

---

---

## Technical Notes

# ARC Family On-Chip Emulation

## Contents

Contents.....	1
1 Introduction .....	2
1.1 ARCAngel2 development platform notes.....	2
2 Emulation options.....	3
2.1 Hardware Options.....	3
2.2 Initialization Sequence .....	4
2.3 JTAG Scan Speed.....	6
3 CPU Setup .....	7
3.1 General Options.....	7
3.2 Debugging Options.....	8
3.3 Advanced Options .....	9
4 Access Breakpoints .....	10
5 Troubleshooting.....	10

# 1 Introduction

The ARC's TangentA4 is 32 bit RISC microprocessor which offers great level of flexibility and performance. It is available as a "soft" core, where the user gets the synthesizable code of the fixed part of the microprocessor and can add some specific part with customer-specific instructions. The core can be synthesizable for FPGA or ASIC platforms and thus allow early stage debugging and prototyping.

## **Debug features:**

- Up to 8 hardware breakpoints
- Unlimited software breakpoints
- Hot attach

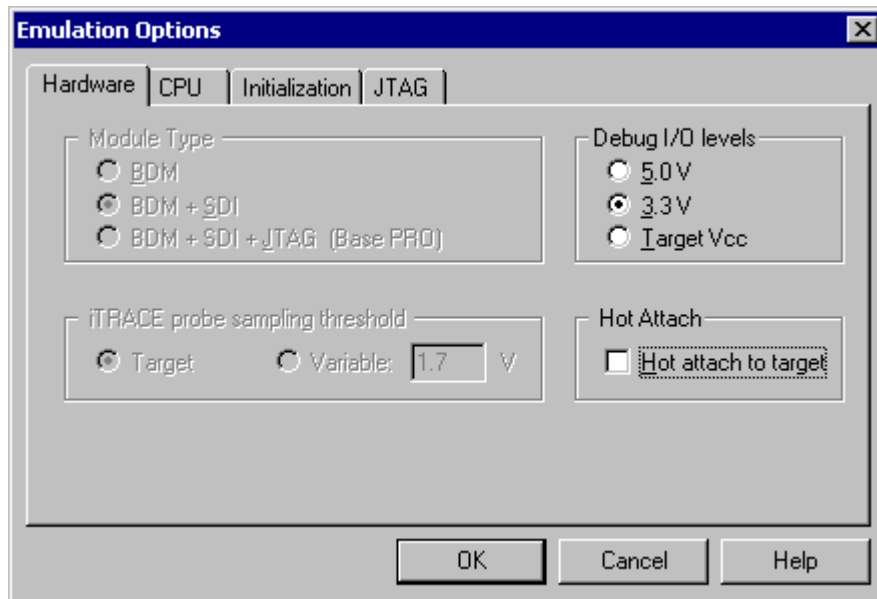
## ***1.1 ARCAngel2 development platform notes***

Whenever a download is performed, first the FPGA configuration file is downloaded. The configuration file must be named 'arcangel.xbf' and must be present in the winIDEA installation directory.

If a new FPGA configuration file is copied to the winIDEA directory, the ARCAngel2 development system should be turned off and on. This ensures that the new FPGA configuration file is downloaded to the development platform.

## 2 Emulation options

### 2.1 Hardware Options



*Emulation options, Hardware pane*

#### **Debug I/O levels**

The development system can be configured in a way that debug (BDM/JTAG) signals are driven at 3.3V, 5V or target voltage levels. When 'Target Vcc' Debug I/O level is selected, a voltage applied to the belonging reference voltage pin (target debug connector) is used as a reference voltage for driving debug (BDM/JTAG) signals. Make sure that the target reference voltage pin is connected when 'Target Vcc' Debug I/O level is selected

#### **Hot Attach**

The JTAG module supports the Hot Attach function. This is a function, which enables the emulator to be connected to a working target device and have all debug functions available.

The procedure for Hot Attach:

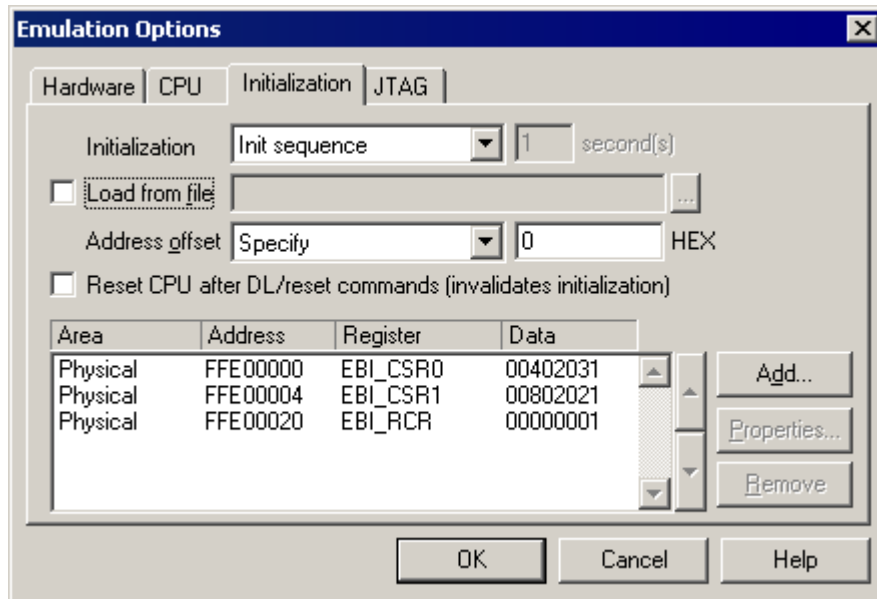
1. The target application should be running.
2. Hot Attach should be selected in the software.
3. A download should be performed, but without the JTAG cable connected. The emulator will be initialized and the ATTACH status will be shown.
4. Connect the JTAG cable.
5. Select the Attach option in the Debug menu. When this option is selected, the emulator tries to communicate through JTAG. If it is successful, it shows the STOP or RUNNING status. At this point, all debug functions are available.
6. When the debugging is finished, the CPU should be set to running and Detach selected from the Debug menu. The status shown is ATTACH. Now the JTAG cable can be safely removed.

## 2.2 Initialization Sequence

Before the flash programming or download can take place, the user must ensure that the memory is accessible. This is very important since there are many applications using memory resources (e.g. external RAM, external flash), which are not accessible after the CPU reset. In that case, the debugger must execute after the CPU reset a so called initialization sequence, which configures necessary CPU chip selects and then the download or flash programming can actually take place. The user must set up the initialization sequence based on his application.

The initialization sequence can be set up in two ways:

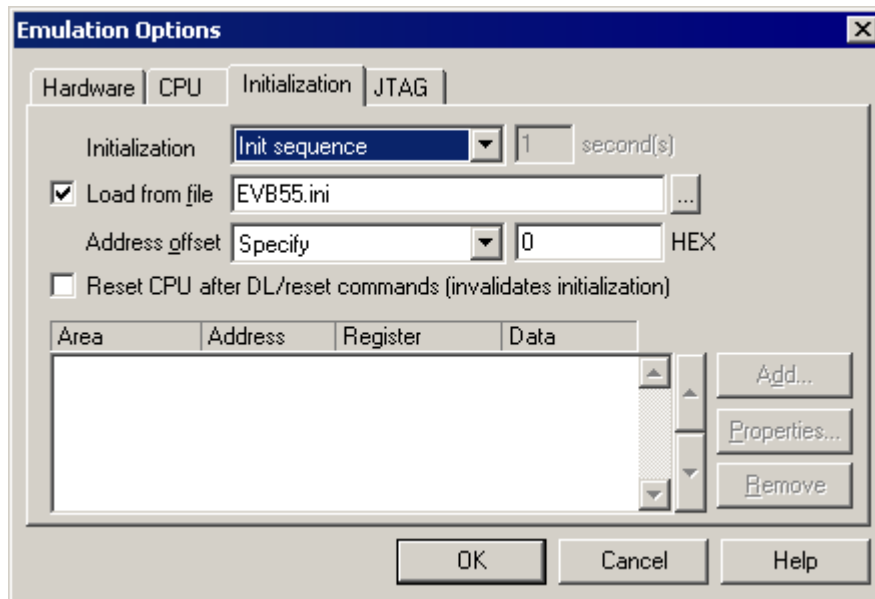
1. Set up the initialization sequence by adding necessary register writes directly in the Initialization page within winIDEA.



2. winIDEA accepts initialization sequence as a text file with .ini extension. The file must be written according to the syntax specified in the appendix in the hardware user's guide.

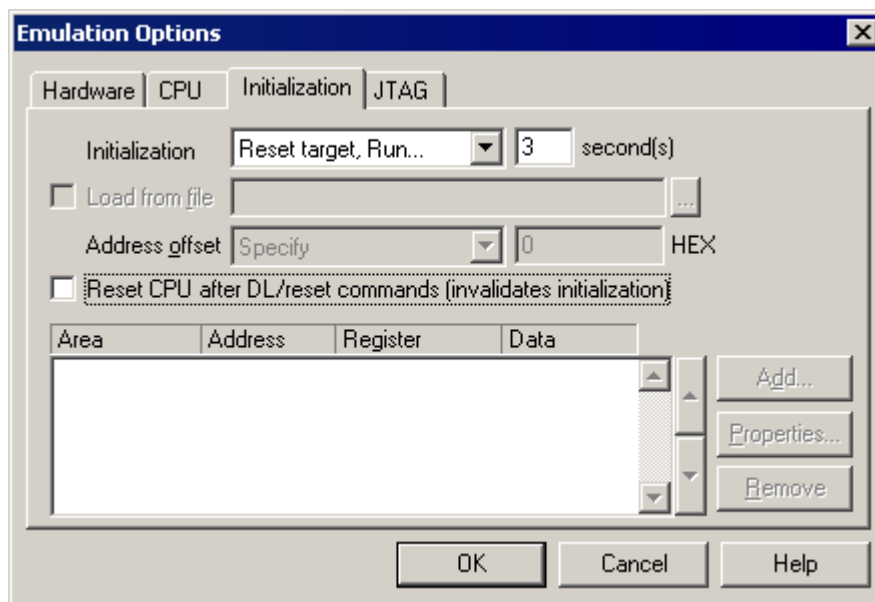
Excerpt from EVB55.ini file for the Atmel AT91M55800 CPU (ARM7TDMI):

```
S EBI_CSR0 L 0x00402031 // CS0 - ext. flash, 4 wait states
S EBI_CSR1 L 0x00802021 // CS1 - ext. SRAM
S EBI_RCR L 0x00000001 // remap internal RAM
```

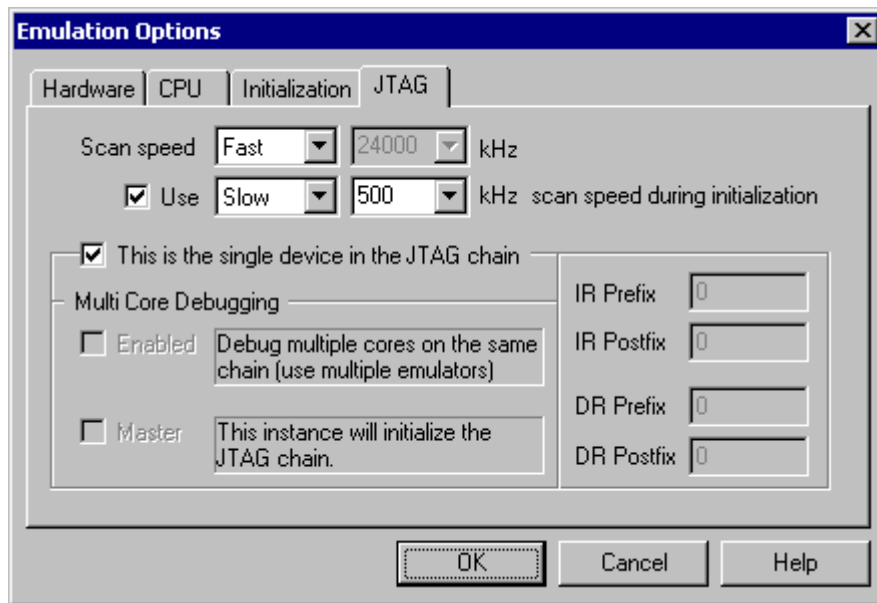


The advantage of the second method is that you can simply distribute your .ini file among different workspaces and users. Additionally, you can easily comment out some line while debugging the initialization sequence itself.

There is also a third method, which can be used too but it's not highly recommended for the start up. The user can initialize the CPU by executing part of the code in the target ROM for X seconds by using 'Reset and run for X sec' option.



## 2.3 JTAG Scan Speed



*JTAG Scan Speed definition*

### **Scan speed**

The JTAG chain scanning speed can be set to:

- Slow - long delays are introduced in the JTAG scanning to support the slowest devices. JTAG clock frequency varying from 1 kHz to 2000 kHz can be set.
- Fast – the JTAG chain is scanned with no delays.
- Burst – provides the ability to set the JTAG clock frequency varying from 4 MHz to 100 MHz.
- Burst+ - provides the ability to set the JTAG clock frequency varying from 4 MHz to 100 MHz

Slow and Fast JTAG scanning is implemented by means of software toggling the necessary JTAG signals. Burst mode is a mixture of software and hardware based scanning and should normally work except when the JTAG scan frequency is an issue that is when the JTAG scan frequency used by the hardware accelerator is too high for the CPU. In general, selecting an appropriate scan frequency usually depends on scan speed limitations of the CPU. In Burst+ mode, complete scan is controlled by the hardware accelerator, which poses some preconditions, which are not met with all CPUs. Consequentially, Burst+ mode doesn't work for all CPUs.

In general, Fast mode should be used as a default setting. If the debugger works stable with this setting, try Burst or Burst+ mode to increase the download speed. If Fast mode already fails, try Slow mode at different scan frequencies until you find a working setting.

---

Note: Burst and Burst+ modes are implemented for PowerPC and ARM CPUs, including XScale.

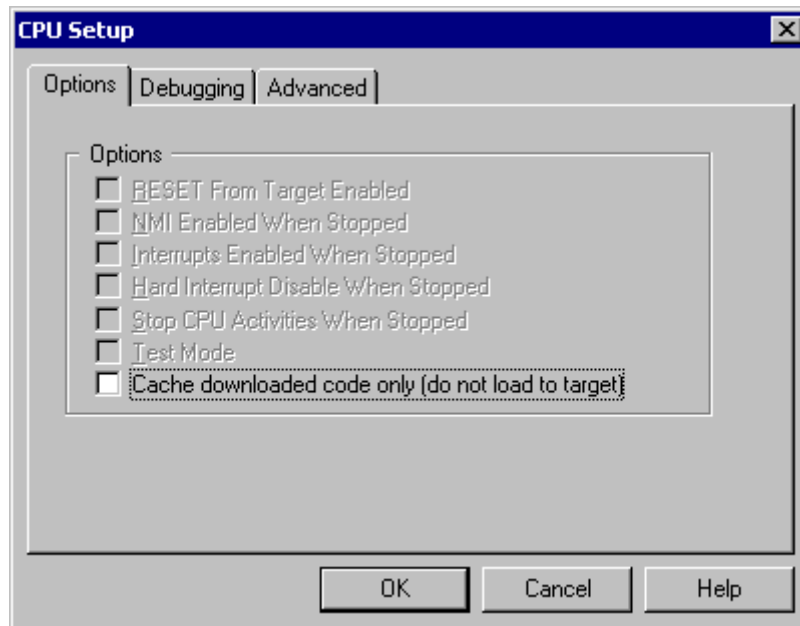
---

### **Use – Scan Speed during Initialization**

On some systems, slower scan speed must be used during initialization, during which the CPU clock is raised (PLL engaged) and then higher scan speeds can be used in operation. In such case, this option and the appropriate scan speed must be selected.

## 3 CPU Setup

### 3.1 General Options



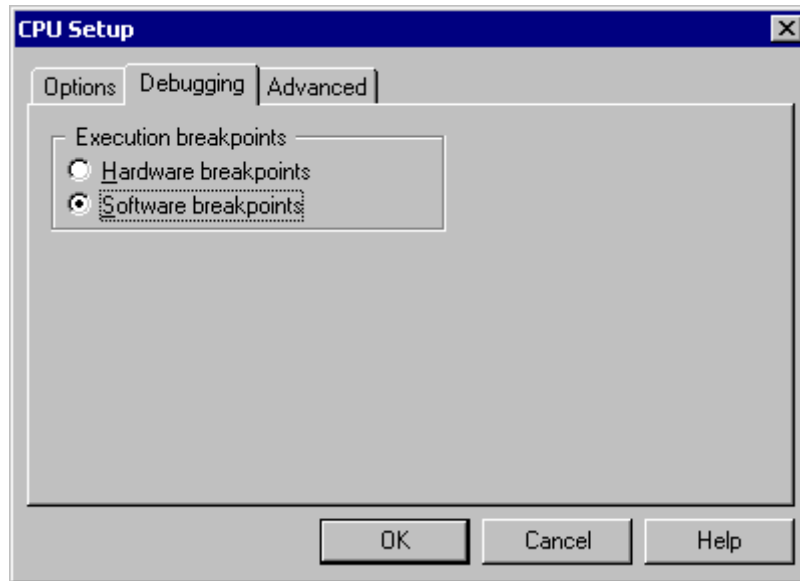
*Debugging options dialog*

#### ***Cache Downloaded Code only (do not load to target)***

When this option is checked, the download files will not propagate to the target using standard debug download but the Target download files will.

In cases, where the application is previously programmed in the target or it's programmed through the flash programming dialog, the user may uncheck 'Load code' in the 'Properties' dialog when specifying the debug download file(s). By doing so, the debugger loads only the necessary debug information for high level debugging while it doesn't load any code. However, debug functionalities like ETM and Nexus trace will not work then since an exact code image of the executed code is required as a prerequisite for the correct trace program flow reconstruction. This applies also for the call stack on some CPU platforms. In such applications, 'Load code' option should remain checked and 'Cache downloaded code only (do not load to target)' option checked instead. This will yield in debug information and code image loaded to the debugger but no memory writes will propagate to the target, which otherwise normally load the code to the target.

## 3.2 Debugging Options



*ARC Debugging options dialog*

### ***Execution breakpoints***

#### *Hardware Breakpoints*

Hardware breakpoints are breakpoints that are already provided by the CPU. The number of hardware breakpoints is limited to eight. The advantage is that they function anywhere in the CPU space, which is not the case for software breakpoints, which normally cannot be used in the FLASH memory, non-writable memory (ROM) or self-modifying code. If the option 'Use hardware breakpoints' is selected, only hardware breakpoints are used for execution breakpoints.

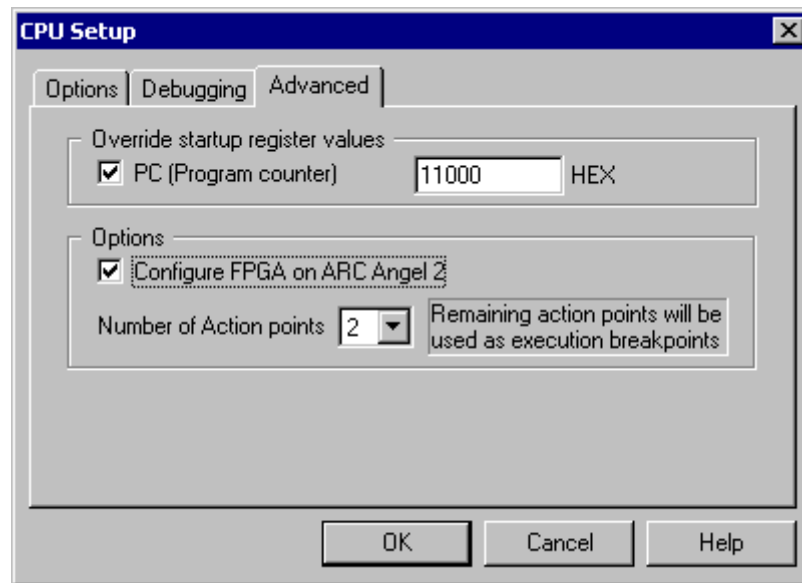
Note that the debugger, when executing source step debug command, uses one breakpoint. Hence, when all available hardware breakpoints are used as execution breakpoints, the debugger may fail to execute debug step. The debugger offers 'Reserve one breakpoint for high-level debugging' option in the Debug/Debug Options/Debugging' tab to circumvent this. By default this option is checked and the user can uncheck it anytime.

#### *Software Breakpoints*

Available hardware breakpoints often prove to be insufficient. Then the debugger can use unlimited software breakpoints to work around this limitation.

When a software breakpoint is being used, the program first attempts to modify the source code by placing a break instruction into the code. If setting software breakpoint fails, a hardware breakpoint is used instead.

### 3.3 Advanced Options



*Advanced On-Chip Emulation Options (ARC)*

#### ***Override startup register values***

This option overrides the default Program Counter reset value with the value set.

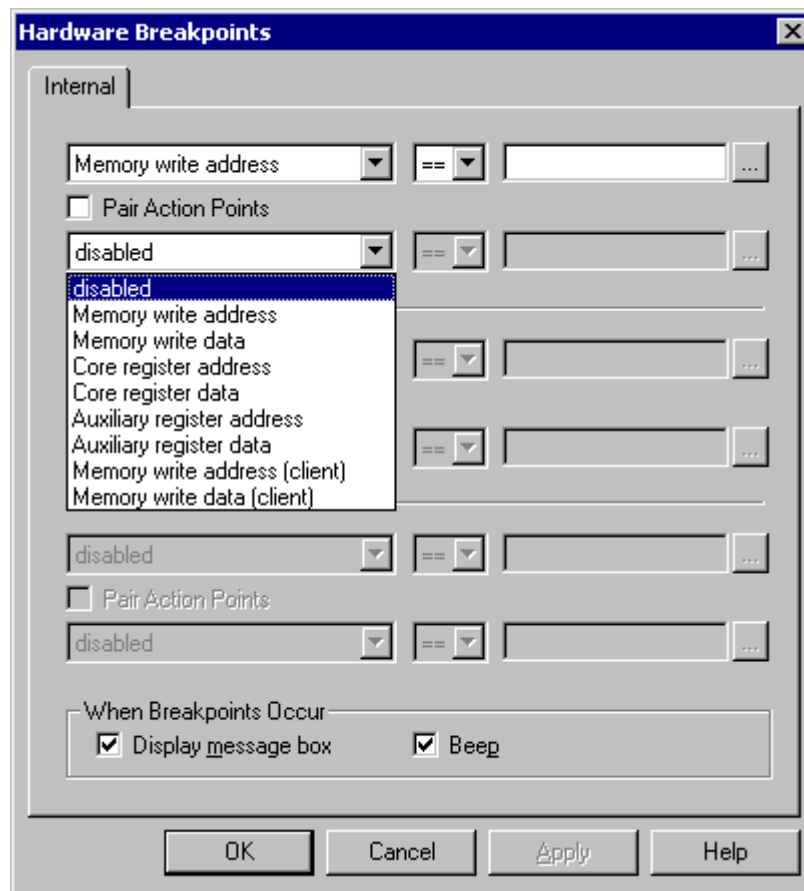
#### ***Configure FPGA on ARC Angel 2***

Please refer to ARC Angel 2 development platform notes for more details.

#### ***Number of Action points***

The specified number of action points will be used as hardware breakpoints, the rest of them will be available for execution breakpoints.

## 4 Access Breakpoints



*ARC Hardware Breakpoints*

Up to four pairs or 8 action points can be used for hardware breakpoints. If the first selection allows to be paired, the “Pair Action Points” option is active – when checked, the second action point in this pair is selected automatically.

## 5 Troubleshooting

Make sure that the power supply is applied to the target debug connector when ‘Target VCC’ is selected for Debug I/O levels in the Hardware/Emulator Options/Hardware tab, otherwise emulation fails or may behave unpredictably.

When performing any kind of checksum, remove all software breakpoints since they may impact the checksum result.

Try ‘Slow’ JTAG Scan speed if the debugger cannot connect to the CPU.

Notes:

Notes: