MB2K2 Firmware Build and Flashing Guide

Introduction

This guide covers the installation of the Xmos xTIMEcomposer toolchain which is used to flash and reprogram the system together with a walkthrough on building the firmware and flashing this to the hardware.

Prerequisites

In order to talk to the XU216's JTAG interface for debugging and programming the on board flash memory it will be necessary to obtain an Xmos 'XA-XTAG' USB debug adaptor. These are really available from component distributors such as Digi-Key for about $20. Octopart has a list of distributors and stock here. The XTAG plugs vertically into the 20 pin header on the MB2K2 PCB.


Step by step guide

It's recommended to use Windows 10 for firmware development on the MB2K2.

1) Download and install version 14.4.1 of the Xmos xTIMEcomposer toolchain from here. the 14.4.1). You will need to register for a free account, make a note of the username/password as these will be used later.

Run the installer and except all the default options. At the end of the process there should be icons on the desktop for the xTIMEcomposer command prompt, the IDE and a link to documentation.

2) Go to the HackaDay page for the MB2K2 project here and follow the link for the MB2K2_V0.90-release. Download the latest version of the distribution using the 'clone or download' button or if preferred clone the repository instead.
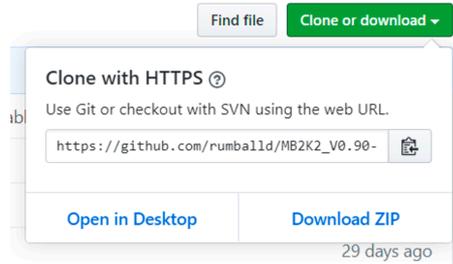
Unzip the dowloaded file to a convenient location, the contents of the unzipped folder should be something like this :-


Note that there is documentation for xTIME composer in the 'firmware/data sheets & app notes etc' folder in the documentation.

   3) Run the xTIME Composer Studio (Community_14.4.1) application (IDE). Note that the IDE needs a 32 bit Java Runtime Environment (JRE) to be installed. Once the IDE has started accept the default workspace location and sign in using the credentials for your Xmos account. Then close the 'XMOS Welcome' tab and select the 'edit' perspective. At this point the window should look something like this :-
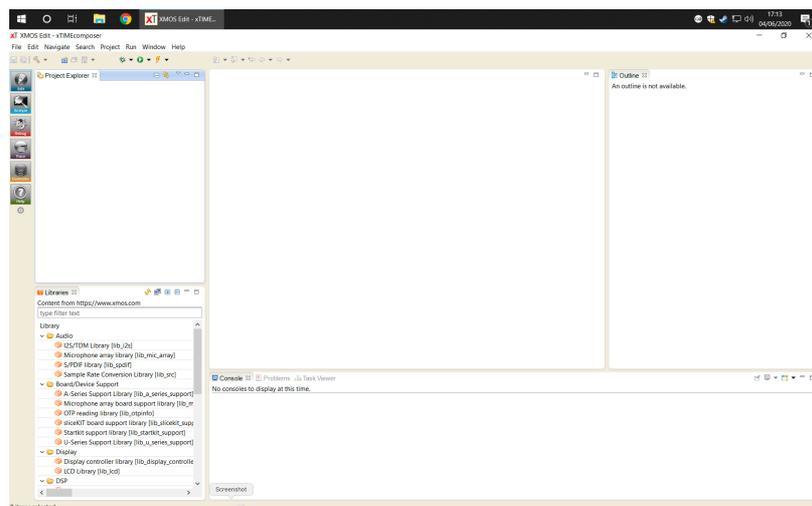
   4) Import the MB2K2 project. Select 'import' from the file menu, then expand the 'general' folder icon and select 'Existing Projects into Workspace' then hit 'next'.
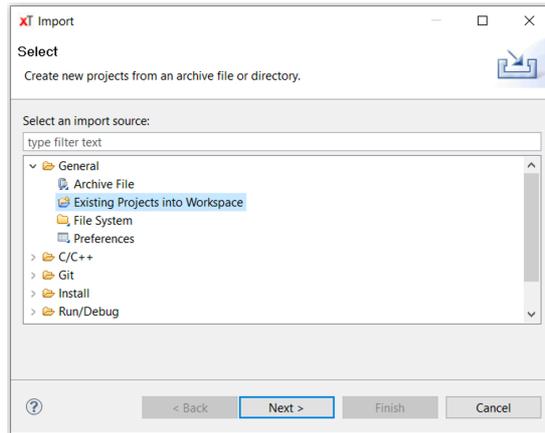
Select the 'archive file' option and browse to the project archive in the 'firmware' folder in the MB2K2 distribution package and select 'open'. There will now be a list of project modules looking something like this :-
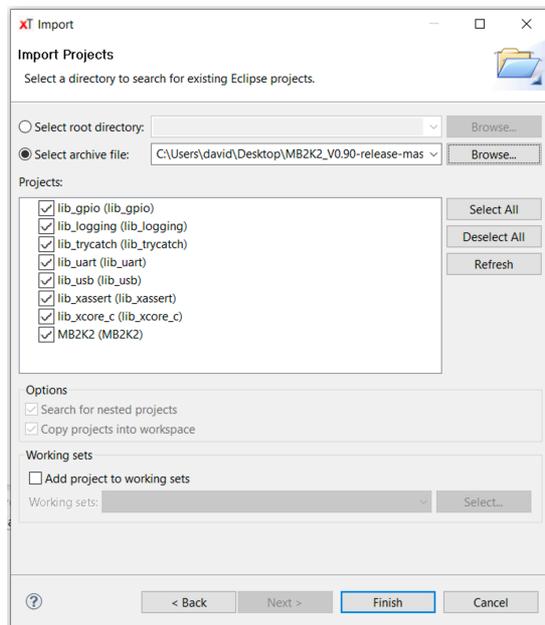
Hit 'finish' and the project will be imported into the workspace. The 'project explorer' tab should now look like this :-
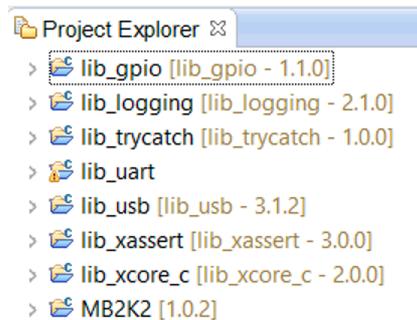
The source file structure consists of a number of Xmos lib projects together with the MB2K2 source folder :-

```
∨ ⚙C > MB2K2 [MB2K2 V1_PCB]
    > ◉ Installed Targets
    > ⚡ Binaries
    > ⓗ Includes
    > 📁 > bin
    > 📁 boot                    - boot image and build tools
    > 📁 > drivers               - drivers for USB CDC class for Windows7
    > 📁 > mon09                 - MON09 source code and build tools
    > 📁 promdisk                - promdisk image DSK (used by xFLASH)
    ∨ 📁 > src
        > 📁 ACIA                - serial port source
        > 📁 CPU                 - 6809 source
        > 📁 GDC                 - uPD7220 source
        > 📁 PIA                 - 6821 and PS/2
        > 📁 promdisk            - PROMdisk source
        > 📁 RAM                 - decode/RAM handler source
        > 📁 ramdisk             - RAMdisk source
        > 📁 RTC                 - RTC and I2C interface source
        > 📁 > USB               - XUD USB interface source
        > 📄 main.xc
      📄 Makefile
    > 🎛 mb2k2_v1.xn             - hardware description file
```
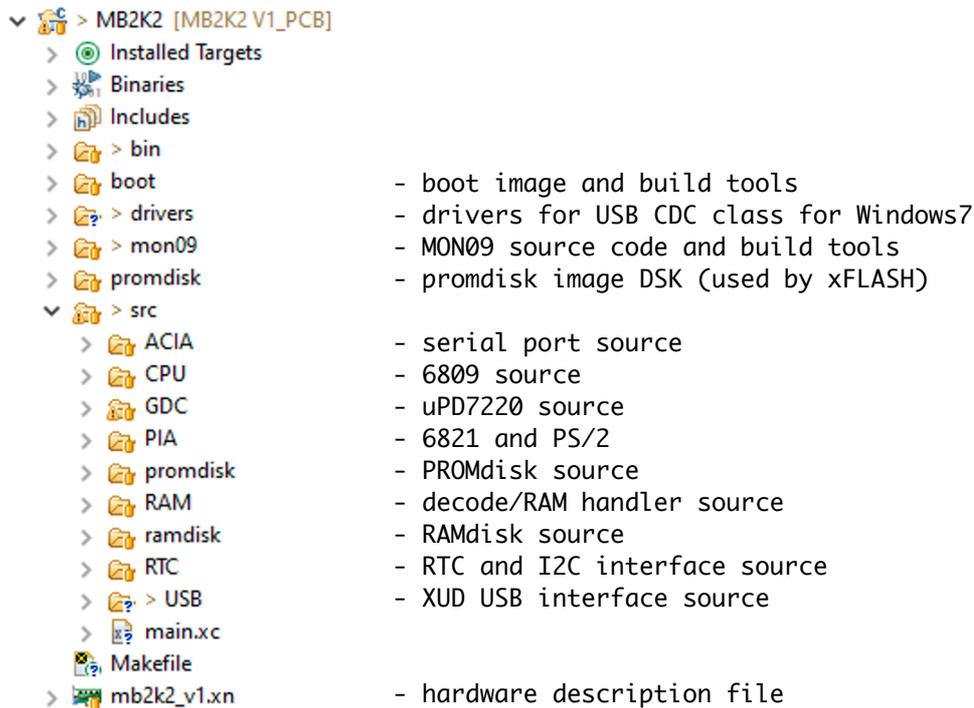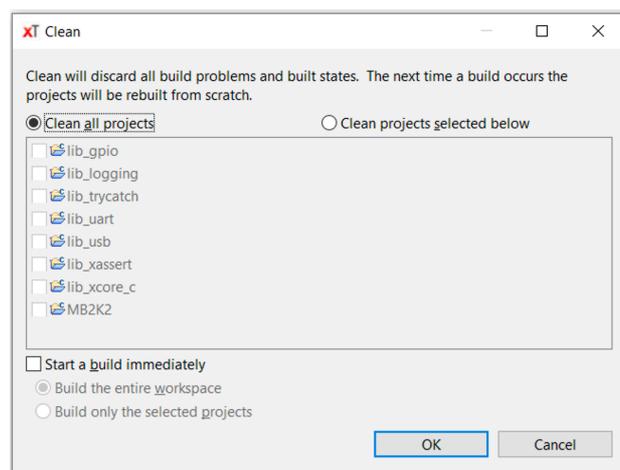
The use of the IDE for editing the code is outside the scope of this document but full details can be found in the 'xTIMEcomposer' User Guide which can be found in the 'firmware/data sheets & app notes' folder in the distribution.

5) Clean the project. Select 'Clean…' from the 'Project' menu. Select 'clean all projects' and unselect 'Start a build immediately' the hit 'OK'. Only for the first time after the project is imported, clean the project a second time.

```
xT Clean                                    —    □    ×

Clean will discard all build problems and built states.  The next time a build occurs the
projects will be rebuilt from scratch.

◉ Clean all projects              ○ Clean projects selected below

  ☐ 📘lib_gpio
  ☐ 📘lib_logging
  ☐ 📘lib_trycatch
  ☐ 📘lib_uart
  ☐ 📘lib_usb
  ☐ 📘lib_xassert
  ☐ 📘lib_xcore_c
  ☐ 📘MB2K2

☐ Start a build immediately
  ◉ Build the entire workspace
  ○ Build only the selected projects

                                   [ OK ]    [ Cancel ]
```

6) Build the project. Expand the 'MB2K2' and then 'src' folders and select 'main.xc'. The build icon (hammer) in the toolbar will then become solid and a build configuration can be selected from the list.
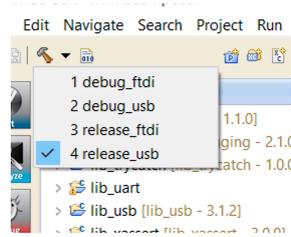
There are two classes of build configurations in the project, 'usb' which uses the USB connection for serial connections and 'ftdi' which replaces the USB interface with a pair of buffered UARTs. Each is further split into 'debug' which has optimisation set to -O0 and adds support for the use of debugPrintf() and 'release' which has optimisation set to -O3 and removes all debugging support. The first class of configurations

The release configs should be used when not debugging as the compiled code runs 3-5 times faster than the debug configs!

For now, use the 'release_usb' configuration. Click on the pulldown arrow next to this and select 'release_usb' and the build will start.

Progress in the build will be shown and status messages/warnings/errors will appear in the Console tab at the bottom of the screen. With a fresh install there should be no errors and the warning messages can be safely ignored.

When the build is complete then a summary of the resource usage in the two XU216 tiles will be shown something like this :-

```
Creating mb2k2_release_usb.xe
Constraint check for tile[0]:
  Cores available:            8,    used:          7 .  OKAY
  Timers available:          10,    used:          7 .  OKAY
  Chanends available:        32,    used:         16 .  OKAY
  Memory available:      262144,    used:     234116 .   OKAY
    (Stack: 5908, Code: 71420, Data: 156788)
Constraints checks PASSED.
Constraint check for tile[1]:
  Cores available:            8,    used:          7 .  OKAY
  Timers available:          10,    used:          9 .  OKAY
  Chanends available:        32,    used:         25 .  OKAY
  Memory available:      262144,    used:     259872 .   OKAY
    (Stack: 35636, Code: 15256, Data: 208980)
Constraints checks PASSED.
xmap: Warning: More than 6 cores used on a tile. Ensure this is not the case on tile running XUD.
xmap: Warning: More than 6 cores used on a tile. Ensure this is not the case on tile running XUD.
Build Complete
```

The map warnings can be ignored but note that the memory for the two tiles is quite full. This should be borne in mind when modifying the firmware. (see the Firmware notes in section 3 of the User Notes).

7) At this point the compiled binary may be loaded onto the hardware or placed into the flash memory and these operations will use the XA-XTAG adaptor to communicate with the XU216 via it's JTAG interface. If not there already the XTAG should be plugged into the MB2K2 and it's microUSB cable connected to the host computer. Note that it's safe to connect and disconnect the XTAG when the MB2K2 is powered on providing that the usual precautions are taken to avoid static discharges.

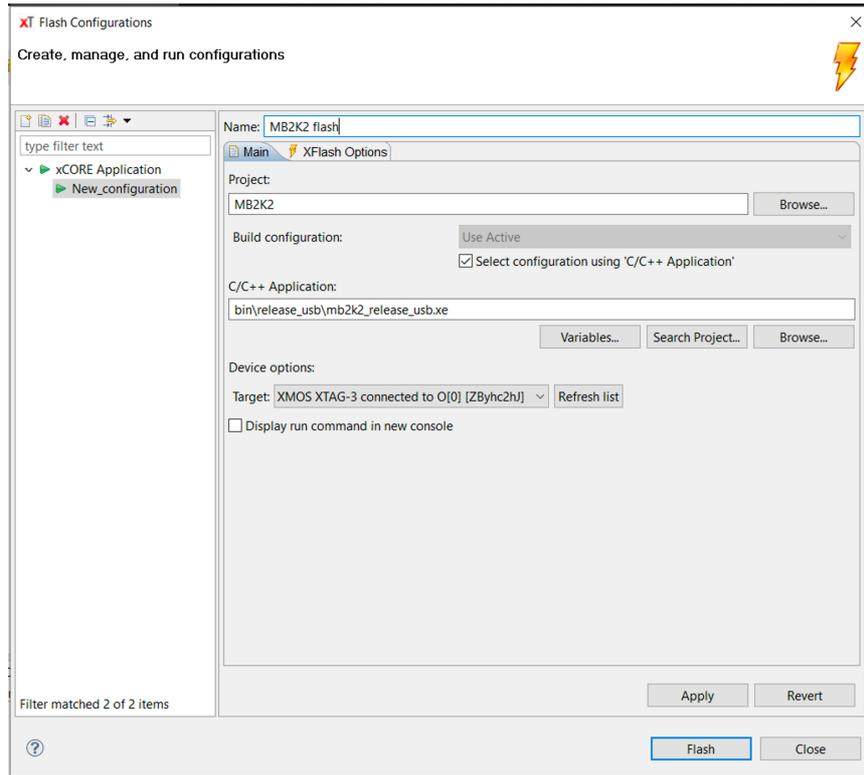## Debugging and flashing the firmware
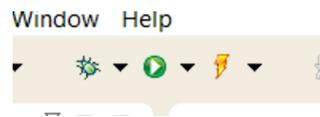
The 'Debug' toolbar icon (on the left hand side above) will load the compiled firmware onto the device's RAM via the XTAG adaptor. Note that as this uses the RAM of the XU216 the firmware will be erased when power is removed. A configuration needs to be set up the first time debug is used. Hit the dropdown beside the debug button and choose 'Debug Configurations…', double click on 'xCORE Application' and name the new config. Many of the settings will be pre filled but it may be necessary in the 'main' tab to browse and select the project and application. Check that the XTAG is selected and showing that it is connected to the target device and lastly hit 'apply' to save the changes.
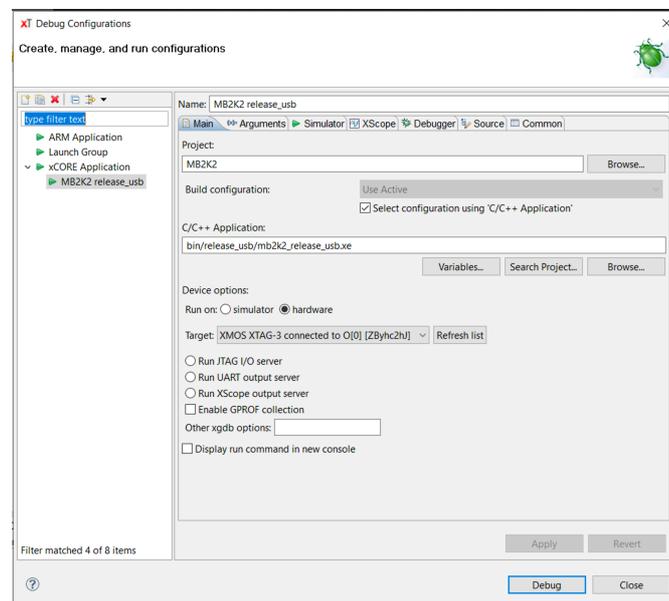

Now hit 'Debug', the console will show the code being loaded via the XTAG and the IDE will switch to the debug perspective. The use of the IDE for debugging is outside the scope of this document but full details can be found in the 'xTIMEcomposer' User Guide which can be found in the 'firmware/ data sheets & app notes' folder in the distribution.

In the same manner as debug, it is necessary to create a flash configuration before this function can be used. Hit the dropdown beside the 'flash' button and choose 'Flash Configurations…' then double click on 'xCORE Application' and name the new config. Unlike the debug config there are many settings to be specified for flashing. In the 'main' tab browse and select the project and application if necessary and check that the XTAG is selected and showing that it is connected to the target device. The 'main tag of the config should look something like this :-
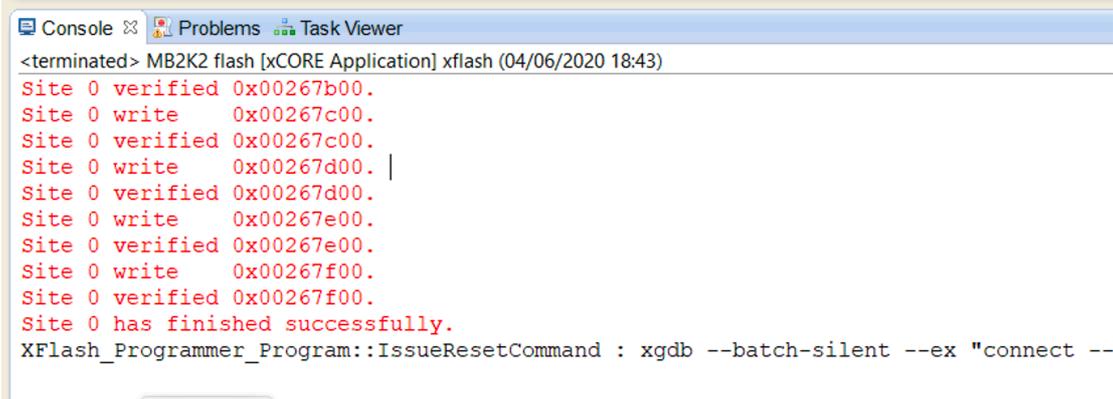
in the 'XFlash options' tab select the 'boot partition size' option and set the value to '0x100000' (1MB). In 'other Flash options' set '--verbose --no-compression --data promdisk/mb2k2pd.dsk'
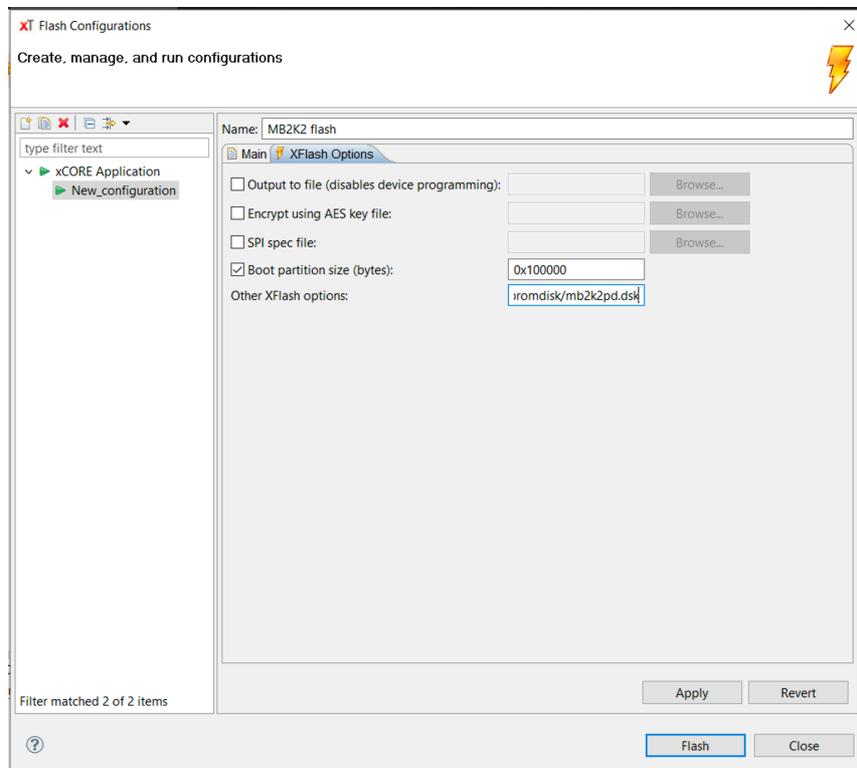
Lastly hit 'apply' to save the changes and then 'flash'. The process will take a few mins to complete displaying progress via a scrolling list of

block addresses as each is written and verified. At the end, the MB2K2 should restart and the MON09 or FLEX prompts should appear.