



GCT General Usage Issues

Contents

- [Adding your own User-Attributes](#)
- [Creating a New NON_ST_BODY \("C"\) Function Block](#)
- [Creating a New Project](#)
- [Handling Multiple GCT Installations](#)

Adding your own User-Attributes

Do the following to add your own user-attributes:

- In the LIBRARY directory under the GCT ROOT directory, create a new directory of the name of the attribute library you wish it to have
- In your new attribute library directory, create a file called LIBRARY.ST
- Edit the LIBRARY.ST file. Add ATTRIBUTE at the top of the file, and END_ATTRIBUTE at the end of the file
- Add your own attributes between the ATTRIBUTE and END_ATTRIBUTE lines as follows: yourattributename : DATA-TYPE
- Edit the DEFAULT.ODB file located in the OPTION directory under the GCT ROOT directory. If no AttributeTypes section exists, add one. Then add each attribute listed in your attribute library, including a short description, a list of data types that that attribute will be available for, its data type, and a default valid. Separate each attribute line with the \ character. See example below

Below is a sample attribute file:

```
ATTRIBUTE
Description : STRING;
RefName : STRING;
DisplayRes : STRING;
TimeUnits : STRING;
END_ATTRIBUTE
```

Below is a sample DEFAULT.ODB file:

```
AttributeTypes:= \
    Description      {Desc := "Text Description",      IdType := {ANY_TIME, ANY_NUM, ANY_BOOL}
    RefName          {Desc := "c reference name",      IdType := {ANY_TIME, ANY_NUM, ANY_BOOL}
    DisplayRes       {Desc := "Display Resolution",    IdType := ANY_REAL}: STRING
    TimeUnits        {Desc := "Time Units",            IdType := ANY_TIME}: STRING
```

Creating a New NON_ST_BODY ("C") Function Block

A NON_ST_BODY function block is one whose execute method is defined directly in "C" instead of "ST". Its INPUT, OUTPUT, IN_OUT and INPUT_OUTPUT variable declarations are still defined in "ST", however internal variables can be defined either in "ST" or "C", or both. **Great care must be taken when creating a NON_ST_BODY function block, as no protection against the user causing fatal application errors is provided.**

To create a new NON_ST_BODY function block, perform the following:

- Select the project view for the project you wish to create the new NON_ST_BODY function block in
- Press the "NEW" button to create a new block
- Select "FUNCTION BLOCK"
- Choose the preferred declarations editor (ST Editor or FBD Editor)
- ALWAYS select ST Editor for the Body Editor
- Specify an appropriate function block name
- Press the "OK" button
- In the Body Editor, entry the following single line: "NON_ST_BODY"
- In the Declarations Editor, add your INPUT, OUTPUT, IN_OUT and INPUT_OUTPUT variables (this can be done at a later date or

In the DECLARATIONS Editor, add your NON_ST, ST, NON_ST and ST variables (this can be done at a later date or updated as required)

You have now created the outline of a NON_ST_BODY function block.

Next, one needs to create the execute and optional cold-start methods for the function block (the execute method contains the function blocks algorithm and is executed each time the function block runs - the cold-start method is run once to initialise any function blocks data)

- `source.c` - contains the execute and optional cold-start methods
- `source.h` - optional file containing definitions required by `source.c` - `source.c` file must explicitly include this file
- `source.dh` - optional file containing additional data declarations to be included in the function blocks data structure - automatically included by the function blocks `exec.h` inside the function block's data structure declaration
- `source.dht` - optional file containing additional data type declarations - automatically included by the function block's `exec.h` before the function block's data structure declaration

source.c

This file must contain the following execute method declaration (this contains the function blocks algorithm and is executed each time the function block runs):

```
void XXXX_execute(XXXX_data_t * z_datap)
```

where XXXX represents the function blocks name.

If a cold-start method is required, the following declaration is required (this contains the cold-start method that is run once to initialise any function blocks data):

```
void XXXX_cold_start(XXXX_data_t * z_datap)
```

where XXXX represents the function blocks name.

If the function block contains non-vol data, then the methods are as follows:

```
void XXXX_execute(XXXX_data_t * z_datap, XXX_nvdata_t * z_dataq)
```

and

```
void XXXX_cold_start(XXXX_data_t * z_datap, XXX_nvdata_t * z_dataq)
```

source.h

This file contains any macros or header information that is required only by the `source.c` file.

source.dh

Any additional data declarations that cannot be declared in ST can be added in `source.dh`. For example, `char` or `char *` need to be added here.

source.dht

Any additional data type declarations that are not defined as standard need to be added in `source.dht`. For example, Windows types should be added here by including the appropriate Windows header file.

Creating a New Project

To create a new project, go to the GCT Project/Library manager and press the "NEW" button. Enter your new project's name, ensuring that it is eight (8) or less characters long (otherwise certain build target compilers will be unable to operate), and then press the "OK" button. You have now created a new empty project.

At this point you may wish to add or change the list of libraries available to your new project. To do this, press the "LIBRARIES" button. This gives you a list of already available libraries. Note the reverse order, where the bottom library is loaded first, and the top library is loaded last - this affects the inter-library dependency. The "GCTATTR" library MUST always be loaded first (bottom).

Use the "ADD", "REMOVE", up and down buttons to manipulate the list of libraries.

Handling Multiple GCT Installations

Only one installation of GCT can be running at a time on a given computer. Attempting to launch a second instance of GCT will eventually lead to failure and possible corruption of function block sources and configuration files.

The installation of GCT that is used then running GCT is determined by the "GCT.INI" file located in the Windows directory. This contains the following information:

```
[DIRS]
GCTROOT=C:\etherm\gct
```

where C:\etherm\gct represents the GCT root directory.

Changing GCTROOT to point to a different valid GCT root directory causes GCT to operate on that configuration with all its libraries and projects.

Therefore, by careful changing of the "GCT.INI" file, one can handle multiple GCT installations.



Last Modified: Monday, 25-Nov-96 16:32:14 GMT

Site Maintained By Adrian Oliver