Infineon XC166/XC2000 Family On-Chip Emulation

Contents

1 Introduction

The Infineon XC166 (C166SV2 core) family includes an innovative debug system, which provides comfortable on-chip debug support (OCDS) to be controlled directly via debug interface pins.

OCDS is used to debug the user software running in the customer's system environment. The OCDS is controlled by an external debugging device via the debug interface.

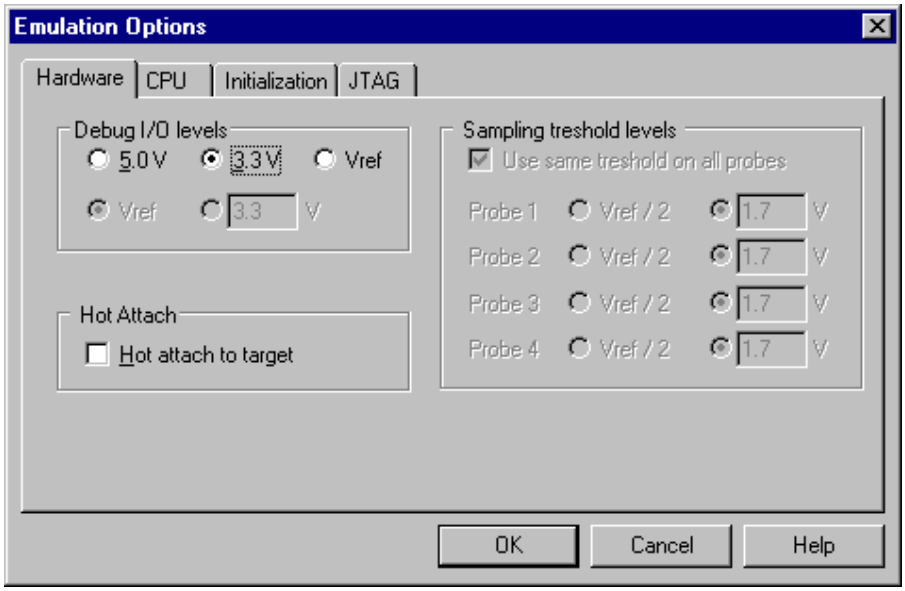
All the debug functions are controlled by the debug interface, the OCDS function control and by the debug IO control (called Cerberus), which provide all the functionality necessary to interact between the debug interface (the external debugger) and the internal system including the OCDS controller.

Debug Features

- On-chip breakpoints
- Unlimited software breakpoints
- Access breakpoints
- Real-time access
- Adjustable debug-mode level (interrupts in background)
- Fast internal/external flash programming

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 the debug JTAG signals are driven at 3.3V, 5V or target voltage level (Vref).

When 'Vref' Debug I/O level is selected, a voltage applied to the belonging reference voltage pin on the target debug connector is used as a reference voltage for voltage follower, which powers buffers, driving the debug JTAG signals. The user must ensure that the target power supply is connected to the Vref pin on the target JTAG connector and that it is switched on before the debug session is started. If these two conditions are not meet, it is highly probably that the initial debug connection will fail already. However in some cases it may succeed but then the system will behave abnormal.

It is recommended to use ‘Vref’ setting only for target Vref voltages 3.3V and below. When debug I/O levels should 5V, ‘5.0V’ should be selected for Debug I/O levels.

Hot Attach

Option must be checked when Hot Attach is used. Refer to the ‘Hot Attach’ chapter for more details on Hot Attach use.

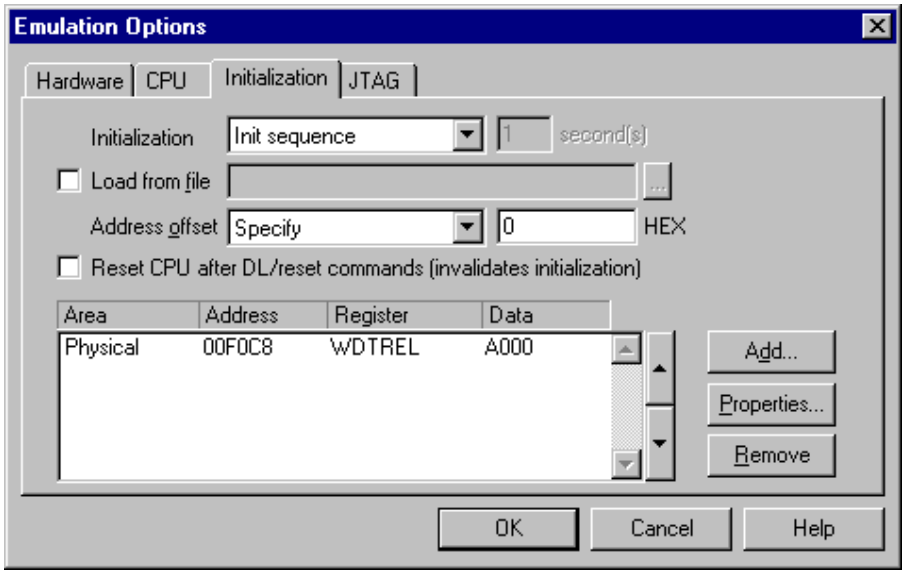
2.2 Initialization Sequence

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.

Note: Keep the default 'Specify' and '0' setting for Address offset within the Initialization tab.

The initialization sequence can be set up in two ways:

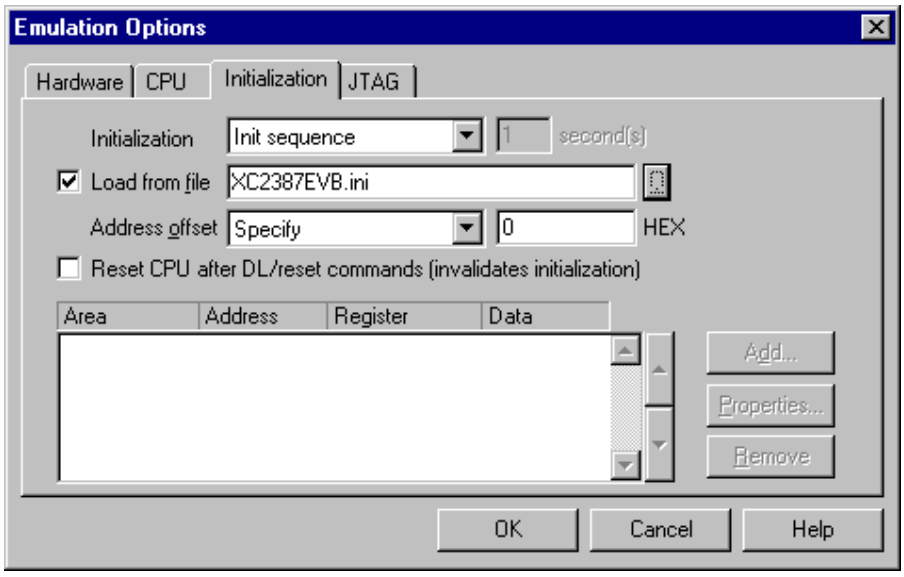
1. Set up the initialization sequence by adding necessary register writes directly in the Initialization page within winIDEA.



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.

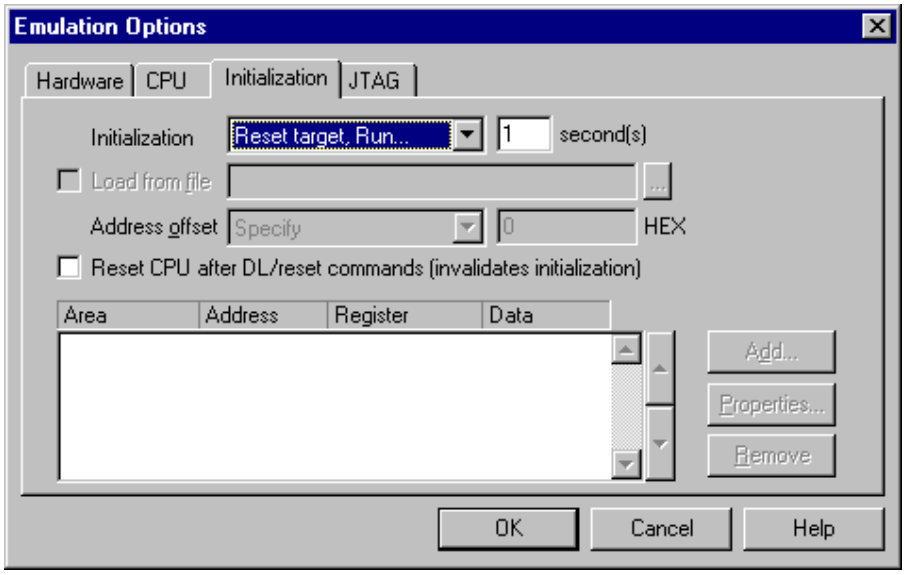
   Excerpt from an example XC2387EVB.ini file for the Infineon XC2387:

   S WDTREL W 0xA000 // watchdog reload value

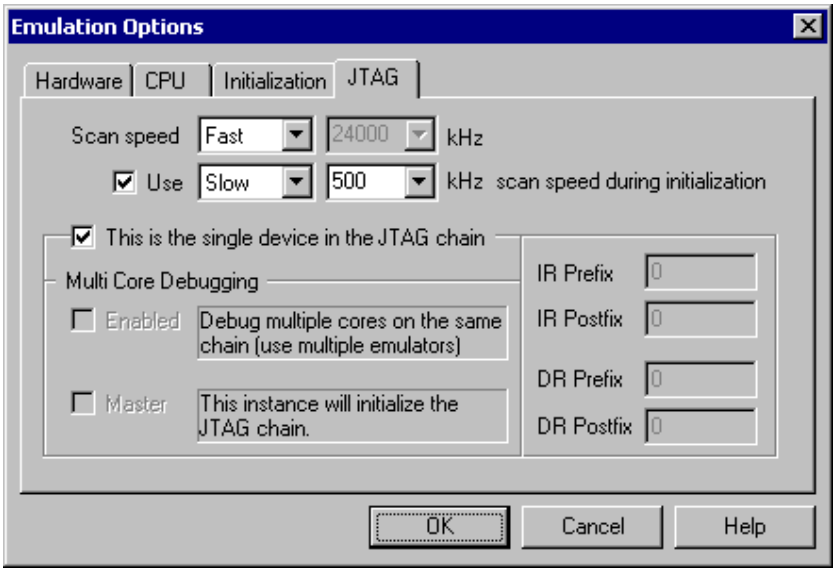


The advantage of the second method is that you can simply distribute your .ini file among different workspaces and users. Additionally, you can easily comment out some line while debugging the initialization sequence itself.

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.
- Other scan speed types (not supported for XC166/XC2000 family) can be seen and are automatically forced to Slow.

Slow and Fast JTAG scanning is implemented by means of software toggling the necessary JTAG signals.

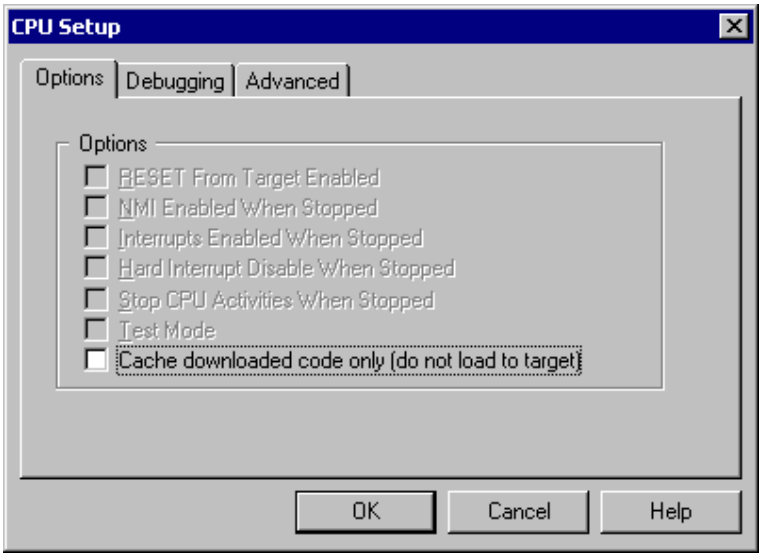
In general, Fast mode should be used as a default setting. If Fast mode fails or the debugging is unstable, try Slow mode at different scan frequencies until you find a working setting.

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



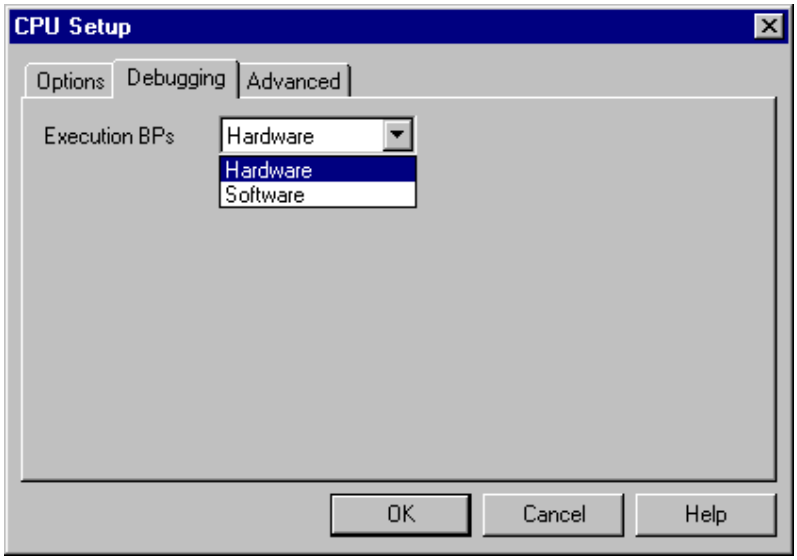
XC166 Family Debugging Options

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



XC166 Family Debugging Options

Execution Breakpoints

Hardware Breakpoints

Hardware breakpoints are breakpoints that are already provided by the CPU. The number of hardware breakpoints is limited to four. 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-writeable 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.

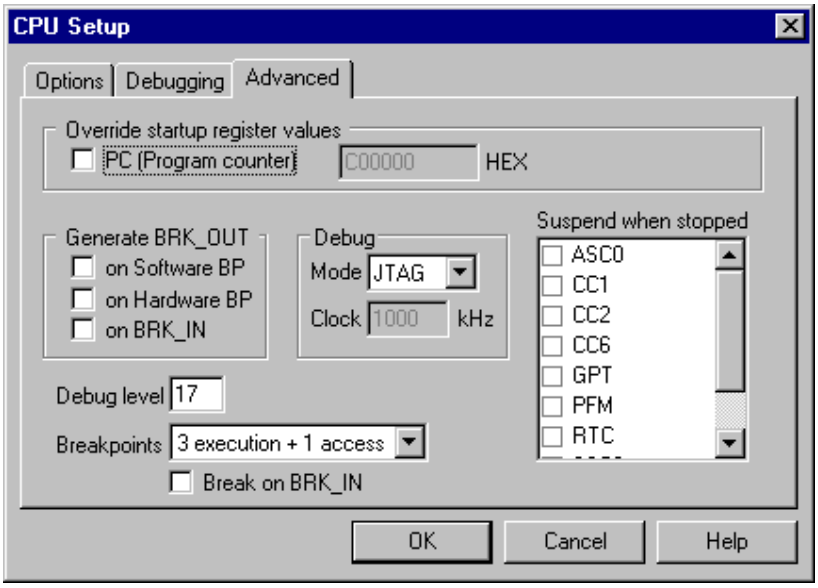
Software Breakpoints

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.

Note: 'Debug mode level' 17 is taken by default (see ‘Advanced’ tab) when software breakpoints are selected. This prevents stacked breaks.

3.3 Advanced Options



XC166 Advanced Emulation Options

Override startup register values

This option overrides the default Instruction Pointer reset value with the value set.

Generate BRK_OUT

Depending on the chosen options, the dedicated BRK_OUT pin is driven on software breakpoints, hardware breakpoints and/or active BRK_IN pin.

Debug

XC166/XC2000 device can feature JTAG, DAP or both debug interfaces. JTAG or DAP debug mode must be selected depending on the debug interface used. Additionally, DAP clock must be selected when DAP debug interface is used.

Note: The debugger must be connected to the target DAP connector (10-pin 1.27 mm pitch) when DAP debug mode is selected and to the target JTAG connector (16-pin 2.54mm pitch) when JTAG debug mode is selected.

Debug level

The on-chip OCDS has an associated debug level. The processor rules are:

- The processor enters debug mode only if the OCDS debug mode level is greater than the current task level
- In debug mode, an interrupt or PEC request is accepted only if its level is greater that the current task level and greater or equal than the OCDS debug level.

As an example, a debug level of 10 is able to stop any task running at level 9 and below. Once stopped, any interrupt of level 10 or higher suspends the debug mode and is executed.

If the processor receives an interrupt or PEC request while it is in debug mode and the request priority is greater or equal to the debug level, it is then accepted. The processor will then return to debug mode at the end of the interrupt routine or once the PEC has been done. This feature allows critical routines/PEC (with a high priority) to run even while the debug of low priority routines is taking place.

Thereby, if the user wants to execute some interrupts in background while the user's program is stopped, the corresponding debug mode level must be set.

Valid values are:

1 to 16: respectively breaks task level 0 to 15

17: break all levels from 0 to 15 and hardware traps

By default, debug mode level 17 is set and should not be changed if no interrupts in background needs to be serviced.

Breakpoints

The user must select the use of on-chip resources due to the fact that the same resources are shared among different debug functionalities.

The user can opt among: 4 execution breakpoints and 3 execution breakpoints + access breakpoint. Execution breakpoints refer to hardware breakpoints.

Break on BRK_IN

The BRK_IN pin is monitored and the execution is breaked if the BRK_IN pin goes to active.

Suspend when Stopped

This option selects the XC166 peripherals to suspend when the CPU is stopped.

4 Real-Time Memory Access

XC166/XC2000 debug module supports real-time memory access. Watch window's Rt.Watch panes can be configured to inspect memory with minimum intrusion while the application is running. Optionally, memory and SFR windows can be configured to use real-time access as well.

5 Access Breakpoints

XC166 Hardware Breakpoints

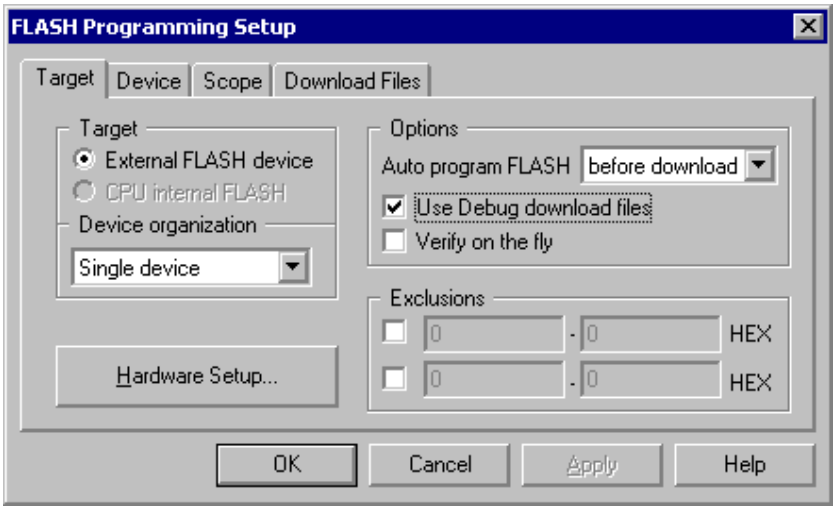
Access breakpoints are implemented by using watchpoints (OCDS).

Access breakpoint can be set on:

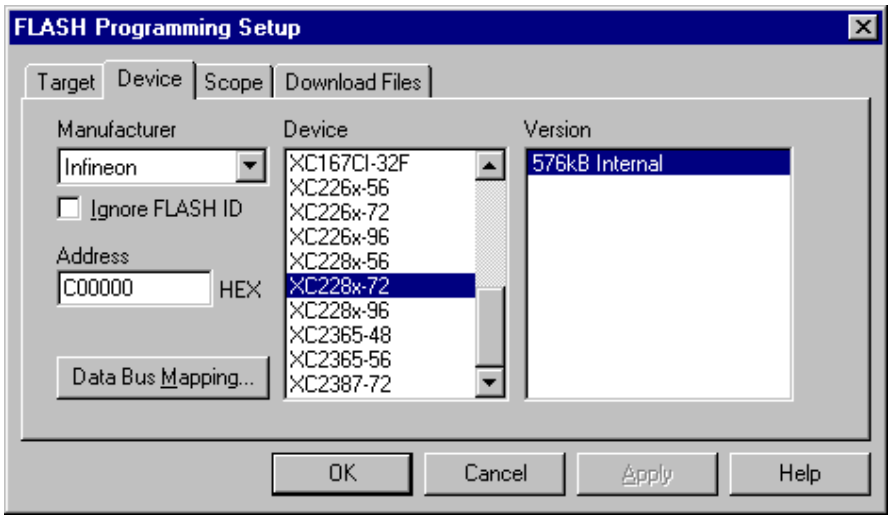
- Data value
- Write address
- Read Address

6 Internal FLASH Programming

The CPU Internal FLASH is programmed through the external flash programming method. The external flash device has to be selected in the 'FLASH Programming Setup' dialog. Next, 'Infineon' must be selected for the manufacturer and **the exact target CPU type.** A proper flash base offset address must be entered too (typically 0xC00000). Finally, make sure that the file being programmed is linked to the flash address. If not, use file offset when adding such a file..

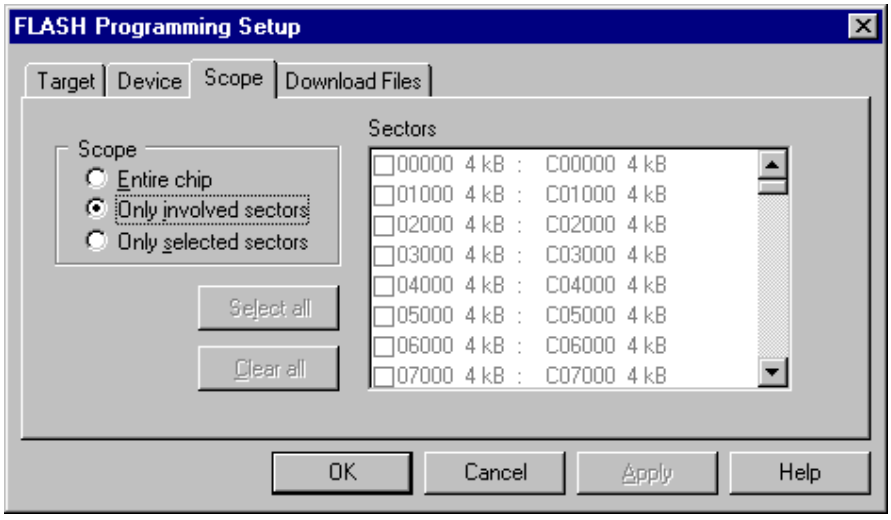


FLASH Programming Target Setup



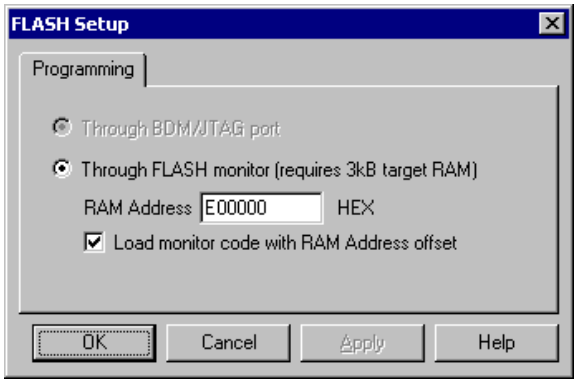
FLASH Programming Device Setup

Check 'Only involved sectors' option in the Scope tab because XC166/XC2000 flash doesn't provide mass erase (entire chip) command.



Internal flash programming configuration

Finally, the user must allocate flash programming monitor ('Hardware Setup' button in the 'Target' tab). It's recommended to allocate it in the internal program SRAM, which resides at address 0xE0 0000.



FLASH Monitor configuration

7 Hot Attach

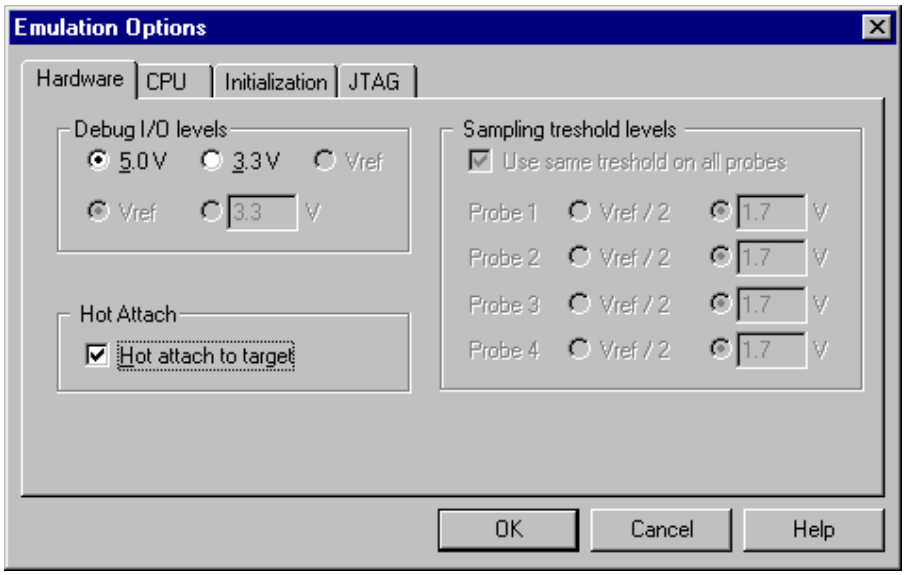
Infineon XC166/XC2000 debug module support includes a Hot Attach function, which allows the emulator to connect to a working target device and have all debug functions available. As such, it is a very convenient troubleshooting tool when the application misbehaves after a longer time. The user can connect to and inspect such an application after the problem pops-up.

Requirements:

- ~4K7 ohm pull-up must be added on the TRST JTAG debug line, which keeps the TRST pin high all the time, when the debugger is connected and when not connected.

To hot attach to a running target with no debugger connected:

- It is presumed that the debugger is not connected to the target and both emulator and the target powered
- Check the ‘Hot attach to target’ option in the ‘Hardware/Emulation Options/Hardware’ tab.
- Execute Download debug command.
- Connect the JTAG debug cable to the target system
- Select the 'Attach' debug command in the ‘Debug’ menu to attach to the target system.



Now, the debugger should display run status and the application can be stopped and debugged.

Select 'Detach' debug command in the ‘Debug’ menu to disconnect from the target application. If the CPU was stopped before detach, it will be set to running.

8 Getting Started

1) Connect the system

2) Make sure that the target debug connector pinout matches with the one requested by a debug tool. If it doesn't, make some adaptation to comply with the standard connector otherwise the target or the debug tool may be damaged.

3) Power up the emulator and then power up the target.

4) Execute debug reset

5) The CPU should stop on reset location, either on 0x0 or 0xC00000 depending on the state of EA pin which is sampled after driving the CPU out of reset.

6) Open memory window at internal CPU RAM location(s) and check whether you are able to modify its content.

7) If you passed all 6 steps successfully, the debugger is operational. Now you may add the download file and load the code to the RAM

8) To program the flash or download the code to the RAM, which is not accessible after reset, make sure you use the initialization sequence to enable the access. First, the debugger executes reset, then the initialization sequence and finally the download or flash programming is carried out.

9 Troubleshooting

- Try 'Slow' JTAG Scan speed if the debugger cannot connect to the CPU.

- Make sure that the power supply is applied to the target JTAG connector when 'Vref' is selected for Debug I/O levels in the Hardware/Emulator Options/Hardware tab, otherwise emulation fails or may behave unpredictably.

- When flash programming fails, double check that proper device, flash base offset address and flash monitor RAM address are selected in the 'FLASH/Setup' dialog.

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