ARM mbed Tutorials

**GETTING STARTED WITH ARM USING MBED**

[**Al Williams**](http://hackaday.com/author/wd5gnr1/)

[**August 11, 2015**](http://hackaday.com/2015/08/11/)



Even though the Arduino was hardly the first 8 bit microcontroller board to support a bootloader and the C/C++ language, it quickly became the *de facto* standard for hobby-level microcontrollers as well as a common choice for one-off or prototype projects. I’m sure there are a lot of reasons why this occurred, but in my mind there were three major reasons: price, availability of lots of library and sample code, and the existence of a simplified GUI IDE that you could install in a few minutes. The build process is simple, too, even though if you ever have to actually figure it out, it is quite ugly. For most people, it works, and that makes it not ugly.

I like the ATMega chips. In fact, I had boards based around the ATMega8 and a bootloader way before there was an Arduino. However, they are fairly small parts. It is true that the Arduino infrastructure has grown to support more ATMega chips, many with more memory and I/O and clock speeds. However, 32-bit processors are getting inexpensive enough that for all but the simplest or highest volume projects, you should be thinking about using 32-bit.

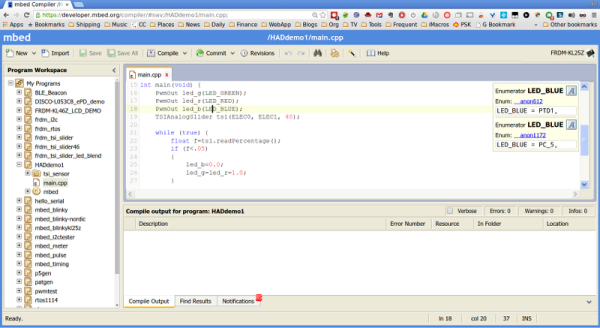
If you’ve tried to go that route before, you’ve probably been daunted by the price, especially the price of development tools. Your alternative is to roll your own tool chain which is very doable (and there are some nice scripts out there that will help you). You also need to worry about libraries and how to integrate them. Not to mention, many of the advanced processors require a lot of setup to get, say, an A/D converter turned on. Most processors keep things they aren’t using turned off, and each pin requires setup to select the 4 or 5 things shared on that pin.

All of this has been a barrier to entry. The vendors have all figured this out, though, and many have tried to build tools aimed at breaking up the Arduino market ranging from inexpensive development boards to code-generating wizards, to full blown IDEs. I want to tell you (and show you, in the video below) how you can make the jump from 8-bit to 32-bit much easier than you might think.

First, let’s talk price. I’m going to focus on [the KL25Z Freedom board](http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=FRDM-KL25Z) from Freescale. It isn’t much bigger than a standard Arduino board and it costs (today) about $13. What do you get for that? A 32 bit ARM processor running at 48MHz. It has 128K of flash, 16K of RAM, a USB port, and several SPI, UART, I2C, and PWM peripherals. It also has 16-bit A/D converters and a 12 bit DAC. The board itself also has an accelerometer and a touch sensor along with a tri-color LED. Not bad for $13.

Of course, you are thinking you can get an ATMega chip for less and plug it in a breadboard. You can use the same techniques I’ll talk about to program an LPC1114FN28 chip. Sure, it only has 4K of RAM and 32K of program storage, but it will easily fit in a breadboard and costs less than $5, even if you are only buying one.

The KL25Z even has an Arduino-compatible daughterboard socket (you know, a place to plug in a shield, as much as it pains me to say shield). You do have to be careful as the board is 3.3V and not 5V. Some 32-bit devices have 5V compatible I/O, but you should always make sure before directly connecting anything.

[](https://hackadaycom.files.wordpress.com/2015/08/mbed0.png)A $13 development board with a lot of memory and other goodies; sounds good, right? But surely the development tools are expensive or difficult to set up, right? Actually, no. ARM provides the mbed website that offers a full-blow C/C++ IDE in your browser (see picture to the left). No software to install at all. Version control, sharing libraries between users, documentation. Everything you might want in an IDE except debugging (but don’t forget the Arduino IDE doesn’t have debugging either). You don’t have to use the online IDE if you prefer to use a local toolchain. However, the IDE is very pleasant and has a lot of productive features.

Maybe the catch is you need an expensive device programmer? Again, no. The board looks like a USB drive. You plug it into your Windows PC (or anything that will recognize a USB drive; I use Linux) and it pops up as a disk drive. The IDE produces files that you just copy to the drive and that programs the microcontroller. The bootloader is built into the chip.

The mbed library (and also third party libraries, including those you create) let you sidestep complexity the same way you do with the Arduino. For example, if you want to create a PWM output, you simply use a PWMOut object:

   PwmOut led\_r(LED\_RED);

This takes care of turning on the right parts of the chip, enabling the PWM outputs, and whatever other voodoo it takes to make that work on the ARM chip you are using. Just about any other I/O task you want is just as easy. The IDE lets you search for libraries ranging from math solutions to an entire real time operating system.

So if you are running into the limits of an 8 bit processor — or even if you just want to step up for the sake of stepping up–you can be up and running on a 32-bit development platform in a few minutes for about $13. Once you get used to having 128K of code space, a ton of RAM and executing code at about 48 MIPS, you’ll find it hard to go back.

Want to know more? The video below will get you started. You can also check out the documentation on the[mbed website](http://mbed.org/). We’ve also talked about the [BBC Micro:Bit](http://hackaday.com/2015/07/07/the-bbc-microbit/) which will work with mbed and even looked at an [ZX81 recreated on an mbed-compatible processor](http://hackaday.com/2014/12/22/zx81-emulated-on-an-mbed/), if you want to get some more ideas.

<https://youtu.be/cKbMyXl3yBA>

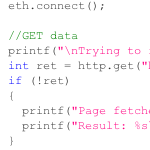
This information taken from <https://developer.mbed.org/handbook/Homepage>

**xplore mbed**

The mbed platform provides free software libraries, hardware designs and online tools for professional rapid prototyping of products based on ARM microcontrollers.

The platform includes a standards-based C/C++ SDK, a microcontroller HDK and supported development boards, an online compiler and online developer collaboration tools.

Here are some details of what you'll get for basing your next ARM microcontroller powered product on the mbed platform:

[](https://developer.mbed.org/handbook/mbed-SDK)

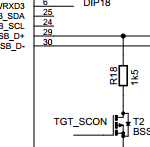
**Software Development Kit (SDK)**

The mbed Software Development Kit (SDK) is an open source C/C++ microcontroller software platform relied upon by tens of thousands of developers to build projects fast. We've worried about creating and testing startup code, C runtime, libraries and peripheral APIs, so you can worry about coding the smarts of your next product.

The SDK is licensed under the permissive Apache 2.0 licence, so you can use it in both commercial and personal projects with confidence.

The mbed SDK has been designed to provide enough hardware abstraction to be intuitive and concise, yet powerful enough to build complex projects. It is built on the low-level ARM CMSIS APIs, allowing you to code down to the metal if needed. In addition to USB and Networking libraries, a cookbook of hundreds of reusable peripheral and module libraries have been built on top of the SDK by the mbed Developer Community.

* [mbed SDK](https://developer.mbed.org/handbook/mbed-SDK)

[](https://developer.mbed.org/handbook/mbed-HDK)

**Hardware Development Kit (HDK)**

The mbed Hardware Development Kit (HDK) provides full microcontroller sub-system design files and firmware for building development boards and custom products that benefit from the native support of the mbed SDK and free mbed Online Compiler and mbed Developer Platform.

The HDK specifies all support components and circuits including the mbed Onboard Interface design that provides simple USB drag-n-drop programming and CMSIS-DAP debug interface for the target microcontroller.

Development boards that are already based on the HDK are the quickest way to get started with the mbed platform. We manufacture official mbed Microcontroller modules that are specifically optimised for flexible rapid prototyping, and are available from distributors worldwide. Our partners are now also creating mbed-enabled hardware such as ultra low-cost ARM evaluation boards in the popular Arduino form-factor.

* [mbed HDK](https://developer.mbed.org/handbook/mbed-HDK)
* [mbed Hardware](https://developer.mbed.org/handbook/mbed-Hardware)

**Free Online Development Tools**

The mbed Compiler is a powerful online IDE that is free for use with hardware implementing the mbed HDK, and tightly integrated with the mbed SDK and Developer Website. Under the hood, it relies on the industry standard ARM professional C/C++ compiler, pre-configured and tested to generate fast, efficient code without fuss.

Login anywhere to get instant access to your development environment, on Windows, Mac, Linux. You can even work from tablets!

Whilst the mbed Compiler provides you your own private workspace, it is also fully integrated with the mbed.org Developer Website so you can easily import libraries and examples. If you choose to, publishing your own code and collaborating with other mbed users is just a few clicks too. The mbed Compiler also supports full export to different toolchains, in case your project demands it as you go to production.

* [mbed Compiler](https://mbed.org/handbook/mbed-Compiler)
* [Collaboration](https://mbed.org/handbook/Collaboration)
* [Export](https://mbed.org/handbook/Exporting-to-offline-toolchains)

**Worldwide Developer Community**

Using mbed means a huge shared context with other developers, and that means when you have a question, there is less pre-amble, less explanation and less time reproducing issues, and more time getting answers. We're proud that this has helped us grow an active and friendly community of skilled developers that are collectively helping get prototypes made even faster.

But where it really gets interesting is with code. Our developers are sharing thousands of open source repositories and building an extensive cookbook of recipes that you can reuse to build your products.

We've also made contributing back is easy; you can publish a library to mbed.org with a few clicks in the IDE, and let others build on your hard work. In fact, this is how some of our users end up collaborating on hard problems, and even getting contract work.

## [mbed SDK](https://developer.mbed.org/handbook/mbed-SDK)

#### Table of Contents

1. [Hello World!](https://developer.mbed.org/handbook/mbed-SDK#hello-world)
2. [High-level Peripheral APIs](https://developer.mbed.org/handbook/mbed-SDK#high-level-peripheral-apis)
3. [Support for Multiple Targets](https://developer.mbed.org/handbook/mbed-SDK#support-for-multiple-targets)
4. [Support for Multiple Toolchains](https://developer.mbed.org/handbook/mbed-SDK#support-for-multiple-toolchains)
5. [Open Source](https://developer.mbed.org/handbook/mbed-SDK#open-source)
6. [See also](https://developer.mbed.org/handbook/mbed-SDK#see-also)

The mbed Software Development Kit (SDK) is a C/C++ microcontroller software platform relied upon by tens of thousands of developers to build projects fast. We've worried about creating and testing startup code, C runtime, libraries and peripheral APIs, so you can worry about coding the smarts of your next product.

The SDK is licensed under the permissive Apache 2.0 licence, so you can use it in both commercial and personal projects with confidence.

The mbed SDK has been designed to provide enough hardware abstraction to be intuitive and concise, yet powerful enough to build complex projects. It is built on the low-level ARM CMSIS APIs, allowing you to code down to the metal if needed. In addition to RTOS, USB and Networking libraries, a cookbook of hundreds of reusable peripheral and module libraries have been built on top of the SDK by the mbed Developer Community.

## Hello World!

Startup code, check. C Library integration, check. Peripheral libraries, check. We've worked hard to help you get to the point:

#### [Import program](https://developer.mbed.org/compiler/#import:/teams/mbed/code/mbed_blinky/)[mbed\_blinky](https://developer.mbed.org/teams/mbed/code/mbed_blinky/)

The example program for mbed pin-compatible platforms

*Last commit 9 days ago by*[*mbed*](https://developer.mbed.org/teams/mbed/)

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 | #include "mbed.h"    DigitalOut myled(LED1);    int main() {      while(1) {          myled = 1;          wait(0.2);          myled = 0;          wait(0.2);      }  } |

## High-level Peripheral APIs

The mbed SDK gives you an API-driven approach to microcontroller coding.

We've done all the hard work of implementing drivers for the different [mbed platforms](https://developer.mbed.org/platforms), so you don't have to. It is liberating to fire up an interface, knowing it'll just work!

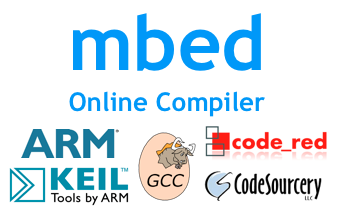
You can code using meaningful abstract objects and API calls, so you don't need to learn the microcontroller hardware details to get going. There is even a "Hello World!" example for every peripheral, just to get you started before you know it.

Take a look at some of these interfaces to get a feel of how it works: [DigitalOut](https://developer.mbed.org/handbook/DigitalOut), [AnalogIn](https://developer.mbed.org/handbook/AnalogIn), [SPI](https://developer.mbed.org/handbook/SPI), [USBMouse](https://developer.mbed.org/handbook/USBMouse), [Timer](https://developer.mbed.org/handbook/Timer), [CAN](https://developer.mbed.org/handbook/CAN)

But if needed, you can always bypass the APIs and talk directly to the microcontroller hardware using the low-level Cortex Microcontroller Software Interface Standard (CMSIS) APIs. Ideal when they are fine for most of your project, but one aspect needs specific low-level control.

For all the mbed C/C++ SDK APIs, see the [mbed Handbook](https://developer.mbed.org/handbook)

To read more about mbed SDK coding style, see the [mbed SDK coding style](https://mbed.org/teams/SDK-Development/wiki/mbed-sdk-coding-style)



## Support for Multiple Targets

The abstraction provided by the mbed SDK APIs enables libraries and example code to be reused by any microcontroller target that the mbed SDK targets.

## Support for Multiple Toolchains

Our goal with the mbed Compiler and mbed SDK is to enable a consistent and stable fully integrated development platform that just works. This helps provide a consistent context for development, code sharing, and questions and answers with other developers that helps you be more productive, especially when prototyping.

However, the mbed C/C++ SDK used with the mbed Online Compiler is also compatible with a number of other popular ARM microcontroller toolchains!

If you'd like to use the [mbed platforms](https://developer.mbed.org/platforms) or mbed C/C++ SDK with an alternate tool, or simply migrate to one as your project develops past prototype, you can choose to export an mbed project to the toolchain of your choice by right-clicking on them in the IDE.

You can read more about this on the [Exporting to offline toolchains](https://developer.mbed.org/handbook/Exporting-to-offline-toolchains) handbook page.

## Open Source



The mbed SDK is licensed under the permissive Apache 2.0 open source licence.

We wanted to make sure the license we chose made it possible to use the SDK in both commercial and personal projects with confidence, including no obligations to open source your own code if you didn't want to. Whilst we encourage sharing of code and experience to be reusable by others, we certainly don't want to enforce it, and a permissive license provides that freedom for our users to keep the options open.

If you are interested in delving in the depth of the mbed SDK implementation you can take a look at the documentation of the [mbed library internals](https://developer.mbed.org/handbook/mbed-library-internals).

and you can use the mbed library sources, instead of one of its builds:

#### [Import library](https://developer.mbed.org/compiler/#import:/users/mbed_official/code/mbed-src/;mode:lib)[mbed-src](https://developer.mbed.org/users/mbed_official/code/mbed-src/)

mbed library sources

*Last commit 30 Sep 2015 by  [mbed official](https://developer.mbed.org/users/mbed_official/)*

If you are interested in [porting the mbed SDK](https://developer.mbed.org/handbook/mbed-SDK-porting) to a new target we provide the sources of all the official mbed libraries, tests and [tools (build and test system)](https://developer.mbed.org/handbook/mbed-tools) in [this github repository](https://github.com/mbedmicro/mbed).

## [mbed HDK](https://developer.mbed.org/handbook/mbed-HDK)

#### Table of Contents

1. [Microcontroller Sub-systems](https://developer.mbed.org/handbook/mbed-HDK#microcontroller-sub-systems)
2. [CMSIS-DAP interface](https://developer.mbed.org/handbook/mbed-HDK#cmsis-dap-interface)
3. [Benefits of the HDK](https://developer.mbed.org/handbook/mbed-HDK#benefits-of-the-hdk)
4. [Source](https://developer.mbed.org/handbook/mbed-HDK#source)

The mbed Hardware Development Kit (HDK) provides full microcontroller sub-system design files and firmware for building development boards and custom products that benefit from the native support of the mbed SDK and free mbed Online Compiler and mbed Developer Platform.

The mbed HDK specifies all support components and circuits including the CMSIS-DAP Interface design that provides simple USB drag-n-drop programming and CMSIS-DAP debug interface for the target microcontroller.

Development boards that are already based on the mbed HDK are the quickest way to get started with the mbed platform. We manufacture official mbed Microcontroller modules that are specifically optimised for flexible rapid prototyping, and are available from distributors worldwide. Our partners are now also creating mbed-enabled hardware such as ultra low-cost ARM evaluation boards in the popular Arduino form-factor.

#### Creating your own mbed-enabled platforms

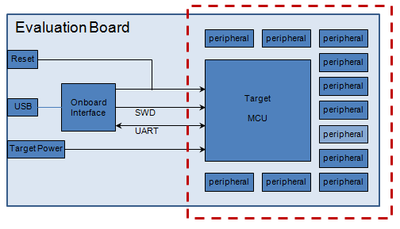
Are you a hardware company that would like to create an mbed-enabled platform of your own that is supported within the mbed platforms database and tools? If so, then please email us at [support@mbed.org](mailto:support@mbed.org) and we can help you with your questions and support you through the process.

## Microcontroller Sub-systems

Each of the subsystems designs include

* Hardware design schematics (Eagle format)
* Interface binary for the CMSIS-DAP interface

An example of how a microcontroller sub-system might be used to build an evaluation board.

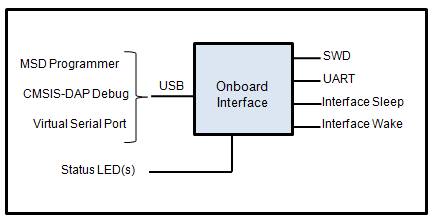


## CMSIS-DAP interface

The CMSIS-DAP Interface is a microcontroller based single chip solution that provides, Drag and Drop programming, [CMSIS-DAP](https://developer.mbed.org/handbook/CMSIS-DAP) debugger and USB serial interface to a range of Cortex-M based microcontrollers.

The small footprint, low number of passive components and rich feature set provide a low cost, low overhead solution that can easily be integrated on a PCB.

The firmware required to turn the low cost microcontroller into a powerful programming, debug and communication interface, is included with the HDK and can be used freely, including for use in commercial products.



The CMSIS-DAP interface provides three main functions over a single physical USB connection :

* USB Disk “drag and drop” programming - ideal for fast turn around prototyping, or in-field upgradable products
* Debug interface using the [CMSIS-DAP](https://developer.mbed.org/handbook/CMSIS-DAP) - Provides full debug capability with tools like [Keil MDK](https://developer.mbed.org/handbook/CMSIS-DAP-MDK)
* USB Serial interface between the host computer and the target

## Benefits of the HDK

There are various benefits to building a custom design onto of the mbed HDK. The ready made schematics are a great short cut, so you can get started on all the things that make your design, without worrying if you've correctly implemented all the "necessary bits" of the design. The mbed HDK incorporates the CMSIS-DAP interface. This provides USB drag and drop programming, [CMSIS-DAP](https://developer.mbed.org/handbook/CMSIS-DAP) debugging and USB serial communication. The [mbed SDK](https://developer.mbed.org/handbook/mbed-SDK) supports each of the exact configurations of HDK designs, and libraries that have been written to the APIs in the mbed SDK are highly reusable. Lastly, the mbed community has developed a wealth of libraries, applications and code examples using the SDK/HDK, and this active community offers a lot of opportunities for support and even hiring in required skills.

## Source

The mbed HDK, complete with PCB Layout files and schematics, can be downloaded from the repository:

* [mbed-HDK](https://developer.mbed.org/teams/mbed/code/mbed-HDK/)

The sources of the CMSIS-DAP Interface Firmware:

* [CMSIS-DAP Interface Firmware](https://developer.mbed.org/handbook/cmsis-dap-interface-firmware)

For support, design review and other help making your platform, email:

* [support@mbed.org](mailto:support@mbed.org)

[**mbed Compiler**](https://developer.mbed.org/handbook/mbed-Compiler)

**The mbed Tools**

1. [Introduction](https://developer.mbed.org/handbook/mbed-Tools)
2. [The mbed Website](https://developer.mbed.org/handbook/mbed-Developer-Website)
3. [**The mbed Compiler**](https://developer.mbed.org/handbook/mbed-Compiler)
   1. [Getting Started](https://developer.mbed.org/handbook/mbed-Compiler-Getting-Started)
   2. [Shortcuts and controls](https://developer.mbed.org/handbook/Compiler-shortcuts-and-controls)
   3. [Touch support](https://developer.mbed.org/handbook/Guide-to-mbed-Compiler-on-tablet-device)
4. [Importing code](https://developer.mbed.org/handbook/Importing-code)
5. [Collaboration](https://developer.mbed.org/handbook/Collaboration)
6. [API Documentation](https://developer.mbed.org/handbook/API-Documentation)
7. [Publishing code](https://developer.mbed.org/handbook/Publishing-code)
8. [Exporting code](https://developer.mbed.org/handbook/Exporting-to-offline-toolchains)
9. [Getting help](https://developer.mbed.org/handbook/Help)

**Instant access to your lightweight C/C++ microcontroller development environment**

The mbed Compiler provides a lightweight online C/C++ IDE that is pre-configured to let you quickly write programs, compile and download them to run on your mbed Microcontroller. In fact, you don't have to install or set up anything to get running with mbed. Because it is a web app, you can log in from anywhere and carry on where you left off, and you are free to work on Windows, Mac, iOS, Android, Linux, or all of them.

**It is online and lightweight, but it is also powerful.**

The compiler uses the professional ARMCC compiler engine, so it produces efficient code that can be used free-of-charge, even in commercial applications. The IDE includes [workspace version control](https://mbed.org/handbook/Compiler-Version-Control), code formatting and [auto-generation](https://developer.mbed.org/cookbook/Documenting-a-Library) of documentation for published libraries. The mbed tools are focused on prototyping and are designed for fast experimentation, and complement other professional production-level tools; you can even export directly to [other toolchains](https://developer.mbed.org/handbook/Exporting-to-offline-toolchains) if you choose, as you progress to productise your design.

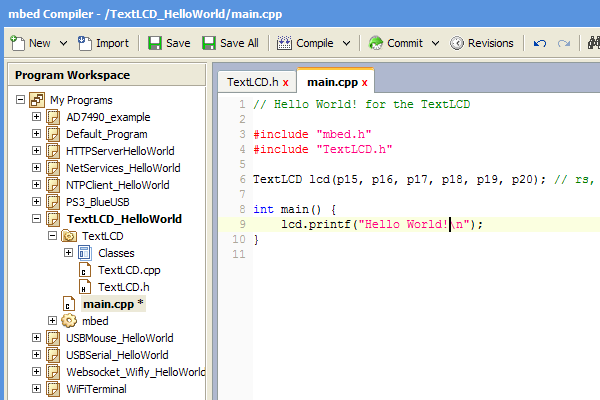
You can [publish projects](https://mbed.org/handbook/Publishing-code) directly from your Compiler workspace to the mbed.org website to [share code](https://developer.mbed.org/code/) with others, and pull existing libraries in to your workspace to get a head start.

**Feature Highlights**

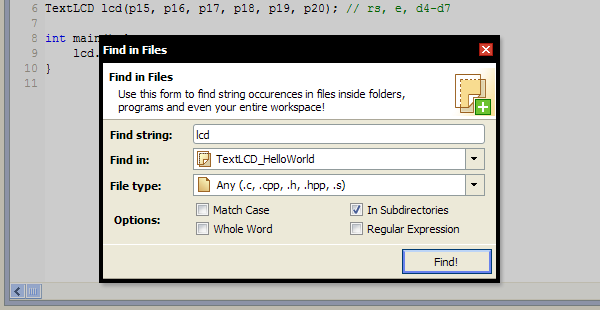
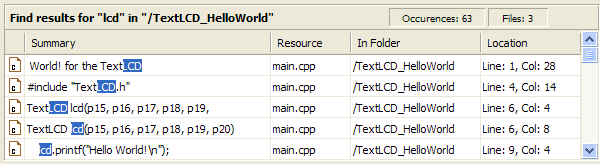
**Online Compiler IDE**

Every mbed user account gets their own private Compiler workspace which contains their programs. This is private to you, and available wherever you login to mbed.

The IDE includes a full code editor including syntax highlighting, standard editor keyboard shortcuts, undo/redo, cut/copy/paste, tabs, block/line comment, and even a code auto-formater. This is where you work on your personal workspace, with multiple files, folders, programs, including a drag and drop folder interface:

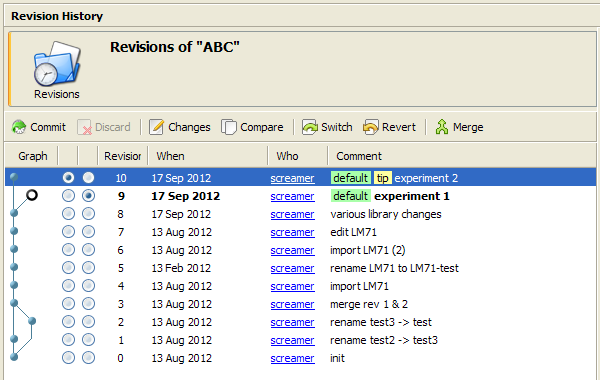


The editor also includes features like find and searching across multiple files and filetypes; for example, searching across your whole program. When you search, the results will appear as a list in the compiler output window where you can jump to any of them with a click:

**Integrated Version Control**

You can use the built-in version control features to let you version, branch and merge code, with a nice representation of the state of your project history:



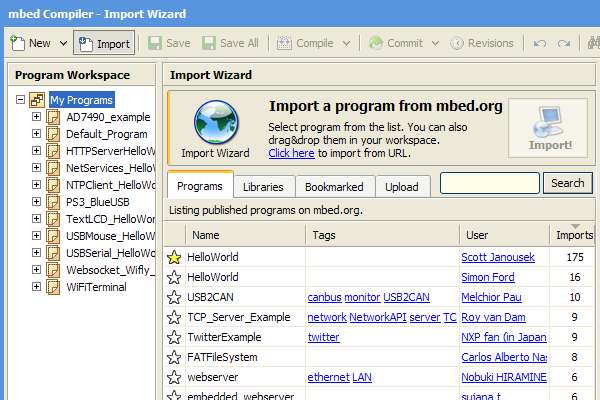
The approach should be familiar to those of you with experience of distributed version control models (as used by mercurial/git); each program has its own local repository, so you can commit and perform actions on it within your own workspace (such as updating, branching and showing changes).

The main things you can do include:

* Commit a version of your project, and view the revision history
* View changes a version made, and compare changes between versions
* Update or revert to a different version
* Branch and merge
* See also [Version Control](https://developer.mbed.org/handbook/Compiler-Version-Control)

**Importing Libraries or Example Programs**

The Import Wizard allows you to import programs and libraries published by mbed users. This is useful for importing code that has been packaged as a reusable library component (e.g. a class for a peripheral), so you can quickly pull in the building blocks for your project.

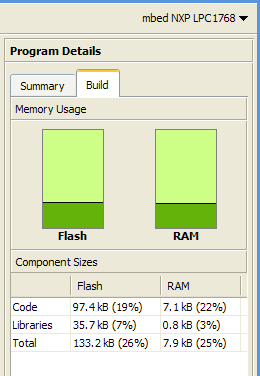


* See also [Importing code](https://developer.mbed.org/handbook/Importing-code)

**Compilation**

To perform the actual compilation the mbed Compiler uses the industry standard [ARM RVDS 4.1](http://www.arm.com/products/tools/software-tools/rvds/arm-compiler.php) compiler engine, in the default configuration, to give excellent code size and performance. There are no limitations on code size (apart from the limits of the device itself!), and the generated code can be used freely for commercial and non-commercial use.

When you compile a program, you'll get a display of the memory usage. This shows the size of program code and any constant (const) variables that will end up in FLASH, and size of data variables that end up in main RAM.

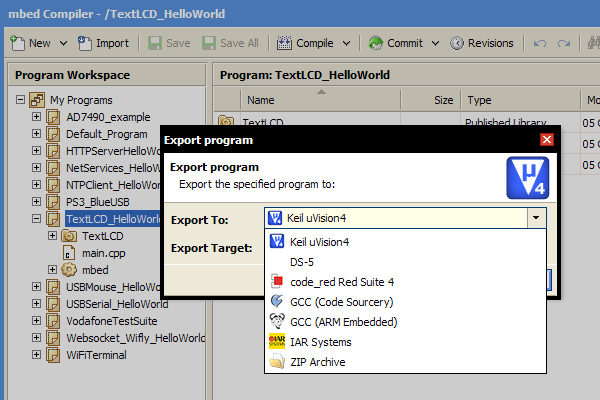


Note, this doesn't include the runtime allocated variables (i.e. the heap and stack), which live in any remaining RAM.

* See also the mbed [Memory Model](https://developer.mbed.org/handbook/Memory-Model)

**Export to Offline Toolchains**

The [mbed C/C++ SDK](https://developer.mbed.org/handbook/mbed-SDK) used with the mbed Online Compiler is also compatible with a number of other popular ARM microcontroller toolchains, so we've also built in the ability to export directly to these toolchains! For example, if you'd like to migrate to a different toolchain as your project develops past prototype, you can choose to export an mbed project by right-clicking on it:



* See also the [/handbook/mbed-SDK](https://developer.mbed.org/handbook/mbed-SDK) and [Exporting to offline toolchains](https://developer.mbed.org/handbook/Exporting-to-offline-toolchains)

**Feature Overview**

Code IDE

* All the core features you expect from a code editor including syntax highlighting, standard [editor keyboard shortcuts](https://developer.mbed.org/handbook/Compiler-shortcuts-and-controls), copy/paste, etc
* Personal workspace with multiple files, folders, programs, including drag and drop folder interface
* Code auto-formatter, print-friendly code preview

Compile Engine

* Pre-configured compile engine that "just works", delivering .bin binary file to save to mbed microcontroller
* Switch between different mbed targets with a drop-down selector
* Output of compile-time messages, including click to go to error and error message wiki
* Build information including graphical display of code size and RAM usage

Built-in Version Control and Collaboration tools

* [Built-in version control](https://developer.mbed.org/handbook/Compiler-Version-Control) (DVCS)
* Publish, fork, push and pull code in [collaboration-enabled environment](https://developer.mbed.org/handbook/Collaboration)
* View graphs, diffs, change sets

Importing and Exporting

* Import programs from online catalogue of [published programs](https://developer.mbed.org/code/)
* Publish your code directly from the mbed Compiler to the mbed Developer Website
* Import from and export to local source files and zip archives
* [Export directly](https://developer.mbed.org/handbook/Exporting-to-offline-toolchains) to other popular ARM toolchains

Accessibility

* Access the mbed Compiler on all major browsers, on all modern operating systems
* Develop and prototype right on your [tablet device](https://developer.mbed.org/handbook/Guide-to-mbed-Compiler-on-tablet-device) (Android, iOS) with the integrated [touch support](https://developer.mbed.org/blog/entry/compiler-touch-support/)
* Backward compatible up to Internet Explorer 6

## [SerialPC](https://developer.mbed.org/handbook/SerialPC)

#### Table of Contents

1. [Serial Communication with a PC](https://developer.mbed.org/handbook/SerialPC#serial-communication-with-a-pc)
2. [Hello World!](https://developer.mbed.org/handbook/SerialPC#hello-world)
3. [Host interface and terminal applications](https://developer.mbed.org/handbook/SerialPC#host-interface-and-terminal-applications)
4. [Terminal Applications](https://developer.mbed.org/handbook/SerialPC#terminal-applications)
5. [Details](https://developer.mbed.org/handbook/SerialPC#details)
6. [Examples](https://developer.mbed.org/handbook/SerialPC#examples)

## Serial Communication with a PC

The mbed Microcontroller can communicate with a host PC through a "USB Virtual Serial Port" over the same USB cable that is used for programming.

This enables you to:

* Print out messages to a host PC terminal (useful for debugging!)
* Read input from the host PC keyboard
* Communicate with applications and programming languages running on the host PC that can communicate with a serial port, e.g. perl, python, java and so on.

## Hello World!

#### Hello World!

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 | #include "mbed.h"    Serial pc(USBTX, USBRX); // tx, rx    int main() {      pc.printf("Hello World!\n");  } |

## Host interface and terminal applications

Your mbed Microcontroller can appear on your computer as a serial port. On Mac and Linux, this will happen by default. For Windows, you need to install a driver:

#### Windows

See [Windows-serial-configuration](https://developer.mbed.org/handbook/Windows-serial-configuration) for full details about setting up Windows for serial communication with your mbed Microcontroller

It is common to use a terminal application on the host PC to communicate with the mbed Microcontroller. This allows the mbed Microcontroller to print to your PC screen, and for you to send characters back.

* [Terminals](https://developer.mbed.org/handbook/Terminals) - Using Terminal applications to communicate between the Host PC and the mbed Micrcontroller

Some terminal programs (e.g. TeraTerm) list the available serial ports by name. However, if you do need to know the identity of the serial port so that you can attach a terminal or an application to it:

* Windows - Look under the "Ports" section in "Device Manager" (''Start -> Control Panel -> System -> Hardware -> Device Manager''). The name will be ''mbed Serial Port (COMx)'', where ''x'' is the number of the COM port allocated.
* Mac OS X - Use the command ls /dev/tty.usbmodem\*
* Linux - Use the command ls /dev/ttyACM\*

## Terminal Applications

## Details

Communication over the USB Serial port simply uses the standard [Serial](https://developer.mbed.org/handbook/Serial) Interface, specifying the internal (USBTX, USBRX) pins to connect to the Serial Port routed over USB.

The Serial Interface defaults to a 9600 baud standard serial connection (8 bits, 1 stop bit, no parity), so your host program should be set to the same settings. If you want to communicate at a different standard baud rate, ensure you modify the settings of both the Serial Interface and the Host PC application!

## Examples

#### Echo back characters you type

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 | #include "mbed.h"    Serial pc(USBTX, USBRX);    int main() {      pc.printf("Echoes back to the screen anything you type\n");      while(1) {          pc.putc(pc.getc());      }  } |

#### Connect to your mbed Microcontroller with a Terminal program and uses the 'u' and 'd' keys to make LED1 brighter or dimmer

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 | #include "mbed.h"    Serial pc(USBTX, USBRX); // tx, rx  PwmOut led(LED1);    float brightness = 0.0;    int main() {      pc.printf("Press 'u' to turn LED1 brightness up, 'd' to turn it down\n");        while(1) {          char c = pc.getc();          if((c == 'u') && (brightness < 0.5)) {              brightness += 0.01;              led = brightness;          }          if((c == 'd') && (brightness > 0.0)) {              brightness -= 0.01;              led = brightness;          }        }  } |

#### Pass through characters in both directions between the PC and Serial Port

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 | #include "mbed.h"    Serial pc(USBTX, USBRX);  Serial uart(p28, p27);    DigitalOut pc\_activity(LED1);  DigitalOut uart\_activity(LED2);    int main() {      while(1) {          if(pc.readable()) {              uart.putc(pc.getc());              pc\_activity = !pc\_activity;          }          if(uart.readable()) {              pc.putc(uart.getc());              uart\_activity = !uart\_activity;          }      }  } |

#### The C stdin, stdout and stderr file handles are also defaulted to the PC serial connection

|  |  |
| --- | --- |
| 1 2 3 4 5 | #include "mbed.h"    int main() {      printf("Hello World!\n");  } |

#### Read in to a buffer

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 | #include "mbed.h"    DigitalOut myled(LED1);  Serial pc(USBTX, USBRX);    int main() {      char c;      char buffer[128];        pc.gets(buffer, 4);        pc.printf("I got '%s'\n", buffer);  } |

#### Troubleshooting

If you have having difficulties with USB serial communication:

* Make sure you have installed the driver if you are working on Windows - [Windows Serial Configuration](https://developer.mbed.org/handbook/Windows-serial-configuration)
* Learn how to use the [Serial](https://developer.mbed.org/handbook/Serial) port
* Read up on using [Terminals](https://developer.mbed.org/handbook/Terminals) programs

**If you have any problems, or think this tutorial could be improved, please tell us in the**[**Forum**](https://developer.mbed.org/forum)**!**