Using Hardware Breakpoints for HC12

The present technical note is related to the usage of hardware breakpoints. It covers topics like:

- Defining breakpoints
- Enabling Hardware Breakpoints
- Restrictions applying to hardware breakpoints

Setting Breakpoints :

Setting breakpoints in RAM

When your application is loaded in RAM, the debugger inserts some special break instructions (SWI, TRAP, BGND…) in the code to set the standard breakpoints. When the instruction is reached, execution of the application stops. This works fine as long as the code is located in RAM.

Setting Breakpoints in ROM, in EEPROM or in Flash

If your application is loaded in EEPROM or in Flash EPROM, it is not possible to use the standard breakpoints (it is not possible to insert the special instructions in the code). In order to be able to set breakpoints, the debugger needs some assistance from the hardware.

Debugging on an Emulator

When you are using an emulator, you usually do not have any problem to define breakpoints in Flash. This is directly supported through the emulator triggering system.

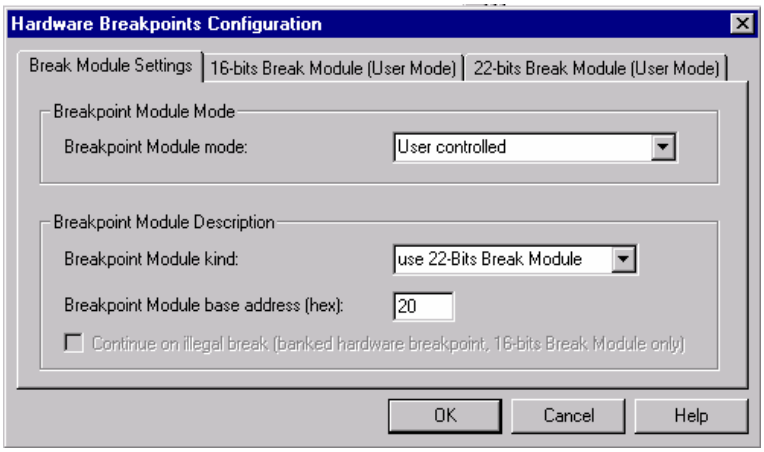
Debugging on a BDM or Monitor Debugger

When you are using a ROM monitor debugger or a debugger communicating through a special Background Debug connector (BDM, ONCE, JTAG, …) usage of hardware breakpoint is only possible when the hardware is equipped with a hardware breakpoint module. Typically, HC05, HC11 and HC16 MCUs are not equipped with such a module.
The rest of this Technical Note is related to the usage of hardware breakpoints on this kind of debugger.

Setting Hardware Breakpoints:

Using a Dedicated Dialog box (Not available for all target):

Some target interface provides a dedicated dialog box to enable and configure hardware breakpoints. Click on the target menu and choose the entry ‘Set hardware BP’. The ‘Hardware Breakpoint Configuration’ dialog pops up:



With this dialog, it is easily possible to

- Enabled/disable the Hardware breakpoint using the “Breakpoint Module mode” combo box. Usually we recommend you to select the mode Automatic (Controlled by the software). If one select the mode “user controlled”, he has to configure also the dialog in the “16-bits Break Module (User Mode)” or “22-bits Break Module (User Mode)” Tab.
- Specify which kind of breakpoint device is available (16-bit Break Module is available for HC12B32, HC12D60 and HC12DG128, 22-bits Break Module is available for Star12 MCUs).
- Specify the location of the Hardware Breakpoint Device using the “Breakpoint Module base address” edit box.

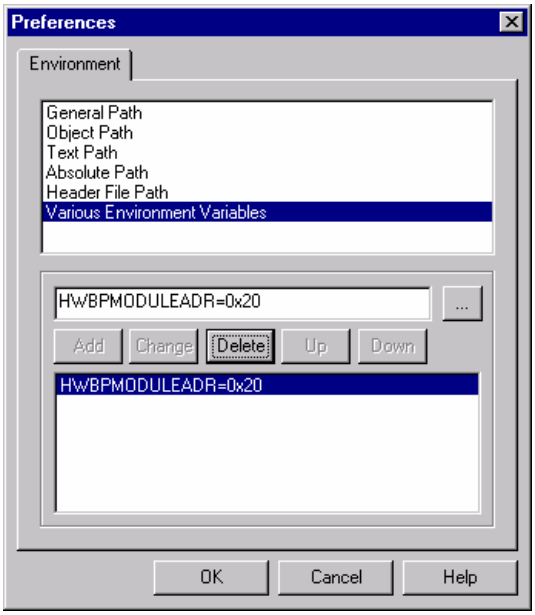
The tabs “16-bits Break Module (User Mode)” or “22-bits Break Module (User Mode)” provide access to the advanced feature of the Hardware Breakpoint module. Please check the target interface manual for more details about this dialog.

Using an Environment Variable:

In 'File' menu of HI-WAVE, click on 'Configuration' and in the 'Preference' dialog choose the entry "Various Environment Variables" and enter the following line:

```
HWBPMODULEADR=0x20
```

Example:



Be Careful:

- HWBPMODULEADR must be written in capital letters.
- Make sure that there is no space, no tabs or any other characters between the address and the equal symbol or between the variable and the equal symbol.
- Hardware breakpoints are not available for every MCU (for example the HC12A4); please check carefully your hardware documentation.
- You have to set the same environment variable to use hardware watchpoints. However there are some restrictions. Please refer to the section "Limitations and restrictions of the hardware breakpoints" below.

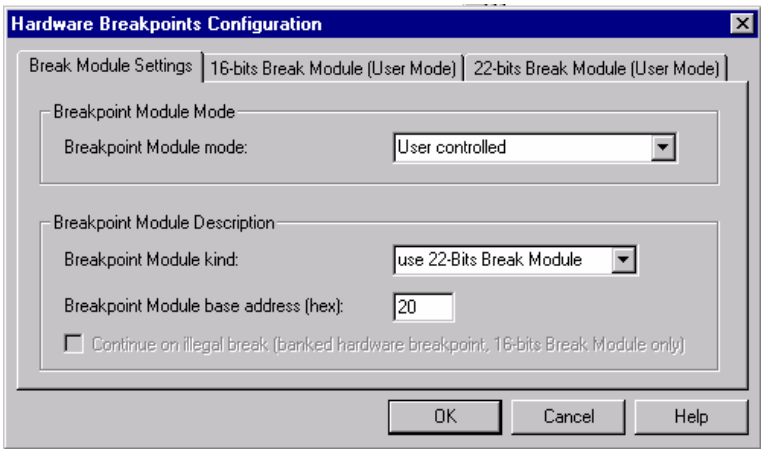
Limitations and restrictions of the hardware breakpoints:

- The number of hardware breakpoints is limited and depends on the hardware (for the MC68HC912B32, only 2)
- In order to use hardware watchpoints, your CPU must a least have 2 breakpoint control registers.

Example:

- For the MC68HC912B32 (2 breakpoint control registers). You can set 2 hardware breakpoints **or** (only) one watchpoint

- On a MC68HC12DG128, the breakpoint control registers are 16-bit wide. If you are working in banked memory model you may have some trouble using hardware breakpoints. When a hardware breakpoint is set on address 0x8200 on bank 0, execution of the application will be stopped as soon as the PC reaches address 0x8200 on any of the banks (0 to 7).
- This can be avoided by checking the "Continue on illegal break …" check box.



Note: When "Continue on illegal break …" is checked, the application does not run in real time any more!!!

About the Register mapping:

- If, in your application, you have re-mapped the I/O registers, the address of the hardware breakpoint module must be set according to the new register block start address.

Example:

Suppose that the breakpoint module address is 0x20 and that the original register mapping: 0x0000 to 0x01FF.
After the re-mapping, the registers are set from 0x0800 to 0x09FF, so you have to tell the debugger the hardware breakpoint module starts at address 0x820

WARNING!

Be careful, you will not be able to set hardware breakpoints ***before the instruction*** which re-maps the registers.

Hidden Hardware Breakpoint

During a debug session (when the code is loaded in flash), the following operations are using a hardware breakpoint:

- "Run to cursor"
- "Step out"
- "Step over"

If you want to use these features, you have to let at least one hardware breakpoint free!!!