Q▪Kernel™

Quick Start Guide

Version 6.0-3363

Q▪Kernel™ is a product of QuasarSoft Ltd.

License

Q-KernelFree is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License (version 3) as published by the Free Software Foundation and modified by the QuasarSoft Ltd. exception.

> The QuasarSoft Ltd. EXCEPTION
>
> You may not exercise any of the rights granted to You in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Program for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.

Q-KernelFree is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the full license text at the following link <http://www.quasarsoft.com/license.html>

For the purpose of applying the license to this document, I consider "source code" to refer to this document source (.docx) and "object code" to refer to the generated file (.pdf).

QuasarSoft Ltd

312-5th Avenue Suite No. 354

Cochrane Alberta T4C 2E3

Canada

Tel. +1 (403) 450 3482

www.quasarsoft.com

Starting with Q▪Kernel™

The simplest way to start with *Q▪Kernel*™ is to run the Blinky example and play with it. To do this first install the code and then use MPLABX to run the code.

Installing *Q▪Kernel*™

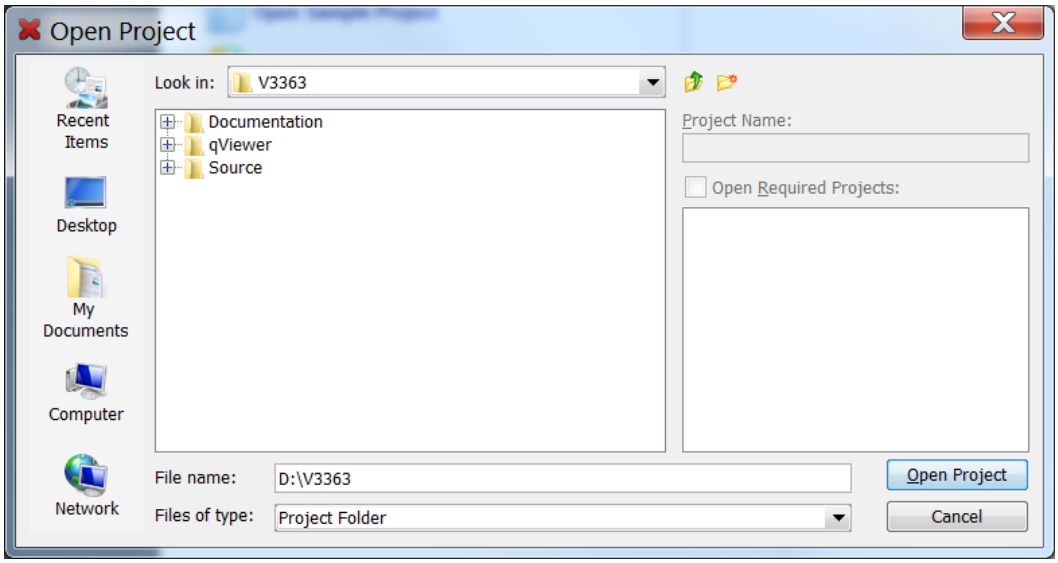
Download the code from the Website. This is a zip file. Extract the zip file with your favorite tool and place it in a directory structure you prefer, for example "c:\qKernel\". The unzip operation creates a directory for the version and under this 3 directories. The structure would look as follows:

V3363
-------- Documentation
-------- qViewer
-------- Source
-------- ------Pic24_MPLAB.X
-------- ------Pic32_MPLAB.X
-------- ------qCrc32c.c
-------- ------qDateTime.c
.....
.....
.....
..... All other sources
.....
.....
-------- ------qUSec.c

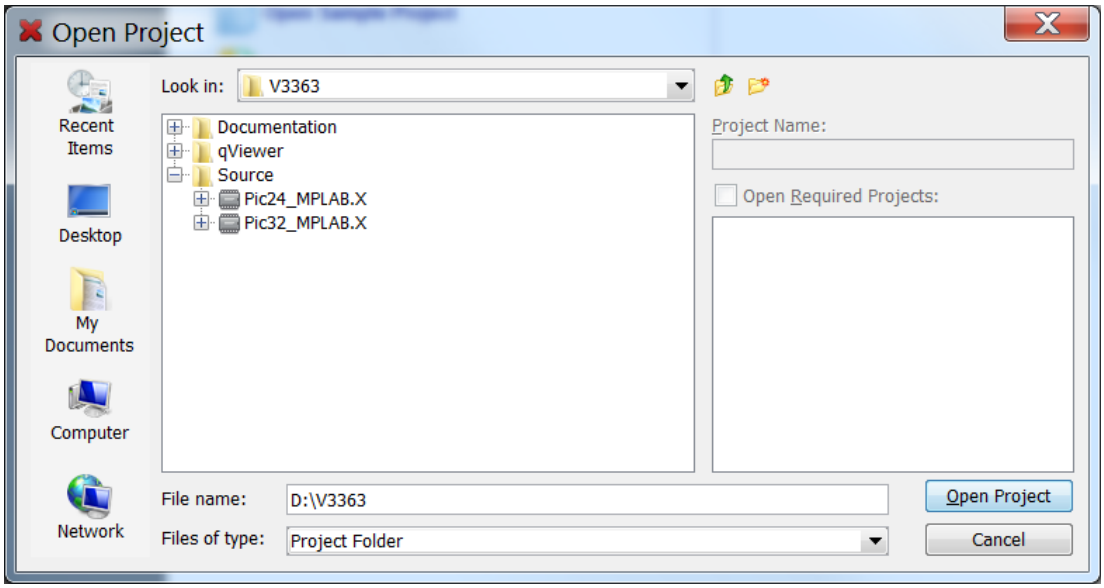
You can find the documentation under the documentation folder.

Running Blinky

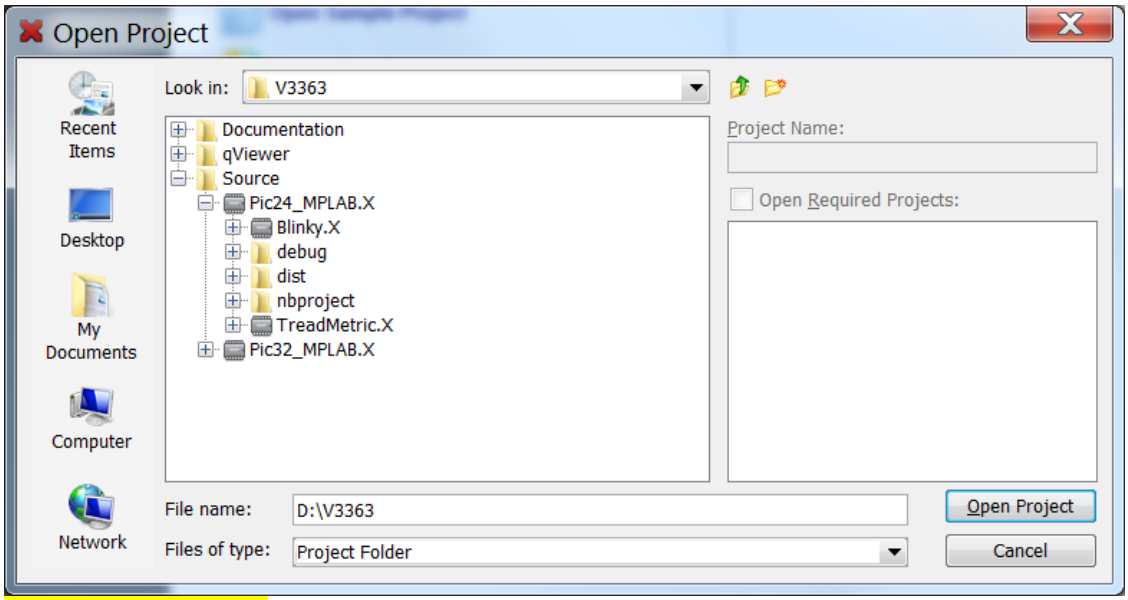
Open MPLAB-X and go to open project, select the installation directory and select first the source, then extract the port directory as shown in the next screen shots.



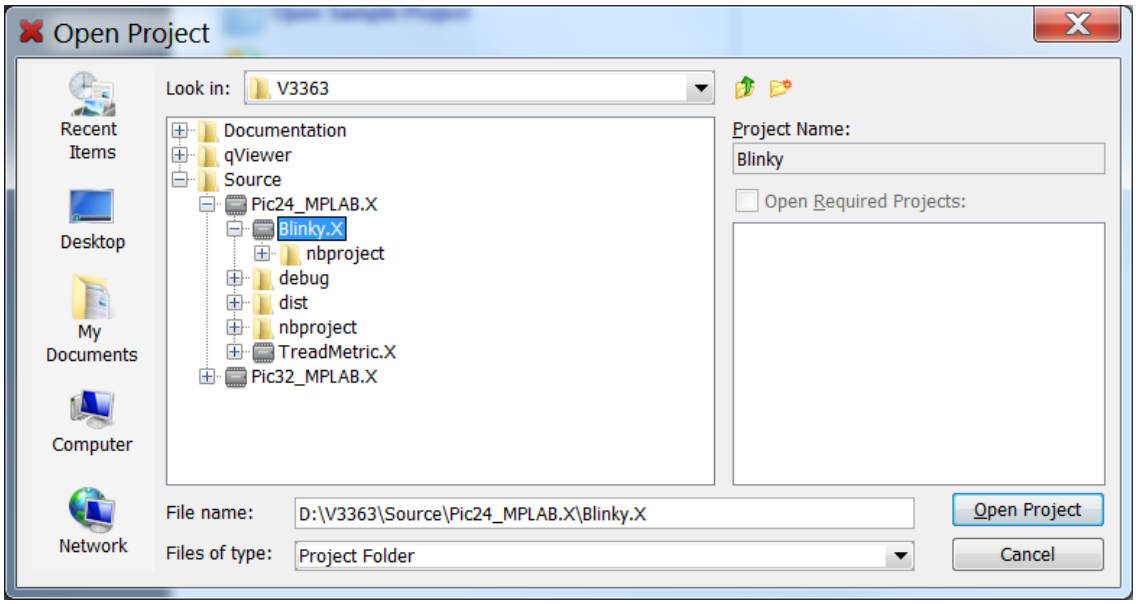
Now click the + to open the source



Now click the + to open the PIC24 or PIC32 port



Now select Blinky.X



And press the open Open Project button.

You are ready to look at your first program.

How to continue?

There are a number of manuals that you should read. Start with the User Guide. Read up about threads, how to create them, sleep, etc. It is very strait forward.

Try to use another mechanism. The most use mechanism is the Mutex, which stands for Mutual Exclusive. If you want to protect access to a specific memory variable like a counter, you have to lock the mutex, increase the counter and unlock it. If you have a multi-threaded application you have to protect your global variable. Don't forget that

If you have an existing application have a very good look to all your global variable. A well written application does not have much global variable and they are protected. Often you will see that global variable don't have to be global, they can be static to a peace of code.

If you have never worked with a RTOS it feel a bit complicated but over time you will learn that it is actually very simple.

Look at Blinky. There are two threads and both threads don't have to know the timing of the other. If you don't have an RTOS you have to design the timing of your system. While Blinky is simple it shows that every thread only cares about its own timing.

Blinking 2 LED's shows how a RTOS can help because every LED thread takes care of its own timing.

Without a RTOS the user has to manage the timing of the system as a whole.

Write the Blinky functionality without a RTOS and you will see that a RTOS helps you to get the timing right.

If you have done that change the program to a blink time of 214 uSec for LED1 and 34523 uSec for LED2. Also make the same changes in Blinky with Q-Kernel.