



---

## Using DoCD debugger with Keil uVision2/3/4 tool

---

1. Overview	2
2. Supplied files	2
3. Installation	2
4. Preparing to work with uVision2/3/4 interface	3
5. Debugging an example project	8
6. Notes on Breakpoints	11
7. FLASH programming	12
8. References	16
9. Contacts	16

## 1. OVERVIEW

The DoCD debugger fully supports Keil uVision2/3/4 AGDI interface to hardware debuggers. It allows Keil software users to work within uVision2/3/4 environments without switching between Keil compiler and external DoCD software. Everything - compilation, simulation, debugging - is under control of a single application. This document describes how to install and use DoCD debugger with programs written using Keil C/ASM tools.

## 2. SUPPLIED FILES

Delivered package contains a complete set of files, needed to run DoCD debugger inside uVision2/3/4 interface. The following directories can be found after successful installation of package:

.\C51\BIN\	: Contains DoCD debugger DLL Target driver file
.\C51\FLASH\	: Contains DoCD debugger FLASH configuration files
.\C51\INC\DCD\	: Contains DCD's 8051/80390 include files
.\C51\EXAMPLES\DoCD\	: Contains example C/ASM 8051 application
.\DATASHTS\DCD\	: Contains DCD's 8051/80390 related PDF files
.\UV2\	: Contains DCD's 8051/80390 uVision2/3 database

This package is delivered to all existing DCD's customers. You may also order the DoCD FPGA evaluation kit by emailing us directly [info@dcd.pl](mailto:info@dcd.pl), and receive complete debug system to start working with our 8051/80390 IP Cores.

## 3. INSTALLATION

**Manual installation of DoCD target driver is no longer required, because package is delivered as executable program, and all steps are performed automatically by setup program.** The description provided below is left for informational purposes.

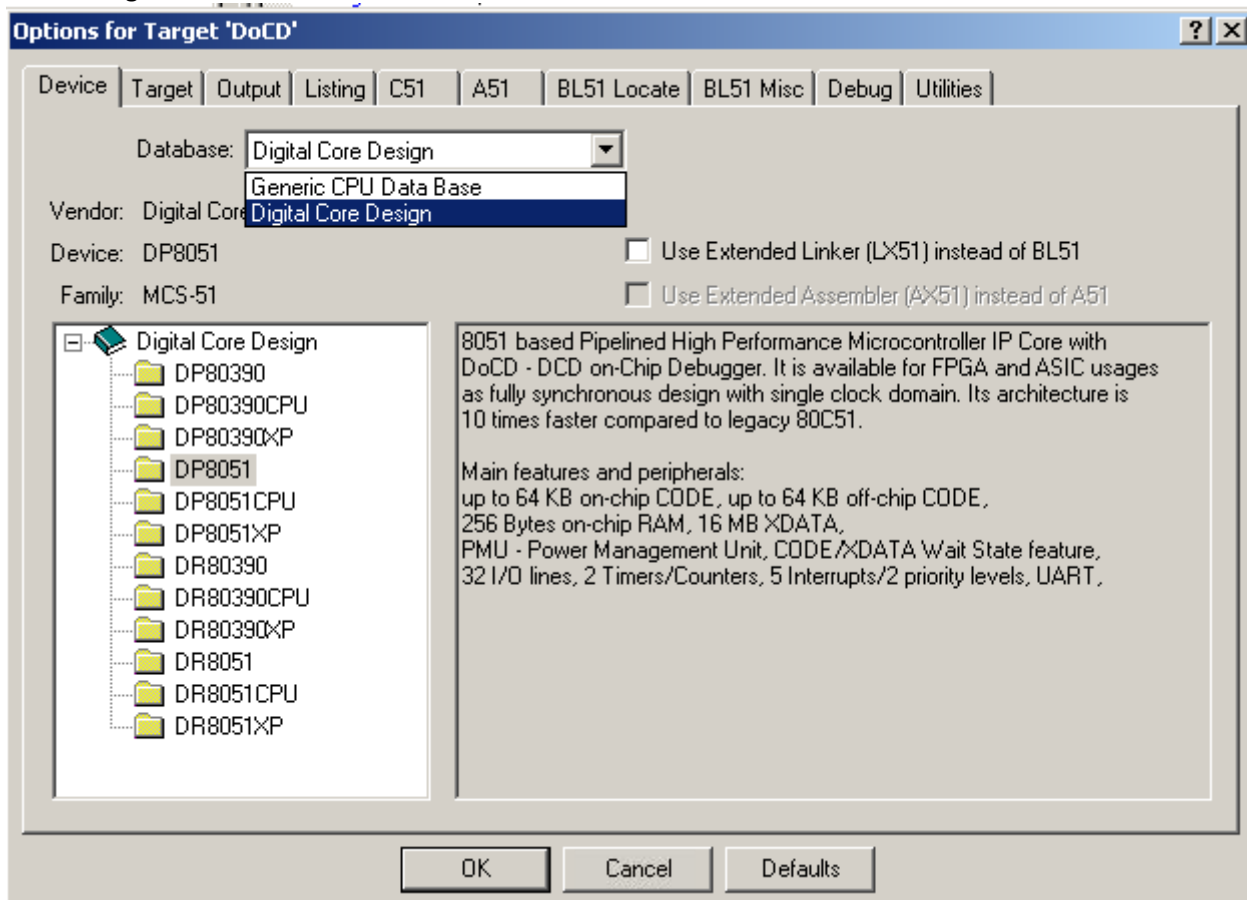
In order to check DoCD target driver follow the steps:

- Install uVision2/3/4 and the C51 Compiler on your machine. You may try to use evaluation version of 8051 Keil software at <http://www.keil.com/demo/eval/c51.htm>
- Check if new files appeared inside `<Keil install path>\C51\BIN`, `<Keil install path>\C51\FLASH`, `<Keil install path>\UV2` and `<Keil install path>\C51\EXAMPLES\DoCD` directories
- Check if the following lines are in `TOOLS.INI` file located inside `<Keil install path>\` folder:
  - section `[C51]`:  
`TDRV0=BIN\DCD_DoCD.DLL ("DoCD 8051 Target Driver")`  
`TDRV1=BIN\DCD_DoCD_USB.DLL ("DoCD 8051 USB Target Driver")`  
*Note: if TDRV0, TDRV1 are already in use, then use the next free number, for example TDRV2 TDRV3*
  - section `[UV2]`:  
`CDB0=UV2\DCD.CDB("Digital Core Design")`  
*Note: if CDB0 is already in use, then use the next free digit, for example CDB1*

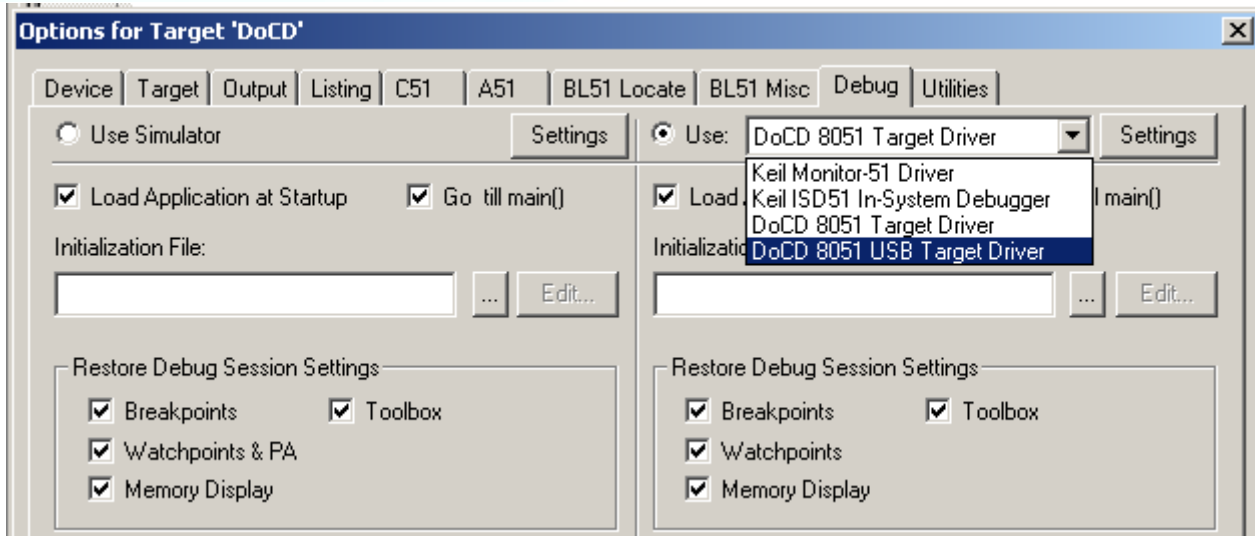
## 4. PREPARING TO WORK WITH UVISION2/3/4 INTERFACE

In order to run your application inside hardware using DoCD target driver follow the steps:

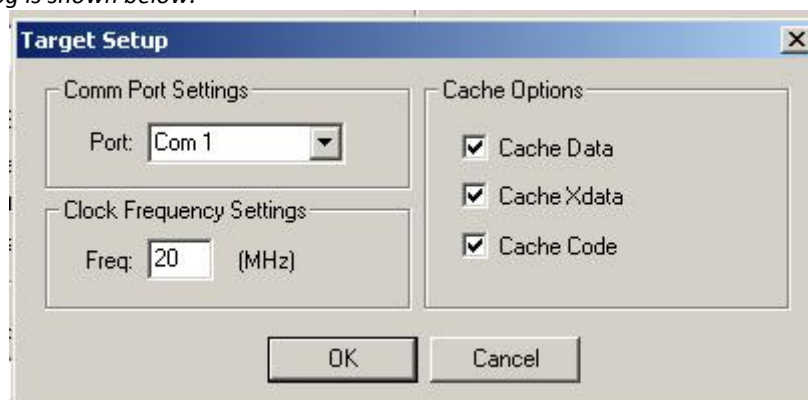
- Run  $\mu$ Vision2/3/4. Select 'Project – Open Project', the Select Project dialog comes up. Select the 'DoCD.uv2' project. It can be found normally in the folder - *<Keil install path>\C51\EXAMPLES\DoCD*.
- Select 'Options for Target – Devices'. From the 'Database' list-box, select "Digital Core Design" which is our devices database. Select required microcontroller type you actually have. If everything is right, then the dialog should look like this:



- Select 'Rebuild all target files' to build the project.
- Select 'Options for Target – Debug'. From the combo-box, select "DoCD 8051 Target Driver" which is our RS-232 DoCD driver or select "DoCD 8051 USB Target Driver" which is our USB DoCD driver. Make sure that the 'Use:' radio button is checked. If everything is correct, then the dialog should look like this:



- Select 'Settings' button. The 'Target Setup' dialog comes up.
- *RS-232 setup dialog is shown below.*

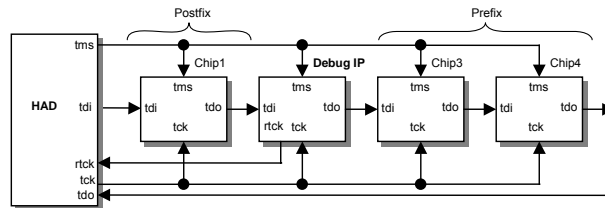


Choose a COM port used by HAD board, enable at least CODE 'Caching Options'. Enter the actual clock frequency applied on-board to global CLK pin of 8051 IP Core. Allowed values are 1 MHz to 250 MHz. It means that DoCD will be working properly with CLK clock frequencies 2 times lower and 2 times higher than entered value.

- *USB setup dialog is shown below.*

There are three options to select JTAG, DTAG or TTAG. It depends which version of Debug IP core you have in your system. Generally DTAG has 3 wires for communication, JTAG has 5 wires and TTAG has 2 wires.

**JTAG** – The DoCD™ debug IP Core v 4.00 and above can be used as standalone device, as well as plugged into JTAG chain. It means that standard JTAG pins can be used, and other JTAG devices can be controlled along with DoCD Debug IP. Such solution saves off-chip pins of ASIC/FPGA device.



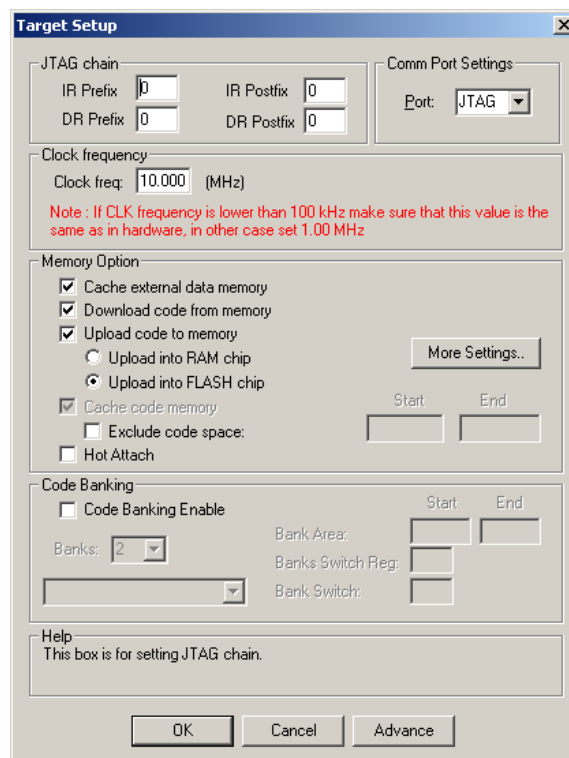
The example target shown in figure above consists of DoCD Debug IP and three devices being fully JTAG compliant. The Chip1 has 5-bit long IR (Instruction Register), Chip3 IR's has 3-bit long, and Chip4 has 4-bit long IR. A DR (Data Register) is always 1-bit long for each JTAG device.

The following values should be written into JTAG port configuration window:

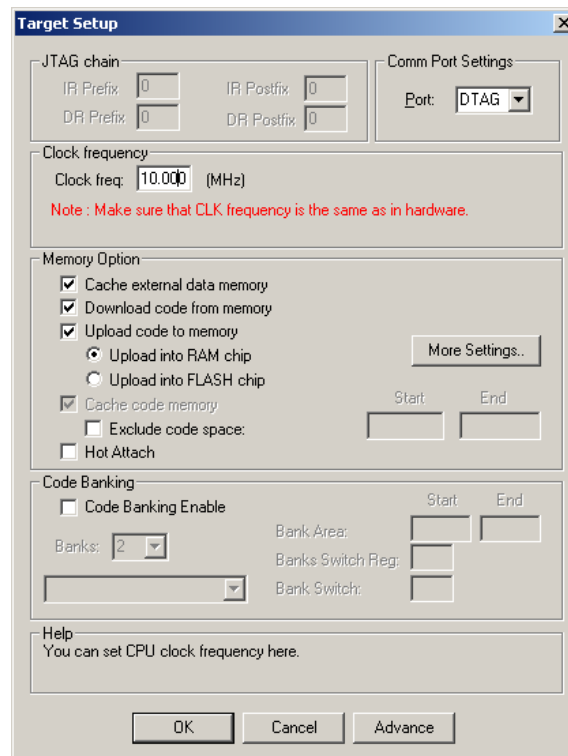
IR–prefix (3+4), DR–prefix (1+1)

IR–postfix (5), DR–postfix (1)

The 0 value should be written in an appropriate IR and DR field, in case where there would be no any prefix or postfix devices.



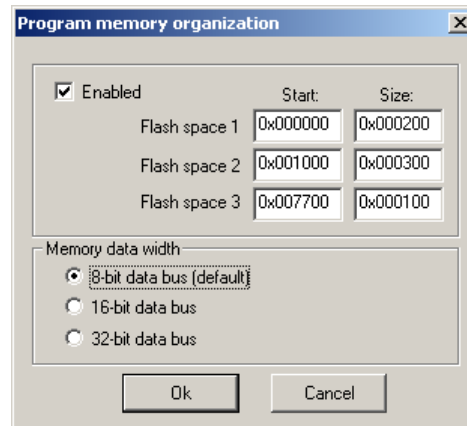
**DTAG** – Enter the actual clock frequency applied on-board to global CLK pin of 8051 IP Core. Allowed values are 1 MHz to 250 MHz. It means that DoCD will be working properly with CLK clock frequencies 2 times lower and 2 times higher than entered value.



**TTAG** – Similarly as like in DTAG interface enter the actual clock frequency applied on-board to global CLK pin of 8051 IP Core. Allowed values are 1 MHz to 250 MHz.

**Memory option** group is used to properly configure DoCD driver work with memories. The following settings can be specified:

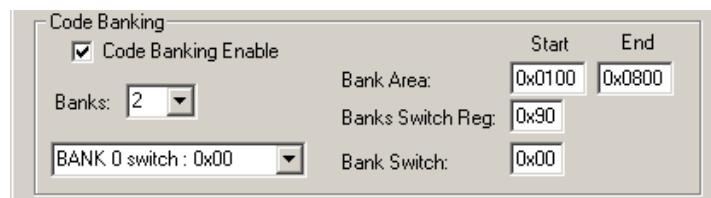
- **Cache external data memory** – checking this box allows driver to cache XDM memory content to improve debugging performance.
- **Download code from memory** – checking this box allows driver to read-back CODE memory content into Keil uVision2/3/4 debugger.
- **Upload code to memory** – checking this box allows driver to write all CODE memory content into hardware. It is obvious that ROM memory or any other memory without write signal connected properly cannot be written. *Please note when using FLASH memory you need to configure FLASH driver and use "More Settings" option to inform debugger where FLASH area is located.* Follow to next section to obtain FLASH programming details.
  - **Upload into RAM chip** –select this option to programming RAM memory
  - **Upload into FLASH chip** – select this option to use FLASH programmer to write program memory
- **Cache code memory** – checking this box allows driver to cache CODE memory content inside Keil uVision2/3/4 debugger. It significantly improves debugging process. This checkbox can be modified at any time during normal debugging process. **It is recommended to always have this box checked.** In case when you have self modified code then use **Exclude code space** parameter.
- **Exclude code space** – checking this box allows driver to update specified code space directly from hardware. Exclude code space is limited to 2kB. This option is used when you have self modifiable program.
- **Hot Attach** – checking this box allows driver to pass over reset CPU while initialization process
- **More Settings** – is used for managing a program spaces implemented in FLASH device. Up to three spaces are available. A following dialog box appears:



- **Enabled** – enables a program memory spaces in FLASH device. *Needs to be used when FLASH memory is used in system.*
- **Flash space 1,2,3** – definition of program memory spaces. The spaces should be separated and space with higher number must follow space with lower number (see example above). At least one space should be defined when this option is enabled. The remaining not used spaces should have 0x0 values.
- **Memory data width** – specifies width of program memory data bus

**Code Banking** group is used to properly configure DoCD driver work with hardware banked memory. Usually this section is disabled. When **Code Banking Enable** is checked then you can specify following settings:

- **Banks** – choose number of supported banks by your program memory. It should be the same number as configured in Keil C/ASM compiler.
- **Bank Area** – it specifies bank start and end addresses
- **Banks Switch Register** – defines address of register which switch memory banks (register can be located only in SFR memory space)
- **Bank Switch** – configures a combination of bits which activates selected bank. The bits combination depends on your actual hardware configuration. Repeat this step for each bank selecting it by drop-down list. located at the left side.



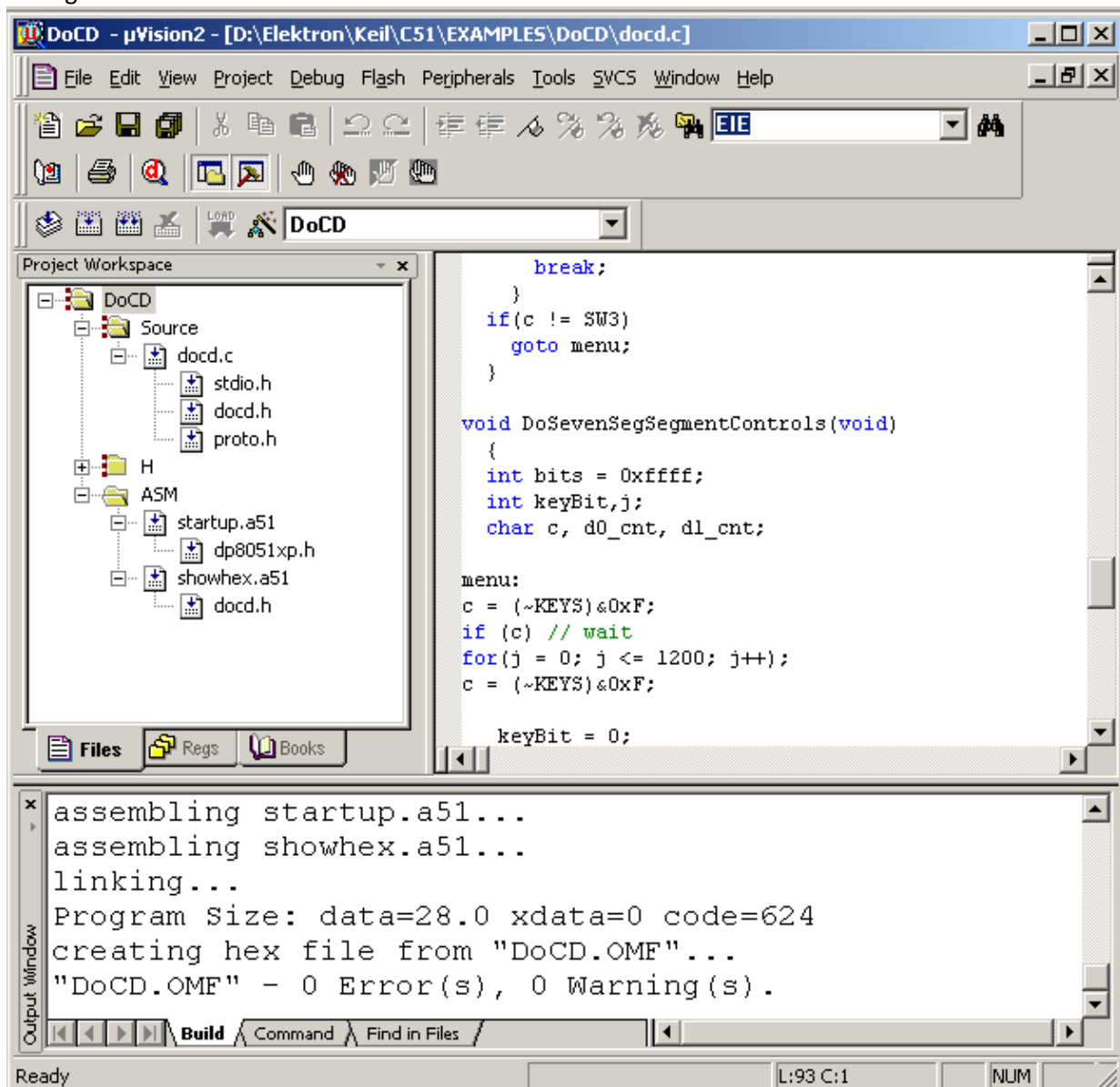
- Close the both dialogs using 'OK' button. To discard all changes use 'Cancel' button.

## 5. DEBUGGING AN EXAMPLE PROJECT

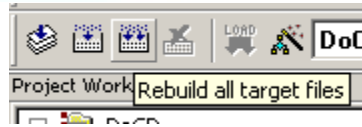
This chapter describes how to use DoCD debugger with 8051 C/ASM application basing on example DoCD project. It is assumed that the installation and configuration steps had been performed. ***You should also have DCD's 8051 core with Debug IP running inside some hardware prototype board equipped with FPGA or ASIC chip.***

Please follow the steps below:

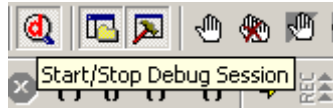
- Run  $\mu$ Vision2/3/4. Select 'Project – Open Project', the Select Project dialog comes up. Select the 'DoCD.uv2' project. It can be found normally in the folder - *<Keil install path>\C51\EXAMPLES\DoCD*. The dialog should look like this:



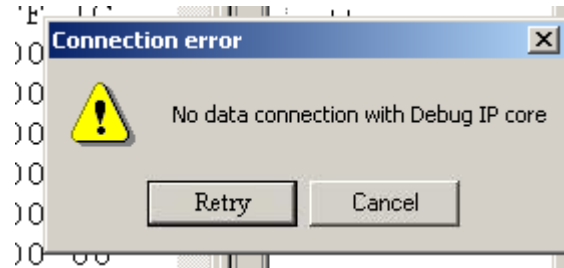
- Select 'Rebuild all target files' to build the project. The build window shown above should report no warnings and errors.



- Start debugging by selecting 'Start/Stop Debug session'

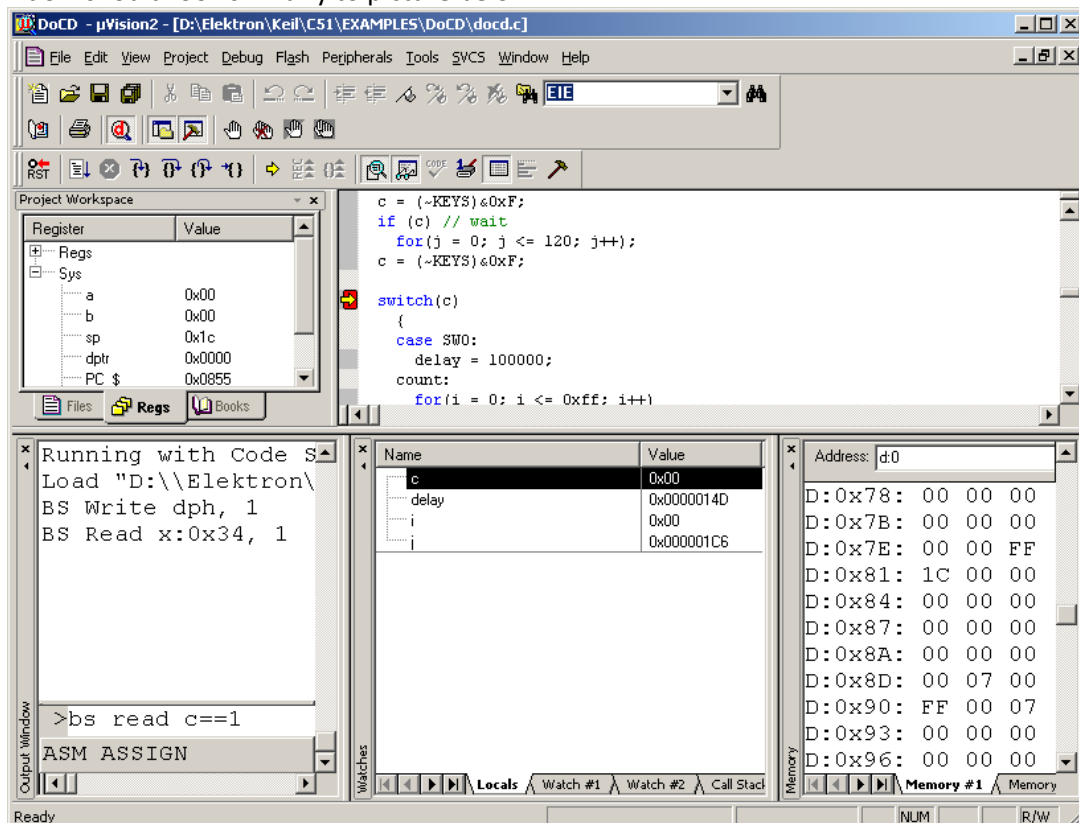


- A warning dialog box may possibly be displayed after 5 seconds, if some communication problems with hardware board would be encountered.

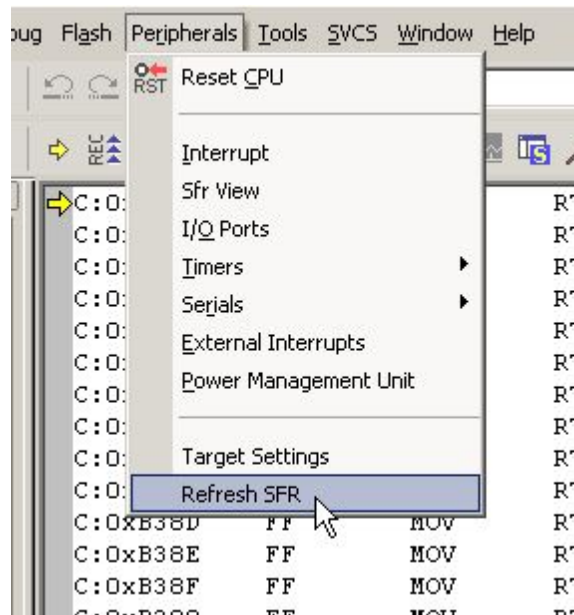


In this case the cables should be checked for proper connection, and 'Retry' button pressed. 'Cancel' button closes debugging session.

- When hardware is properly configured, and program successfully uploaded/downloaded, then the debugger window should look similarly to picture below:



- At this point debugging process is similar as in uVision2/3/4 simulator. The program can be run, halted, run step by step, breakpoints can be set/cleared. Variables can be watched, memory areas read/written/modified.
- The SFR registers can be update during microprocessor work by use 'Refresh SFR' item from 'Peripherals' option in main menu.



## 6. NOTES ON BREAKPOINTS

The following limitations are applied to the uVision2/3/4 debugger:

1. There are 4 independent complex hardware breakpoints available. They can be set using command line option or graphical boxes of uVision2/3/4 tool while debugging session is active:

- at CODE space (PC value). Example command line:

[BS main](#) - breakpoint at [main](#) address

At least one breakpoint is always active regardless of CODE memory type (SRAM, FLASH). A red square displayed at the left side of C/ASM source editor signalizes active CODE breakpoint. Since Debug IP version 3.50 and above number of CODE breakpoints is unlimited, and the can be set on RAM type of memory. CODE memory type is automatically detected by driver, and assures true CODE breakpoint activation.

- at DATA space (READ, WRITE only). Example command line:

[BS READ delay](#) - break when [delay](#) variable is being read

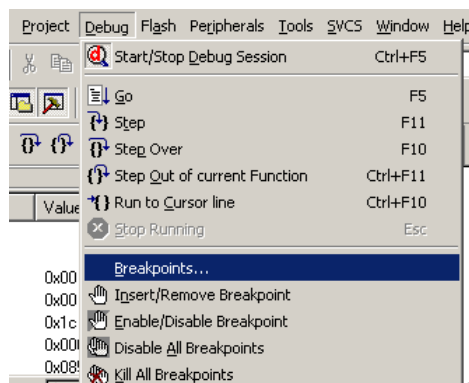
- at SFR space (READ, WRITE only). Example command line:

[BS WRITE DPH](#) - break when DPH SFR register is being written

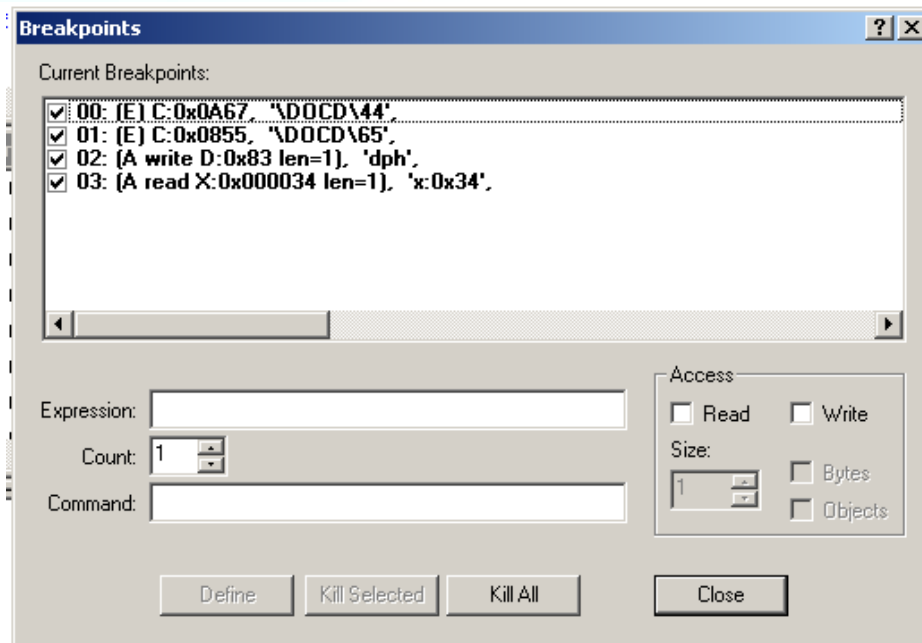
- at XDATA space (READ, WRITE only). Example command line:

[BS WRITE x:0x34](#) - break when 0x34 address is being written

Please refer to uVision2/3/4 Debug Commands help file. Conditional breakpoints are not supported. To manage breakpoints select 'Debug | Breakpoints...' menu:



You should see the Breakpoints window. The selected breakpoint can be enabled, disabled, defined, or deleted.

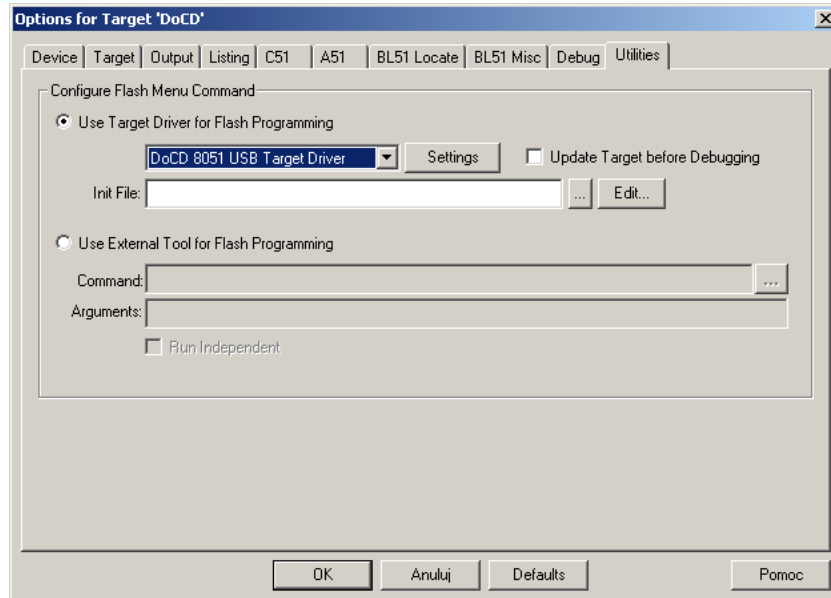


2. Code is automatically uploaded by uVision2/3/4 while debugging session is being started, and is assumed that CODE memory is writable as SRAM. If you have FLASH chips then configuration of FLASH driver is needed. Refer to next chapter to do this. Please check DCD's 8051 IP Cores specification for more details about Program Memory writes.
3. In case of read only CODE memory (ROM, FLASH), user is responsible for proper image loading into physical chip holding CODE. The CODE inside hardware memory must be identical with CODE used in uVision2/3/4 software. In other case debugging won't work correctly, and may crash with unexpected errors.

## 7. FLASH PROGRAMMING

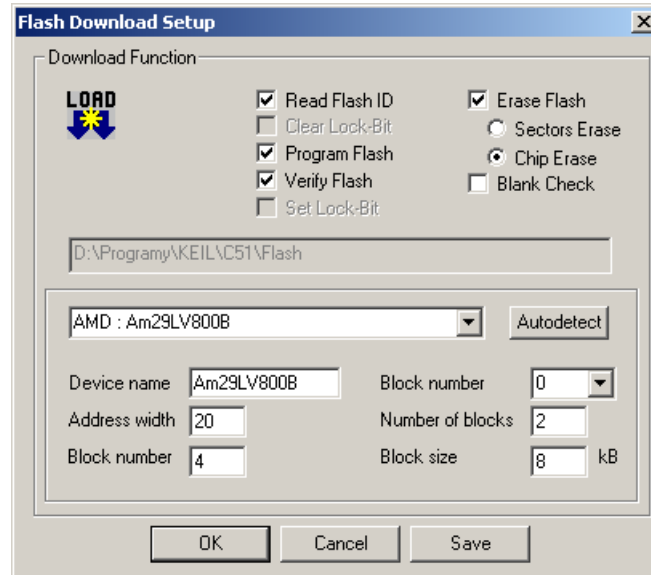
DoCD debugger fully supports programming of all FLASH memory devices. Such support is assured by configurability of FLASH programming algorithm, and supported devices database. New FLASH device can be easily added to existing base using build-in editor. DoCD debugger allows user to simply perform in-system programming of its FLASH memory without using any external equipment. In order to use Flash Device Programming driver follow the steps:

- Select 'Options for Target – Utilities'. From the combo-box, 'DoCD 8051 USB Target Driver' which is DCD USB DoCD driver. Make sure that the 'Use Target Driver for Flash Programming:' radio button is checked. If everything is correct, then the dialog should look like this:



FLASH programming task is performed directly within Debug software, and after uploading of code, it is ready for debugging. Programming time is very short, because of HAD2 support. This feature saves time, and makes usage of DCD's debugger very comfortable and flexible.

- Select 'Settings' button. The 'Flash Download Setup dialog' comes up.



This dialog allows you to configure DoCD to work with your FLASH device. There are two way of configuration:

**Autodetect** – automatically finds FLASH device when you have connected DoCD hardware board (FPGA or ASIC) to your workstation.

**Manually** - you can choose device manually from actual database list. You can specify FLASH options such as:

- **Read FLASH ID** – by selecting this driver compares selected device ID with the real ID on your hardware. If they don't match then programming stops with error message.
- **Clear Lock-Bits** - It clears sector Lock-Bits set in FLASH memory. Must be selected to allow erase of chip before uploading of new program. Not all chips support this.
- **Erase FLASH** - If you work on non-blank device you **must** select this option to erase. It almost always happens when you developing code and successfully uploading its newer versions.
- **Erase Sectors** - erases sectors one-by-one.
- **Erase Chip** - erases the whole chip – uses a single command if supported. This option is recommended when Flash specified area are used .
- **Blank Check** - selecting this the driver will check if the **whole** chip has been erased. Not recommended to use in case of large FLASH chips.
- **Program FLASH** - this allows you to program FLASH chip in hardware with program code. This option **must** be selected to enable programming.
- **Verify FLASH** - it verifies each programmed byte only (does not check other spaces of FLASH). It is recommended to enable this option along with **Program FLASH**.
- **Set Lock-Bits** - this option disables accidental programming of chip. Not all chips support this.

You can also specify FLASH device settings, in case when it is not available in current database. All required data can be found in FLASH datasheet:

- **Device Name** - setting new ID value and name adds a new device to program database. If file <Device Name>.INI exist then it will be overwritten.
- **Address bus width** - set or modify an address bus width.
- **Block number** - set or modify number of blocks.
- **Number of blocks** - enter number of identical sections.
- **Block size** - enter size of identical sections.

- **Save button** - allows save all made changes or add new device into database. Physically file is stored in C51\FLASH subdirectory as a text file, and you can edit it in any text editor.
- Close the both dialogs using 'OK' button. To discard all changes use 'Cancel' button.
- Run 'Download to Flash Memory' command from the Toolbar.

## 8. REFERENCES

The DoCD uVision2/3/4 debugger package is delivered by DCD free of charge upon customer request. The customer should have DCD's 8051 IP Core equipped with DoCD on chip debugger, and HAD communication board. DoCD debug system is available at DCD web-site <http://www.dcd.pl>.

The Keil 8051 uVision2/3/4 tool suite is also required. It can be obtained from Keil GmbH at <http://www.keil.com>.

## 9. CONTACTS

For any questions or comments please contact DCD.

**Headquarters:**

Wroclawska 94

41-902 Bytom, POLAND

*e-mail:* [info@dcd.pl](mailto:info@dcd.pl)

*tel.* : +48 32 282 82 66

*fax* : +48 32 282 74 37

**Distributors:**

Please check <http://www.dcd.pl/apartn.php>