# Release Notes for MON390

**Monitor for Dallas Contiguous Mode**

This file contains release notes and last minute changes that are not found in the printed manuals.

Information in this file, the accompany manuals, and software is

Copyright � Keil Software, Inc and Keil Elektronik GmbH.

All rights reserved.

## Contents

1. [MON390 Overview](#_gjdgxs)
2. [How MON390 Works](#_30j0zll)
3. [Setup MON390 on Your Hardware](#_1fob9te)
4. [Configure �Vision3 for MON390](#_3znysh7)
5. [Tool Configuration for MON390](#_2et92p0)
6. [Debug with MON390](#_tyjcwt)
7. [Troubleshooting](#_3dy6vkm)
8. [Known Problems](#2s8eyo1)
9. [Technical Support](#_1t3h5sf)
10. [Contact Details](#_4d34og8)

## MON390 - Monitor for Dallas Contiguous Mode

The MON390 is a Monitor program that allows you to debug applications on your target hardware using the Keil �Vision3 Debugger. You connect the �Vision3 Debugger with a serial cable to a target board that uses a Dallas CPU (like DS80C390, or DS80C400) running in Contiguous Mode. In this mode up to 16MByte memory are supported.

To get started, you must:

* Configure and install the MON390 on your target hardware as described under [Setup MON390 on your hardware](#_1fob9te).
* Select the correct Debug Driver as explained under [Configure �Vision for MON390](#_3znysh7).
* You need to adjust the project settings for your application as explained under ...

We recommend that you test your MON390 installation with an example project available in the Keil\C51\MON390 folder before you start debugging with your own test program.

### Features

* Controlled program execution with single-stepping and multiple software breakpoints.
* Viewing and changing CPU registers and memory.
* Access to Special Function Registers (SFRs).
* Execution of user programs in full speed when no conditional breakpoints are set.
* MON390 is transparent for a user program's interrupt system.

### Requirements

* One UART (can be shared with the user application in non-interrupt mode). MON390 uses a serial interface with a timer as baudrate generator.
* 2K bytes of code memory for MON390 program code.
* 32 bytes of xdata memory (von Neumann mapped) for MON390 data.
* 8 bytes stack space in the user program to be tested.
* Code space for the user application must be configured as von-Neumann memory.

## How MON390 Works

MON390 connects to the �Vision3 Debugger via the serial interface. MON390 is fully transparent for the user application. There are only a few issues to keep in mind when you are using MON390:

* MON390 requires that the programs you debug are located in RAM space. This is necessary because breakpoints are set by replacing instructions in your program with ACALL instructions. This operation, while completely transparent, may have side effects that affect the operation of your target program. Refer to Breakpoint Side Effects later in this section for more information.
* You may need to relocate your startup code, program code segments, and interrupt vector table.
* You may enable or disable the HALT command of the �Vision3 Debugger. This is configured with the **Enable Serial Break** under **Options for Target - Debug: Use: MON390: Dallas Contiguous Mode - Settings**. If this is enabled the �Vision3 Debugger and the MON390 use the serial interrupt vector to signal that the target program should stop running.

### Breakpoint Side Effects

Breakpoints are required to stop the user application for checking of the system variables. To perform a breakpoint MON390 inserts a ACALL instruction in the user program. This method avoids extra hardware for the breakpoint logic. However, breakpoints can only be set in RAM memory. Another drawback is that a ACALL instruction occupies up to three bytes. Therefore breakpoints may have side effects when set before a label (jump target) on a one or two byte CPU instruction. The following example demonstrates this problem.

**User Program**

8000 E4 CLR A  
8001 74 03 MOV A,#3  
 :    :       :  
  
   
 :    :       :  
  
   
8010 80 ED SJMP 8001

First, the user program is executed until address 8010 with a Debugger command like **G, 8010**.

Then, a breakpoint is set at address 8000. The breakpoint is realized by writing a ACALL instruction into the user program; this means that the user program is modified by the breakpoint.

**User Program Modified Due to Breakpoints**

8000 XX ACALL ; An ACALL instruction is  
8001 XX XX ??? ; written at address 8000.   
 :    :       :  
  
   
 :    :       :  
  
   
8010 80 ED SJMP 8001

When the user program continues execution at address 8010 the program jumps to address 8001. However, this memory location is overwritten with the second op-code byte of the ACALL instruction that is inserted due to the breakpoint at address 8000. Therefore, the MOV A,#3 instruction will be not executed and the behavior of the user program is unpredictable.

In real C applications such conditions are rare, however, you should keep in mind that the ACALL instruction might influence the user program. In such situations, you may single step in the user program. During single stepping all instructions are executed without conflicts, since the debugger takes care about conflicts.

### Program Execution

When only exection breakpoints are defined, the user program is executed in full speed.  You may also use **access** or **conditional** breakpoints. In this case, the application is executed in single steps and not in real time. Single step execution is at least 1000 times slower.

## Setup MON390 on Your Hardware

Preconfigured Mon390 version are available in the folder **Keil\C51\MON390\**. If your target board is not available, you need to configure the **User Monitor** for your target hardware.

The following steps are required to customize the **User Monitor** project:

1. Open the **Keil\C51\MON390\User Monitor\MON390.uv2** Project.
2. Open **Install.A51** with the editor.
3. Check the configurations settings in **Install.A51** and modify them, if necessary, to match your target hardware.
4. Customize the baudrate initialization code.  You may use the �Vision3 simulator to verify your settings.
5. Build target and load it into the target hardware device with the appropriate tool (depends on your hardware), or burn it to the boot ROM.
6. When you have Verify that the monitor starts up, by checking a blinking sequence on Port1, which the monitor generates after the initialization.

Program that may be used to verify the settings for a user Mon390 can be found in the folder **Keil\C51\MON390\Examples**.

## Configure �Vision3 for MON390

Once you have installed the MON390 firmware to the target hardware you may configure the �Vision3 Debugger for communication with the Dallas Contiguous Mode target system.

1. Start MON390 on the target system (typically with a CPU reset) before you start the uVision3 Debugger.
2. In �Vision3, open the dialog **Project - Options for Target - Debug.**
3. Select **Use: MON390: Dallas Contiguous Mode**.
4. Enable **Load Application at Startup** to load the user program to the target hardware.
5. You may enable **Go till main** to execute the user program until the main() function is reached.
6. Configure the options below with **MON390: Dallas Contiguous Mode - Settings**:

### COM Port Settings

* **Port**: PC COM port used to connect the target hardware.
* **Baudrate**: Communication baud rate with the target hardware.

### Cache Options

The MON390 driver implements data caches to speed-up screen updates.

* Disable the cache options to view actual (un-cached) memory content of DATA (and SFR), IDATA, or XDATA space when you halt program execution. This way you can be certain you are viewing the current values of I/O ports, timers, or memory-mapped peripherals.
* Enable the cache options to obtain maximum performance. This enhances especially the code disassembly and displaying of huge memory areas.

### Target Options

* **Enable Serial Break** allows you to the HALT program exection. When this is enabled MON390 uses the serial interrupt vector and the serial line cannot be shared with the user program I/O routines. When this is disabled the user program may use putchar and getkey routines to perform character I/O via the serial interface in non-interrupt mode.
* **Test Monitor Data RAM** will execute a small RAM test of the MON390 data area (256 Bytes beginning with MON\_RAM\_START). This may help you to solve problems with wrong configured MON390 firmware.
* **Ignore Handshake Lines** avoids that the monitor driver toggles the DTR or RTS lines of the serial interface. This is especially helpful, if your target board uses one of these lines for special functions like a hardware reset.  
  *Note: This option is required for the Dallas TINI m400.*

## Tool Configuration for MON390

When using MON390, the complete target application is stored in von-Neumann mapped RAM. This means that the **code** memory and **xdata** memory are accessing the same physical memory space. This is required, since the target hardware is not able to write into code space and the MON390 changes the program code to set breakpoints in your application.

Therefore the EPROM and RAM areas that are entered in the dialog **Project - Options for Target � Target � Off-chip Memory** must be non-overlapping physical memory areas. These ranges are supplied to the Linker if you have enabled the option **Use Memory Layout from Target Dialog** in the **LX51 Locate** dialog page. Therefore you should also check that this option is set.

Refer to the �Vision3 projects in the folder **Keil\C51\MON390\Examples** and review the tool configuration of this applications if you are not sure about the tool configuration.

### Additional Settings when MON390 is Install at Address 0

When the Monitor is installed at address 0 interrupt vectors will be redirected as specified with the **INT\_ADR\_OFF** definition in the **INSTALL.A51** file. Therefore you need to modify the START390.A51 startup code and you may need to set an interrupt vector offset. The initial **LJMP** instruction needs to be exchanged by an **AJMP** instruction. The Monitor-390 already switches the CPU into contiguous mode, and therefore the **AJMP** instruction can have a jump target anywhere in the 64KB CODE memory class that is used in the **START390.A51**. The modification in the START390.A51 is as shown below:

?C\_STARTUP LABEL NEAR  
 ;DB 02H ; LJMP in Classic 8051 Mode  
 ;DW WORD0 STARTUP1  
 AJMP STARTUP1 ; AJMP in Contigous Mode

The other modifications depend on the redirection address. If you just change the interrupt vector address offset to a different 64KB segment (for example C:0x010000) you need to enter the start address of the under **Project - Options for Target - Target - Off-Chip Code memory**. For example: EPROM #1: Start: 0x10000 Size: 0x40000. The uVision2 IDE converts this into a LX51 Linker/Locater CLASSES directive that will contain a CODE memory class that starts at C:0x10000. If you redirect the start address to a page offset (for example to C:0x008000 or C:0x018000), you need in addition also a change for the ?C\_CPURESET?0 segment in START390.A51 as shown below:

?C\_CPURESET?0 SEGMENT CODE OFFS 8000H  
 RSEG ?C\_CPURESET?0

## Debugging with MON390

Once the �Vision3 Debugger is configured, you may start debugging with **Debug - Start/Stop Debug Session**. The �Vision3 Debugger connects to the Dallas target system via the **MON390: Dallas Contiguous Mode** driver.

MON390 supports most �Vision3 debugger features. For instance, you may single-step through code, set breakpoints, and run your application. Variables may be viewed and modified using the standard debugger features. Changing the values of SFRs and memory of any type can be done in the Watch Windows or the the Register tab.

More information about debugging with MON390 can be found in the User's Guide *Getting Started with �Vision3*, **Chapter 5. Testing Programs** and **Chapter 11. Debugging with Monitor-51**.

### �Vision3 Restrictions When Using MON390

There are several restrictions you must consider when using MON390 and the �Vision3 Debugger.

* It is not possible to use Debug - Memory Map to change the memory mapping of the 8051 system. This is because the memory mapping of the 8051 target system depends on hardware components.
* The Performance Analyzer, Call Stack, Code Coverage, and Step Out command are not available with MON390.
* The Periodic Window Update Option may not be used with MON390.

## Troubleshooting

The following section gives you tips to solve common problems.

### If the Debugger does not connect...

... it is typically a problem of MON390 code and xdata locations or the initialization of the serial interface.

You should first verify that the target system's serial interface is correctly configured. You may check this by using the �Vision3 Simulation as follows:

1. Open the �Vision3 project that you have used to configure MON390, i.e. **Keil\C51\MON390\User Monitor\MON390.uv2**.
2. Select **Project - Options for Target - Debug: Use Simulator**.
3. Start program simulation with **Debug - Start/Stop Debug Session**. This loads the monitor program in simulation mode.
4. Start running the simulation (Debug - Go).
5. Verify the settings of the UART with the Serial Port Dialog (Peripherals - Serial). The baudrate is displayed correctly if you have entered the correct XTAL frequency in **Project - Options for Target - Target - Xtal**. Typically, the baudrate will not match the PC baud rate 100%, but it should be within 2.5% tolerance.
6. Check serial communications:
   * Open the serial communication window (View - Serial Window).
   * Right-click in the Serial Window and select Hex Mode from the context menu.
   * Enter **s0in=0xFF**<ENTER> in the Command Window to Input a 0xFF character in the serial input stream.

The �Vision3 should show the Monitor identification 0xFE, 0xA5 in the serial window now.If everything is correctly configured and you are still unable to connect to the hardware, you'll try to link the following test-program with your monitor install code and load it into your hardware. It shoud echo all inputs on the keyboard back in your Hyperterminal./\*  
 \* Test Function: verify serial communication with HyperTerminal  
 \*/  
#include <REG390.H>  
  
void main (void) {  
 char c = 'A';  
   
 // init serial interface 19200 baud @ 20MHz  
 PCON |= 0x80;  
 SCON0 = 0x50;   
 TMOD |= 0x21;  
 TH1 = 0xF8;  
 CKCON |= 0x10;  
 TCON = 0x50;  
  
 TI = 1;  
 while (1) {   
 if (RI) {   
 c = SBUF;   
 RI = 0; }  
 while (!TI);  
 TI = 0;  
 SBUF = c;  
 }   
}

If the serial channel is correctly configured and you are still unable to connect to the target hardware, the MON390 code or xdata space might not be accessable.

### If MON390 stops working or behaves strange during debugging...

... your application is most likely overwriting the user application program. This might happen when the user application makes xdata write accesses to the program code locations. Code and xdata memory must be non-overlapping areas, since the MON390 requires von Neumann wired code space, which means that code and xdata space are physically the same memory area. You should therefore check the XDATA and CODE MEMORY MAPPING that is listed in the Linker MAP file and verify that code and xdata space are not overlapping.

If MON390 does not single step CPU instructions or if you cannot read or write SFR data locations the MON390 xdata memory area cannot be accessed from code space. The MON390 data memory must be also von Neumann mapped xdata/code space.

During operation, MON390 may report the following errors:

**CONNECTION TO TARGET SYSTEM LOST**: �Vision3 has lost the serial connection to the MON390 program. This error might occur because your program re-initializes the serial interface used by MON390. This error also occurs when you single step in the serial I/O routines of your application.

**NO CODE MEMORY AT ADDRESS xxxxx**: you try to download code into ROM space or non-existing memory. The code memory must be von-Neumann mapped xdata/code RAM.

**CANNOT WRITE INTERRUPT VECTOR**: MON390 program cannot install the interrupt vectors for the serial interface. This error occurs when the code memory at address 0 cannot be accessed. Most likely this space is not von-Neumann mapped.

## Technical Support

At Keil Software, we are dedicated to providing you with the best development tools and technical support. That's why we offer numerous ways you can get the technical support you need to complete your embedded projects.

* [**Technical Support Knowledgebase**](http://www.keil.com/support)  
  More than 1500 technical support questions and answers are available in the Support Solutions Knowledgebase. When a new question arises, it is added to the knowledgebase which is continuously published to the Web. This enables you to get technical support at times when our support staff is unavailable.
* [**Application Notes**](http://www.keil.com/appnotes)  
  Numerous Application Notes help you decipher complex features and implement robust applications.
* [**Example Programs and Files**](http://www.keil.com/download)  
  Utility programs, example code, and sample projects are regularly added to the Download File section of the web site.
* [**Discussion Forum**](http://www.keil.com/discuss)  
  Post questions, comments, and suggestions to the Keil Software Discussion Forum and interact with other Keil users around the world.

Many of the features of our Technical Support Knowledgebase and Web Site are the results of your suggestions. If you have any ideas that will improve them, please [give us your feedback](http://www.keil.com/support/feedback.asp)!

## Contact Details

If you experience any problems or have any questions about this product, contact one of our [distributors](http://www.keil.com/distis) or offices for assistance.

| **In the USA...**  **Keil Software, Inc.**  1501 10th Street, Suite 110  Plano, TX  75074  USA  800-348-8051 - Sales  972-312-1107 - Support  972-312-1159 - Fax  [sales.us@keil.com](mailto:sales.us@keil.com) - Sales E-Mail  [support.us@keil.com](mailto:E-Mailsupport.us@keil.com) - Support E-mail |  | **In Europe...**  **Keil Elektronik GmbH**  Bretonischer Ring 15  D-85630 Grasbrunn  Germany  +49 89 456040-0 - Sales  +49 89 456040-24 - Support  +49 89 468162 - Fax  [sales.intl@keil.com](mailto:sales.intl@keil.com) - Sales E-Mail  [support.intl@keil.com](mailto:E-Mailsupport.intl@keil.com) - Support E-Mail |
| --- | --- | --- |

Copyright � Keil Software, Inc. and Keil Elektronik GmbH.

All rights reserved.

Visit our web site at [www.keil.com](http://www.keil.com).