

“Poor-Man’s” ARM Flash-Programming using a Wiggler-type JTAG-Interface, Keil’s μ Vision3 and Twentyone’s H-JTAG

A small tutorial by Martin Thomas, Kaiserslautern, Germany <mthomas@rhrk.uni-kl.de>

Copyright © 2006 Martin Thomas

This text or parts of this text can be copied freely but the copyright and author’s name must not be removed.
(Text Version 1.0)

Introduction

The following tutorial describes programming of the flash-memory in ARM7TDMI-Controllers with free software (free as no cost, *not* free as “free beer” and *not* free as “open source”). It will show how the μ Vision/H-JTAG/Wiggler-combination can be used as flash-programming-solution for binaries created with a GNU ARM-toolchain (i.e. my WinARM-collection).

The used software:

- H-JTAG Version 0.2.1 (beta 20060402) made by “Twentyone” (<http://twentyone.blogchina.com/>)
- μ Vision V3.30a from Keil (an ARM-company) – An evaluation-version can be downloaded from www.keil.com
- WinARM Version 20060331 (www.siwawi.arubi.uni-kl.de/avr_projects/arm_projects)

The μ Vision evaluation-version and H-JTAG are only free to use under certain conditions. Check the Web-pages for further information.

The used hardware for this tutorial:

- A Wiggler-type parallel-port JTAG-adaptor. The Wiggler-clone from Olimex has been used for this tutorial - be aware the Olimex-wiggler is not a “one-chip” device. It’s something different with “extra connections” (cit. from Sparkfun/Olimex Support-Forum). But the method described in this tutorial should work with all Wiggler-clones and of course with the original Wiggler from Macraigor..
- Keil MCB2130 evaluation-board with Philips LPC2138 ARM7TDMI-controller
- PC with PII-400 Processor, Intel Mainboard with Intel BX-Chipset, MS-Windows 2000, Parallel Port set to EPP in PC’s BIOS

The method is not limited to the LPC2138 or LPC2000-targets. μ Vision offers flash-programming for a lot of different controllers.

Software-Installation

Download the H-JTAG and μ Vision from the URLs given above. The μ Vision evaluation-version will do. If you are lucky you have the full-version of the good Keil IDE but I guess you would not read this text if you could afford the full-version.

- Install H-JTAG (start H-JTAG-[Version].exe which is a packed installer) as Administrator
- Install μ Vision as Administrator

The following steps can be done with non-administrator privileges.

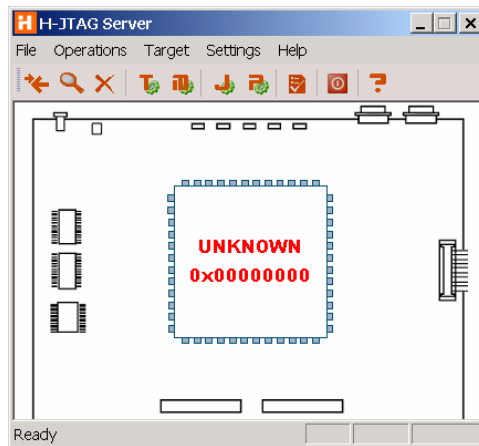
Hard- and Software-Configuration

Hardware-Connection

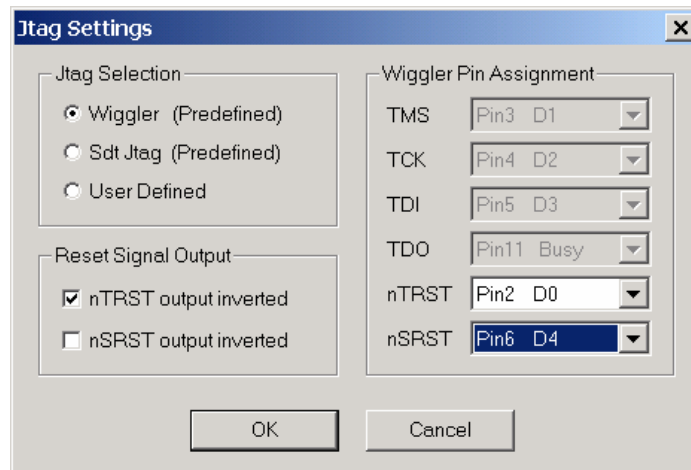
Connect your Wiggler(-clone) with the PC's parallel-port and the target-board. Verify that the JTAG-interface of the Microcontroller is enabled (here: JTAG-Jumper on MCB2130 set). Make sure CRP does not disable the JTAG-interface (do a chip-erase with the UART-bootloader and ISP-software if in doubt).

Configuration of H-JTAG

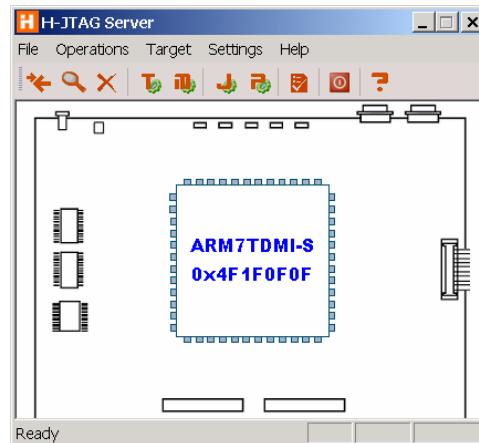
Start H-JTAG.exe (located in /Program Files(Programme)/H-JTAG/). The following window should show-up:



Obviously there is something more to configure since the target-device has not been detected correctly. The following configuration might only be needed for the Olimex Wiggler-clone. The original Wiggler from Macraigor or other clones may work "out of the box". To set-up the Olimex Wiggler-clone select Settings→Jtag-Settings and modify the settings as shown in the following screenshot.

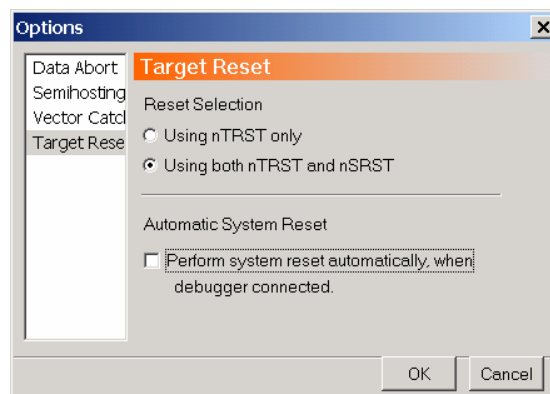


Select Menu Operations→Detect Target. For the LPC2138 which is an ARM7TDMI-S you should now see that it has been detected successfully:



For other controllers like i.e. the AT91SAM7S-series which are ARM7TDMI-controllers (no -S) the output will be different.

Since I had a “LED-blink”-firmware installed on the board I checked the reset-function Operations→Reset Target. This did not reset the target with the default configuration. To enable the reset-function choose Settings→Options→Target Reset and activate “Using both...” in the dialog.



Configuration of μ Vision

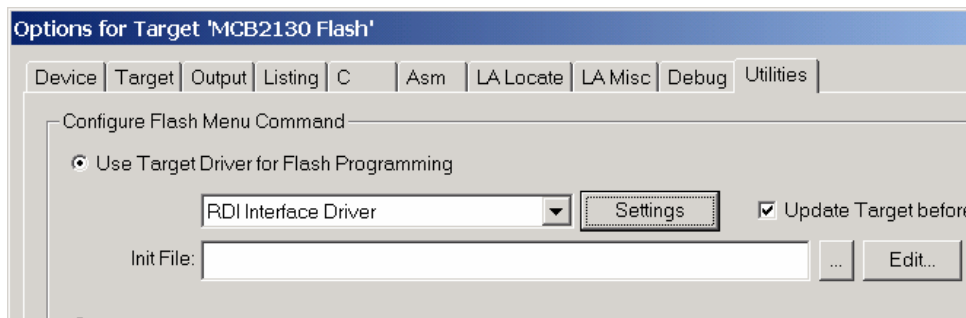
An existing example-project which comes with μ Vision is used to verify the connection: start μ Vision and select Project→Open Project→browse to c:\Keil\ARM\Boards\Keil\MCB2130\Blinky and select Blinky.uv2. The projects-workspace will be opened. Select Projects→Components... from the main-menu and select MCB2130 flash in the dialog's Project Targets list and hit the Set As Current Target button. Select Projects→Rebuild all Target Files which should compile the source-code, the output should be something like:

```
Build target 'MCB2130 Flash'
assembling Startup.s...
compiling Blinky.c...
compiling Serial.c...
linking...
Program Size: data=1199 const=65 code=5796
“.\Flash\Blinky” - 0 Error(s), 0 Warning(s).
```

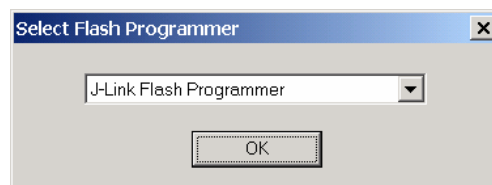
Now μ Vision has to be configured to use H-JTAG and the Wiggler. Since H-JTAG provides an RDI-interface μ Vision will be set-up to use this interface. First of all: make sure H-JTAG is started (window: H-JTAG Server, executable /program files/H-JTAG V0.2.1/H-JTAG.exe):



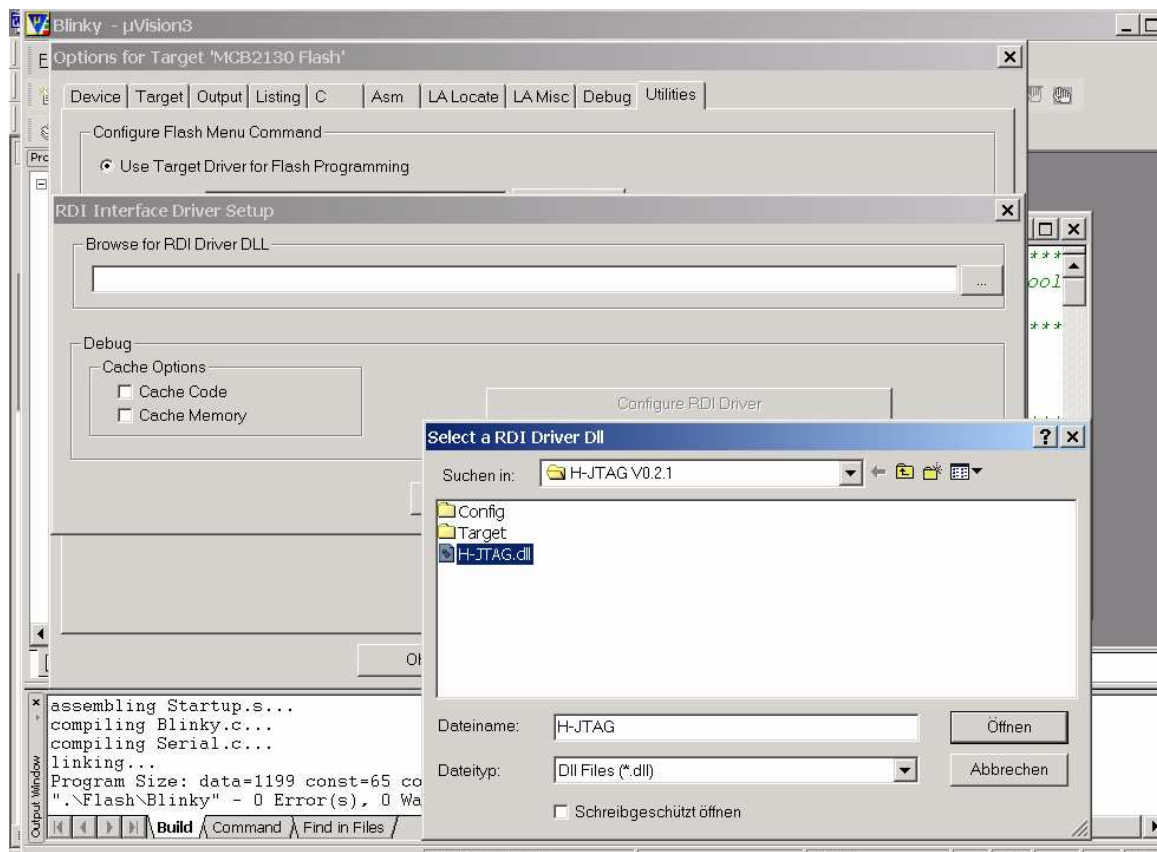
Select Project \rightarrow Options for Target 'MCB2130 flash' from the μ Vision main-menu. In the dialog select sheet/tab "Utilities". Select RDI Interface Driver:



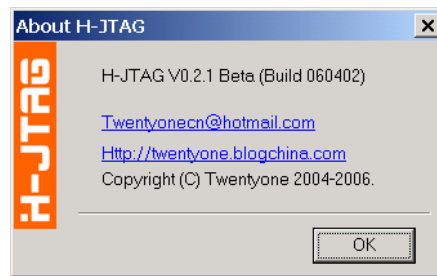
Press the Settings button, select J-Link Flash Programmer:



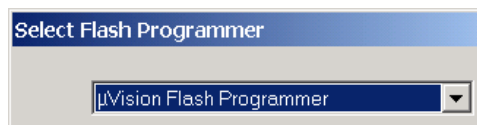
Press the [...] -button and browse for the H-JTAG RDI Driver-DLL:



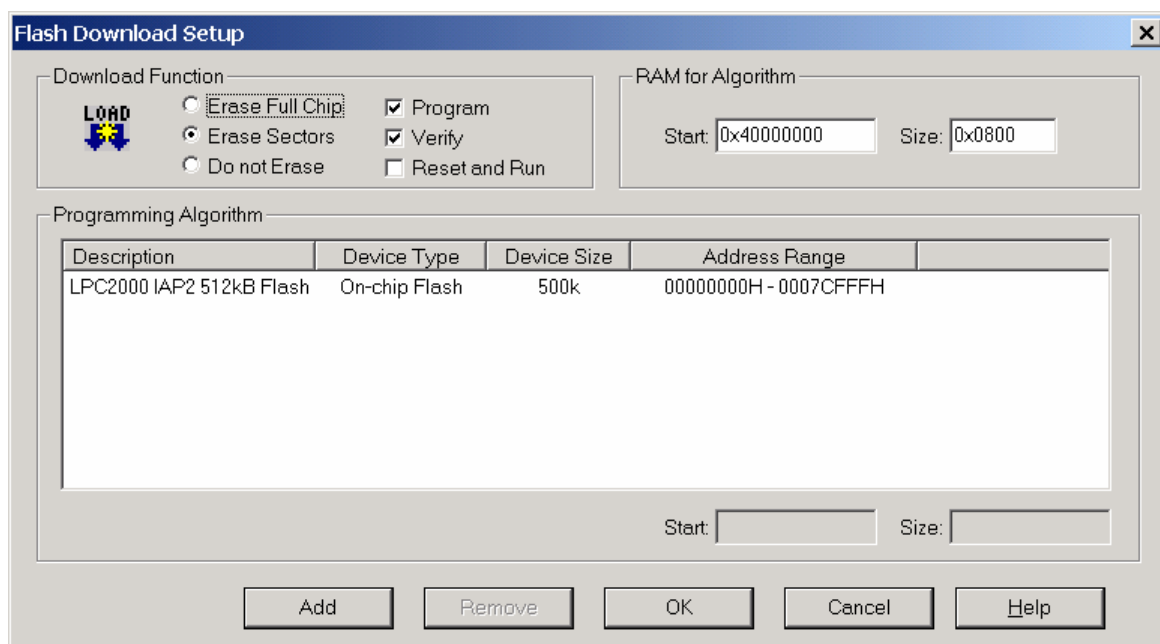
Selecting the [Configure RDI driver]-button will bring up Twentyone's "About Dialog". The H-JTAG-configuration is done from the H-JTAG window, so there is nothing to configure here.



Press [OK] in the RDI Interface Driver Setup and press [Settings] again but now select μ Vision Flash Programmer from the list:



Hit [Add] and select LPC2000 IAP2 512kB Flash from the list for the LPC2138. Modify RAM for Algorithm Size to 0x0800:

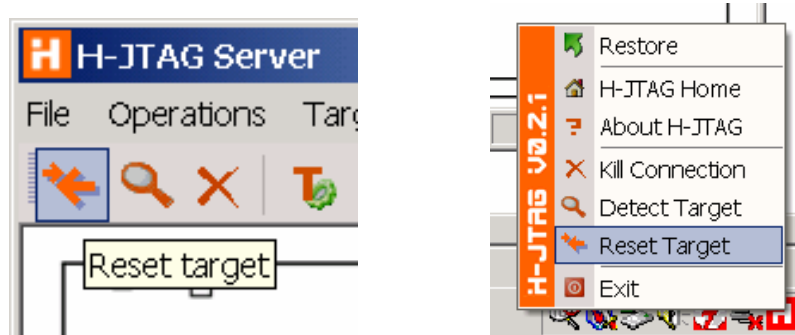


(If another controller should be programmed the matching algorithm for this controller has to be selected from the list and start and size of the working-area in RAM have to be adapted to the used target.)

Close all dialogs with [OK]. Now the flash-programming can be tested with Flash→Download from the μ Vision main-menu. The output should be like this:

```
Load "C:\\Keil\\ARM\\Boards\\Keil\\MCB2130\\Blinky\\Flash\\Blinky.ELF"
Erase Done.
Programming Done.
Verify OK.
```

Now the target can be reset from the H-JTAG-Window of the Tray-Icon (Right-Click→Reset Target) and the application gets started.



The option Reset and Run in the μ Vision Flash Download set-up seems to be ignored by the soft-/hardware used for this tutorial but give it a try if you are using a newer version of H-JTAG.

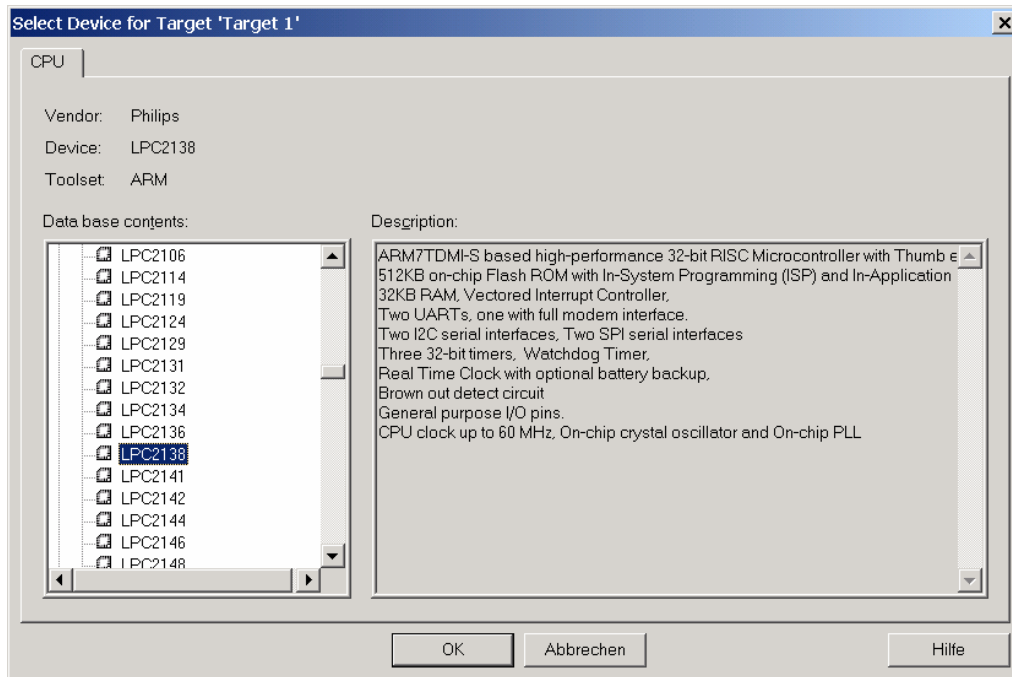
Using the Tools as Flash-Programming Solution for arm-elf-gcc

The method how a hex-file can be flashed with appropriate configuration-files and command-line options for μ Vision is described in the μ Vision online-help as “Command Line Invocation” (if remembered correctly, still can’t read the online-help on my electronics-development-PC).

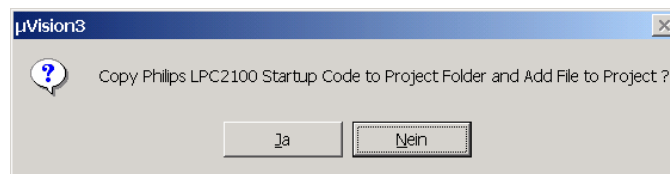
At first a μ Vision-workspace(“Project”) has to be created. Than the makefiles for the arm-elf-gcc project gets modified to call μ Vision with the appropriate command-line-options to upload a hex-file to the target-controller on “make program”.

Create a μ Vision Project for hex-File-Programming

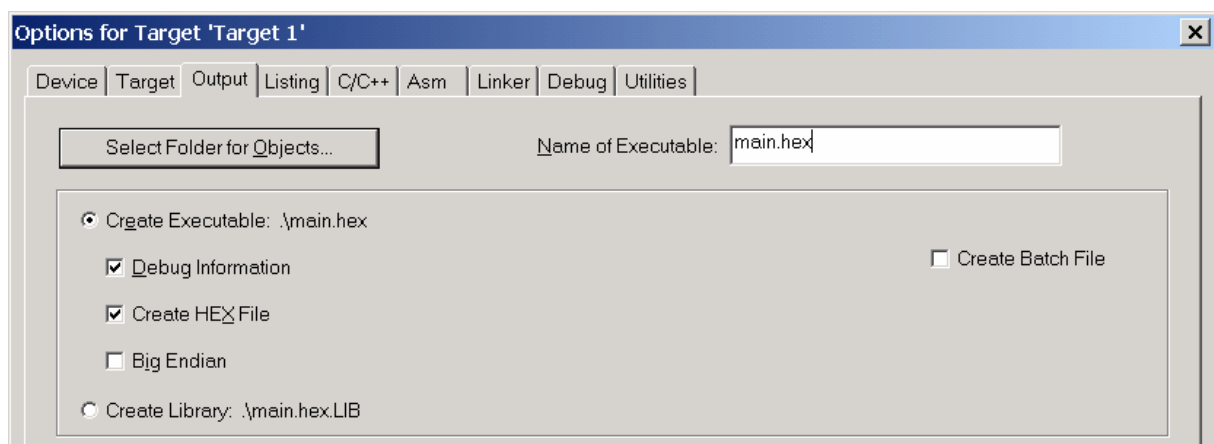
The "hex-flash"-project created in this section keeps the settings needed by μ Vision, it does not contain any source-code. First close all projects and files in μ Vision. Select Project→New Project, choose a directory and a name for the project. In this example the project is placed in c:/WinARM/examples/uvisionflash and the project's name (name of the uv2-file) is also uvisionflash. The select device dialog appears, select your target-controller (LPC2138 in this example):



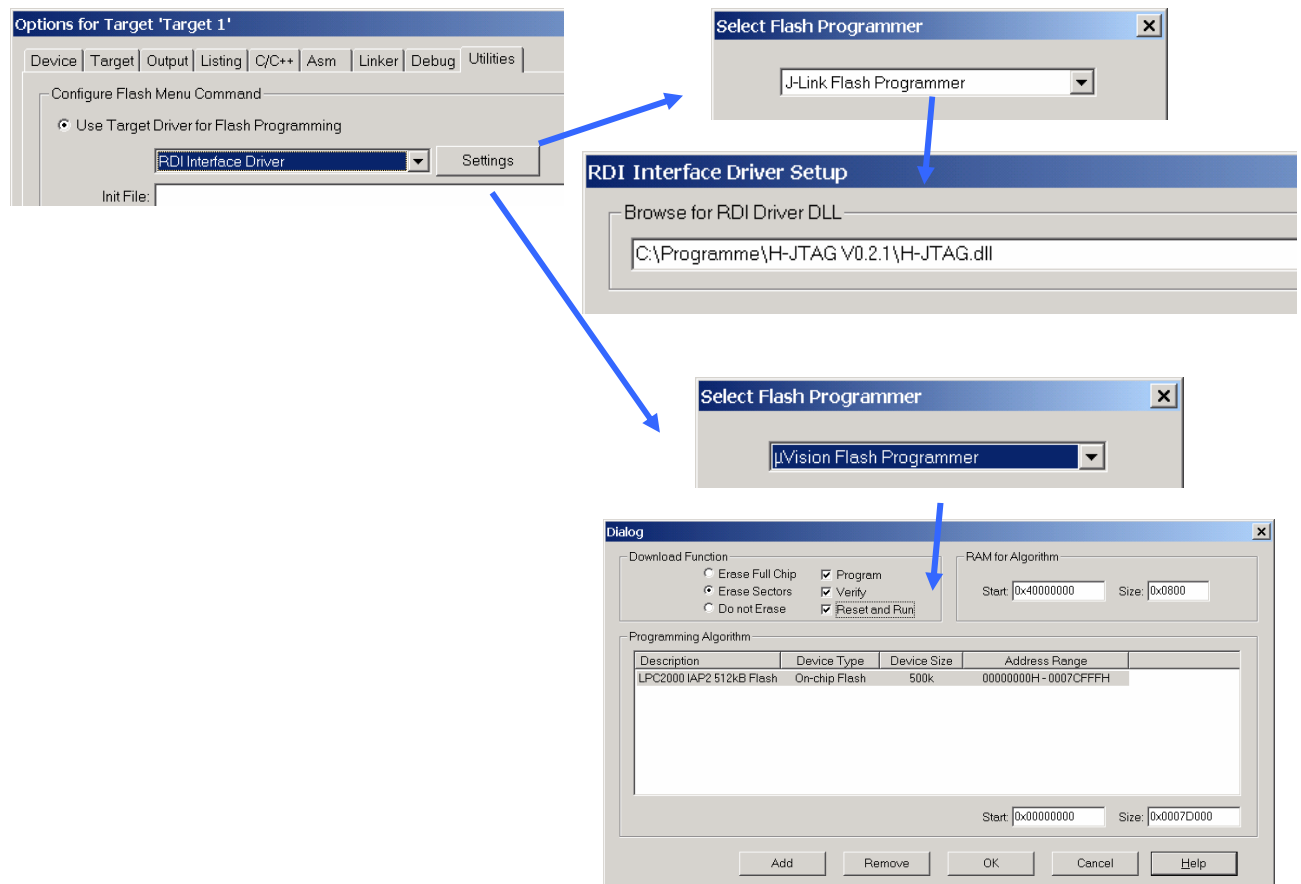
The startup-code is not needed so select [No] in the dialog:



Select Project→Options for Target 1 from the main-menu, select sheet Output and check Create HEX file, type main.exe for name of executable:



Select sheet utilities and configure the Flash Programming as described in the previous section. Some pictures as reminders:



Save the project and close µVision. There are now four files (maybe additional .bak-files) in the project directory:

Verzeichnis von C:\WinARM\examples\urovisionflash

```
[.]                [..]                main.hex.plg
urovisionflash.Opt  uvisionflash.plg  uvisionflash.Uv2
                   4 Datei(en)                3.572 Bytes
```

The files needed for the next steps are uvisionflash.Uv2 and uvisionflash.Opt.

Flash-Programming Test

The example used here is a simple LED-switch application for a LPC2138 which has been created based on the WinARM-examples and my WinARM template-makefile. First of all verify that the project is created (linked) for flash and the build-process produces a hex-File (in a makefile based on the WinARM-template: RUN_MODE=ROM_RUN, FORMAT = ihex)

The name of the target has to be the same as in the options for target dialog of the µVision project without extension. Since main.hex has been set in µVision the target-name has to be main (makefile: TARGET = main), the rules in the makefile automatically add the file-extension hex. Some excerpts from the used makefile:

```

MCU      = arm7tdmi-s
...
SUBMDL   = LPC2138
...
RUN_MODE=ROM_RUN
...
TARGET   = main
...
FORMAT   = ihex

```

A "make all" should now produce a file main.hex:

```

"make" all

----- begin (mode: ROM_RUN) -----
arm-elf-gcc (GCC) 4.1.0 (WinARM)
...
Creating load file for Flash: main.hex
arm-elf-objcopy -O ihex main.elf main.hex
...

```

Copy the two files uvisionflash.Uv2 and uvisionflash.Opt created in the previous section into the source-directory. All files in the gcc-project directory after a "make all":

```

Verzeichnis von C:\WinARM\examples\lpc2138_mcb2130

06.04.2006  13:51      <DIR>      .
06.04.2006  13:51      <DIR>      ..
06.04.2006  13:43      <DIR>      .dep
06.04.2006  13:43          14.425 crt0.lst
06.04.2006  13:43          3.012 crt0.o
13.05.2005  03:30          7.675 crt0.S
20.10.2005  02:17          3.693 LPC2138-RAM.ld
05.04.2006  02:26          3.759 LPC2138-ROM.ld
05.04.2006  02:42           350 lpc2138_testings.pnproj
05.04.2006  02:42           69 lpc2138_testings.pnps
08.03.2006  18:01         24.965 LPC214x.h
05.04.2006  02:56          2.892 main.c
06.04.2006  13:43         39.462 main.elf
06.04.2006  13:43          2.374 main.hex
06.04.2006  13:43         20.013 main.lss
06.04.2006  13:43         11.640 main.lst
06.04.2006  13:43          8.135 main.map
06.04.2006  13:43          3.656 main.o
06.04.2006  13:43          1.191 main.sym
06.04.2006  13:50         13.362 Makefile
05.04.2006  02:36           22 registers.h
05.04.2006  02:43          3.458 sysconfig.h
05.04.2006  02:46          3.327 sysTime.c
05.04.2006  02:44          3.820 sysTime.h
06.04.2006  13:43         14.295 sysTime.lst
06.04.2006  13:43          4.104 sysTime.o
06.04.2006  13:50           992 uvisionflash.Opt
06.04.2006  13:26          2.380 uvisionflash.Uv2
                25 Datei(en)          193.071 Bytes

```

Now that everything is configured and created the flash-programming can be tested. Make sure that the H-JTAG server is started and type the following line at the command prompt with the project directory being the active directory (all in one line):

```
C:\WinARM\examples\lpc2138_mcb2130>C:\Keil\uv3\Uv3.exe
-f uvisionflash.Uv2 -ouvisionflash.txt
```

µVision will now start and flash the contents of main.hex into the LPC2138 flash-memory. The output-filename (option -o) has been selected as uvisionflash.txt, after a successful programming the file has the content:

```
Load "main.hex"
Erase Done.Programming Done.Verify OK.Application running ...
```

To start the application select Reset Target from the H-JTAG menu (the µVision Reset and Run option did not work with the used hard- and software).

Modify a Project's Makefile for "µVision-Hex-File-Flashing"

Since the programming can be started from the command-line the call can be integrated easily into the makefile. I have kept both programming-methods in the makefile and have added an option to switch between µVision-programming and lpc21isp-programming.

An additional variable is used to select the option and the program make-target is modified to take the selection into account. The modifications in the makefile:

```
...
# select flash-tool:
FLASH_TOOL = UVISION

# MCU name and submodel
MCU      = arm7tdmi-s
...

...
# Program the device.
ifeq ($(FLASH_TOOL),UVISION)
# Program the device with Keil's uVision
program: $(TARGET).hex
    @echo
    @echo "Programming with uVision"
    C:\Keil\uv3\Uv3.exe -f uvisionflash.Uv2 -ouvisionflash.txt
else
# Program the device with Martin Maurer's lpc21isp
program: $(TARGET).hex
    @echo
    @echo $(MSG_LPC21_RESETRINDER)
    $(LPC21ISP) $(LPC21ISP_OPTIONS) $(LPC21ISP_DEBUG)
$(LPC21ISP_FLASHFILE) $(LPC21ISP_PORT) $(LPC21ISP_BAUD) $(LPC21ISP_XTAL)
endif
```

The modification can be tested easily by "make program". A complete "make all program" for this example:

```
C:\WinARM\examples\lpc2138_mcb2130>make all program

----- begin (mode: ROM_RUN) -----
arm-elf-gcc (GCC) 4.1.0 (WinARM)
Copyright (C) 2006 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Assembling (ARM-only): crt0.S
arm-elf-gcc -c -mcpu=arm7tdmi-s -I. -x assembler-with-cpp -DROM_RUN -Wa,-adhlns
=crt0.lst,-gdwarf-2 crt0.S -o crt0.o

Compiling C: main.c
arm-elf-gcc -c -mcpu=arm7tdmi-s -I. -gdwarf-2 -DROM_RUN -Os -Wall -Wcast-align
-Wimplicit -Wpointer-arith -Wswitch -Wredundant-decls -Wreturn-type -Wshadow
-Wunused -Wa,-adhlns=main.lst -I./include -Wcast-qual -MD -MP -MF .dep/main.o.d
-Wnested-externs -std=gnu99 -Wmissing-prototypes -Wstrict-prototypes -Wmissin
g-declarations main.c -o main.o

Compiling C: sysTime.c
arm-elf-gcc -c -mcpu=arm7tdmi-s -I. -gdwarf-2 -DROM_RUN -Os -Wall -Wcast-align
-Wimplicit -Wpointer-arith -Wswitch -Wredundant-decls -Wreturn-type -Wshadow
-Wunused -Wa,-adhlns=sysTime.lst -I./include -Wcast-qual -MD -MP -MF .dep/sysTi
me.o.d -Wnested-externs -std=gnu99 -Wmissing-prototypes -Wstrict-prototypes -W
missing-declarations sysTime.c -o sysTime.o

Linking: main.elf
arm-elf-gcc -mcpu=arm7tdmi-s -I. -gdwarf-2 -DROM_RUN -Os -Wall -Wcast-align -
Wimplicit -Wpointer-arith -Wswitch -Wredundant-decls -Wreturn-type -Wshadow -Wu
nused -Wa,-adhlns=crt0.lst -I./include -Wcast-qual -MD -MP -MF .dep/main.elf.d
crt0.o main.o sysTime.o --output main.elf -nostartfiles -Wl,-Map=main.map,
--cref -lc -lm -lgcc -TLPC2138-ROM.ld

Creating load file for Flash: main.hex
arm-elf-objcopy -O ihex main.elf main.hex

Creating Extended Listing: main.lss
arm-elf-objdump -h -S -C main.elf > main.lss

Creating Symbol Table: main.sym
arm-elf-nm -n main.elf > main.sym

Size after:
main.elf :
section      size      addr
.text        836        0
.bss         8      1073741824
.stack      2048      1073742080
.comment     54         0
.debug_aranges 96         0
.debug_pubnames 108        0
.debug_info  850        0
.debug_abbrev 536        0
.debug_line  524        0
.debug_frame 144        0
.debug_str   318        0
.debug_loc   180        0
.debug_ranges 176        0
Total       5878

Errors: none
----- end -----

Programming with uVision
C:\Keil\uv3\Uv3.exe -f uvisionflash.Uv2 -ouvisionflash.txt

C:\WinARM\examples\lpc2138_mcb2130>
```

The files uvisionflash.Uv2 and uvisionflash.Opt can be re-used in other projects. The contents of the files basically "tell" μ Vision to flash a file main.hex on a LPC2138 thru the RDI-interface.

If the target-name should be something other than "main" the file uvisionflash.Uv2 can be modified with a text-editor: open the file, search for OutName and modify the name from main.hex to the name of your hex-File (the target-name plus extension, i.e. TARGET = superapp in makefile and OutName(superapp.hex) in the Uv2-file).

Conclusion

The described method is a cost effective (=cheap) solution to program the flash-memory of different controllers.

This tutorial has been made for beginners and people with very limited budget. Commercial use is restricted: Respect the μ Vision and H-JTAG license conditions.

I hope this information is useful for you. Please send feedback, bug-reports and suggestions by e-mail.

Cheers,
Martin Thomas
mthomas@rhrk.uni-kl.de