MSP430 Flasher Manual

Version 1.3.2

September 2014

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

MSP430 Flasher is a user-friendly, shell-based interface providing easy access to MSP430 devices via JTAG or Spy-Bi-Wire (SBW) by porting the most common functionality of the MSP Debug Stack to the command line.

The typical *MSP430 Flasher* execution flow consists of the following steps. Optional steps can be activated/deactivated by using special triggers/parameters (see section 'Triggers and Arguments').

- Initialize FET debugger
- Perform FET recovery (in case a corrupted FET firmware is detected)
- Update FET firmware (in case of a mismatch between firmware and MSP Debug Stack)
- Power up the target MSP device
- Configure the target MSP for JTAG or SBW communication
- Connect to the target MSP and display device information
- Optional: erase (parts of) the target device's memory
- Optional: load target code into the device from a TXT or HEX file
- Optional: verify target code transfer
- Optional: read out device memory and dump it into a TXT or HEX file
- Optional: set up hardware breakpoints
- Optional: reset the device
- Optional: lock JTAG access
- Optional: reset the device
- Optional: power down the device
- Optional: start target code execution and run to hardware breakpoints
- Disconnect from the target MSP
- Close FET connection

Status reports are logged into a text file called *log.txt* in the subdirectory Log, which has to be in line with the *MSP430 Flasher* executable. If the directory does not exist, it will automatically be created. New instances are appended to the logfile so old logs are never overwritten.

2 Compatibility

MSP430 Flasher officially supports the following operating systems:

- Windows 7 32bit/64bit
- Windows 8 32bit/64bit
- Windows XP 32bit/64bit
- Ubuntu 12.04 32bit

NOTE: ***MSP430 Flasher for Linux does not support eZ430 development tools.*** This includes the eZ430 Value Line Launchpad, eZ430 Chronos and older MSP-EXP430 experimenter's boards with eZ430 onboard emulation.

NOTE: Building *MSP430 Flasher* on Linux is currently only supported on 32bit systems.

MSP430 Flasher requires a hardware interface to communicate with MSP target devices. The following TI flash emulation tools (FETs) are supported:

- MSP-FET430UIF
- eZ-FET & eZ-FET lite
- eZ430 (incl. LaunchPad)

NOTE: **Do not disconnect the JTAG or emulator USB cable while MSP430 Flasher is running.** Wait until MSP430 Flasher execution is finished before disconnecting the debugger or target device.

NOTE: In order to differentiate between multiple eZ430 tools (i.e. two or more Value Line LaunchPads connected to the same host PC) consider connecting each tool individually and look out for the unique identifier being reported by *MSP430 Flasher* ("Found USB FET @ **HID0xxx:COMxxx**"). Use this identifier with the **–i** switch whenever more than one eZ430 debugger is connected.

3 Triggers and Arguments

MSP430 Flasher runs from an executable file called *MSP430Flasher.exe*. This file accepts a number of triggers and arguments necessary to give the user access to the full capabilities of the software. Table 1 lists all available triggers and arguments.

Trigger	Arguments	Description / Additional Information
-h / -?	N/A	Usage information (displays this table of command line switches)
-x	N/A	Displays available exit specifications (see trigger –z)
-i	(TI)USB - **default** COM*n* – (i.e. COM15) HID*n*:COM*n*	Communication port for the FET debugger – TIUSB (or USB) is the default. Use COM*n* to choose a debugger connected to COM port *n*. Use HID*n*:COM*n* for eZ430 tools (see note in section 'Compatibility').
-m	AUTO - **default** JTAG SBW2 SBW4	This trigger specifies which communication mode will be used. - AUTO for automatic communication mode detection (default) - JTAG for devices which **only** support 4-wire JTAG. - SBW2 for 2-wire Spy-Bi-Wire communication - SBW4 for 4-wire JTAG communication with devices which **also** support Spy-Bi-Wire
-n	Device name	Optional - The name of the device being accessed (prompt in case of mismatch between found and selected device)
-r	[Filename, mem_section]	This switch triggers a read operation in target device memory section specified by *mem_section.* The memory content will be written to a file specified by *Filename.* Available memory sections are: MAIN – the device's main memory INFO – info memory (see trigger –u) BSL – bootstrap loader memory (see trigger –b) RAM – random access memory *0x****-0x**** - custom memory section
-w	Filename	This switch triggers a memory write operation. The accepted formats are TXT (TI-txt) or HEX (Intel-hex).
-v	N/A	Triggers a verification of the device memory after programming
-u	N/A	This trigger unlocks locked flash memory (InfoA) for writing
-b	N/A	This trigger unlocks the BSL memory for writing
-e	ERASE_ALL - **default**	Triggers an erasure of the device's MAIN memory

	ERASE_MAIN	(ERASE_MAIN) or MAIN & INFO memory including INFOA segment if unlocked (ERASE_ALL)
	ERASE_SEGMENT	See section "Segment Erase" below. Use only with –w switch!
	ERASE_TOTAL	**Applicable for FR5xx/FR6xx families only (except FR57xx)!** Triggers a complete erase of the target device's memory overriding and resetting any memory protection settings.
	ERASE_USER_CODE	**Applicable for FR4xx devices only!** Overrides and clears FRAM memory protection (see SLAU445 user's guide) and erases INFO and MAIN memory.
-d	[breakpoint addresses]	Set hardware breakpoints. Use "," as delimiter.
-t	Timeout_in_ms	Specifies a timeout for breakpoints (counted in milliseconds).
-p	JTAG password	This trigger specifies the JTAG password that should be used to open a password protected target device (supported on FRAM devices only). The user is prompted if the password is incompatible with the password length specified by trigger -l.
-l	password_length	Optional - The length of the JTAG password specified via trigger –p. Can be used for double-checking the entered password. Length is counted in 16-bit words! Default = 0 – Use decimal format
-o	L	Operating mode for L092 devices.
	C	L – L092 mode (with external EPROM) C – C092 mode
-f	N/A	Permanently locks JTAG access to the target MSP. **Warning:** The device will no longer be accessible via JTAG or SBW. This action is irreversible.
-g	N/A	This trigger disables the logging mechanism
-s	N/A	This trigger suppresses the FET firmware update user prompt. In case of a mismatch between MSP Debug Stack and FET firmware, an update will be forced.
-q	N/A	Triggers QUIET mode – No system messages will be displayed (except for errors and user prompts)
-z	[exit_spec,…]	This trigger specifies the state of the device after programming. For available exit specs see Table 2. Use "," as a delimiter.
Omitted arguments will be replaced by the default options if possible or the user will be prompted to provide them later.		

Table 1 Available Triggers and Arguments

4 Exit specifications

The user can select the desired state for the device to be set to when *MSP430 Flasher* finishes its operation. This can be done using the trigger -z and passing the argument(s) [exit_spec,…], where *exit_spec* is a valid exit specification (see table below).

NOTE: The specifications are delimited with the ',' (comma) character and enclosed by square brackets.

Exit specification	Description
default (-z not used)	The device does not receive a 'hard' reset and is powered down after programming. Target code execution does not start.
-z [VCC]	VCC is set to the default value of 3000mV. Target code execution starts.
-z [VCC=*3600*]	The target VCC is set to a custom value (specified in millivolts). Valid voltages range from 1800 to 3600 mV. Target code execution starts. Please note that eZ430 and eZ-FET debuggers do not support target voltages other than 3000 mV!
-z [RESET]	The device receives a 'hard' reset (using the RST/NMI pin) after programming and is powered down.
-z [VCC(=*x*), RESET] **-z [RESET, VCC(=*x*)]**	The device receives a 'hard' reset (using the RST/NMI pin) after programming and VCC is left on. Target code execution starts

Table 2 Available (combinations of) exit specifications

5   Firmware Update

During runtime, in case *MSP430 Flasher* detects a conflict in the existing firmware versions it will prompt the user to choose whether to let *MSP430 Flasher* update the firmware or not:

>> The firmware of your FET is outdated.
>> Would you like to update it? (Y/N): _

Entering Y will update the firmware of the FET, displaying timely status reports, and on success will continue execution of the *MSP430 Flasher* routine. Entering N will resume the running instance with the outdated firmware. In case of an error during the update the user will be prompted with:

>> Update failed. (R)etry/(C)ancel? _

Entering R will repeat the attempt to update while entering C will resume the running instance with the outdated firmware.

NOTE: It's possible to suppress this user prompt by using the –s switch. In case of a mismatch between FET firmware and MSP Debug Stack version, a firmware update will be done automatically.

---

6   Segment Erase

As of version 1.2.2, *MSP430 Flasher* supports erasure and reprogramming of a single memory segment while the rest of the device's memory is left untouched. In order to use this feature, the **–e** switch has to be used with the ERASE_SEGMENT option.
The user has to provide a TI-txt or Intel-hex file which contains the target code in one continuous block only. The start address of this memory block defines the segment which should be erased.

NOTE: The size of the memory block that will be programmed must not exceed the size of the segment in which it should be programmed. Memory segments are either 256, 512 or 1024 Byte of size and have fixed addresses inside the main memory depending on the MSP430 device. Please check the device user's guides for the segment size and location for a specific target device.

NOTE: The entire segment will be erased prior to programming, even if the memory block to be programmed is smaller than the memory segment size.

---

7 Example Cases

7.1 Loading and executing target code from a TXT file

Details:

- Device: MSP430F5438A
- Interface: USB
- Password: N/A
- File: file.txt (in the same directory as the executable)
- Erase Type: ERASE_ALL
- Verification: TRUE
- VCC: ON

NOTE: To load a TI .txt or Intel .hex file, the user must first ensure that the file to be loaded is in the same directory and level as the executable or a valid path is specified.

The command line to use in this case would be:

MSP430Flasher.exe -n MSP430F5438A -w file.txt -v -z [VCC] (-i USB) (-e ERASE_ALL)

NOTE: Triggers -p and -l are not used because the device does not require a password. Triggers –l and -e may be used but are unnecessary since USB and ERASE_ALL are the default settings for these parameters respectively.

Figure 1 shows the console output on entering the command line above into Windows command prompt if the selected device is in fact connected via the specified COM port.
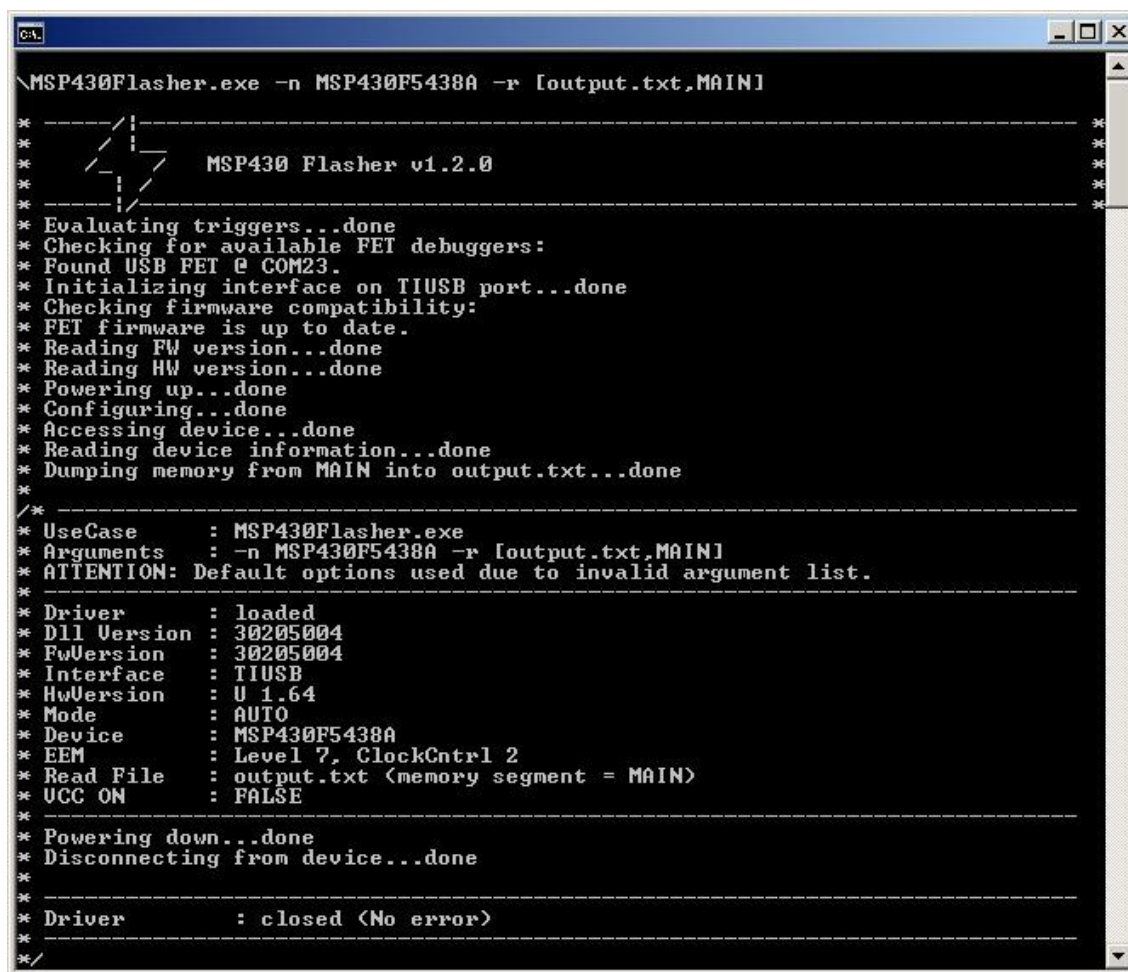
Figure 1 Loading and executing target code from a .txt file

7.2 Reading out device memory

MSP430 Flasher provides the user with the ability to read out any section of the device memory into a file of choice. The four memory sectors are MAIN, INFO, RAM and BSL. In this example the MAIN memory of an MSP430F5438A is dumped into a file called output.txt.

MSP430Flasher.exe -n MSP430F5438A -r [output.txt,MAIN]

Figure 2 shows the console output on entering the command line above.

Figure 2 Reading out device memory

7.3 Setting hardware breakpoints

As a sanity check after programming an MSP430 device, *MSP430 Flasher* offers the possibility of setting hardware breakpoints anywhere in the MAIN memory of the device. If the user entered breakpoints using the –d trigger, *MSP430 Flasher* will execute the program code, reading and displaying the CPU registers each time a breakpoint is hit. Once a breakpoint is hit, it will be cleared.

Details:
- Device: MSP430F5438A
- Interface: USB
- File: file.txt
- Erase Type: ERASE_ALL
- Verification: TRUE
- Breakpoints: [0x03F88,0x5C18,0x13F66]
- Breakpoint timeout: 1 second
- VCC: ON

NOTE: BP address #1 (0x03F88) is invalid, as it is not in the device's MAIN memory.
BP address #3 (0x13F66) will not be reached as it's not an address inside the boundaries of the target code.

MSP430Flasher.exe –n MSP430F5438A –w file.txt –v –d [0x3F88,0x5C18,0x13F66] –t 1000 –z [VCC]

Figure 3 shows the console output on entering the command line above.

NOTE: BP address #1 (0x3F88) generates an error message because of its invalid address

The first valid breakpoint entered will be referred to as "Breakpoint #1".

The breakpoints are numbered according to their input order.

Breakpoint #1 (0x5C18) was hit and cleared. The CPU state was displayed.

Breakpoint #2 (0x13F66) timed out. Despite its valid address, it couldn't be reached during target code execution.

```
C:\                                                                  _ □ ×

\MSP430Flasher.exe -n MSP430F5438A -w file.txt -v -d [0x3F88,0x5C18,0x13F66]
 -t 1000 -z [VCC]
* -----/!---------------------------------------------------------------- *
*    / !__                                                                 *
*   /_  /    MSP430 Flasher v1.2.0                                         *
*    ! /                                                                   *
* -----!/--------------------------------------------------------------- *
* Evaluating triggers...done
* Checking for available FET debuggers:
* Found USB FET @ COM23.
* Initializing interface on TIUSB port...done
* Checking firmware compatibility:
* FET firmware is up to date.
* Reading FW version...done
* Reading HW version...done
* Powering up...done
* Configuring...done
* Accessing device...done
* Reading device information...done
* Initializing EEM...done
* Loading file into device...done
* Verifying transfer...done
* Setting Breakpoints...
* Error creating breakpoint @ 0x3f88: invalid address!
*
/* -----------------------------------------------------------------------
* UseCase      : MSP430Flasher.exe
* Arguments    : -n MSP430F5438A -w file.txt -v -d [0X3F88,0X5C18,0X13F66] -t 100
0 -z [VCC]
* ATTENTION: Default options used due to invalid argument list.
* -----------------------------------------------------------------------
* Driver       : loaded
* Dll Version  : 30205004
* FwVersion    : 30205004
* Interface    : TIUSB
* HwVersion    : V 1.64
* Mode         : AUTO
* Device       : MSP430F5438A
* EEM          : Level 7, ClockCntrl 2
* Prog.File    : file.txt (ERASE_ALL, verified = TRUE)
* BSL Unlock   : FALSE
* InfoA Access : FALSE
* VCC ON       : TRUE
* Breakpoints  : 2 set
* -----------------------------------------------------------------------
* Running Program in RUN_TO_BREAKPOINT mode...
  Waiting for ->   Breakpoint hit   (timeout 1 sec)...
 MESSAGE: Breakpoint hit...
  EVENT: occured
* Hit breakpoint #1 @ 0x5c18...
* Clearing Breakpoint #1...   ||   Processor State: CPU Registers
  ||                  PC=0x05C18 R4 =0x6BFBE
  ||                  SP=0x05BFA R5 =0x6BFBE
  ||                  SR=0x00001 R6 =0x6BFBE
  ||                             R7 =0x6BFBE
  ||                             R8 =0xFFFFF
  ||                             R9 =0xFFFFF
  ||                             R10=0xFFFFF
  ||                             R11=0xFEFFF
  ||                             R12=0x00000
  ||                             R13=0x00002
  ||                             R14=0x00182
  ||                             R15=0x005F4
* Running Program in RUN_TO_BREAKPOINT mode...
  Waiting for ->   Breakpoint hit   (timeout 1 sec)...
  EVENT: Timeout occured
* Disconnecting from device...done
*
* -----------------------------------------------------------------------
* Driver        : closed (No error)
* -----------------------------------------------------------------------
*/
```
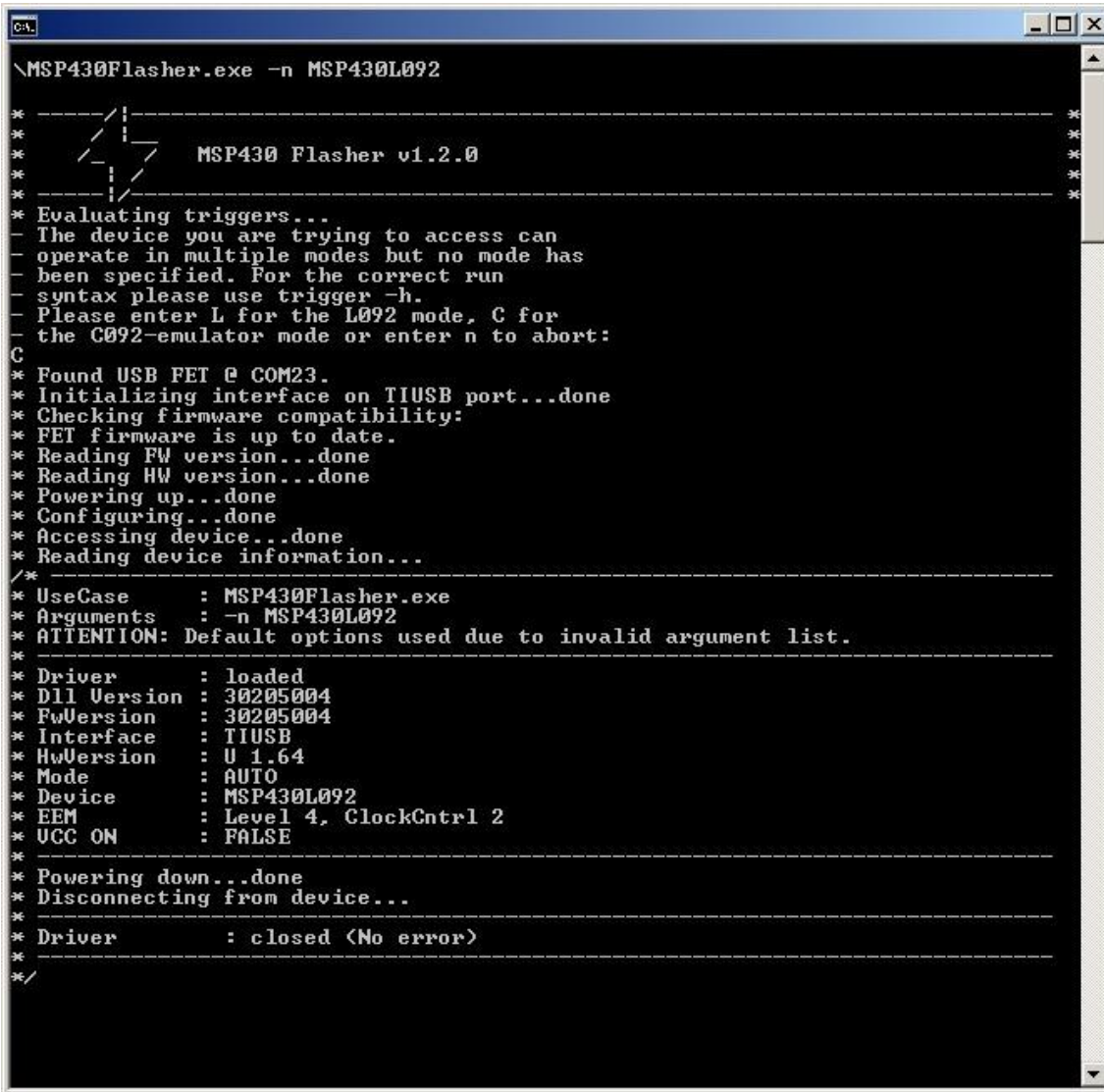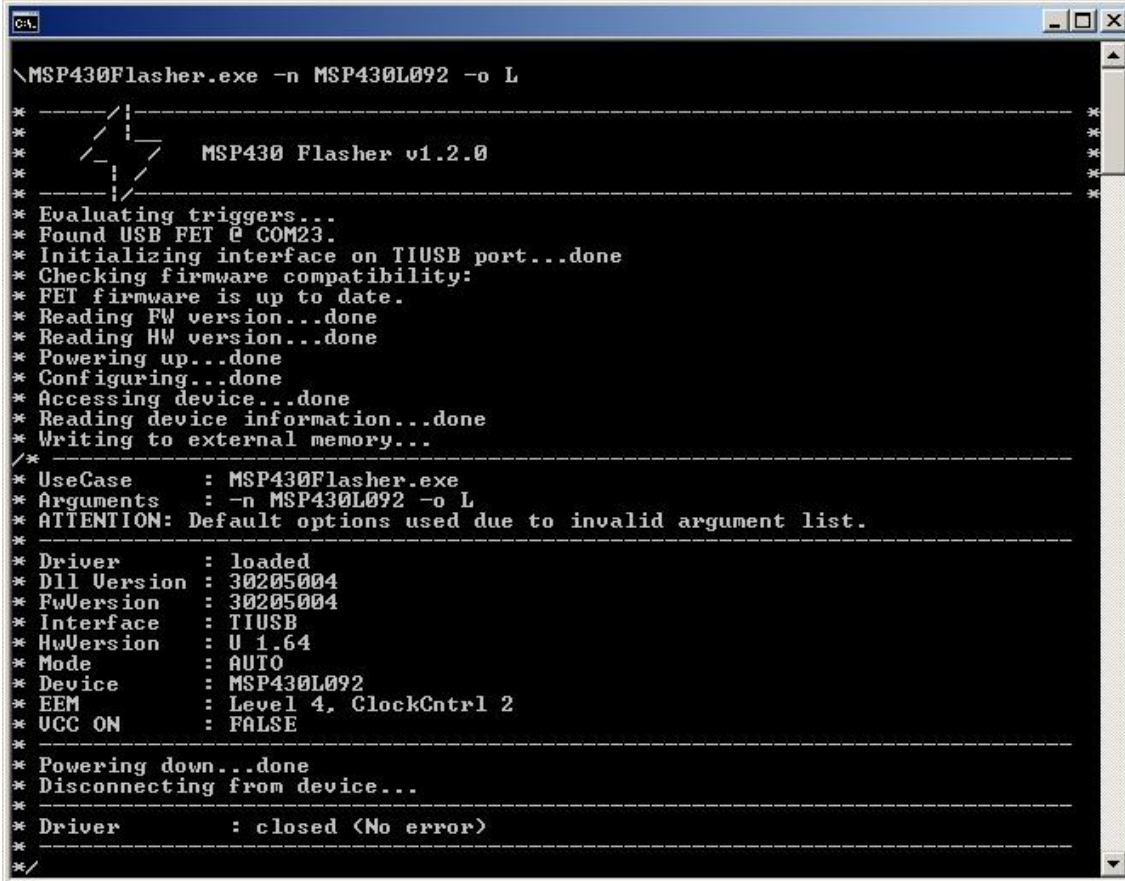
Figure 3 Setting hardware breakpoints

7.4 **Accessing a device with a device activation code**

Some devices require a device activation code in order to be operable. Devices of this kind, such as the MSP430L092 will cause an error in *MSP430 Flasher* in case the provided activation code is incorrect or no activation code is provided. *MSP430 Flasher* provides the necessary *Activation Code* internally but the user must specify the desired operating mode using the -o trigger. As can be seen below, this switch takes the argument 'L' when the L092 operating mode (with external memory) is desired and 'C' when the C092 operating mode (without external memory) is desired.

Figure 4 shows the console output if the following command line is entered:

MSP430Flasher.exe –n MSP430L092


```
\MSP430Flasher.exe -n MSP430L092

*  -----/¦--------------------------------------------------   *
*      / ¦__                                                    *
*    /_  /    MSP430 Flasher v1.2.0                             *
*     ¦ /                                                       *
*  -----¦/--------------------------------------------------   *
* Evaluating triggers...
- The device you are trying to access can
- operate in multiple modes but no mode has
- been specified. For the correct run
- syntax please use trigger -h.
- Please enter L for the L092 mode, C for
- the C092-emulator mode or enter n to abort:
C
* Found USB FET @ COM23.
* Initializing interface on TIUSB port...done
* Checking firmware compatibility:
* FET firmware is up to date.
* Reading FW version...done
* Reading HW version...done
* Powering up...done
* Configuring...done
* Accessing device...done
* Reading device information...
/* --------------------------------------------------------
* UseCase      : MSP430Flasher.exe
* Arguments    : -n MSP430L092
* ATTENTION: Default options used due to invalid argument list.
* --------------------------------------------------------
* Driver       : loaded
* Dll Version  : 30205004
* FwVersion    : 30205004
* Interface    : TIUSB
* HwVersion    : U 1.64
* Mode         : AUTO
* Device       : MSP430L092
* EEM          : Level 4, ClockCntrl 2
* VCC ON       : FALSE
* --------------------------------------------------------
* Powering down...done
* Disconnecting from device...
* --------------------------------------------------------
* Driver       : closed (No error)
* --------------------------------------------------------
*/
```

Figure 4 Accessing a L092 device without specifying an operating mode

The user is prompted to select and operating mode when the device name is found to be MSP430L092 and no mode has been selected. When 'C' is entered as the device's operating mode, the external memory is not accessed.

Figure 5 shows the console output if the same command line is entered with an additional –o switch specifying the operating mode in advance.

MSP430Flasher.exe –n MSP430L092 –o L

Figure 5 Accessing a L092 device

The L092 mode was selected from the start so the user was not prompted for additional input. Note also that the *MSP430 Flasher* wrote to the external memory: "*Writing to external memory…*"

7.5 Locking JTAG access to the target device

MSP430 Flasher offers the option of permanently locking JTAG access to the target device by specifiying the –f switch. For older MSPs from the 1xx, 2xx and 4xx families this triggers blowing a device internal poly fuse, thus making the JTAG interface physically and irreversibly unusable. For newer MSPs (i.e. from the 5xx and 6xx families) the –f programs the so called 'electronic fuse' or 'soft fuse' which has the same effect (see family user's guide for more details).

NOTE: Breakpoint functionality is disabled when the –f switch is used.

MSP430 Flasher cannot blow the JTAG security fuse of MSP430L092 devices.

MSP430Flasher.exe –n MSP430F5438A –f

Figure 6 shows the console output on entering the command line above.

Figure 6 Locking JTAG access to the target device

MSP430Flasher.exe –n MSP430F5438A –r [out.txt,MAIN]

Figure 7 shows the console output on entering the command line above in order to try reading the device's main memory after blowing the JTAG security fuse.
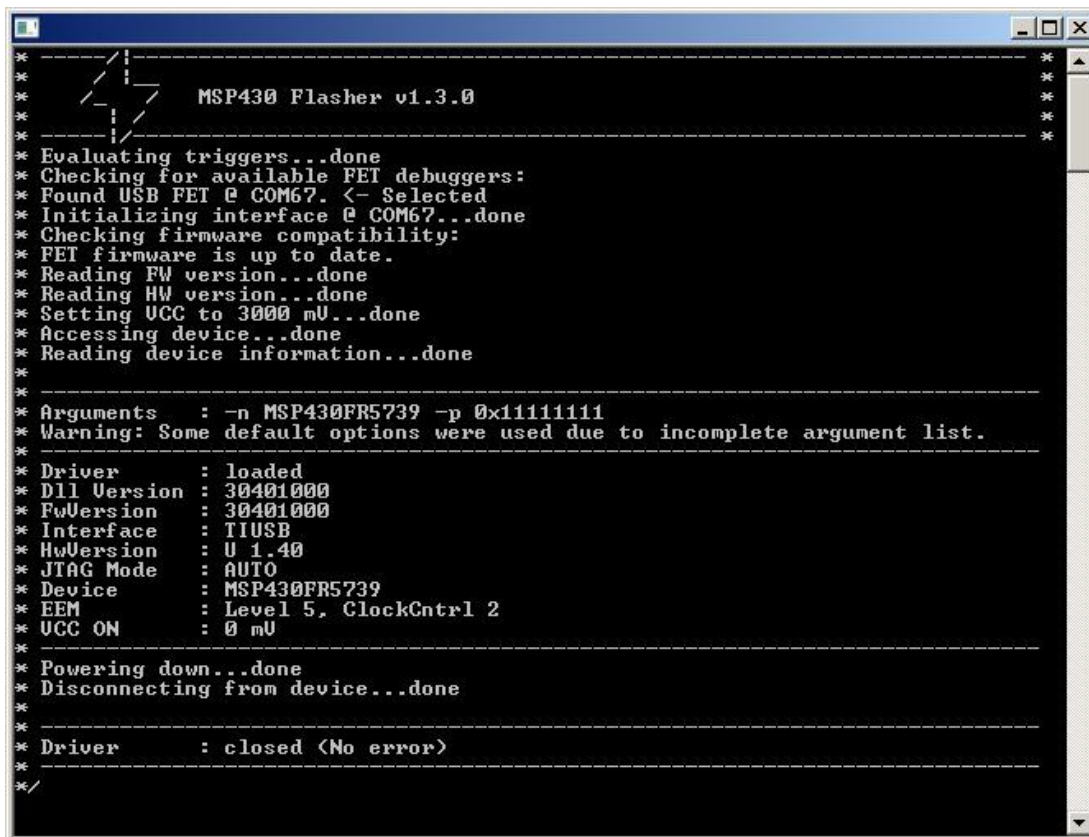

Figure 7 Trying to access a device with a blown JTAG security fuse

7.6   Unlocking a password protected target device

Newer MSP devices from the FRxx families support a JTAG lock mechanism which can be reversed by specifying a password (see device family user's guide – i.e. SLAU272 – section 1.13.2). This mechanism is not to be confused with the electronic fuse which locks the JTAG interface permanently!
In order to unlock a password protected device, use the –p switch to provide the correct JTAG password (leading with "0x") as shown below.

MSP430Flasher.exe -n MSP430FR5739 -p 0x11111111

Figure 8 Unlocking a password protected target device