CCStudio 3.1 Migration and Key Improvements

Last Modified: 3 April 2008

Agenda

- Migration Issues for CCS 3.1
- Key Fixes/Improvements in CCS 3.1

Migration Issues

Path/Name Changes

- CSL no longer integrated with BIOS (CCS 3.1+ and 2.40)
  - CSL resides in separate folder $(Install_dir)\<ISA>\csl
- C67x mthlib
  - fastrts67x.lib renamed to fastmath67x.lib
  - Location moved to $(Install_dir)\C6000\cgtools\lib
- Profile Based Compilation (PBC) renamed to CodeSizeTune (CST)

Changes to Debug Format

- STABS is deprecated in favor of DWARF
  - Both formats fully supported by CCS 3.1 for debug
- Output skeletal DWARF by default
  - No impact on optimization
  - No longer need **`-gp`** for non-BP based profiling

Old (CCS 2.3-)	**New** (CCS 2.4+)	**Comment**
Default	**`--symdebug:none`**	**No debug info**
`-g, -gt`	**`--symdebug:coff`**	**STABS format**
`-gp`	**`--symdebug:profile_coff`**	**STABS profiling**
`-gw`	**`-g, --symdebug:dwarf`**	**DWARF format**
N/A	**`--profile:breakpt`**	**DWARF profiling (BP based)**
N/A	**Default**	**DWARF profiling (non-BP based)**
N/A	**Default, `--symdebug:skeletal`**	**Skeletal DWARF**

- Note default and **`-g`** act differently
- **`-gt, -gw, -gp`** act the same, but are deprecated

Object Format vs. Debug Format

- COFF and ELF are object file formats
- STABS and DWARF are debug information formats
- Unfortunately, sometimes the term COFF is used to refer to STABS debug format
- Object file formats
  - All current tools support COFF
  - Future tools will support ELF
- Debug file formats
  - Old tools supported only STABS
  - Recent tools support STABS (default) and DWARF
  - New tools support DWARF (default) and STABS (deprecated)
  - Future tools will support DWARF only

Target Connectivity

- CCS 2.40+ supports connect/disconnect
  - CCS starts up disconnected to the target by default
    - CCS still calls GEL **`StartUp()`** function when CCS is launched
      - Any behavior that attempt to access the target in GEL **`StartUp()`** will fail
      - Use **`OnTargetConnect()`** for target access initialization

CCS 3.1: OnTargetConnect()

- New built in GEL callback **'OnTargetConnect()'**
  - CCS 3.1: Acceptable to place all target access initialization steps that was done in **StartUp()** to here if desired

```
// DM642EVM.gel for CCS 2.21

StartUp()
{
    setup_memory_map();
    GEL_Reset();
    init_emif();
}
```

2.21 to 3.1

```
// DM642EVM.gel for CCS 3.1

StartUp()
{
    setup_memory_map();
}

OnTargetConnect()
{
    // place target initialization steps here
    GEL_Reset();
    init_emif();
}
```

- More information:
  - App Note SPRAA74: Creating Device Initialization GEL Files

CCS 2.40/3.0: `OnTargetConnect()`

- **`OnTargetConnect()`** behavior differs between CCS 3.1 and CCS 2.40/3.0

- CCS 2.40/3.0 **`OnTargetConnect()`**
  - Called *before* “re-removal” of the debug state on a target connect
  - Used for bare minimal target initialization to get the target in a good state
  - Built-in target access GEL functions not allowed
    - Interferes with the process of re-removal of the debug state
    - Exception: **`GEL_Reset()`** and **`GEL_IsInRealtimeMode()`**

- CCS 3.1 **`OnTargetConnect()`**
  - Called *after* the “re-removal” of the debug state on a target connect
  - No target access restrictions as was the case in 2.40/3.0
  - Undocumented callback **`OnTargetInit()`** is now called where **`OnTargetConnect()`** used to be called

Keyboard Shortcuts

- Default setting for number of the debug related keyboard shortcuts changed to be more consistent and intuitive.
  - Ex: Setting probe points (Ctrl-F9 to F8), stepping (Step Into: F8 to F11), etc...

- Option to change back if via the Customize->Keyboard dialog

ARM9TDMI simulator

- ARM9TDMI simulator configurations have been removed
  - Were not suitable for debugging purposes (SDSsq39154 , SDSsq40103, etc…)
- To use these simulators for regression purposes it is possible to create a configuration use the ‘Create Board’ option in CCSetup
  - From the Create Board tab select TI Simulator as your connection
  - Add a TMS470R2x processor (where tisim_arm9tdmi.dvr is the driver being used).

Key Fixes/Improvements in CCS 3.1

Robustness

- Editor
  - CodeWright Integration in CCS 2.40
    - Addresses many issues reported against the old CCS integrated editor
    - CodeWright's CodeSense feature more stable that CCS's CodeMaestro
  - Major performance improvements made to CodeSense in 3.1 from 2.40
- Watch Window
  - Improved robustness and functionality

CCS 2.2x Profiler Issues

- *“Sorting within profiler window via (exclusive) columns appears to be screwed up.. i.e. can't sort correctly for current column”*
- *“Closing the window does not save (or remind to save) the profile data. This could result in losing data that took hours to collect.”*
- *“It isn't clear when the profiler can be turned off and on. If you set a breakpoint set near the entry of a function and turn on the profiler, it is possible to miss the fact that the function was called and get very screwed up profile results.”*
- *“It is possible to get cycle counts for a function A that are larger than the inclusive cycle counts for the *only* function B that calls function A.”*
- *“Profiles are clearly wrong... does not account for cycles within subfunctions as part of inclusive total”*
- *“Profile function should profile selected functions.”*

Issues Addressed with New Profiler

- *"Sorting within profiler window via (exclusive) columns appears to be screwed up.. i.e. can't sort correctly for current column"*
  - This bug is addressed with the new profiler
- *"Closing the window does not save (or remind to save) the profile data. This could result in losing data that took hours to collect."*
  - With the new profiler, data collected is preserved when the Profile Viewer is closed and then opened. The only time the data can be inadvertently lost is when the Profiler is still enabled and the application is run again.
- *"It isn't clear when the profiler can be turned off and on. If you set a breakpoint set near the entry of a function and turn on the profiler, it is possible to miss the fact that the function was called and get very screwed up profile results."*
  - The new profiler handles these issues better… i.e. - the profiler still needs to know when a function is entered so if profiling is enabled after you are in a function, it won't know you are in the function. However it will simply not collect any profile data for the function until you re-enter the function. In the old profiler, the behavior was unpredictable… sometimes collecting garbage data that can screw up profile results as mentioned.
- The new profiler has made great improvements in accuracy/stability will help address issues such as ones reported below:
  - *"It is possible to get cycle counts for a function A that are larger than the inclusive cycle counts for the *only* function B that calls function A."*
  - *"Profiles are clearly wrong... does not account for cycles within subfunctions as part of inclusive total"*
  - *"Profile function should profile selected functions."*

Key Improvements with New Profiler

- Utilizes fast simulation features (C55x, C64x)
  - Uses “fast points” (instead of standard BP’s) which do not halt the target
    - 10x-100x improvement in profiling speed
  - Uses special counters provided by simulator (doesn’t use CLK/ctools)
    - Messing with CLK does not screw up profiling.
- ~1.5x speed improvements on HW and old simulators
- Better accuracy
  - Takes better advantage of DWARF debug symbol information
    - DWARF is the default debug symbols generated for CCS 3.1 CGT
    - DWARF contains function exit point information
      - CCS is less reliant on driver
    - Smarter breakpoint/fast-point placement
      - Accuracy was sometimes compromised because of poor placement of BP’s
    - Can profile optimized code (w/o **–gp** option)
      - Default behavior will have minimal information needed for profiling but no impact on optimization (with fast simulators)
      - Need to use –profile:breakpt for BP based profiling
  - “*Calculation algorithms have been greatly improved*”
  - Takes in consideration of places where some cycle counts might be missing and compensates
- Easier to setup profiling of loops
- Better usability

DWARF Overlay Debug Support

- Key OFS fix for DWARF overlay debug support:

Improved ARM GCC Debug Support

- Key improvements for better support of ARM GCC generated output

CCS 3.1 Release Notes

- Check the CCS 3.1 Release Notes for a full list of all fixes, general notes, known issues and workarounds