

## Preface

The purpose of this guide is to give a brief overview of the real-time monitoring and debugging capabilities of the QuickCores Pro8051+ CPU soft core and to outline the configuration procedure that must be followed to enable Keil's μVision2 IDE access to, and use of, these features. Once μVision2 is properly configured, you will be able to use μVision2 to develop and debug applications which use one or more QuickCores Pro8051+ soft core(s) implemented in an Actel ProASIC<sup>PLUS</sup> or Axcelerator FPGA.

Configuring μVision2 for use with the Pro8051+ basically consists of three steps:

- (1) downloading and installing the “**keilqc8051.exe**” self-extracting dll install program from QuickCores web site: [www.quickcores.com](http://www.quickcores.com) ;
- (2) enabling the hardware debug feature by selecting “QuickCores Pro8051” in the Debug menu of μVision2's “Target Options” menu; and
- (3) configuring “Target Setup” for either “Direct” or “TCP/IP” connection and choosing which type of JTAG connection (RS-232, USB, etc) will be used for real-time monitoring and debugging.

---

## Table of Contents

1.00 Introduction.....	2
2.00 Pro8051 Overview .....	2
2.01 Pro8051 Real-Time Monitor (RTM).....	3
2.02 Using the Pro8051+ RTM in Your End Application .....	4
3.00 Enabling the Pro8051+ RTM for use with Keil μVision2 IDE .....	4
3.01 Direct Connection .....	6
3.02 TCP/IP Connection.....	6
3.03 BoxServer .....	7
3.04 JTAG Configuration .....	7
3.05 Querying for, and Selecting, Target CPU.....	8
3.06 Selecting/Changing Target CPUs .....	9
3.07 Activating the μVision2 Debugger .....	9
3.08 QuickCores Multi-Core (modeless) Dialog .....	10
3.09 μVision2 and Pro8051 Real-Time Monitoring .....	11
4.00 Flash Programming Support.....	11

## 1.00 Introduction

As a result of the availability of FPGAs which have an ever increasing number of usable gates, embedded RAM blocks, programmable PLLs and other analog functions, it has now become economically feasible to roll into your FPGA design one or more 8-bit microcontrollers. In the past, one of the biggest hurdles to embedding a microcontroller in an FPGA was the availability of professional quality “C” language development and debug tools suitable for microcontrollers implemented in an FPGA.

QuickCores has addressed these issues in its new Pro8051+ and Pro8051<sup>HYPERCORE</sup> IP soft cores by first incorporating into their architectures a JTAG-accessible real-time monitoring and debug capability, and then by making available to software tool companies the required dlls and related API for accessing these real-time monitoring and debug features.

With their  $\mu$ Vision2 Integrated Development Environment (IDE), Keil Software, Inc. is the first 8051 embedded development tool company to offer a complete software tool suite to support the real-time monitor and debug capabilities of QuickCores Pro8051 family of IP soft cores for both Actel's ProASIC<sup>PLUS\*</sup> and Axcelerator\* family of FPGAs.

Using the self-extracting install program available for free download at QuickCores web site, configuring  $\mu$ Vision2 to work seamlessly with the Pro8051 on-chip real-time monitor and debug capabilities is straightforward and results in a complete embedded development tool package for developing and debugging FPGA-based real-time applications.

## 2.00 Pro8051 Overview

The Pro8051+ and Pro8051<sup>HYPERCORE</sup> are synthesizable (Verilog) IP soft cores which can be easily implemented in an FPGA. The Pro8051 soft core is unique in the industry in that it is built on a proprietary real-time monitor architecture. Whereas in the past, both a software monitor (embedded within the user application) and a dedicated serial port were required to monitor what was happening internal to the CPU, memory, registers, etc., of the standard variety of 8051 microcontroller, QuickCores Pro8051 requires absolutely no software monitor code or dedicated serial port in order to perform real-time monitoring and/or debug functions.

In fact, because of this capability, the Pro8051 is usable without any program memory or dedicated serial port whatsoever. Via the device's JTAG connection, all memory, including direct data, indirect data, external data, program memory, CPU registers, SFRs, etc., are accessible in real-time, whether the CPU is running in real-time or halted at a breakpoint.

When implemented in Actel's ProASIC<sup>PLUS</sup> (APA family) FPGAs, the Pro8051 real-time monitor and debug function is accessed through the device's existing TAP controller and JTAG pins. When implemented in an Actel Axcelerator FPGA, access is

by way of a secondary TAP and JTAG pins implemented from internal programmable logic and user I/O pins.

One way to look at the architecture of the Pro8051 is to visualize a hardware shell whose primary function is to receive, decode and execute instructions presented to it primarily through a JTAG scan path (whether halted at a breakpoint or not) wherein its secondary function is to fetch, decode and execute instructions from a traditional program memory. In short, the Pro8051 CPU is actually primarily a JTAG-accessible real-time (hardware) monitor that just so happens to execute standard 8051 instructions from a secondary instruction bus (the traditional program bus) when it's not doing anything else (which is most of the time).

## 2.01 Pro8051 Real-Time Monitor (RTM)

As previously mentioned, the Pro8051+ is actually a hardware real-time monitor which just so happens to execute standard 8051 instructions (from a 8-bit program memory) when it's not doing anything else. The Pro8051+ real-time monitor receives its monitor instructions via a JTAG scan path.

Just like standard 8051 instructions, the Pro8051+ monitor instructions are specific to which memory/register space is being accessed. Each monitor instruction requires anywhere from three to five CPU clock cycles to fetch, decode and execute, depending on which memory space is being accessed. The Pro8051 real-time monitor does not use DMA or interrupts but rather is a function of the CPU architecture itself.

Some of the functions provided by the Pro8051+ real-time monitor include:

- Read any program memory location
- Write any program RAM location
- Read or write any direct data memory location
- Read or write any SFR location
- Read or write any indirect data memory location
- Read or write any external data memory location
- Force a hardware breakpoint
- Single-step
- "Go From" new PC address
- Call monitor subroutine
- Reset CPU
- Read monitor status

## 2.02 Using the Pro8051+ RTM in Your End Application

Because the Pro8051+ RTM requires zero software overhead in the target side application, and because it does not require any traditional 8051 hardware resources (such as the standard 8051 serial port and/or timers), the Pro8051+ RTM is essentially “free” in terms of traditional 8051 resources. This fact makes it ideal for use in end user applications where a real-time monitoring capability is required.

With the RTM capability, you can actually access and use resources on your target board which may be available but which your target application may or may not actually be using such as an A/D or D/A, serial port, thermometer, digital potentiometer, just to name a few. Since RTM access to these resources don’t require any software overhead on the target application side, you can retrieve and modify the contents of these resources in real-time, without the target application even knowing about it. Once retrieved, you can display the information graphically on the screen of a PC in the form of a pressure or temperature gauge, tachometer, voltage meter, etc., using any number of commercially available PC-hosted instrumentation and control packages such as National Instruments LabView\* or The MathWorks MathLab\* and Real-Time Works\* for example.

To facilitate such use, QuickCores has provided at its web site an API and corresponding “dll” which give you access to these lower level real-time monitor functions. All that is required is a JTAG connection to a PC. This can be accomplished with either a QuickCores FLASH-232+ or FLASH-USB+ pod.

## 3.00 Enabling the Pro8051+ RTM for use with Keil μVision2 IDE

The μVision2 debugger can be configured for use with one or more Pro8051 CPUs implemented on a single FPGA device by downloading and installing the “keilqc8051.exe” self-extracting install program from the QuickCores web site, [www.quickcores.com](http://www.quickcores.com).

During installation, the install utility will automatically self-extract and place the required dlls in the appropriate directories.

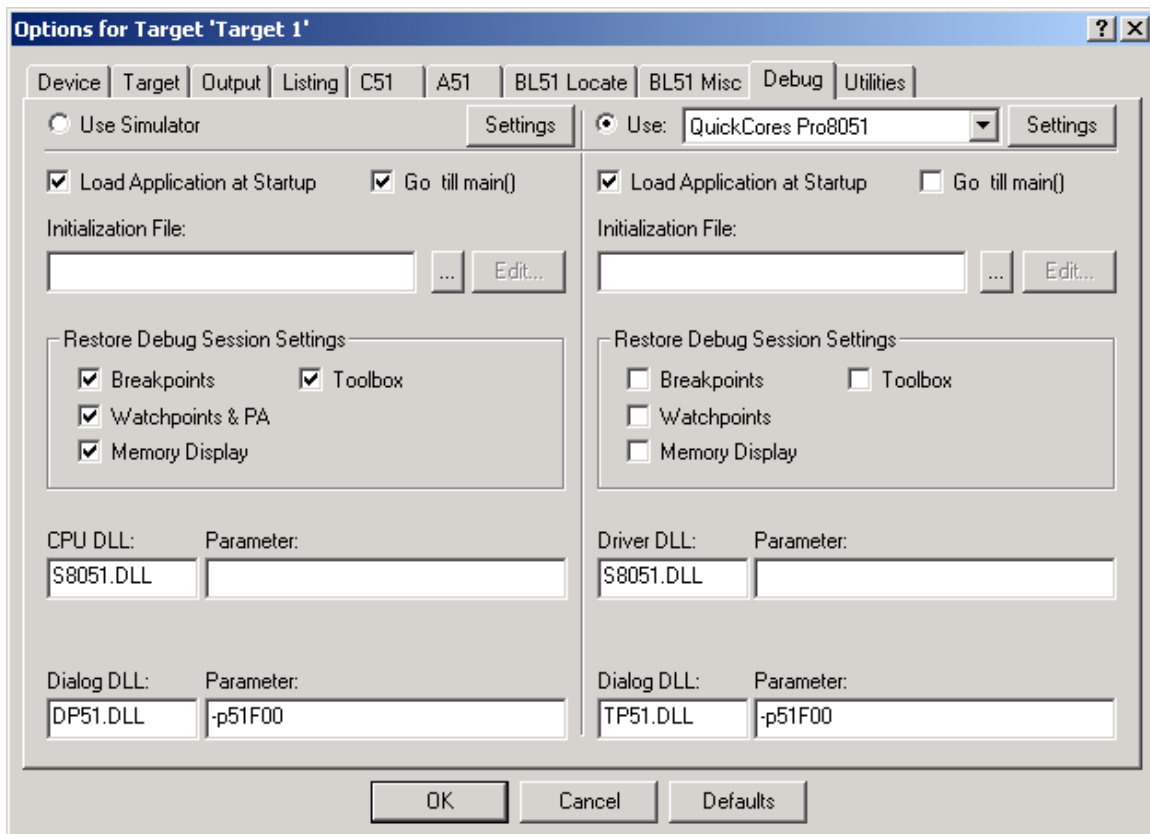
Currently there are four dlls and they are named and described as follows:

- keilqc8051.dll This dll is a “wrapper” which goes around the ppemud32.dll and translates Keil μVision2 API specified commands into equivalent QuickCores real-time monitor/debug commands
- ppemud32.dll This dll provides all lower level access to the target CPU via one of QuickCores debug pods
- sbwsd32.dll This dll is required if you plan to use BoxServer\* for remote monitoring/debugging or want to enable debugging of multiple CPUs by multiple users at different locations. To

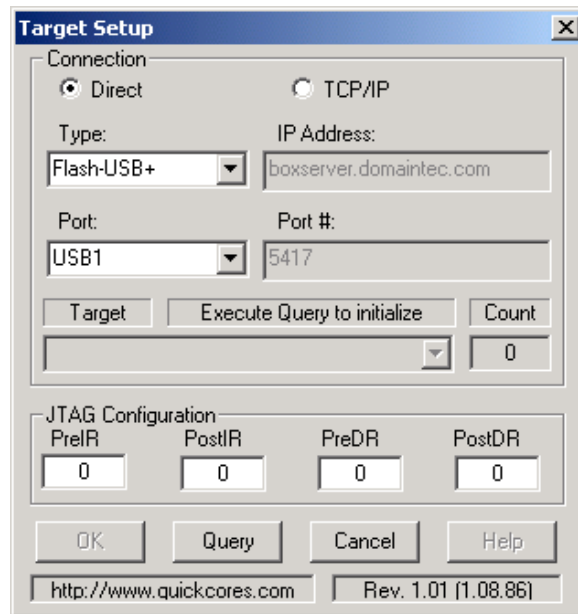
make use of this capability, you must install BoxServer which is the client/target server application

- ftd2xx.dll This dll is required if you are using the QuickCores FLASH-USB+ JTAG device programmer/debug pod

Once the above dlls are properly installed by the installation program, the next step is to launch  $\mu$ Vision2 and click on the “Project” menu item and select “**Options for Target**” to bring up the debugger configuration menu. Within the “Debug” menu, click on the “Use” button and select “QuickCores 8051” from the list as shown in the following example:



Next, click on the “Settings” button to bring up the QuickCores “Target Setup” GUI as shown below:



The Target Setup GUI allows you to properly configure  $\mu$ Vision2 for access to the target Pro8051 CPU(s) via the proper type connection, either Direct or TCP/IP, and the desired physical interface, i.e., USB, RS-232, etc. Once configured, you will be able to access the Pro8051 internals in real-time, regardless of whether the target is running its application or stopped at a breakpoint. The Target Setup GUI also provides a means for configuring the JTAG scan chain in cases where there is more than one device (JTAG TAP/IR register) in the chain. In such cases, the debugger will need to “pad” (preamble/post-amble) each scan with the appropriate number of “bypass” bits.

### 3.01 Direct Connection

Use “Direct” connection if your JTAG programming/debug pod is connected to the same computer as the one you will be running Keil  $\mu$ Vision2 from and if you plan to monitor/debug one CPU at a time.

### 3.02 TCP/IP Connection

Use “TCP/IP” connection if:

- The JTAG programming/debug pod is attached to a computer (server) remotely located on a network;
- There are several target (CPUs) embedded in the target device and there is more than one user; or
- You want to concurrently monitor/debug more than one CPU embedded in the same or different target devices.

To set up  $\mu$ Vision2 for remote access, select the “TCP/IP” and enter the IP address of the workstation on which BoxServer is running. If you are using the “server”

also as your workstation, then just type in the word “local” in the IP Address field of the Target Setup menu.

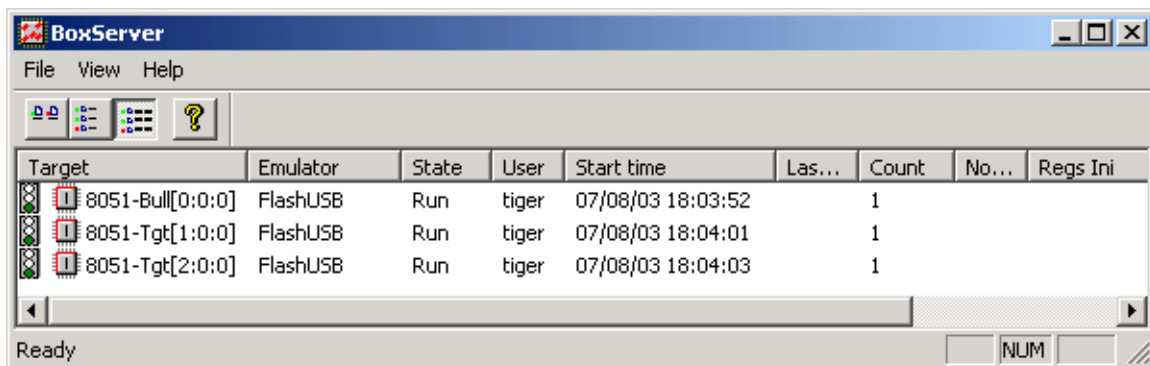
If the IP address field is configured for “local” mode, then the “Port #” is defaulted to “5417”. Otherwise, if you are remotely accessing the target from a different computer, then you will need to enter the Port # for the TCP/IP connection to the server on which BoxServer is running.

### 3.03 BoxServer

If your connection is “TCP/IP”, you must first install and then launch BoxServer. BoxServer is an application that functions as a “server” that manages requests between one or more users and a target device comprising one or more target CPUs. The JTAG programming/debug pod must be connected to the same computer as the one running BoxServer.

When BoxServer is launched, it will perform a JTAG scan of the devices in the scan chain and will list the devices (target CPUs) it has detected. Next to each target listed are readings which show the current status of each device (i.e., running or halted) and whether or not the device is available (i.e., it's not being monitored/debugged/used by someone else).

Below is an example BoxServer window showing three target CPUs.



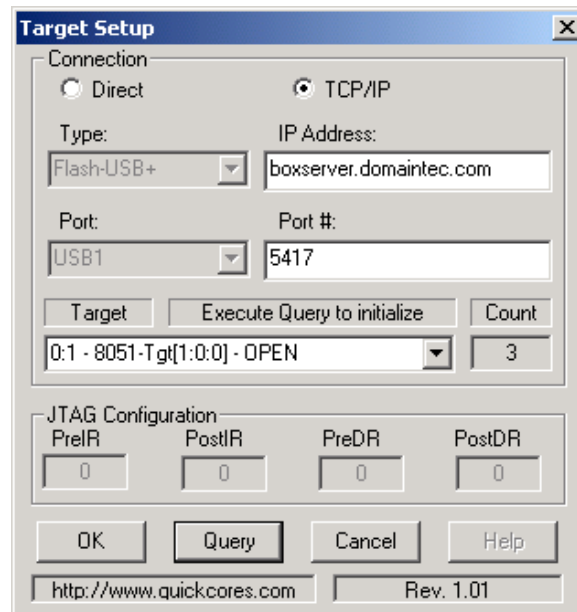
### 3.04 JTAG Configuration

The “JTAG Configuration” block in the Target Setup GUI is used to set up the appropriate number of bypass bits used for each scan. It is not within the scope of this current document to go into JTAG scan theory. For a more detailed explanation, you can download the QuickFLASH user guide from QuickCores web site.

If you plan on monitoring/debugging only one Pro8051+ CPU or one QuickCores multi-CPU Pro8051<sup>HYPERCORE</sup> and it's the only device in the JTAG scan chain, then you can just leave the JTAG Configuration settings to their default value of “0” which is the proper setting for a single device.



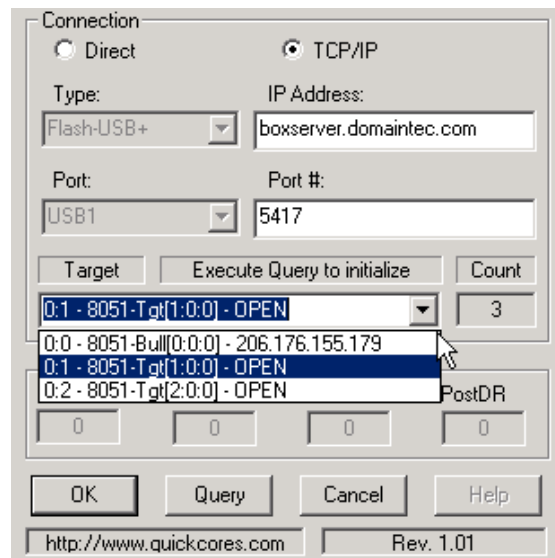
If you are accessing the target device from a remote location (that is, from a work station different from the one in which BoxServer is running), then the fields in the JTAG Configuration block will be grayed out as shown in the following example:



### 3.05 Querying for, and Selecting, Target CPU

The QuickCores Pro8051 dll supports real-time monitoring and debugging of one or more Pro8051 CPUs arranged within the same JTAG scan chain. In order that  $\mu$ Vision2 may know how many and what type CPUs are included in the JTAG scan chain, the “Query” function must be activated by clicking on the “Query” button. When the JTAG query is completed, the number of CPUs that were detected is displayed in the “count” field of the Target Setup menu, and a pull down list of available targets is made available from which you can select the desired target CPU to be monitored or debugged when the  $\mu$ Vision2 debug window is opened as shown in the following example GUI. In this case, three target CPUs have been detected:






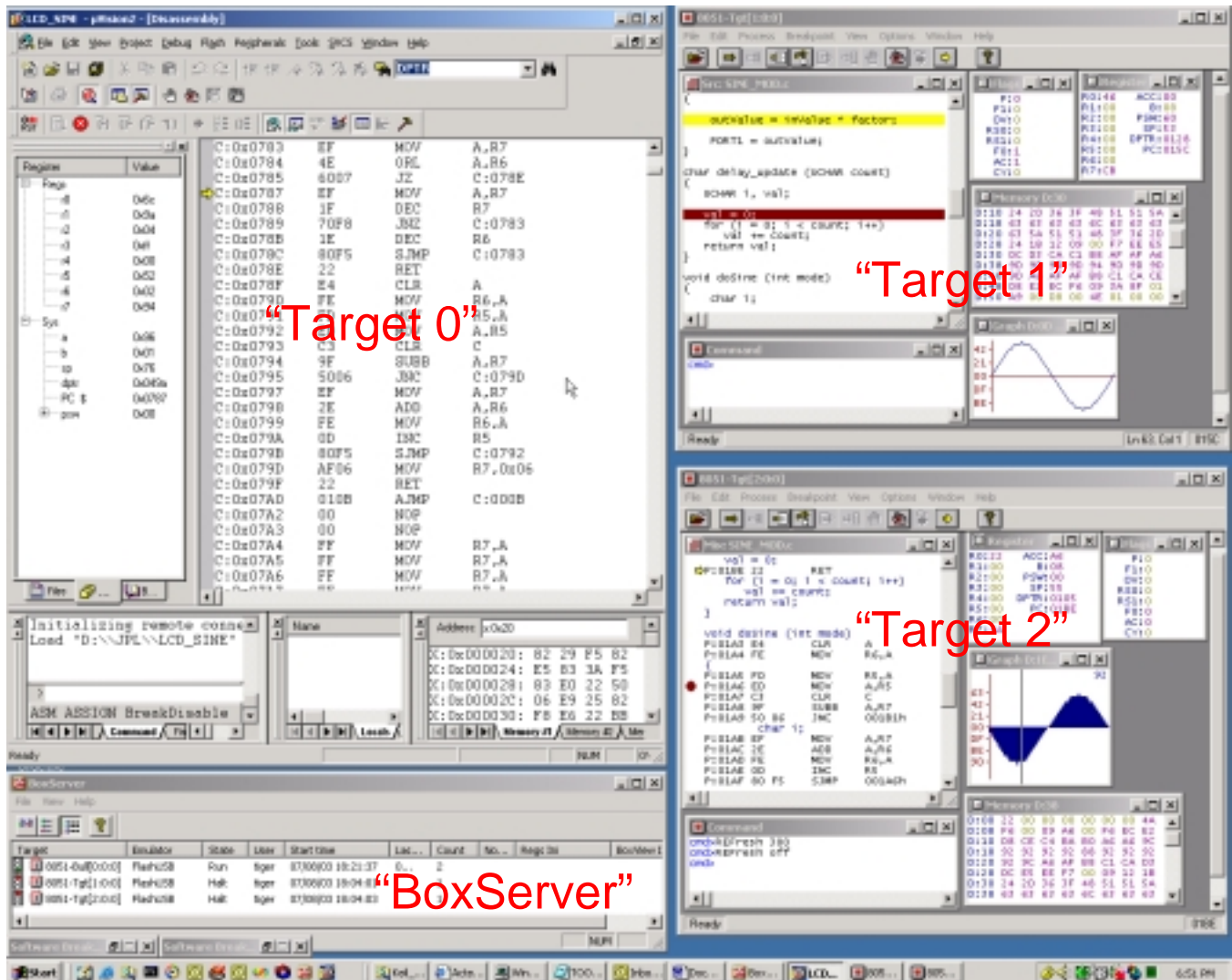
### 3.06 Selecting/Changing Target CPUs

In cases where there is more than one target in the JTAG scan chain and you want to select a target different from the one currently selected, just click on the “Target” pull down menu shown above and select the desired target. When selected, click on the “OK” button.

### 3.07 Activating the $\mu$ Vision2 Debugger

Once  $\mu$ Vision2 is properly configured, just click on the  “debug” button located on the  $\mu$ Vision2 tool bar. After clicking on the debug button,  $\mu$ Vision2 will bring up the debug window for the selected target device as shown in the example given below.

Note that as of this writing,  $\mu$ Vision2 can only display one target CPU at a time. If you need to concurrently debug more than one target CPU at a time, in the BoxServer window (shown in the lower left window of the example image below) simply right-click on the target device you wish to display and a BoxView\* debugger session will be opened automatically for the selected target as shown in the example screen capture below.

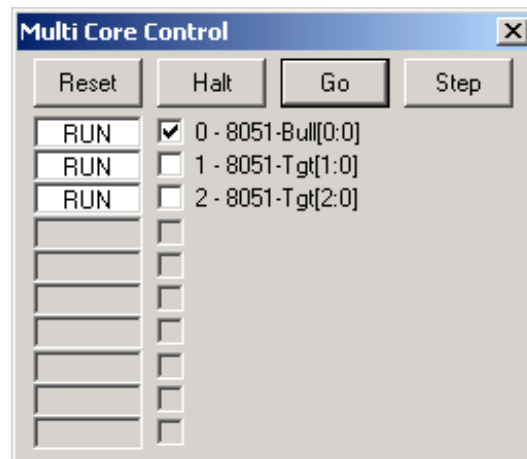


### 3.08 QuickCores Multi-Core (modeless) Dialog

QuickCores has provided a modeless dialog in its dll that will enable you to issue commands to any target CPU in the JTAG scan chain. This comes in handy in cases where you are not using BoxServer or the BoxView debugger but you still want to be able concurrently control multiple CPUs during a monitoring/debug session.

The multi-CPU control menu provides a convenient means for issuing “Reset”, “Halt”, “Go” and “Step” commands concurrently to selected targets. This will allow you single-step multiple CPUs in lock-step with each other.

The multi-core modeless dialog is located in the  $\mu$ Vision2 “Peripherals” menu, and when selected, brings up the following multi-core control GUI:



The above example shows three targets, “Bull”, “Target 1”, and “Target 2” which are all in “RUN” mode. To issue one of the four commands provided in the dialog, click the check box adjacent to the desired target CPU and then click on one of the four command buttons.

If any of the target CPUs are halted, either by way of responding to the “Halt” button, or by way of encountering a previously set breakpoint, the corresponding target’s “RUN” status message will be replaced with that target’s current program counter value.

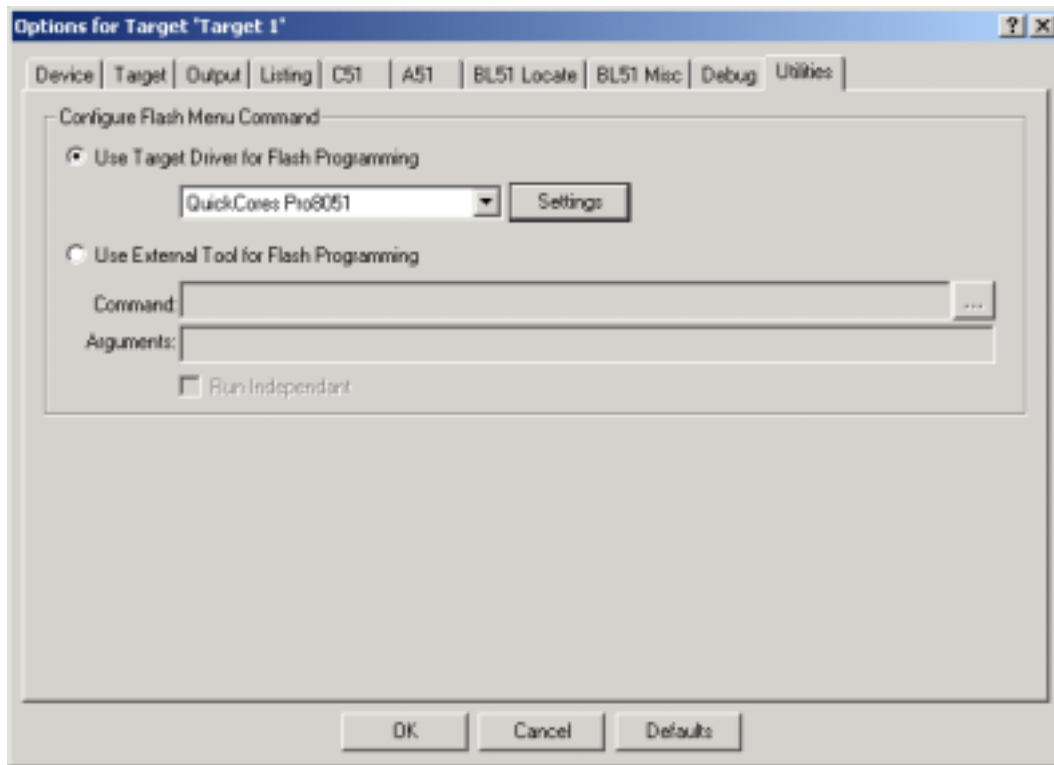
### 3.09 $\mu$ Vision2 and Pro8051 Real-Time Monitoring

As previously mentioned, the Pro8051+ and Pro8051<sup>HYPRCORE</sup> are built on a hardware real-time monitor architecture. This means that you do not have to halt the target CPU prior to changing a value in target device’s memory space. To update/refresh a memory window (including program memory and even while the target is running) just grab the window’s slide bar and slide it up or down. You can also enable the “Periodic Window Update” feature in the within the “View” menu.

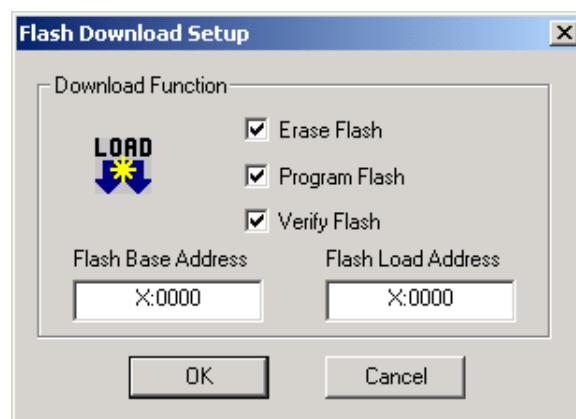
You can also make edits to memory/register contents, including Pro8051 program RAM, on-the-fly, without having to first halt the target.

### 4.00 Flash Programming Support

QuickCores provides a number of FPGA target boards that incorporate a Flash memory device external to the Pro8051 and FPGA. To configure  $\mu$ Vision2 so that you can program the target board’s Flash memory, under the  $\mu$ Vision2 “Project” menu, select “Options for Target” and then click on the “Utilities” tab as shown below:



Click on the “Use Target Driver for Flash Programming” button and select “QuickCores Pro8051 from the drop-down list provided. Next, click on the “Settings” button to bring up the “Flash Download Setup” GUI as shown in the following example:



At the bottom of the Flash Download Setup GUI, two fields are provided: “Flash Base Address” and “Flash Load Address”. The Flash Base Address field allows you to specify where the first location (base address) of the Flash device is physically mapped.

The Flash Load Address allows you to specify the offset from the address specified in the hex records being downloaded where you want the file to be actually loaded.

In most cases you can just leave these fields configured with their default settings of X:0000. If your application is a multi-CPU application wherein you are boot-loading several targets from a single external Flash memory, then you will most likely need to configure the Flash Load Address to have an appropriate offset for storage of each target's application.

Revision 0.02  
July 10, 2003

Copyright © 2003  
**QuickCores IP**  
All rights reserved

QuickCores IP  
811 E. Plano Parkway  
Suite 115  
Plano, TX 75074  
Phone: (972) 578 1121  
Fax: (972) 578 1086  
<http://www.quickcores.com>

---

### **\*Statement on Trademarks**

QuickCores, QuickFLASH, MUSKETEER, FLASH-232+, FLASH-USB+, Pro8051+, Pro8051, and Pro8051<sup>HYPERCORE</sup> are trademarks of QuickCores IP.

BoxView and BoxServer are trademarks of Domain Technologies, Inc.

Actel, ProASIC<sup>PLUS</sup>, Axcelerator and Libero are trademarks of Actel Corporation.

Keil and  $\mu$ Vision2 are trademarks of Keil Software, Inc.

LabView is a registered trademark of National Instruments Corporation.

MathLab and Real-Time Works are registered trademarks of The MathWorks, Inc.

All other trademarks referenced herein are the property of their respective owners.