Profiling Support Tools (V1.3)

The time profiling tool relies on a program called GetRegions.exe which reads the program's executable image (.axf extension, ELF) file and extracts each function's name, starting code address, and code size. GetRegions can support an RTOS or bare-metal system, based on the symbol USING_RTOS defined in profile.h. With this information it can create a C file (region.c) with profiling support:

- A constant table (RegionTable) containing name, starting and ending addresses for each function's code.
- A table (RegionCount) to track the access count for each region.
- An initialized constant variable NumProfileRegions.

GetRegions can instead create a list of all functions and their size in bytes, allowing the user to determine which functions use the most code space as a first step in code size optimization.

Limitations: GetRegions will generate incorrect symbol name information for ELF files which are large enough to use indirect symbol name information.

Windows Installation

No installation is needed. Simply run GetRegions.exe with the proper command line arguments.

Command Line Options

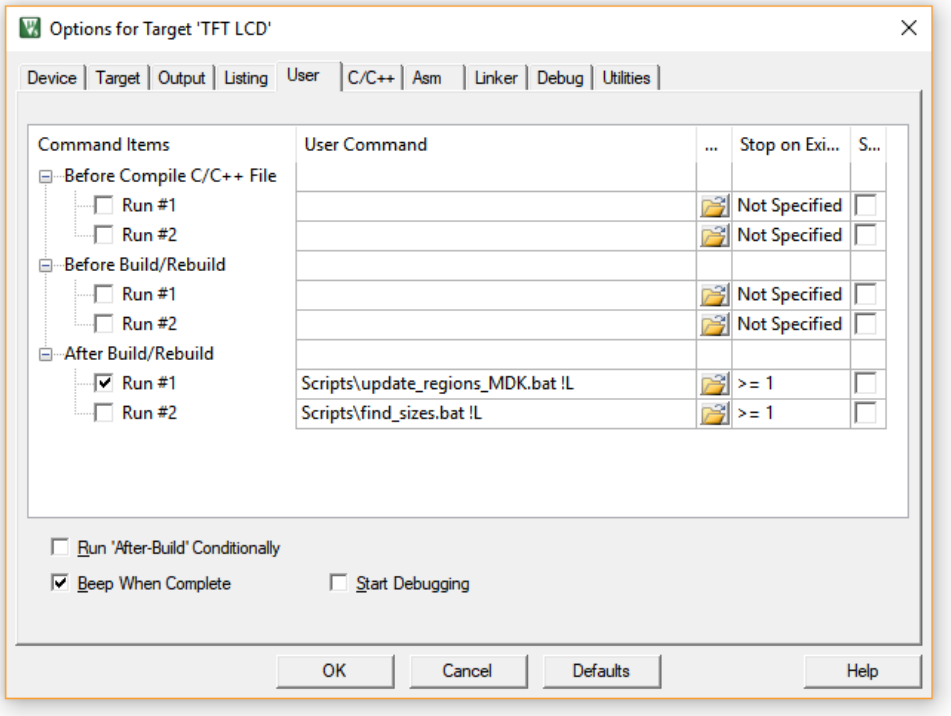
```
Usage : GetRegions filename [options]
  -c Generate C code with region information for time profiling
  -s Generate C code with data structure to support sorting regions by time
  -xprefix Exclude all functions beginning with prefix. Multiple OK: -x_ -xos
  -z Generate text with region size information
  -o Output file name
  -d Print debug information
```

Time Profiling Procedure

Use one of the two following procedures:

- Manual procedure
  - Build your program in µVision.
  - Run **update_regions_manually.bat** (in the Scripts folder) to create a new region.c file (which is placed in the src folder). This file will have the correct number of entries but the addresses may be wrong (this is ok – a later pass will fix this).
  - Rebuild your program in µVision to generate an executable with the correct size region table. **Note that this table still uses the old function addresses, which will be wrong if the previous region table had a different number of entries.**
  - Run **update_regions_manually.bat** to create a new region.c file with the correct function addresses and the correct number of entries.
  - Rebuild your program in µVision to generate an executable with the correct region addresses.
- Automated procedure

- Change the project options. As shown in the figure below, select the User tab and under "Run User Programs After Build/Rebuild" check box "Run #1" and add the command **Scripts\update_regions_MDK.bat** !L to the text box. Click OK to save these changes. (You can optionally add the command shown below to "Run #2" to find function sizes as well.)



- Now every time you build your program the regions will be updated.
- To ensure the region information is correct, you may need to build the project up to three times if you have changed the source code significantly (e.g. both the number and size of the functions).

- You can download and execute your program on the target hardware. This will populate the RegionCount table to indicate the execution time profile.

Warnings and Troubleshooting

- You must regenerate region.c and rebuild your executable every time you change your program in a way that might change functions sizes or locations.
- There is a chicken-and-egg problem: in order to create region.c, we need the map file. But the linker will not create the map file if region.c does not exist. So the solution is to copy the region.c file from this distribution into your project. The linker will be able to build the executable, which GetRegions can analyze to create the correct region.c.
- If µVision complains about not being able to reload region.c, try closing and reopening the region.c window.
- Large programs may use so many functions that the region table becomes too large. To reduce the region table size, tell GetRegions to exclude functions with names beginning with specified prefixes. Do this by passing the prefixes as command line arguments in the update regions bat files.
  - `GetRegions.exe ..\Objects\TFT-TS-BL-Profiler-RTX-CMSIS.axf -c -s -xos -x_ -xOS -xsvc -xMessageQueue -o..\Source\Profiler\region.c`

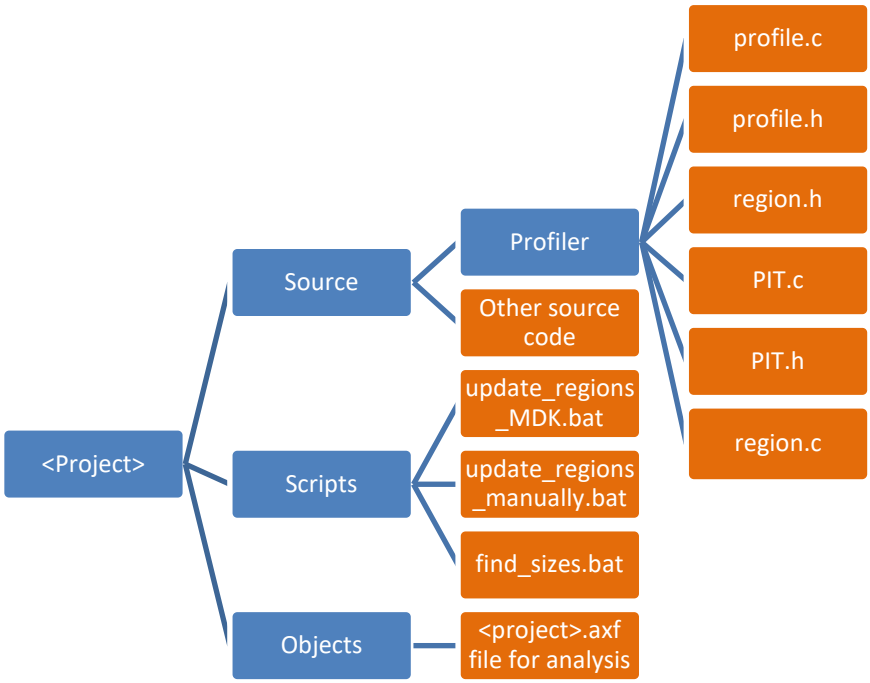
Code Space Profiling Procedure

To find the functions which use the largest amount of code space, use this process:

- Build your program in µVision.
- Run **find_sizes.bat** (in the Scripts folder) which will create **sorted_function_sizes.txt** in the obj folder. This is a list of functions sorted by decreasing size.
- You can automate this step by adding it to the project build user options in µVision.

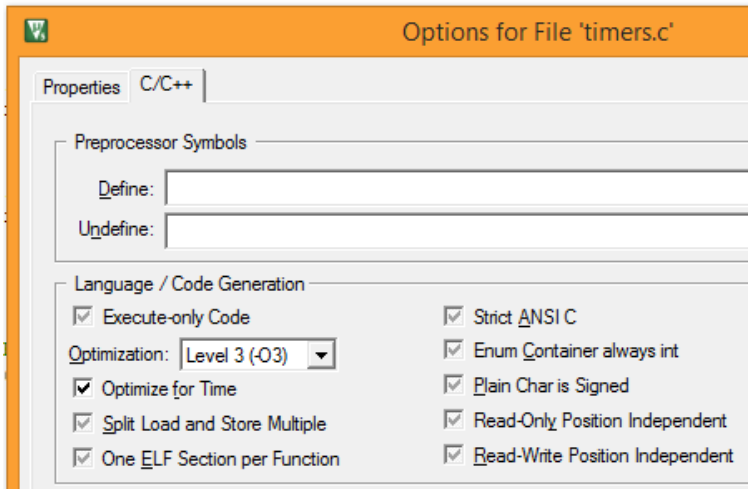
Adding Profiling Support to a Project

- The following instructions assume this directory structure for your MDK ARM project:



- Copy the bat files in Scripts into a new folder called Scripts in your project folder. The scripts are written expecting the .axf file to be located in a folder called Objects, which is at the same level as the Scripts folder.
    - You may need to modify the scripts to match your folder structure and file locations. For example, set PATH to include the location of your GetRegions.exe file.
- Copy these files from Profiler directory to the source code directory in your project
    - profile.c
    - region.c
    - PIT.c
- Copy these files from Profiler directory to the include file directory in your project
    - profile.h
    - region.h
    - PIT.h
- Edit profile.h to indicate whether the project uses an RTOS or not, which affects how the return addresses are determined. Comment out the definition **#define USING_RTOS** if the project does not use an RTOS.
- Add these files to your project in MDK:

- region.c
- PIT.c and profile.c – Note that you need to override default compiler optimization settings for these files so they will always be compiled with -O3 optimization for time. Do this by right clicking on the PIT.c or profile.c entry in the Project window of MDK and selecting "Options for File …".
  Then select the C/C++ tab, set it as shown below (but for PIT.c and profile.c), and select OK.



Configuring Profiling Support

Configure the profiler in the profile.h file:

- Sampling frequency
- RTOS or bare-metal mode
- LCD support
- Serial port printf support

Rebuilding GetRegions.exe

You can modify and recompile GetRegions.exe. Source code and documentation are included in the ToolSource directory. This program is compiled with Microsoft Visual C++ 2010 Express.

Changes from V1.2

- Switched from Cygwin to Visual Studio
- Added support for scheduler-based code, using PSP.