Updated: Oct. 10, 2012

Doxygen Rules

This document lays out the procedures and rules for properly documenting your code using Doxygen.

Basic use of Doxygen Tags

The Doxygen comment block goes under the MODULE/DECK line in GAMESS, but before the "SUBROUTINE" line.

All tags should begin with the comment sign "C" or "!" followed by a greater-than sign, ">". This document uses the Fortran 90 convention of "!". All tags begin with "@", so the line would look like:

!*MODULE … DECK …
!> @brief …
!> @*tagname ...*
SUBROUTINE …()

There will be an example code at the end illustrating the proper use of the tags.

Required Tag for ALL code

This tag *must* be included for **all** modified code, even a small change in an existing subroutine including bug fixes.

- **@brief**

Required Tags for all code except bug fixes; recommended for bug fixes

The following tags should be used if you have additional information about the subroutine being modified. If you do not know anything other than a brief description of the subroutine, the @brief tag is all that is needed.

- **@author**

- **@details**
- **@param, if you added a new argument to a subroutine, otherwise this is not required; it is, however, highly recommended**

Required Tags for all new code

If a subroutine was not in GAMESS until this submission, the following must be included in addition to the above rules.

- **@param**

Optional Tags

This is a short list of some commonly used additional tags that are not required, but encouraged when applicable. You are welcome to use tags not included on this list.

- **@note**
- **@warning**
- **@todo**
- **@bug**
- **@see**
- **@date**

Descriptions of the Tags

Here are the descriptions of the tags along with examples of use. All tags should begin with the comment sign “C” or “!” followed by a greater-than sign, “>”. This document uses the Fortran 90 convention of “!”. All tags begin with “@”, so the line would look like:

!> *@tagname*

The term “user” refers to anyone writing code that uses your code, not the end user of GAMESS.

@brief

Provides a brief description of the subroutine. The text following this tag should only be one sentence long. Any further information should go in the details section.

@author

Take credit for your work! If you made significant contributions to a subroutine, put your name along with the date and a brief description of the contribution made. Bug fixes should use the “date” tag discussed below. If you know the original authors’ names, put them at the top of the “author” tag. If you don’t know the original author’s name, ask Mark or Mike if they know. The format should be:

!> @author *Original Author*
!> @author *Your Name*
!> - *date- comment about change*

For example, say a subroutine was written by John Doe in July of 2010, and edited by Jane Doe in September of 2012. Then, John Doe made an additional change in October of 2012. This would be the example comment block:

!> @author John Doe
!> - July, 2010- Subroutine written
!> - October, 2012- Added additional loop to account for a larger variety of cases.
!> @author Jane Doe
!> - September, 2012- Cleaned up excess code to make the routine more general.

@date

This should be used to account for bug fixes or for changes not significant enough to warrant an “author” addition. The dates given should be in order, with the earliest being at the top. The format for the “date” tag is as follows:

!> @date *Month, Year Author’s Name*
!> - *Changes made*

For example, if John Doe made a small bug fix with the indexing variable in October of 2012, the comment block would look like this:

!> @date October, 2012
!> - Bug fix with indexing variable

@details

For additional information the “details” tag should be used. This tag should be used for any information that would be useful for a user of the subroutine. It can be as long as needed and can contain relevant information such as paper references, discussion about the algorithm, background information, etc. If following other tags, especially “@brief”, an additional line should be used to separate “@details” from “@brief” like so:

```
!> @brief This subroutine changes the coordinates from Z-Matrix to Cartesian.
!>
!> @details Since GAMESS uses Cartesian coordinates for most of the calculations, there is a
!> need for the conversion from Z-Matrix to Cartesian coordinates. This routine is based on
!> Model-Builder written by Mark Gordon.
```

@param

If you are editing a subroutine that was written by someone else, the “@param” tags are not required, but strongly recommended if you know what the arguments mean. Even if you only know the meaning of a sub-set of the arguments, include them with an explanation. If you add a new argument to a subroutine, you must include this tag. The format for this tag is

!> @param *argument_1 Description of argument_1*
!> @param *argument_2 Description of argument_2*
…

For example, if you had a subroutine with 4 arguments:

SUBROUTINE DISPLACE(COORDS,X_DISPLACE,Y_DISPLACE,Z_DISPLACE)

The resulting comment block would be:

```
!> @param COORDS Input coordinates
!> @param X_DISPLACE Displacement in the X direction
!> @param Y_DISPLACE Displacement in the Y direction
!> @param Z_DISPLACE Displacement in the Z direction
```

@note

The “@note” tag should be used to highlight information for the user. This could be anything you would want the user to be aware of. For example, you could remind users of a particular way you intended the code to be used. The syntax is:

!> @note *Your note*

For example, if you made a routine that should only be called at the beginning when the input file is being read, this would be the comment block:

!> @note Should only be called when the input is being read in.

@warning

Should be used when what you have to say is of more importance than a simple "@note" statement. A good use of the "@warning" tag would be: If the contents of the tag are not followed, a bug could occur. The syntax is the same as @note.

@todo

Whenever there is additional work to be done on a subroutine, the "@todo" tag should be used. The "@todo" tag will place the line into a special list that includes all of the todo statements in one place. This tag is helpful when you need to release a subroutine with basic functionality, but you have additional functionality in mind. For example:

!> @todo Make this routine run in parallel.

@bug

Any known, but not easily fixable bugs should be documented with this tag. However, this tag should not be used as a substitute for debugging or error reporting. This tag should let users know that under certain conditions a subtle bug may appear, so they can keep that in mind when they use this routine. Just like "@todo" this line is placed in a special list.

@see

This tag places a "see also" section in the documentation page. It is useful for linking similar parts of the code, so users won't have to search for it themselves. Using this tag saves people time and can help you by preventing code duplication.

Example code

Here is an example subroutine that uses many of the tags previously discussed. This will give you an idea of how Doxygen can be incorporated into GAMESS.

Updated: Oct. 10, 2012

```
!*MODULE MATH DECK MULTIPLIER
!> @brief Multiplies two arrays and returns the result
!>
!> @detail Takes in 2 3x3 arrays and multiplies
!> by multiplying the rows of Array1 with
!> the columns of Array2
!>
!> @author Random Grad
!> - September, 1985
!> @date January, 2012- Another Grad
!> - Bug Fix
!> @date October, 2012- John Doe
!> - Different bug fix
!> @param Array1 First array
!> @param Array2 Second array
!> @param Out Returned array
!> @todo write the rest of the subroutine
Subroutine Multiplier(Array1,Array2,Out)
   Implicit none
   Double Precision, Intent(in) :: Array1(3,3),Array2(3,3)
   Double Precision, Intent(out) :: Out(3,3)
…
```