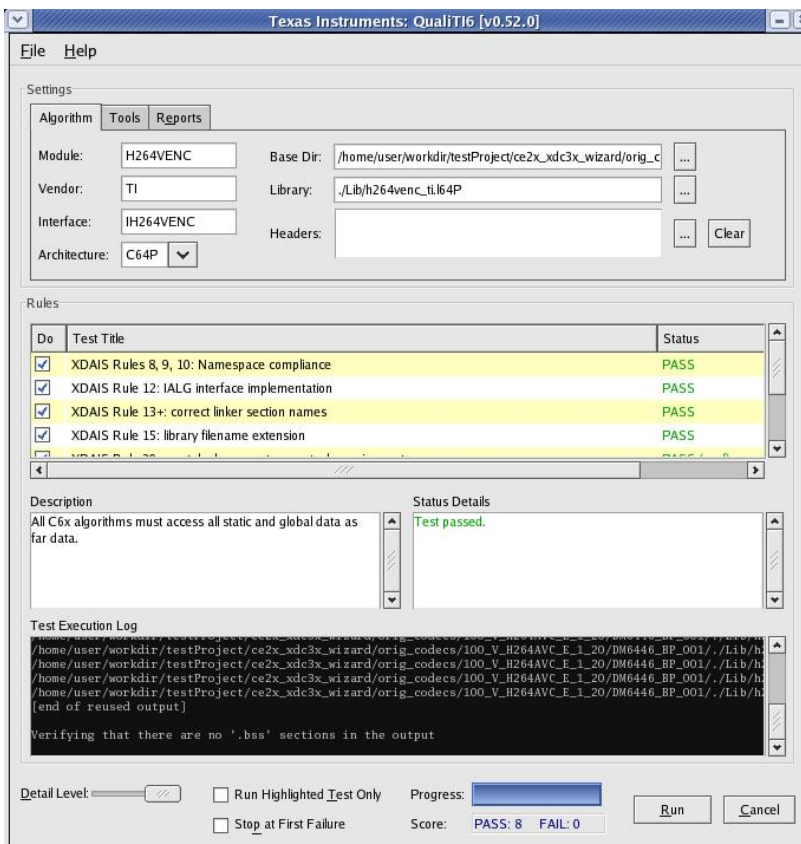# QualiTI

QualiTI Introduction

QualiTI is a tool from TI to verify the XDAIS compliance of a certain piece of code, e.g. a codec library. The tool was first published with XDAIS 6.00 and improvements are made with each XDAIS release [1]. Currently, the tool supports C64x+ (64P), C674, and ARM platforms. An older, alternative approach for ARM codec developers non-QualiTI-based approaches is still available.

It is designed as an optional but recommended helper for porting existing free form codec libraries (e.g. open source ones) to the more encapsulated XDAIS or XDM design for making that code correctly plug into any XDAIS-based framework (e.g. Codec Engine).

• Sample QualiTI XDAIS-complaince check screenshot:

At present the QualiTI compliance tool only tests a subset of the XDAIS rule-set - however it does check most of the typical problems that cause integration issues. For example: -

• flags global symbols that do not meet the XDAIS namespace requirements and could clash with other algorithms
• checks for poorly named sections which could cause code or data to be mis-categorized and linked to inconsistent memory sections causing crashes.
• checks for the appropriate IALG, IMODULE, IDMAx (if implemented), IRES (if implemented) interface symbols.

For each rule tested it will emit **PASS** or **FAIL**. For each failure, it will indicate precisely what failed, possible reasons for the failure and suggested ways to fix it.

Some details on the tests performed follow: -

Details on tests performed

Rules 8, 9, 10: Namespace compliance

These rules are bundled together simply because we run the same tool to check compliance for all 3.

The rules themselves are: -

- (Rule 8) All external definitions must be either API identifiers or API and vendor prefixed.
- (Rule 9) All undefined references must refer either to the operations specified in Appendix B (a subset of C runtime support library functions and a subset of the DSP/BIOS HWI API functions), or TI's DSPLIB or IMGLIB functions or other XDAIS-compliant modules.
- (Rule 10) All modules must follow the eXpressDSP naming conventions for those external declarations disclosed to the client.

What does *namespace compliance* mean?

Well, if we run the symbol listing tool `nm6x` on an algorithm as follows: -

```
      C:\CCStudio_v3.3\C6000\cgtools\bin\nm6x.exe -g fir_ti.l64P
```

This yields an output something like: -

```
ARCHIVE:  fir_ti.l64P
00000000 T _FIR_TI_exit
00000000 T _FIR_TI_init
00000064 T _FIR_TI_filter
00000000 T _dummy1
00000000 U _memcpy
00000000 T _FIR_TI_activate
00000000 T _FIR_TI_alloc
00000000 T _FIR_TI_control
...
```

Anything with `FIR_TI_` prefix is good, `_memcpy` is ok since it comes from the allowed RTS functions list in Appendix list of spru352. `_dummy1` is bad - 'T' represents 'text' so there's a global function without a decent namespace. This is not *namespace compliant*.

How to fix this?

- See Kbase article 51141 [2] and 36239 [3]
- Or, manually fix symbols that dont have `_MOD_VEN` prefix

QualiTI tells you this too.

Rule 12: IALG interface implementation

- All algorithms must implement the IALG interface

Here we check for the presence of required XDAIS function-table interface entry points. XDAIS requires the following 2 symbols to be exposed: -

- `MOD_VEN_IALG` - the interface to the list of IALG functions
- `MOD_VEN_IMOD` - the interface to the list of IMOD functions which extend the IALG functions

Internally again QualiTI uses `nm6x` to check for the existence of such symbols

Absence of either of these symbols is non-compliance - there should be no characters after the `_IALG` or `_IMOD` characters

How to fix this?

- read spru352 & spru360 - they show you how to create an alg with the required interface entry points
- cut/paste/modify a decent 'template' alg like FIR_TI in the Reference Frameworks [4]

QualiTI tells you this too.

Rule 13+: correct linker section names

- Each of the IALG methods implemented by an algorithm must be independently relocatable

This rule requirs that each IALG function be in its own section - this enables clients to link/place them anywhere they want - as opposed to lumping them in with `.text`

We need a different tool than nm6x for this - we need something that shows the list of sections in the archive.

QualiTI therefore leverages the Codegen XML Perl Scripts Utility package [5]. More specifically it runs `sectti.exe`

This is the same utility that the XDC Codec Package Wizard [5] uses.

It produces something like: -

```
Reading from stdin ...
Library,Filename,Section,Type,Size
fir_ti.l64P,fir_ti.o64,.text:init,CODE,32
fir_ti.l64P,fir_ti.o64,.text:exit,CODE,32
fir_ti.l64P,fir_ti_filter.o64,.text,CODE,32
fir_ti.l64P,fir_ti_filter.o64,.cinit,DATA,24
fir_ti.l64P,fir_ti_filter.o64,.tables,UDATA,16
fir_ti.l64P,fir_ti_filter.o64,.text:filter,CODE,224
fir_ti.l64P,fir_ti_ialg.o64,.text:algNumAlloc,CODE,32
fir_ti.l64P,fir_ti_ialg.o64,.text:algMoved,CODE,32
fir_ti.l64P,fir_ti_ialg.o64,.text:algInit,CODE,64
fir_ti.l64P,fir_ti_ialg.o64,.text:algAlloc,CODE,128
fir_ti.l64P,fir_ti_ialg.o64,.text:algFree,CODE,64
fir_ti.l64P,fir_ti_ialg.o64,.text:algDeactivate,CODE,32
fir_ti.l64P,fir_ti_ialg.o64,.text:algControl,CODE,64
fir_ti.l64P,fir_ti_ialg.o64,.text:algActivate,CODE,32
fir_ti.l64P,fir_ti_vt.o64,.cinit,DATA,48
fir_ti.l64P,fir_ti_vt.o64,.far,UDATA,40
```

Observe: -

- list of sections differentiated by code, const data or uninitialized data
- Subsections for each of the IALG functions are indeed present so we are in compliance.

However: -

- NOTE that we see an entry

```
fir_ti.l64P,fir_ti_filter.o64,.tables,UDATA,16
```

This is **bad**. Absent the sectti script we wouldn't know whether this was code, uninitialized data or const data. Hence the linker may arbitrarily place this in the wrong type of memory (eg L1PSRAM) if not explicitly placed - this is a frequent source of crashes.

QualiTI therefore emits a FAIL if sections found are not subsections of the main compiler sections (.text, .far, .const etc). Since this is an **addition** to Rule 13 we call this *Rule13+*.

How to fix this?

- Add #pragma CODE_SECTIONS for each of your IALG fxns
- Rename your sections to be subsections of main compiler sections e.g. `.const:tables`
- See this example partial link linker command file to find out how to expose the IALG sections Rule 13 needs to see.

Rule 15: library filename & filetype

- Each eXpressDSP-compliant algorithm must be packaged in an archive which has a name

that follows a uniform naming convention

Pretty straightforward - your archive name format needs to match Rule 15 e.g. `fir_ti.l64P` is ok but `myLib.lib` is not

It also has to be an archive - not a partial link or executable.

How to fix this?

- Rename your lib to match the rule

Rule 20: must declare worst-case stack requirements

- Rule 20 - All algorithms must characterize their worst-case stack space memory requirements (including alignment).

This rule is ever more important to specify accurately now because Frameworks like Codec Engine have methods to specify stack size and the BIOS TSK stacks are allocated on this basis - so if you specify a value too low in your XDAIS algorithm documentation your Codec Engine application will crash.

QualiTI uses the `call_graph.exe` script to determine worst-case stack usage of the algorithm. It trawls the algorithm's DWARF information to find the cumulative worst-case function and reports it.

NOTE - worst-case stack calculation is dependent on a number of factors due to TI Codegen limitations - if you get a warning from running this script on your alg, *please* read the documentation of the call_graph.exe (docs\ofd\call_graph.htm) script to determine the cause. A common example is if you build your codec with `-ml3`. This causes all calls to be far calls - some deficiencies in the generated assembly from Codegen then make it impossible to differentiate far calls .v. indirect calls .v....hence the stack size number reported is wrong. **The right way to fix this is not to use -ml3**. Far calls were made redundant 3 years ago when linker trampolines became available. Full details on these limitations can be found here.

NOTE - Building your algorithm with -g (debug info) also causes problems in calculating the worst-case stack usage at present. The background to this can be found here. This is typically not a major problem since -g inhibits optimization thus most production libraries do not use -g.

Rules 21, 22: must characterize static data & program memory requirements

- Rule 21 - Algorithms must characterize their static data memory requirements.
- Rule 22 - All algorithms must characterize their program memory requirements.

For rules 21 & 22 the static data & program memory requirements, `sectti.exe` is again used, this time to get the footprint. The output is folded into the final HTML report.

Rule 25: All C6x algorithms must be supplied in little-endian format

Again this can be determined statically. The information is in each of the object files.

If any file in the archive is not little-endian, this rule will fail.

Rule 26: All static/global data must be far on c6x

There should be no `.bss` in your algorithm if its for c6x - it is a nightmare for ROMing and also has some total-application-size limits since .bss must be < 32Kb (on c6x).

How to fix this?

- Find the offending data and make it far or const
- Or, rebuild with

```
--mem_model:data=far
```

How to run the tool?

The README.txt in packages\ti\xdais\qualiti\ shows how to run the tool. In a nutshell its a RTSC [6] package, so if you know how to run one XDCScript product, you know how to run them all. QualiTI supplies: -

- startqti.bat : batch file that runs the app (Windows)
- startqti.sh : shell script that runs the app (Linux)

Edit startqti and modify the path to match your XDC installation directory.

Once you have setup the environment and specified the MODULE, VENDOR, INTERFACE, and tool-settings, the script will run the tests and generate an HTML report, presenting a brief summary on standard output.

Requirements

The requirements to run this tool are:

XDAIS	XDC Tools	TI cgtools	TI CG XML tools [7]
6.10 to 6.22.01	3.00.05 or higher	Yes*	1.20 or higher
6.23 or higher	3.05 or higher	No	2.10 or higher

* TI codegen tools (on Windows they are typically found in C:\CCStudio_<version>\C6000\cgtools)

Note that more info on the CG XML product is available here [7]

Feature Additions

- XDAIS 6.10
  - initial support for C64P devices
- XDAIS 6.22.01
  - adds support for C674 devices
  - adds -g option for launching the QualiTI GUI with an input .qti file
- XDAIS 6.23
  - adds support for Arm devices

Download

QualiTI is distributed with the XDAIS product here [1]. Newer versions have more features, so it's recommended to download the most recent release.

See Also

- XDAIS API Reference [8]
- XDAIS Rules and Guidelines [9]
- Top 10 Recommendations for codec producers
- XDAIS download site [1]

References

[1] http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/xdais/index.html

[2] http://www-k.ext.ti.com/srvs/cgi-bin/webcgi.exe?Company={5761bcd8-11f5-4e08-84e0-8167176a4ed9},kb=dsp,case=51141,new

[3] http://www-k.ext.ti.com/SRVS/CGI-BIN/WEBCGI.EXE/,/?St=167,E=0000000000001197558,K=94,Sxi=17,Case=obj(36239)

[4] https://www-a.ti.com/downloads/sds_support/targetcontent/RF/index.html

[5] https://www-a.ti.com/downloads/sds_support/applications_packages/index.htm

[6] http://wiki.eclipse.org/DSDP/RTSC

[7] http://processors.wiki.ti.com/index.php/Code_Generation_Tools_XML_Processing_Scripts

[8] http://www-s.ti.com/sc/techlit/spru360

[9] http://www-s.ti.com/sc/techlit/spru352

Article Sources and Contributors

QualiTI *Source*: http://processors.wiki.ti.com/index.php?oldid=49814 *Contributors*: Alanc, BradGriffis, ChrisRing, DanRinkes, DavidMeixner, Jcusano, Markg, Xyvonned

Image Sources, Licenses and Contributors

Image:qualiti_ScreenShot043.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:Qualiti_ScreenShot043.jpg *License*: unknown *Contributors*: Alanc, Alexander.stohr, Cem8101, Dvnwlms

License

License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

1. Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
2. Each time You Distribute or Publicly Perform an Adaptation, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.
3. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
4. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
5. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.
6. The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.