Running ModusToolbox™ from the Command Line

Version 1.1

Introduction

The ModusToolbox SDKs allow you to use a GNU make build system for building applications via command line make. This document describes what is required to run the build system and how to use it. The document is structured as follows:

- Environment Setup – Prerequisites for setting up the build environment
- Building Applications – Instructions on building (compiling and linking) applications
- Creating an Application – Anatomy of ModusToolbox applications and how to create one

For information about programming and debugging from the command line, refer to the following:

- **PSoC 6:** Cypress Programmer 2.1 OpenOCD CLI Guide
- **CYW20819 Bluetooth:** WICED Hardware Debugging

Note When building using the command line interface (CLI), the expectation is that the entire SDK is open for modification to suit your needs. Use source control or make copies of the SDK elements that are modified so that the SDK can be reverted to a known operating state.

Note This document is targeted at users who want to run from the command line. Using applications with the IDE and the command line interchangeably is not supported. That is, a ModusToolbox application created from the IDE cannot then be built from the command line.

Environment Setup

In order to build via command line make, the SDK expects the following pre-requisites:

- Existence of GNU make.
- Environment variable called “CYSDK”, which points to the SDK location.
- All appropriate drivers from the SDK installed (described in Windows/macOS and Linux sections)

Windows

On Windows, Cypress recommends using standard make distributions, such as Cygwin.

Setting up the environment can be done in a number of ways. Assuming that the SDK was installed in the default location, type the following on the command line:

MSYS:

```
export CYSDK=C:/Users/<USER NAME>/ModusToolbox_1.1
```

Cygwin:

```
export CYSDK=/cygdrive/c/Users/<USERNAME>/ModusToolbox_1.1
```

You may specify this path in the Windows system environment variables for convenience.

You may also find it easier to map a virtual drive for the ModusToolbox directory. Furthermore, having a shorter path will prevent issues with path limits on Windows. Use the `subst` command in a console window (as your current user) to map or remove a virtual drive, as follows:

```
subst <drive letter>: C:\Users\%USERNAME%\ModusToolbox_1.1
set CYSDK=<drive letter>:
```

As for drivers for ModusToolbox on Windows, all necessary packages are included with the installation and no further action needs to be taken.

macOS and Linux

For macOS and Linux, the SDK relies on the GNU make distribution provided by the operating system. Therefore, there should be no extra steps required to set up GNU make.

On macOS, the CYSDK environment variable can be configured via command line as follows:

```
export CYSDK=/Applications/ModusToolbox_1.1/
```

On Linux, the installation location is where the archive was extracted. The environment variable can be configured as follows:

```
export CYSDK=<Extracted root directory>/ModusToolbox_1.1/
```

In order to install all relevant drivers, refer to the ModusToolbox Installation Guide.

Note macOS and Linux users are required to launch the ModusToolbox IDE once after installation before attempting to build via command line.

Building Applications

By default, the SDKs provide a number of examples that can be built either in the ModusToolbox IDE or via command line make. These projects are located in the respective library's example directory. For example:

```
ModusToolbox_1.1/libraries/psoc6sw-1.1/examples
```

or

```
ModusToolbox_1.1/libraries/bt_20819A1-1.0/examples/
```

Note You can obtain additional examples from the Cypress GitHub website.

Open a shell/terminal and navigate to an example directory. Pick an example of interest and navigate to the directory where the *modus.mk* file is located. You may now run the make commands on the project:

```
make [VARIABLE] [TARGET]
```

Commonly used VARIABLEs and TARGETs are documented here for easy reference. Note that multiple VARIABLEs can be specified via command line. This will override the default make VARIABLEs defined in the build system and allows customization from the command line. For platform-specific VARIABLEs and TARGETs, refer to the platform specific help documentation by running the *help* target. A platform is a collection of settings to define the hardware, such as power and debug settings. This could also refer to a development kit.

Common Make Targets

TARGET	Description
all build	Build all artifacts for the application
clean	Clean just the platform and configuration.
cleanplatform	Clean all configurations in the target platform.
scrub	Clean all configurations and platforms.
program	Build the application then program the artifacts into the target device. Note that the programming operation does not support parallel builds.
quickprogram	Program the artifacts as they exist on disk into the device. No build is performed and if the artifacts on disk are out of date, these out of date artifacts will be programmed into the device. Note that the programming operation does not support parallel builds.
help	Print the (platform-specific) help documentation.

Common Make Variables

When make is invoked, the following make variables can be set on the command line to change the build process. You can also use `make help` for more options.

VARIABLE	Description
CYSDK	This is generally set with an environment variable, but can be supplied on the make command line. This variable defines the home directory of the ModusToolbox installation (for example, C:\Users\<username>\ModusToolbox_1.1\).
PLATFORM	The target platform for the build process. Refer to the CY_VALID_PLATFORMS variable in your application *modus.mk* file for a complete list of supported platforms. Platforms include: PSoC6_cm0p, PSoC6_cm4_dual, PSoC6_cm4_single, PSOC6_DUAL_CORE, PSOC6_SINGLE_CORE, CYW920819EVB-02, 20819, CYBT-213043-MESH, and CYW920819EVB-02.
DEVICE	**PSoC 6 Only** For platforms where the device is not implied from the PLATFORM, this make variable gives the target device for the application. If CY_VALID_DEVICES is defined, the DEVICE value must be one of these values in the CY_VALID_DEVICES list. The IDE New Application wizard contains a list of devices. For example, CY8C6347BZI-BLD53. See also: https://www.cypress.com/products/32-bit-arm-cortex-m4-psoc-6
CY_BUILD_LOCATION	The build directory for the build process. It defaults to a subdirectory named build in the current directory.
VERBOSE	Prints information about the build process. [true, false]
DEBUG	Prints information about the Makefiles. [true, false]
WHICHFILE	Prints the Makefiles as they are read. [true, false]

Sample PSoC Build

The following shows an example command list for building and programming a PSOC6_DUAL_CORE BlinkyLED design on a CY8CKIT-062-BLE kit.

```
cd libraries/psoc6sw-1.1/examples/BlinkyLED
make PLATFORM=PSOC6_DUAL_CORE DEVICE=CY8C6347BZI-BLD53 -j4
make quickprogram
```

You may then optionally run the *clean*, *cleanplatform*, or *scrub* targets to clean the artifacts.

Sample Bluetooth Build

The following shows an example command list for building and programming a Beacon design on a CYW920819EVB-02 kit.

```
cd libraries/bt_20819A1-1.0/examples/BT-SDK/20819-A1_Bluetooth/apps/demo/beacon
make PLATFORM=CYW920819EVB-02 program
```

You may then optionally run the *clean*, *cleanplatform*, or *scrub* targets to clean the artifacts.

Changing Devices

If you need to change devices for your application, Cypress recommends that you always do a clean before invoking make. Otherwise, there is a chance that stale object files that are incompatible with the target device may get pulled into the application.

Creating an Application

An application in ModusToolbox is a set of application sources, a top-level *modus.mk* Makefile and a *design.modus* file. There is also an optional file called *makefile* that references *modus.mk* that is primarily present to provide the convenience of being able to just type *make* on the console without any arguments. It is also used in certain designs to be able to provide default make VARIABLEs that differ from those defined in either *modus.mk* or the platform.

File	Description
modus.mk	Top-level makefile that describes the design. It contains the design make VARIABLEs that are passed to the build system.
design.modus	Graphical configurator design file used in IDE builds to describe the application's hardware configuration. This file is opened by the build system during a build process and generates source files that can be used in your application.
makefile	Wrapper makefile that includes modus.mk.
{source files}	Source files such as .c and .h files used in the design.

An application can exist anywhere on disk and does not need to be located in the SDK's example directory. It is recommended (especially in Windows) to shorten the path to the application as much as possible to avoid line length complications.

Depending on the chosen application library type, the make VARIABLEs in *modus.mk* file will be different. Refer to the following sections to see the list of available make VARIABLEs that can be used in *modus.mk*.

PSoC 6 Application

The following table summarizes the most common make VARIABLEs of interest in the *modus.mk* file targeting a PSoC 6 application. Note that these can be overridden when called from the command line.

VARIABLE	Description
TOOLCHAIN	Toolchain used to build the design. For example, GCC, IAR.
OPTIMIZATION	Toolchain-specific optimization flag. For example, Og, Os.
CONFIG	Build configuration – Release, Debug.
VFP_FLAG	Vector Floating-point flag (soft/hard) selection

These are additional make VARIABLES available, but you will likely not need to change these.

VARIABLE	Description
APP_MAINAPP_CM0P_DEFINES	Defines specific to the CM0+ application
APP_MAINAPP_CM0P_FLAGS	Compiler flags specific to the CM0+ application
APP_MAINAPP_CM0P_INCLUDES	Includes specific to the CM0+ application. Use CY_LOCAL_INCLUDE_CM0P if the include path should show in the ModusToolbox IDE.
APP_MAINAPP_CM0P_LIBS	Other libraries (.a) needed by the CM0+ application
APP_MAINAPP_CM4_DEFINES	Defines specific to the CM4 application
APP_MAINAPP_CM4_FLAGS	Compiler flags specific to the CM4 application
APP_MAINAPP_CM4_INCLUDES	Includes specific to the CM4 application. Use CY_LOCAL_INCLUDE_CM4 if the include path should show in the ModusToolbox IDE.
APP_MAINAPP_CM4_LIBS	Other libraries (.a) needed by the CM4 application
CY_APP_CM0P_SOURCE	The source code for the CM0+ application
CY_APP_CM4_SOURCE	The source code for the CM4 application
CY_EXAMPLE_DESCRIPTION	Description of the example project
CY_EXAMPLE_NAME	The name of the example.
CY_LOCAL_INCLUDE_CM0P	Prepend CM0+ path to use in includes and custom linker scripts for translating the paths from CLI to IDE.
CY_LOCAL_INCLUDE_CM4	Prepend CM4 path to use in includes and custom linker scripts for translating the paths from CLI to IDE.
CY_MAINAPP_CM0P_LINKER_SCRIPT	Custom CM0+ linker script location (For example, <ABSOLUTE PATH>/customScript.ld). Use CY_LOCAL_INCLUDE_CM0P if the linker script is in the application directory.
CY_MAINAPP_CM0P_OVERRIDE_MAIN	If set to 1, override the default CM0+ main.c file with a local copy provided by the application.
CY_MAINAPP_CM0P_SWCOMP_EXT	Software components to be ignored by the CM0+ application. This is useful when a SW comp has a dependency on another but the sources for that other SW comp is provided locally by the application. For example, config files.
CY_MAINAPP_CM0P_SWCOMP_USED	Software components needed by CM0+
CY_MAINAPP_CM4_LINKER_SCRIPT	Custom CM4 linker script location (For example, <ABSOLUTE PATH>/customScript.ld). Use CY_LOCAL_INCLUDE_CM4 if the linker script is in the application directory.
CY_MAINAPP_SWCOMP_EXT	Software components to be ignored by the CM4 application. This is useful when a SW comp has a dependency on another but the sources for that other SW comp is provided locally by the application. For example, config files.
CY_MAINAPP_SWCOMP_USED	Software components needed by CM4
CY_REQUIRED_SDK	Required SDK for this example.
CY_SHOW_NEW_PROJECT	If this is true, the example is shown as part of the new project dialog in the IDE. If this is false, the project is not shown.
CY_VALID_DEVICES	List of valid devices for this example. If this is empty, this example works for all devices.
CY_VALID_PLATFORMS	List of valid platforms for this example.
CYCONFIG_DESIGN_MODUS	The location of the *design.modus* file.
CYCONFIG_GENERATED_SOURCES	Additional (non-core) set of generated source files such as CapSense and USB configurator generated files.
PLATFORMS_ VERSION	Version of the platform files to use in the build.

Bluetooth Application

The following table summarizes the most common make VARIABLEs of interest in the *modus.mk* file targeting a Bluetooth application. Note that these can be overridden when called from the command line.

VARIABLE	Description
TOOLCHAIN	Toolchain used to build the design. For example, GCC, IAR.
BT_DEVICE_ADDRESS	Bluetooth device address.
UART	Used to specify a serial port to download. The default is AUTO. It can also be COM4 or /dev/ttyWICED_HCI_UART0. It varies by OS.
ENABLE_DEBUG	Flag to enable or disable debug mode.

There are also application-specific settings. Refer to the specific app *readme.txt* file for more information.