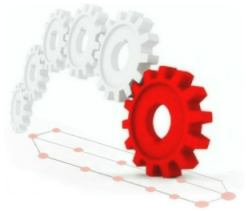
SYS/BIOS with GCC (CortexM)



The SYS/BIOS 6.33.00.19 product release, combined with the XDCtools 3.23.00.32 release, introduces support for building SYS/BIOS and SYS/BIOS applications with the GNU Compiler Collection (GCC) toolchain. This page provides information on: the initially supported target devices, required build options, the build and debug flow, and limitations.

A walkthrough of installation, build, and debug of a few example applications is included.

Contents

Supported devices

All Stellaris and Tiva Cortex M3 and Cortex M4-based devices are supported.

Compile options

SYS/BIOS and the XDC runtime are pre-built with the following GCC compiler options. To successfully link to the pre-built libraries in these products your application builds must use these same compile options.

For M3:

```
-mcpu=cortex-m3     : Cortex M3 CPU
```

For M4:

```
-mcpu=cortex-m4     : Cortex M4 CPU
-msoft-float        : soft floating point support
```

For M4F:

```
-mcpu=cortex-m4     : Cortex M4 CPU
-mfloat-abi=hard    : hardware floating point support
-mfpu=fpv4-sp-d16
```

For M3, M4 and M4F:

```
-mthumb             : Thumb 2 code only
-mabi=aapcs         : ARM Architecture Procedure Call Standard ABI
```

Linker options

The following linker options are used by default when building a SYS/BIOS application:

For M3:

```
-mthumb -march=armv7-m
```

For M4:

```
-mthumb -march=armv7e-m
```

For M4F:

```
-mthumb -march=armv7e-m -mcpu=cortex-m4 -mfloat-abi=hard
-mfpu=fpv4-sp-d16
```

For M3, M4 and M4F:

```
-nostartfiles                          : don't link to the default compiler startup libraries
-static                                : generate fully bound executables, i.e., no shared libraries
-Wl,--gc-sections                      : remove unused code/data sections to reduce app footprint
-Wl,-T,<users-linker-cmd-file>         : user's linker command file
-Wl,-T,<path-to-cfg-dir>/linker.cmd    : SYS/BIOS linker command file
```

NOTE: The assumption is that the makefile uses the gcc driver to invoke the linker and does not directly invoke the linker. The above options may need to be tweaked a little if the linker is invoked directly.

Development flow

SYS/BIOS supports both command-line and Code Composer Studio (CCS) based build flows. If you are using CCS, you can import a Stellaris/Tiva project from TI Resource explorer (Open "View"->"TI Resource Explorer"). The TI resource explorer projects are complete with linker scripts and are ready to build without making any modifications to them.

If building an app on the command-line, GCC, SYS/BIOS, and XDCtools need to be installed into the user's build environment. The command line tools need to be invoked directly, or (preferably) the build can be managed with makefiles. Once built, applications can be loaded and debugged with CCS. In this CCS context, the Runtime Object Viewer (ROV) can be used to browse SYS/BIOS kernel object states, and the Real-Time Analysis (RTA) tools are also available. Alternatively, GDB or another debugger can be used for basic debug as well; ROV and RTA tools will not be available in those contexts.

Configuration overview

SYS/BIOS needs to be "configured" before applications can be compiled and linked with it. Configuration is primarily used for specifying which modules to include, and setting default values for the many tunable parameters in SYS/BIOS.

The input to the configuration tool, named configuro, is a text file with a .cfg extension written in JavaScript (ECMAScript). The output of configuration is a set of options for the C compiler to be used when compiling user applications, and a set of linker options for use when linking applications.

For compiling, the output of the configuration process is a set of include options (-I<dir-to-include>) and pre-processor defines (-D). These are contained a file called compiler.opt that can be provided to gcc via the @ option.

For linking, a list of libraries and object files is produced by configuro and output in a file called linker.cmd. This file should be passed to the linker along with the user's linker command file specifying memory placement and any other linker directives. When using the compiler front-end for linking, the option for specifying the linker command file is -Wl,-T,cfg-out-dir/linker.cmd.

See the sample application package referenced on this page for a concrete example of configuration and linking of the appropriate components. See the SYS/BIOS User's Guide for more details on the configuration process. Also, examples of integrating configuro with various toolchains using make is covered [| here (http://rtsc.eclipse.org/docs-tip/Consuming_Configurable_Content/makefiles)].

Example Walkthrough

This section steps through building three sample SYS/BIOS applications on a Linux host. The three samples are: a simple "hello world" app; one that illustrates use of the SYS/BIOS Clock module; and one that illustrates usage of the Task and Semaphore modules. The application sources, configuration, and makefile are included in the sample application package that can be downloaded from the link below.

1) Install GCC. For this example, Linaro's BareMetal GNU Tools are used (https://launchpad.net/gcc-arm-embedded/4.7/4.7-2013-q3-update). See the readme at the aforementioned link for installation instructions.

2) Install XDCtools by invoking its installer (for example: "./xdctools_setuplinux_3_25_05_94.bin"), and follow the prompts to select an install location.

3) Install SYS/BIOS by invoking its installer (for example: "./bios_setuplinux_6_37_02_27.bin"), and follow the prompts to select an install location.

4) Download and uncompress the sample application package from this link: [| Samples_CortexM.tar.gz (http://processors.wiki.ti.com/images/2/2f/Samples_CortexM.tar.gz)] Install this package into the location you will use for building your applications.

5) Edit the first three lines of the sample Makefile, specifying the appropriate values for your build environment for: M4TOOLS, SYSBIOS, and XDCTOOLS.

6) The samples package includes a linker script for a Tiva TM4C123GH6PM device. If you are building for a different Tiva M4 device, you can copy the tm4c123gh6pm.lds file, and edit the "MEMORY" region specifications at the top of the file, to correspond to the memory map of your device. You can specify your new linker script file as the LINKERCMD file in Makefile.

7) Build the three sample applications by invoking:

```
make
```

You can clean the build by invoking:

```
make clean
```

8) To run the programs (with the .out extensions) on your M4 target with CCS, you can do the following. These instructions are for CCS v6.

- Open CCS, select File->New->Target Configuration File
- Enter a File name, for example: tm4c123.ccxml

- Click “Finish”
- In the “Connection” drop down list select “Stellaris In-Circuit Debug Interface”
- In the "Device" list, check the box next to “Tiva TM4C123GH6PM”
- Click the “Save” button
- From the CCS top-level menu, select View->Target Configurations
- Expand the “User Defined” folder
- Right click on your new target configuration file (e.g., “tm4c123.ccxml”) and select “Launch Selected Configuration”
- Goto "Tools"->"Debugger Options"->"Program/Memory Load Options" and ensure "Enable Semihosting" is selected. Click on "Remember My Settings". This is required for CIO to work.
- In the Debug view, right click on “Disconnected Device” and select “Connect Target”
- Select Target->Reset->SystemReset
- Select Target->Load Program, and then “Browse” and select one of the sample executables (e.g., “task.out”)
- Run and debug the program

You can select Tools->ROV from the CCS top-level menu to bring up the Runtime Object Viewer to browse SYS/BIOS module states.

Migrating existing GCC projects to SYS/BIOS v6.51

SYS/BIOS v6.51 migrated to GCC v6.3.1 codegen tools and switched from newlib to newlib-nano C runtime library. newlib-nano was selected as it is optimized for embedded applications.

In order to migrate an existing GCC project to use SYS/BIOS v6.51, the following updates need to be made:

- Point to latest GCC v6 compiler shipped with CCS (GCC v6.3.1 shipped with CCS v7.2).
- Add "<GCC_INSTALL_DIR>/gcc-arm-none-eabi-6-2017-q1-update/arm-none-eabi/include/newlib-nano" to the compiler include path.
- For Cortex-M3 targets, replace "-L$(SYSBIOS)/packages/gnu/targets/arm/libs/install-native/arm-none-eabi/lib/armv7-m" with "-L$(SYSBIOS)/packages/gnu/targets/arm/libs/install-native/arm-none-eabi/lib/thumb/v7-m --specs=nano.specs" on the link line. "nano.specs" selects newlib-nano as the C runtime library.
- For Cortex-M4 targets, replace "-L$(SYSBIOS)/packages/gnu/targets/arm/libs/install-native/arm-none-eabi/lib/armv7e-m" with "-L$(SYSBIOS)/packages/gnu/targets/arm/libs/install-native/arm-none-eabi/lib/thumb/v7e-m --specs=nano.specs" on the link line. "nano.specs" selects newlib-nano as the C runtime library.
- For Cortex-M4F targets, replace "-L$(SYSBIOS)/packages/gnu/targets/arm/libs/install-native/arm-none-eabi/lib/armv7e-m/fpu" with "-L$(SYSBIOS)/packages/gnu/targets/arm/libs/install-native/arm-none-eabi/lib/thumb/v7e-m/fpv4-sp --specs=nano.specs" on the link line. "nano.specs" selects newlib-nano as the C runtime library.
- Rebuild the project.

FAQs

How do I enable Semi-Hosting for Cortex-M GNU targets ?

In order to enable Semi-Hosting, the app needs to be linked with a special library called librdimon.a and a module called "ti.sysbios.rts.gnu.SemiHostSupport" needs to be pulled in (xdc.useModule'd) the application's cfg file.

Please see http://processors.wiki.ti.com/index.php/SYS/BIOS_with_GCC_(CortexA)#How_do_I_enable_Semi-Hosting_for_Cortex-A_GNU_targets_.3F for detailed steps.

Note: ***An app built with semi-hosting support will not work standalone***. This is because the semi-hosting library (librdimon) adds breakpoint instructions (bkpt #imm) to the code thus requiring a debugger to work properly. The solution in this case is to disable semi-hosting by removing the SemiHostSupport module from the cfg file and linking with libnosys.a library instead of librdimon.a library.

What are the limitations of newlib-nano libc compared to newlib libc ?

See this FAQ.

C++ destructors are not being called in my SYS/BIOS (TI-RTOS Kernel) application ?

See this FAQ

Why is exception handling not working in my C++ application ?

See this FAQ

{{

1. switchcategory:MultiCore=
   - For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum
   - For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum

   Please post only comments related to the article **SYS/BIOS with GCC (CortexM)** here.

Keystone=

- For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum
- For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum

C2000=*For technical support on the C2000 please post your questions on The C2000 Forum. Please post only comments about the article* ***SYS/BIOS with GCC*** DaVinci=*For technical support on DaVincoplease post your questions on The DaVinci Forum. Please post only comments about the article* ***SYS/BIOS with GCC (CortexM)*** *here.*

MSP430=*For technical support on MSP430 please post your questions on The MSP430 Forum. Please post only comments about the article* ***SYS/BIOS with GCC (CortexM)*** *here.*

OMAP35x=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article* ***SYS/BIOS with GCC (CortexM)*** *here.*

OMAPL1=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article* ***SYS/BIOS with GCC (CortexM)*** *here.*

MAVRK=*For technical support on MAVRK please post your questions on The MAVRK Toolbox Forum. Please post only comments about the article* ***SYS/BIOS with GCC***

For technical su please post you questions at http://e2e.ti.cor Please post on comments abo article ***SYS/BI GCC (CortexM***

}}

Please post only comments related to the article **SYS/BIOS with GCC (CortexM)** here.

(CortexM) *here.*

(CortexM) *here.*

Links

Amplifiers & Linear
Audio
Broadband RF/IF & Digital Radio
Clocks & Timers
Data Converters
DLP & MEMS
High-Reliability
Interface
Logic
Power Management
Processors

- ARM Processors
- Digital Signal Processors (DSP)
- Microcontrollers (MCU)
- OMAP Applications Processors

Switches & Multiplexers
Temperature Sensors & Control ICs
Wireless Connectivity