RAM SIZE REPORT

The RAM Size report, which shows the sizes of the various application components, is a diagnostic tool that helps identify components with excessive RAM sizes. This report is derived from C files that are produced by the Code Generator, and it therefore involves operations that are not required for the production of other Rapid reports.

Production of the RAM Size report is a 3-step process:

1. Generating code to produce the data files.
2. Compiling and linking a project that contains the relevant C file to generate the report-producing, executable file.
3. Running the executable file to produce the report.

These steps are explained in detail in the following sections.

Step 1: Generating the Data Files

In addition to information from the standard *.h* files produced during code generation, the RAM Size report also uses files that must be especially generated for this purpose. These files are produced during code generation when the Create Size Report Files check box is selected (in the Code Generation Status dialog box), and are placed in the Size folder in the code generation source output folder.

Step 2: Creating the Executable File

The RAM Size report uses some of the standard files generated by the Code Generator and other files generated specifically for this report.

To create the executable file:

1. Create a project that contains the file named *RAM_report_<appname>.c* from the Size folder in the code generation source output directory. The Size folder, as well as the files in it, is created by the Code Generator when the "Create Size Report Files" check box is selected.

   Include paths to the following files:

   - The *.h* files from the code generation source output directory.
   - The *.h* files from the Rapid codegen directory.
   - The *.h* files from the Size folder directory.

❖ *NOTE: When a component in a RapidPLUS project is generated as "Stand-alone application" or "Empty task," you have to create a separate project for each of these components, because they are not a part of the main application. The Size folder will contain a .c file for each of these separate components. For example, if your Rapid application contains the user object "empty.udo", which is marked as "Empty Task", then the Size folder will contain an additional file named "RAP_report_tempty.c".*

2. (Optional) If your target system supports output to files, and you want to store the RAM Size report in a text file, add the following line in the User Code section at the beginning of the *RAM_report_<appname>.c* file:

```
/******** RapidUserCode BEGIN HEADER_RAM_report_szapp.c ********/
#define STORE_RAM_REPORT_IN_FILE
/******** RapidUserCode END   HEADER_RAM_report_szapp.c ********/
```

   (You need to perform this step in the first code generation session. This code is stored even if you repeat code generation after modification to the Rapid application.)

3. Compile and link the project.

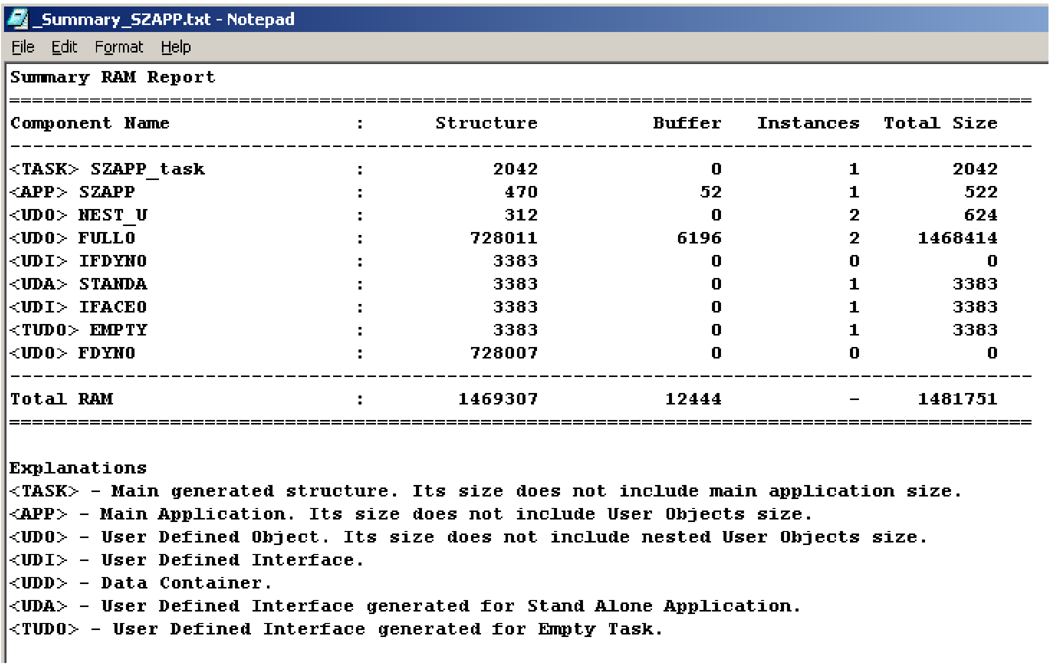
❖ *NOTE: The executable will yield the most accurate report when it is created with the target compiler and linker. If the target system is not capable of printing the report, you can use DOS (for 16-bit targets) or Windows (for 32-bit targets). This may slightly distort the reported sizes, but will not affect their relative values.*

Step 3: Producing the Report

To produce the report, simply run the executable file. The output is displayed on the console and two text files are created:

- *_Summary_<appname>.txt*: presents the RAM size of each component as well as the total RAM used.
- *_Detail_<appname>.txt:* presents RAM size information about each component's C structures.

A sample of the summary report



```
Summary RAM Report
=========================================================================================
Component Name                :     Structure          Buffer   Instances   Total Size
-----------------------------------------------------------------------------------------
<TASK> SZAPP_task             :          2042               0           1         2042
<APP> SZAPP                   :           470              52           1          522
<UDO> NEST_U                  :           312               0           2          624
<UDO> FULLO                   :        728011            6196           2      1468414
<UDI> IFDYNO                  :          3383               0           0            0
<UDA> STANDA                  :          3383               0           1         3383
<UDI> IFACEO                  :          3383               0           1         3383
<TUDO> EMPTY                  :          3383               0           1         3383
<UDO> FDYNO                   :        728007               0           0            0
-----------------------------------------------------------------------------------------
Total RAM                     :       1469307           12444           -      1481751
=========================================================================================

Explanations
<TASK> - Main generated structure. Its size does not include main application size.
<APP> - Main Application. Its size does not include User Objects size.
<UDO> - User Defined Object. Its size does not include nested User Objects size.
<UDI> - User Defined Interface.
<UDD> - Data Container.
<UDA> - User Defined Interface generated for Stand Alone Application.
<TUDO> - User Defined Interface generated for Empty Task.
```

Report Legend

Component Name presents the name of each component and its code generation type.

Structure indicates the amount of RAM used by each component.

Buffer indicates the amount of RAM allocated statically outside the structure.

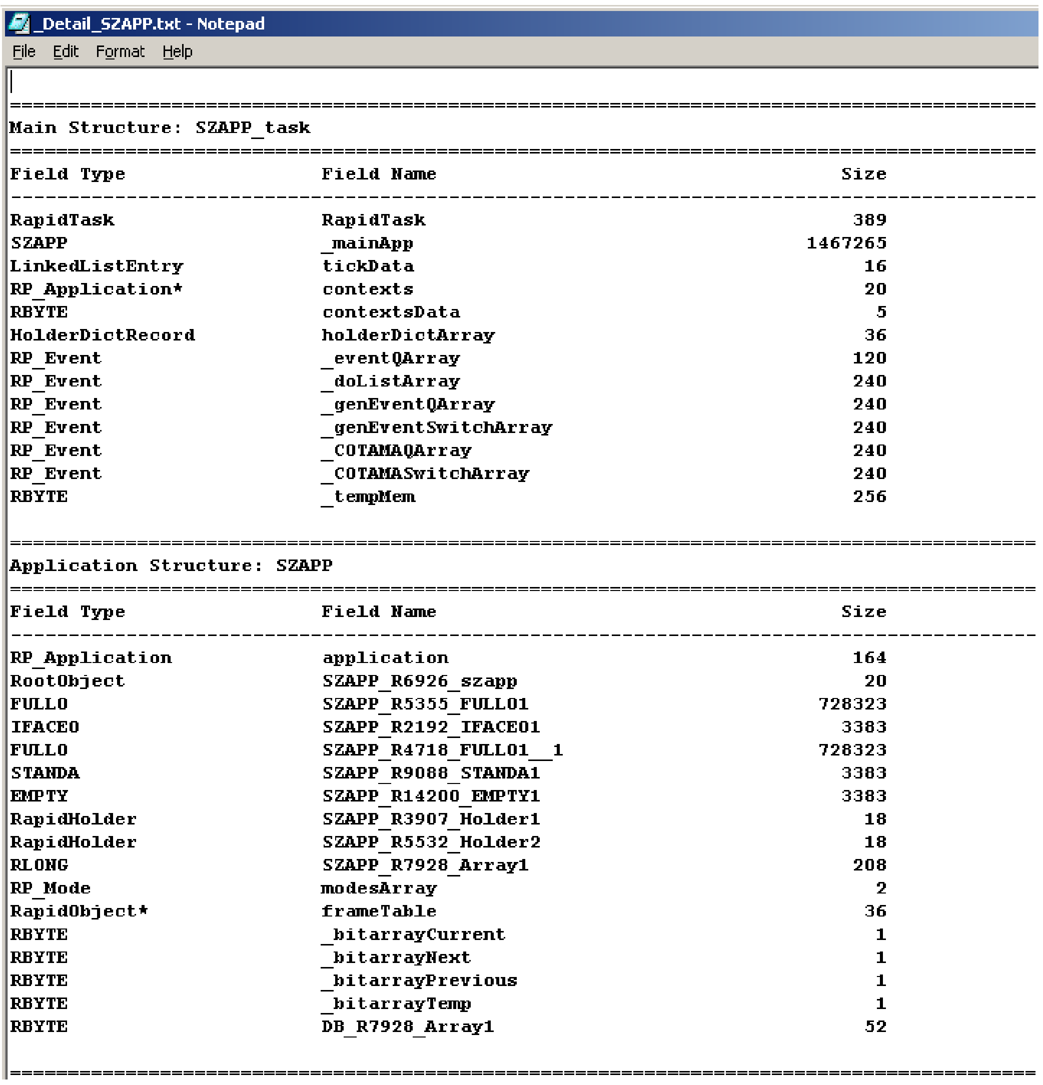
❖ *NOTE: If you have not checked the option "Support compilers with structure size limitation" in the Code Generation Preferences dialog box, Miscellany page, then the item "Buffer" does not appear in the report because the externally allocated memory buffers become a part of the component's structure.*

Instances indicates the number of statically allocated instances of the component.

❖ *NOTE: If the number of statically allocated instances of a user object is 0, its size is still included in the RAM Size report, but it does not influence the total RAM size. It may be useful to estimate additional RAM size when using dynamic memory allocation.*

Total Size equals (Structure + Buffer) × Instances.

A sample of the detailed report



```
==========================================================================================
Main Structure: SZAPP_task
==========================================================================================
Field Type                  Field Name                                   Size
------------------------------------------------------------------------------------------
RapidTask                   RapidTask                                     389
SZAPP                       _mainApp                                  1467265
LinkedListEntry             tickData                                       16
RP_Application*             contexts                                       20
RBYTE                       contextsData                                    5
HolderDictRecord            holderDictArray                                36
RP_Event                    _eventQArray                                  120
RP_Event                    _doListArray                                  240
RP_Event                    _genEventQArray                               240
RP_Event                    _genEventSwitchArray                          240
RP_Event                    _COTAMAQArray                                 240
RP_Event                    _COTAMASwitchArray                            240
RBYTE                       _tempMem                                      256

==========================================================================================
Application Structure: SZAPP
==========================================================================================
Field Type                  Field Name                                   Size
------------------------------------------------------------------------------------------
RP_Application              application                                   164
RootObject                  SZAPP_R6926_szapp                              20
FULL0                       SZAPP_R5355_FULL01                         728323
IFACE0                      SZAPP_R2192_IFACE01                          3383
FULL0                       SZAPP_R4718_FULL01__1                      728323
STANDA                      SZAPP_R9088_STANDA1                          3383
EMPTY                       SZAPP_R14200_EMPTY1                          3383
RapidHolder                 SZAPP_R3907_Holder1                            18
RapidHolder                 SZAPP_R5532_Holder2                            18
RLONG                       SZAPP_R7928_Array1                            208
RP_Mode                     modesArray                                      2
RapidObject*                frameTable                                     36
RBYTE                       _bitarrayCurrent                                1
RBYTE                       _bitarrayNext                                   1
RBYTE                       _bitarrayPrevious                               1
RBYTE                       _bitarrayTemp                                   1
RBYTE                       DB_R7928_Array1                                52

==========================================================================================
```

Report Legend

Field Type presents the types of structure fields.

Field Name presents names of each structure field.

Size indicates the RAM size of each field.