or more parameters from the list supported by the guardbanding initialization file, as shown above.

Note also that the directory and file names used here (*metcal* and *gbinit.ini*) are just examples. You may choose any file name you like, and place it anywhere on a network-accessible drive.

Next, on each of the three MET/CAL workstations, edit the MET/CAL initialization file (*metcal.ini*) and add the line:

```
gb_init = h:\metcal\gbinit.ini
```

You have now configured your system so that all three workstations will use the *rds* guardbanding method whenever guardbanding is enabled. If you wish to change the guardbanding method, you can make the change in a single location (*gbinit.ini*) and all workstations will see the change.

- *gb_mode*

The *gb_mode* parameter allows the table lookup method to be configured. There are two choices:

- *interp*

  Setting *gb_mode* to *interp* causes MET/CAL to linearly interpolate between rows in the guardband table.

- *step*

  Setting *gb_mode* to *step* causes MET/CAL to interpret the guardband table as a step function.

  If the lookup value exactly matches a column 1 value in the in guardband table, the corresponding column 2 guardband table value is used as the guardband factor.

  If the lookup value falls between two rows of the guardband table, the guardband factor value on the first row is used. The guardband table values are internally ordered so that the column 1 values are ascending. Thus, choosing the guardband factor value from the first row is equivalent to choosing the guardband factor value for the row containing the smaller lookup parameter value.

The default value of *gb_mode* is *step*.

gb_mode has no effect unless a table-based guardband method (*tsr*, *tur*, or *ntur*) is used.

- *gb_overflow*

  The *gb_overflow* parameter configures the handling of a guardbanding "overflow" condition when the guardbanding method is *mu*.

  An overflow condition occurs when the calculated uncertainty of the measurement exceeds the tolerance of the test.

  Recall that when the method is *mu* the guardbanded test limits are determined by tightening the specification limits by an amount equal

to the expanded measurement uncertainty. Therefore, if the expanded measurement uncertainty exceeds the test tolerance it is not possible to tighten the test limits by the required amount. This is because a test cannot have a test tolerance less than zero.

There are two situations that, individually or in combination, may cause a guardbanding overflow condition:

- If *nmeas* is greater than 1 and the UUT values vary excessively the standard deviation will be large. A large standard deviation causes a large measurement uncertainty.

- If the reference is inadequate relative to the UUT, the reference accuracy may dominate the calculated measurement uncertainty. In this case the test is not metrologically sound. I.e., the test specification ratio is close to or less than one.

gb_overflow allows the procedure writer or system administrator to configure how MET/CAL handles an overflow condition. There are four choices for the *gb_overflow* value:

- *limit*

  When *gb_overflow* is set to *limit*, an overflow condition causes MET/CAL to set the guardbanded test tolerance to zero. This prevents the test from passing outright, but allows *pass indeterminate*, *fail indeterminate*, and *fail* as possible test results.

- *limit_w*

  limit_w is the same as *limit*, except that a run time warning is generated to notify the operator that a guardbanding overflow condition has occurred.

- *disable*

  When *gb_overflow* is set to *disable*, MET/CAL handles an overflow condition by turning guardbanding off for the test.

- *disable_w*

  disable_w is the same as *disable*, except that a run time warning is generated to notify the operator that a guardband overflow condition has occurred.

The default value of *gb_overflow* is *limit_w*.

In addition to being specified in the procedure in a *VSET* or *TSET* statement, *gb_overflow* may also be specified in the MET/CAL initialization file or in the guardbanding initialization file.

Results Table:

When a guardbanding overflow condition occurs, MET/CAL annotates the guardbanding method name in the database *Results Table* by appending the overflow indicator "*". In other words, the method name in the *Results Table* will be "MU *" instead of "MU". Report writers may make use of the "*" annotation to flag test results which are doubtful due to a guardbanding overflow.

Compatibility:

gb_overflow is not supported in MET/CAL V7.11 SP1 or prior versions. The first released version of MET/CAL which supports *gb_overflow* is V7.20.

- *gb_pts*

  gb_pts is used to control the Post Test Summary window display after a guardbanded test.

  Procedures often include an:

  ASK- F

The effect of this is to cause the Post Test Summary window to be shown after each test with a FAIL result, but not after a test with a PASS result.
When guardbanding is enabled the situation is more complex. The result of a guardbanded test is PASS, PASS INDETERMINATE, FAIL INDETERMINATE, or FAIL. *gb_pts* allows the procedure writer to configure the operation of MET/CAL in this respect.

Legal values for *gb_pts* are:

- *f*

  Setting *gb_pts* to *f* configures MET/CAL so that the Post Test Summary is shown only after a FAIL result. The Post Test Summary is not shown after a PASS, PASS INDETERMINATE, or FAIL INDETERMINATE result.

- *p*

  Setting *gb_pts* to *p* configures MET/CAL so that the Post Test Summary is shown suppressed after a PASS result. The Post Test Summary is shown after a FAIL, FAIL INDETERMINATE, or PASS INDETERMINATE result.

- *pf*

  Setting *gb_pts* to *pf* configures MET/CAL so that the Post Test Summary is suppressed after a PASS or PASS INDETERMINATE result, and displayed after a FAIL or FAIL INDETERMINATE result.

- *result*

  Setting *gb_pts* to *result* causes the *gb_pts* parameter to take the same value as the *gb_result* parameter.

The following table shows how the *gb_pts* value and the test result determine whether the Post Test Summary window is displayed:

	Pass	Pass Indeterminate	Fail Indeterminate	Fail
gb_pts=f	no	no	no	yes
gb_pts=p	no	yes	yes	yes
gb_pts=pf	no	no	yes	yes

gb_pts may be specified in the MET/CAL initialization file or in the guardbanding initialization file, as well as by using *VSET* and/or *TSET* procedure statements.

The default value of *gb_pts* is *result*.

Specifying:

VSET GB_PTS = *

causes the *gb_pts* specification to be as it would be without a procedure specification of *gb_pts*.

gb_pts is not supported in MET/CAL V7.10 and V7.11. You must have V7.11b or later to use the *gb_pts* parameter. In V7.10 and V7.11 the behavior of the software with regard to the Post Test Summary display after a guardbanded test is equivalent to a *gb_pts = p* specification.

gb_pts has no effect on tests to which guardbanding cannot be applied (e.g., EVAL tests, go / no go tests, and operator-terminated tests).

gb_pts also has no effect if the ASK 'A' flag is used to configure the system to always, or never, display the Post Test Summary.

- *gb_result*

  The *gb_result* parameter is used to configure the *guardband procedure result* mode. The guardband procedure result mode determines the overall result of a verification procedure in cases

where the result of one or more individual tests is *indeterminate.* There are three choices:

- *f*

  Setting *gb_result* to *f* indicates that *indeterminate* results are to be treated as *fail* results when the overall procedure result is determined.

- *pf*

  Setting *gb_result* to *pf* indicates that *pass indeterminate* results are to be treated as *pass* results and *fail indeterminate* results are to be treated as *fail* results when the overall procedure result is determined.

- *p*

  Setting *gb_result* to *p* indicates that *indeterminate* results are to be treated as *pass* results when the overall procedure result is determined.

When guardbanding is enabled, there are three possible outcomes for each individual verification test: *pass*, *fail*, or *indeterminate*.

To illustrate these concepts consider a test performed at 100 V, with a UUT specification of ± 1%. The lower specification limit is 99 V and the upper specification limit is 101 V. When guardbanding is not enabled a UUT value between 99 V and 101 V will be considered a *pass*. If the UUT value is less than 99 V or greater than 101 V, the test result will be *fail*.

Now suppose that guardbanding is enabled, and the guardband method is *mu*. Suppose further, as part of this example, that the calculated value of the expanded measurement uncertainty is 0.1 V.

The region within ± 0.1 V of each specification limit is now considered an *indeterminate* region. If the UUT value falls within an *indeterminate* region it not possible to state with certainty whether the test result is *pass* or *fail*.

Continuing with the example, if the UUT result is between 98.9 V and 99.1 V, or between 100.9 V and 101.1 V, the result is *indeterminate*.

Indeterminate results may be further divided into *pass indeterminate* and *fail indeterminate*. A *pass indeterminate* result is an *indeterminate* result that is still within the specification limit. A *fail indeterminate* result is an *indeterminate* result that is not within the specification limit.

Continuing with the example, if the UUT value is 99.05 V, the test result is *pass indeterminate* because 99.05 V is within the specification limit (99 V), but still within the *indeterminate* region (98.9 V to 99.1 V). On the other hand, if the UUT value is 98.95 V the result is *fail indeterminate* because 98.95 is below the lower specification limit (99 V), but still within the indeterminate region (98.9 V to 99.1 V).

With this explanation of *indeterminate*, *pass indeterminate*, and *fail indeterminate* in mind it is easier to understand the purpose of the *gb_result* parameter.

When *gb_result* is *p* (pass), all *indeterminate* results are regarded as *pass* results when the overall procedure result is determined. For example, suppose you run a procedure that contains 10 individual verification tests. If five test results are *pass* and five test results are *indeterminate* the overall procedure result will be *pass*.

Conversely, when *gb_result* is *f* (fail), all *indeterminate* results are regarded as *fail* results when the overall procedure result is determined. For example, suppose you run a procedure that contains 10 individual verification tests. If five test results are *pass* and five test results are *indeterminate* the overall procedure result will be *fail*.

Setting *gb_result* to *pf* (pass/fail) means that *pass indeterminate* results are taken to be *pass* results for the purpose of determining the overall procedure result and *fail indeterminate* results are taken to be *fail* results for the purpose of determining the overall procedure result. In other words, setting *gb_result* to *pf* is equivalent to instructing MET/CAL to use the specification limits, not the guardband limits, for

purposes of determining the overall procedure result. When *gb*_result is *pf*, you will still be able to determine whether the result of an individual test is *indeterminate*, but the overall procedure result will be just as it would have been without guardbanding.

The *gb*_result default value of is *pf*.

- *gb_table*

  The parameter *gb_table* is used to specify the name of the guardband table. The guardband table is required when the guardband method (parameter *gb*) is set to *tsr*, *tur*, or *ntur*. The guardband table is not used with the other guardband methods.

  The guardband table is a simple text file consisting of two columns. The first column is the lookup parameter value. The second column is the associated guardband factor.

  Comment lines may be included in the guardband table file. Any line which begins with a '#' or a ';' is a comment line.

  Here is a simple example that illustrates how to create a guardband table file. Suppose you would like to apply an 80% guardband if the test specification ratio (TSR) is between 3 and 4, and a 60% guardband if the TSR is between 2 and 3. To accomplish this you would create a guardband table file like this:

  ```
  #TSR   GBF
  2.0    0.6
  3.0    0.8
  4.0    1.0
  ```

  Spaces and tabs are ignored. You may use any text editor (*WordPad*, *Notepad*, etc.) to create the file.

  Any name may be used for the file. For example, you could create three different files, *GuardbandTable1.txt*, *GuardbandTable2.txt*, and *GuardbandTable3.txt*, and place them in a directory of your choosing. To switch between them it is necessary to modify the value of the

gb_table parameter. *gb_table* may be specified in the MET/CAL initialization file (*metcal.ini*), a guardbanding initialization file, or at the procedure level in a *VSET* or *TSET* statement (see **Compatibility** note below).

The same guardband table file can be used with different guardband methods. The first column in the example above is labeled "TSR", but that is simply a comment. If you change the guardband method to *tur*, for example, then the first column specifies the lookup values for the test uncertainty ratio (TUR), not the test specification ratio (TSR).

If the value of the lookup parameter is greater than the column one value in the last row of the guardband table, MET/CAL uses the *GBF* given in the last row. For example, using the sample table shown above, if the *TSR* is 20.0, the *GBF* will be 1.0, because 1.0 is the *GBF* value (i.e., the column 2 value) specified on the last row of the table. If the value of the lookup parameter is less than the column one value in the first row of the guardband table, MET/CAL uses the *GBF* given in the first row. For example, again using the sample table shown above, if the TSR is 1.0, the *GBF* will be 0.6, because 0.6 is the *GBF* value from the first row. It is important for the metrologist and/or procedure writer to carefully consider what should happen when the lookup parameter (*TSR*, *TUR*, or *NTUR*) is a small value. Either the guardband table should contain rows at the beginning to cover those cases, or procedures should be carefully reviewed to ensure that such cases (e.g., very low *TURs*) do not occur.

Compatibility:

> Prior to MET/CAL V7.20v it was not possible to specify *gb_table* in a *VSET* or *TSET* procedure statement. A *gb_table* specification could occur only in the MET/CAL initialization file or the guardbanding initialization file. Beginning with V7.20v, *VSET* and *TSET* support the *gb_table* parameter.

- *gb_tol_err*

The *gb_tol_err* parameter determines how MET/CAL calculates the *Tolerance Error* when guardbanding is enabled.

The *Tolerance Error* is the error of a test expressed as a percentage of the *Test Tolerance*.
The *Tolerance Error* is:

- Displayed in the *Post Test Summary* window after each numeric evaluation. The value is on the line labeled *Error % Tol*.
- Shown in the *Test Results* window in the *%Tol* column.
- Written to the database *Results Table* in the column labeled *tol_err*.

gb_tol_err allows the procedure writer or system administrator to configure whether the *Tolerance Error* calculation is based on the specified limits for the test or on the guardbanded pass limits.

There are two choices for the *gb_tol_err* value:

- *spec*

  When *gb_tol_err* is set to *spec* MET/CAL uses the limits based on the UUT's specified tolerance as the basis for the *Tolerance Error* calculation.

- *gb_pass*

  When *gb_tol_err* is set to *gb_pass* MET/CAL uses the guardbanded pass limits as the basis for the *Tolerance Error* calculation.

The default value of *gb_tol_err* is *spec*.

In addition to being able to be specified in the procedure in a *VSET* or *TSET* statement, *gb_tol_err* may also be specified in the MET/CAL initialization file or in the guardbanding initialization file.

Example:

Consider a test where:

gb_tol_err = spec

Reference Value = 10.0 V
UUT Value = 10.1 V
Test Tolerance = 2 %
Expanded Uncertainty = 0.05 V
Guardbanding Method = MU

Then:

Upper Spec Limit = 10.2 V
Guardbanded Upper Limit = 10.15 V
Tolerance Error = 50 %

The *Tolerance Error* value of 50 % comes from the fact that the measurement error (0.1 V) is 50 % of the UUT's specified tolerance (0.2 V).

If *gb_tol_err* is changed to *gb_pass* in the example:

gb_tol_err = gb_pass
Reference Value = 10.0 V
UUT Value = 10.1 V
Test Tolerance = 2 %
Expanded Uncertainty = 0.05 V
Guardbanding Method = MU

Then:

Upper Spec Limit = 10.2 V
Guardbanded Upper Limit = 10.15 V
Tolerance Error = 67 %

The *Tolerance Error* value of 67 % comes from the fact that the measurement error (0.1 V) is 67 % of the guardbanded test tolerance (0.15 V).

Compatibility:

gb_tol_err is not supported in MET/CAL V7.20 or prior versions. Versions of MET/CAL that support guardbanding, up to and including V7.20s, calculate *Tolerance Error* based on

the guardbanded pass limits. This is equivalent to *gb_tol_err=gb_pass*, but without the ability to configure the setting.

- *uconf*

  The *uconf* parameter is used to specify the confidence associated with the UUT's specification. *uconf* is used only when the guardband method is *ntur* (normalized test uncertainty ratio). The specified *uconf* value is used to normalize the UUT specification to a 1-sigma value (standard uncertainty). In MET/CAL V7.10 the *uconf* parameter has no other use than this. (It is not used in the measurement uncertainty calculation.)

  Legal values for *uconf* are:

  - *68.27%*
    1s
    1sigma
  - *90%*
    90.0%
    90.00%
  - *95%*
    95.0%
    95.00%
  - *95.45%*
    2s
    2sigma
  - *99%*
    99.0%
    99.00%
  - *99.73%*
    3s
    3sigma

  The values within each of the bullets above are equivalent forms.

  In some cases it may be difficult to determine the confidence associated with the UUT's specification. MET/CAL uses a default value of *2s* (i.e., 2 σ).

Post Test Window

When guardbanding is enabled MET/CAL's *Post Test* window appends the annotation *(GB)* after a *Pass* or *Fail* test result indicator on the first line of the window. If the result of the test is *Pass Indeterminate* MET/CAL displays *Pass Indeterm* on the first line. In this case it is not necessary to append the *(GB)* annotation because *Pass Indeterm* by itself is sufficient to indicate to the operator that guardbanding is in effect. Similarly, when the result is *Fail Indeterminate* MET/CAL displays *Fail Indeterm* on the first line of the *Post Test* window.

Test Results Window

The MET/CAL Test Results window indicates *Pass Indeterminate* and *Fail Indeterminate* results by using a different color.

When guardbanding is enabled, *Marginal Pass* results are no longer distinguished from other results.

The following table shows the color indications in the Test Results window when guardbanding is enabled:

RESULT	**COLOR**
Pass	Green
Indeterminate Pass	Light Green
Indeterminate Fail	Light Red
Fail	Red
RSLT & Brace Text	Cyan

When guardbanding is not enabled the color mapping is as follows:

RESULT	COLOR
Pass	Green
Marginal Pass	Yellow
Fail	Red
RSLT & Brace Text	Cyan

Terminology: TSR, TUR, TAR

Since its inception in 1988 MET/CAL has used the term *Test Uncertainty Ratio* (*TUR*) to refer to the ratio:

$$\frac{\text{UUT Specification}}{\text{Reference Accuracy}}$$

However, this ratio does not, strictly speaking, involve the total measurement uncertainty. The measurement uncertainty includes the reference accuracy as a component, but typically includes other uncertainty components as well. In many cases, the measurement uncertainty includes an uncertainty component based on the standard deviation of a sequence of measurement values, and also includes a component based on the resolution of the UUT. The on-line *VSET* help file describes MET/CAL's measurement uncertainty calculation in detail.

Therefore, except for the fact that the ratio shown above has traditionally been referred to as the *TUR*, it would be better to refer to the ratio above as the *Test Specification Ratio* (*TSR*). Other vendors and writers have often used the term *Test Accuracy Ratio* (*TAR*) to refer to the Test Specification Ratio.

This terminology confusion has become more troublesome with the introduction of guardbanding support, because MET/CAL now supports guardband methods based both on the *Test Specification Ratio* as well as on the actual *Test Uncertainty Ratio*.

The terminology used in this document, and reflected in the values allowed for the *gb* parameter, therefore differs from the terminology used other

places in MET/CAL. Outside of guardbanding, when MET/CAL uses the term *TUR* it refers to the *Test Specification Ration* (*TSR*). In the guardbanding documentation and parameter values, however, *TSR* is used for *Test Specification Ratio*, and *TUR* refers to the ratio:

$$\frac{\text{UUT Specification}}{\text{Expanded Measurment Uncertainty}}$$

Similarly, the *Normalized Test Uncertainty Ratio* (*NTUR*) refers to the ratio:

$$\frac{\text{Normalized UUT Specification}}{\text{Normalized Measurement Uncertainty}}$$

Is Guardbanding Enabled?

In some cases it may be difficult to determine whether guardbanding is enabled for a particular test or set of tests. The reason that guardbanding may sometimes not be enabled even though the procedure writer intended to enable it, is that it is possible to enable guardbanding, but fail to provide all the required information.

An example of failing to provide required information would be to choose the *Measurement Uncertainty* guardband method (i.e., set *gb* to *mu*), but then fail to set *nmeas* to a value greater than zero to enable the measurement uncertainty calculation.

The best way to determine if a test has been guardbanded is to look at the data in the database *Results Table.* If the row corresponding to the test in question includes values for the *Guardband Method*, and the guardbanded lower and upper test limits, then guardbanding was enabled when the test was run. The *Results Table* column headers are *guardband_meth*, *guardband_ll*, and *guardband_ul*, respectively.

The *Post Test* window also indicates whether guardbanding is enabled. (However, many procedures are written so that the *Post Test* window is not shown when the test result is *Pass*, or, in some cases, not shown at all.) It is also important to be aware that *Post Test* will indicate that guardbanding is in effect even when the guardband factor (*GBF*) is exactly 1.0. Technically, guardbanding is, in fact, in effect, but the guardband limits are the same as the specification limits. (This can be seen by looking at the data in the *Results Table*.)

The *Test Results* window also indicates guardbanded tests when the test result is *Indeterminate Pass* or *Indeterminate Fail*. However, when the test result is *Pass* or *Fail* the *Test Results* window does not indicate whether guardbanding is in effect for the test.

UUT Test Tolerances in the Procedure

MET/CAL V7.10 is designed to allow guardbanding to be enabled without any, or very little, modification of the MET/CAL procedure source code. The UUT test tolerances entered into the procedure remain the standard specification limits defined by the UUT's verification procedure.

The guardbanded test limits cannot be determined by examining the procedure source files. The guardbanded limits are shown in the result data generated by procedure execution.

Changing Guardbanding Parameters During Procedure Execution

As with all *VSET* parameters, the MET/CAL procedure language allows the procedure writer to modify the value of a parameter at any point during procedure execution.

One may, for example, change the guardband method during the course of procedure execution by changing the value of the *gb* parameter.

This capability should be used with care, however. In general, the best policy is to choose a specific guardbanding approach, and apply it consistently.

If your policy is to guardband tests only when the TSR falls below a specified threshold, that can be accomplished by setting up a suitable guardband table, where the guardband factor is 1.0 for all tests with a TSR above the threshold. It is not necessary to specifically enable and disable guardbanding on a per-test basis within the procedure.

One parameter that you should generally not change in mid-stream during procedure execution is *gb_result*. *gb_result* controls how MET/CAL determines the overall procedure result, based on the mix of individual test results (some *Pass*, some *Indeterminate*, and some *Fail*). As an example of what not to do, consider what would happen if, for the first half of a procedure, you set *gb_result* to *p*, and then, for the second half of the procedure, you changed *gb_result* to *f*. If you did that you would be telling

MET/CAL to handle *indeterminate* results in the first half of the procedure as *Pass* results, but to handle *indeterminate* results in the second half of the procedure as *Fail* results. This is not illegal, and MET/CAL will do exactly what it was instructed to do, but it is probably difficult to justify this from a metrological point of view.

Results Table

The database *Results Table* contains three columns that store guardbanding data.

- *Guardbanded Lower Limit*

  The *Guardbanded Lower Limit* is the lower limit of the test based on the specified guardband method.

  The column header in the database table is *guardband_ll*.

- *Guardbanded Upper Limit*

  The *Guardbanded Upper Limit* is the upper limit of the test based on the specified guardband method.

  The column header in the database table is *guardband_ul*.

- *Guardband Method*

  The *Guardband Method* is a string field, which indicates the specified method. If the field is blank, guardbanding was not used for the test. Otherwise, the field indicates the method using the same string values used for the *gb* parameter:

Column Value	Guardband Method
direct	Direct
mu	Measurement Uncertainty
ntur	Normalized Test Uncertainty Ratio
rds	Root Difference Square
tsr	Test Specification Ratio
tur	Test Uncertainty Ratio

In addition to the three columns described above, the *Test Status* result value also indicates, for some tests, that guardbanding was in effect. The *Results Table* column header for this result quantity is *test_status*. *Test Status* is a string field, which will be one of the values: *Pass, Pass Indeterminate, Fail,* or *Fail Indeterminate.*

Legacy Results

Guardbanding does not affect "legacy results". Thus, there is no indication in the *CalResults Table* in the database of whether guardbanding was in effect, or of whether a particular test result was *Pass Indeterminate* or *Fail Indeterminate*.

Example

This example illustrates how to write a procedure in which the *rds* guardband method is used.

Consider the following procedure:

```
Fluke                                                MET/CAL Procedure
================================================================
INSTRUMENT:             guardbanding rds test
DATE:                   December 07, 2004
AUTHOR:                 Fluke
REVISION:
ADJUSTMENT THRESHOLD:   70%
```

```
NUMBER OF TESTS:       1
NUMBER OF LINES:       18
CONFIGURATION:         Fluke 5500A
================================================================
 STEP    FSC    RANGE NOMINAL  TOLERANCE MOD1      MOD2  3  4 CON
  1.001  ASK+           K
  1.002  VSET         gb = rds; nmeas = 2
  1.003  5500         1.00000000V             0.0002U         2W
```

The *VSET* statement (1.002) that sets *gb* to *rds* establishes the guardband method. Since the *rds* method depends on the measurement uncertainty calculation, the *VSET* statement also assigns a value greater than zero to the *nmeas* parameter. This turns on the measurement uncertainty calculation. Notice that the *ASK* FSC 'K' flag is set. For stimulus-type FSCs (like *5500*) the 'K' flag must be set when measurement uncertainty is enabled.

To try this example, use the *Run Time* application to execute the procedure shown above. Run the procedure in *demo mode*. It's not necessary to have an actual Fluke 5500A, although you will have to make sure that a 5500A is configured on the workstation. (Choose a UUT asset number reserved for testing.)

Since *nmeas* is set to 2, MET/CAL will prompt twice for the UUT readings. Enter "1" in response to both prompts. When the *Post Run* window appears save the results and close the *Run Time* application.

Next, use MET/TRACK to view the *Results Table* data for the sample calibration performed above. The table should contain data values like these:

```
expanded uncertainty  = 3.4883721E-5 V
lower limit           = 0.9998 V
guardband lower limit = 0.999803066 V
guardband upper limit = 1.000196934 V
upper limit           = 1.0002 V
```

To understand these numbers, recall that using the *rds* method entails calculating the guardbanded test tolerance as:

$$\sqrt{tol^2 - emu^2}$$

where *tol* is the UUT specification and *emu* is the expanded measurement uncertainty.

Plugging the numbers shown above into this formula produces:

$$\sqrt{0.0002^2 - 0.000034883721^2}$$

$$= \sqrt{0.00000004 - 0.000000001216873990 80}$$

$$= 0.00019693431902337388$$

In other words, the original UUT test specification is ± 200 μV.

With guardbanding in effect, using the RDS method, the test tolerance is tightened to approximately ±196 μV.

(The expanded uncertainty value used in this example is based on the 90-day specs of the 5500A. If you use the 1-year specs (for example), the calculated value of the expanded uncertainty may change, and it will be necessary to adjust the numbers in this example accordingly.)