Embedding it better...

µTasker Document

µTasker Ethernet Loader

Table of Contents

1. Introduction

Many projects can benefit from having a standalone boot loader programmed into them so that the user can load new code simply and efficiently using everyday techniques without requiring special programs or training. The µTasker Ethernet Loader is one possibility that fulfils these requirements based on fast Ethernet transfer using standard Web Browser technology. This comfortable solution is easy to use and occupies between 16k and 32k code space, depending on the processor and compiler used.

The Ethernet Loader can be pre-programmed into production devices so that boards can be manufactured with self-test and application loading capabilities. Since the Ethernet Loader can always be forced (usually by a switch that is depressed when the device starts) it can always be used to recover a device that has otherwise been programmed with invalid (in-operational) software. The Ethernet Bootloader usually has fixed MAC and IP addresses so that it can always be contacted on a known IP address.

The Ethernet Loader can be combined with Flash protection techniques to protect it from un-intentional deletion or for Flash protection.

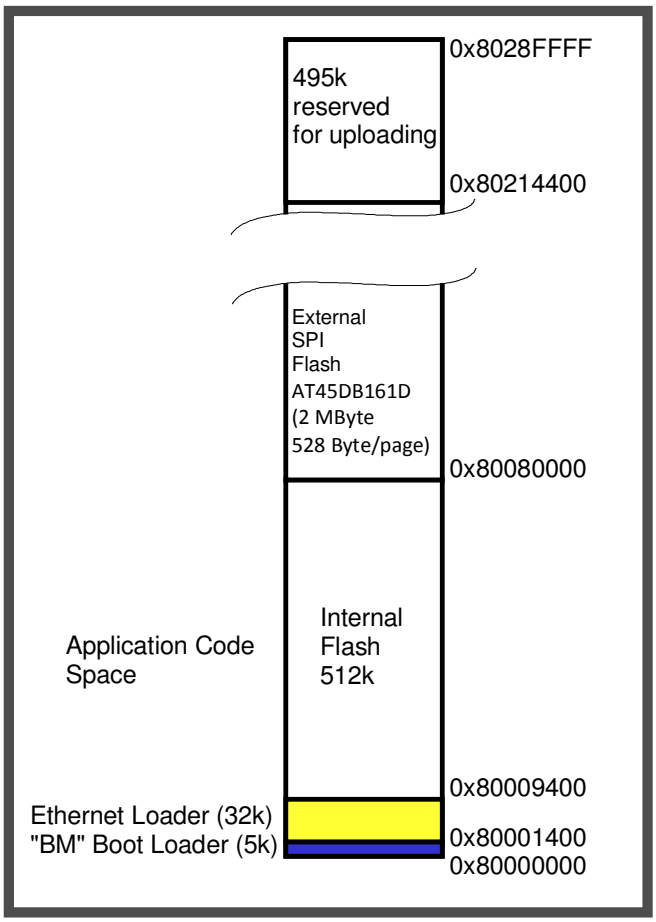
This document gives full details about working with the Ethernet Loader on various hardware platforms with various Software tools.

The µTasker Ethernet Loader can be used together with applications from other projects.

2. Example of the use of the Ethernet Loader together with "Bare-Minimum" Loader to provide encrypted Uploading of Applications and Ethernet loader Revisions

It is possible to combine the Ethernet Loader with the "Bare-Minimum" Loader to allow the application to update code directly (using the "Bare-Minimum" Boot Loader functionality) and also have a stand-alone Loader for production or fall-back use. This allows either new application software or new Ethernet Loader versions to be loaded. The encryption capabilities of the "Bare-Minimum" Loader can also be used with intermediate storage in either internal or external Flash.

The following example shows how an AVR32 project with external SPI Flash is configured to achieve this operation, based on SPI Flash as intermediate storage space. The system Flash memory map is show in the following diagram:



The "BM" Boot Loader is located at the start of internal flash and is always booted to. The "BM" project is configured for the processor (512k type assumed) with SPI support. The SPI Flash connections for the ATMEL AT45DB161D are configured with the upload location set to 0x80216c00 – *note that this is a virtual address since the SPI Flash is not physically memory mapped in the system but is assumed to be addresses from the area immediately following the end of internal Flash*.

Encryption is enabled and the option ADAPTABLE_PARAMETERS enabled so that the upload area can be used for uploading both application and Ethernet Loader code.

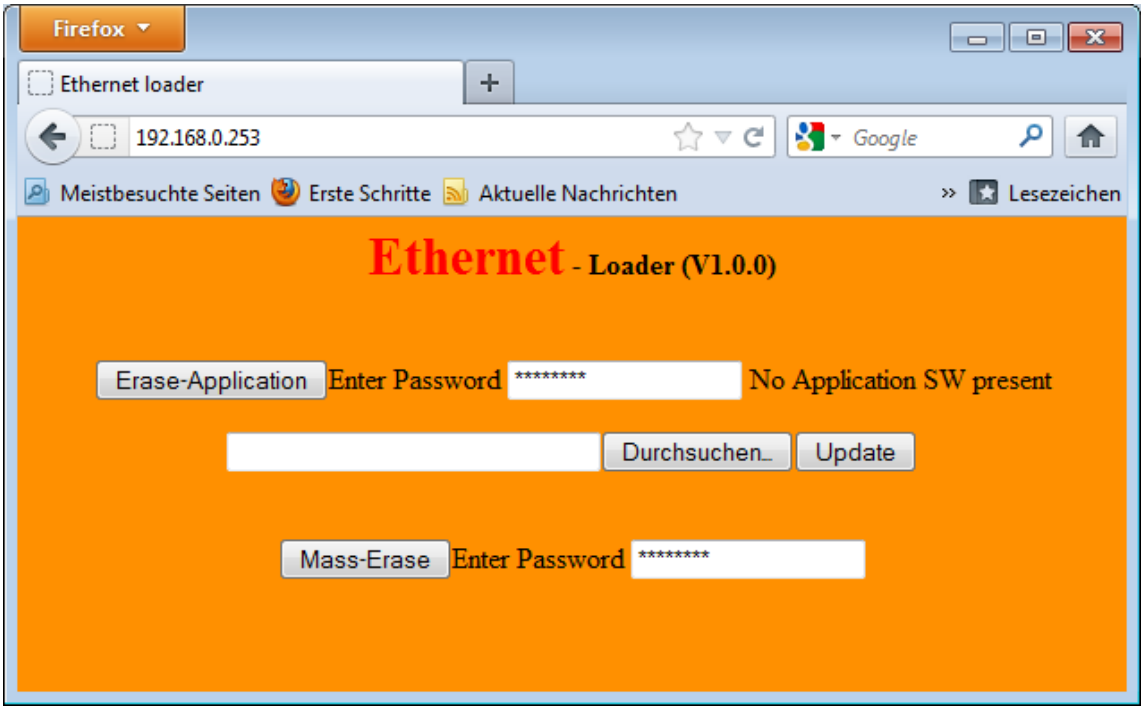
3. Conclusion

This document is in progress. See the appendix for provisional details about working with it in particular projects.

Modifications:
- V0.1 Preview with AVR32 AS6 guide

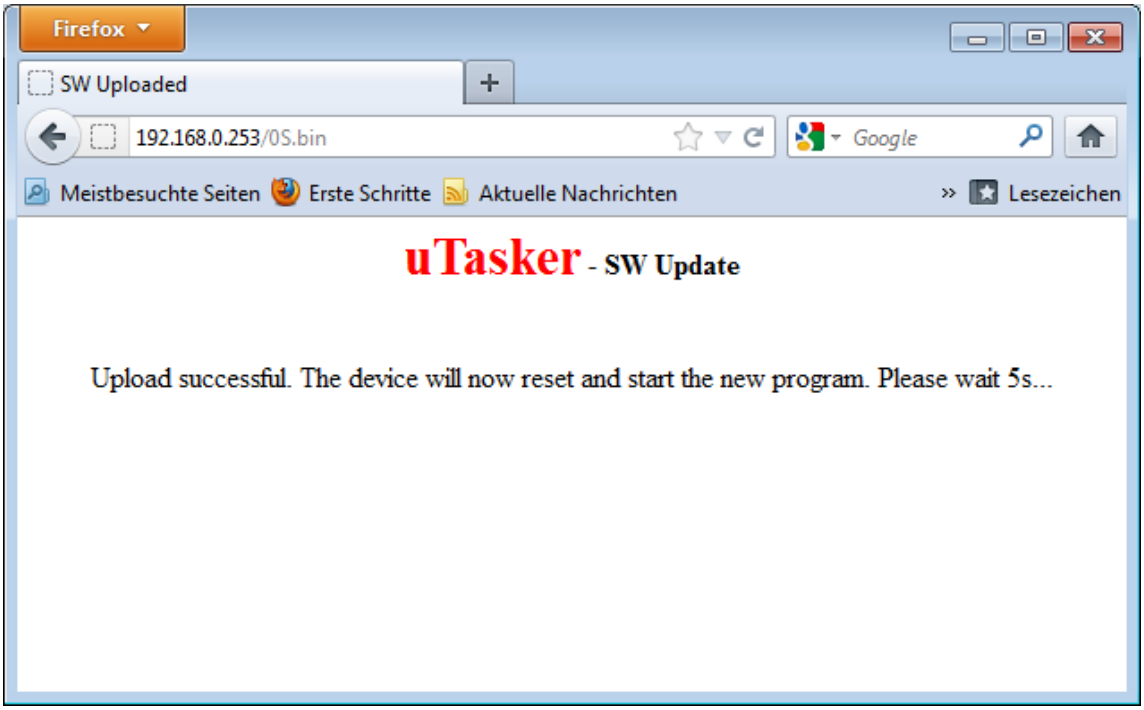
Coldfire – CodeWarrior 7

1) Build the Ethernet Loader project (with CW7.1)
   #define M52235EVB was used for tests but M52236_TOMMY is foreseen for the final board (not verified)
   The default IP and MAC addresses are set in application.c. These are fixed values and not parameters so you can set a fixed MAC address from your block reserved for this use.
   IP 192.168-.0.253 is default if not changed.
2) Delete internal Flash memory, load to the target and start – the Blink LED runs at 5Hz speed to show that the board is operating and the Ethernet loader is operating. Since there is no application loaded the boot loader always runs.
3) Browse to the board at the defined IP address to see the following web page:



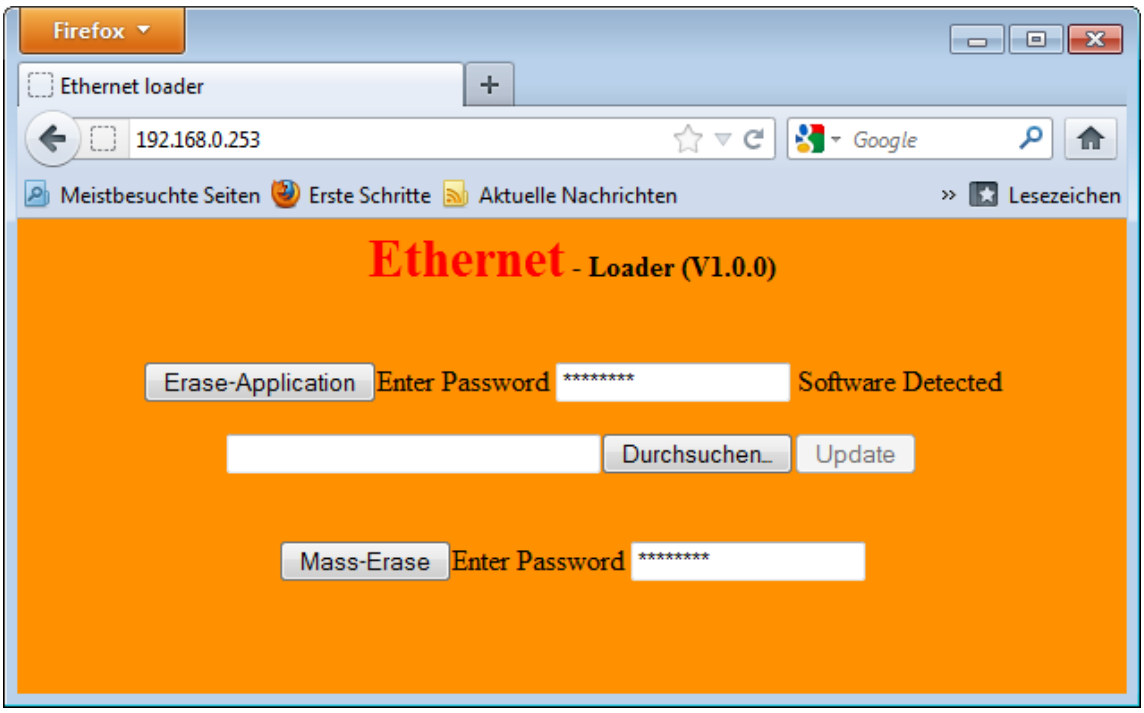
4) Since there is no application loaded an upload can be performed.
   Build therefore the uTasker main application, using the M5223X_BM_ROM target. This has been configured to link at the starting address 0x8004 (the offset of 4 bytes is due to the upload format and 0x8000 would not be correct!)
   The output "uTasker_BM.bin" is generated but this is in MOTOROLA binary format (speciality of CodeWarrior) and so is not suitable for loading just yet. Execute the bat file "BM-MakeBin.bat" which executes a conversion program in the uTasker tools directory which converts it to 'plain' binary as required by the internal Flash.
   WARNING: After each new application build one must not forget to execute the conversion. If this step can be integrated in the IDE's build sequence it is good to do it (unfortunately not possible with CW7.1)

5) Select the file “uTasker_BM_Upload.bin” which was created in the last step for the upload and then press the “Upload” button.
6) After about 2s the following screen should show that the upload was successful:



The new application should run almost immediately – the uTasker application flashes the LED at 2.5Hz and so it is clear that it is running rather than the Ethernet loader (5Hz).
The uTasker application has a default IP address at 192.168.0.3 so it can be browsed to there.

7) In order to return to the Ethernet loader the switch SW1 can be held when the board is reset and the web browser contacted again on its default IP address.
   This time the page shows that SW has been detected and the upload formula is disabled



8) In order to perform an upload the software must be deleted with the "Erase-Application" button. This is however only accepted when the password is correct. The default password is
66ED-90dk7WW65ss
and can be changed in webInterface.c
9) Once the application deletion is complete – this may take 2 or 3 seconds, depending on the size of the area being deleted, the image is the same as in 3) and so the software upload can be executed again as described there.

General set up details:

A) The start of the application is defined by APPLICATION_START_ADDRESS. This is 0x8000 in the Coldfire project since the Ethernet Loader occupies about the first 25k of Flash. The link address of the application is however 0x8004.
B) When the application is deleted an area of size FILE_SYSTEM_SIZE is deleted from the start of the application (APPLICATION_START_ADDRESS). If the system has parameters (often at the end of the Flash or sometimes between the end of the Ethernet boot-loader and the start of the application the size of FILE_SYSTEM_SIZE should be set to only delete the area for the new application and not the parameter area.

General Application requirements:

A) The application must be linked to start at the correct address (eg. 0x8004 rather than 0). The start address starts with the initial stack pointer value and then the first program counter value, as when the code were to be used for booting.
B) Since the application cannot use interrupt vectors in the boot area it must configure the interrupt vector table to be in SRAM (0x20000000) and it must then enter its interrupt into this area (0x400 bytes at the beginning of SRAM must be reserved for this).

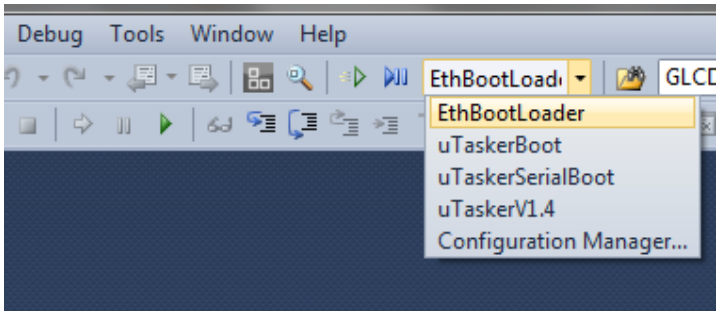
The uTasker project can be used as reference if an application is used based on another system. It shows how the vector base register is changed [mcf5xxx_wr_vbr()] and also how interrupts are entered [fnSetIntHandler()] in mcf52235_init() in M5223X.c.

C) Note that the Ethernet Loader powers down peripherals by default (see fnInitHW() in M5223X.c). This can be modified if required or else the application can power up each one that it uses as needed.

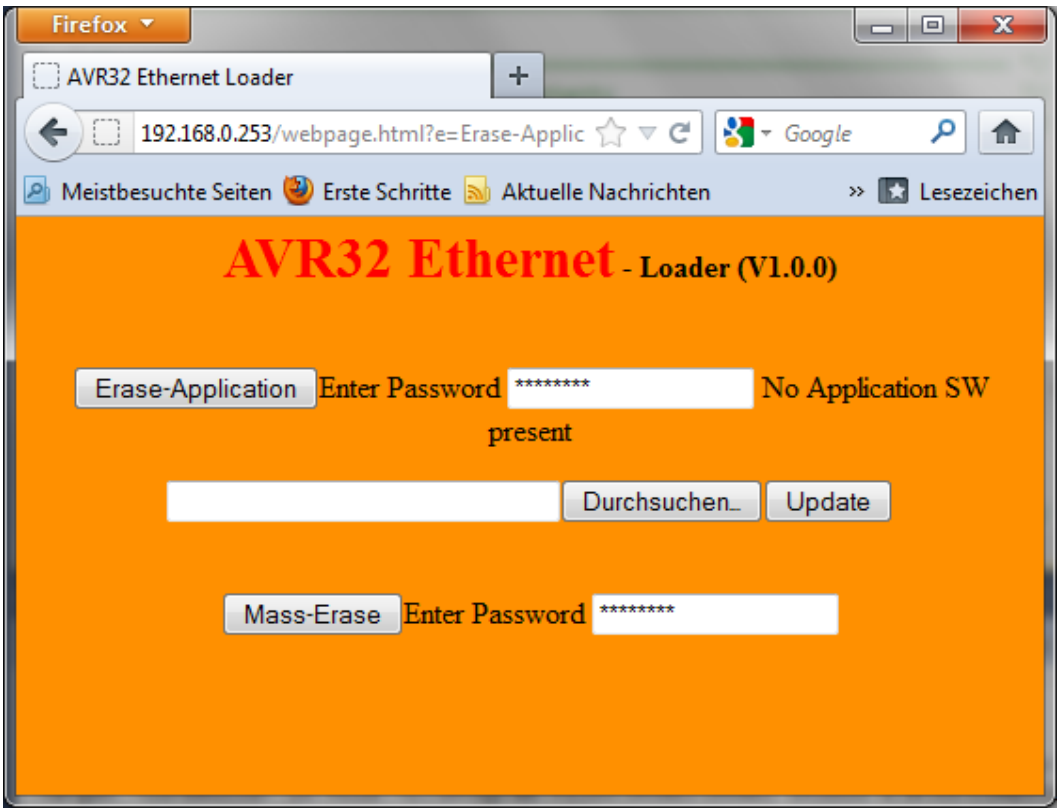
D) The Ethernet loader always configures the PLL to its defined speed (also when it doesn't stay in the Ethernet loader mode). The default M5223X speed is 60MHz (or 50MHz for processors that cannot run at 60MHz). The application doesn't need to do this but can also overwrite it as required.

AVR32 – ATMEL Studio 6

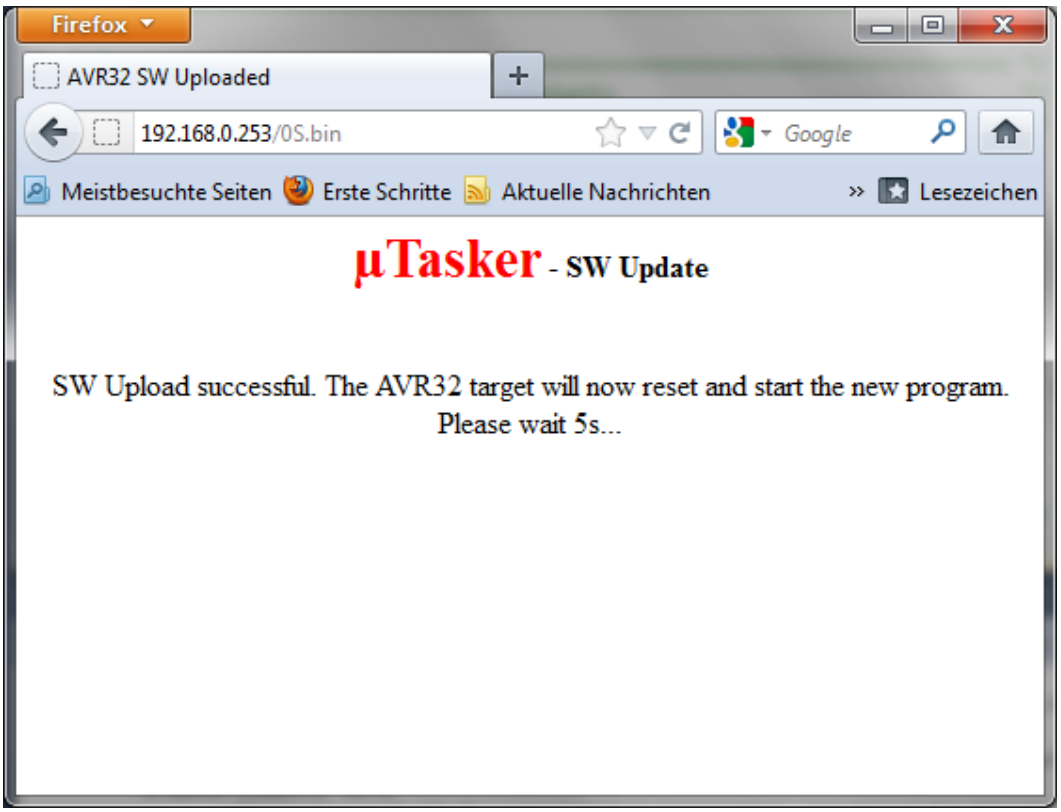
1) The AS6 project can be opened by double clicking on the project file uTaskerV1.4.atsln at the top level of the µTasker project structure.
2) There are 4 projects that can be selected in the Build configuration manager. The project called "EthBootLoader" can be selected as shown in the diagram below:



3) The board to be built for can be selected in config.h. For example, to build for the EVK1100 the define AVR32_EVK1100 is set and other board possibilities removed.
4) The project can be built by executing "Build | Build Solution" or pressing F7
   The default IP and MAC addresses are set in application.c. These are fixed values and not parameters so you can set a fixed MAC address from your block reserved for this use.
   IP 192.168-.0.253 is default if not changed.
5) It is possible to load the object to the board from within AS6 or else execute the bat file "Program_Target_with_JTAGICE_MkII.bat" in the directory "\Applications\EthBootLoader\GNU_AVR32" which will quickly load "EthBootLoader.bin" to the hardware via JTAG ICEII.
   The Blink LED runs at 5Hz speed to show that the board is operating and the Ethernet loader is operating. If there is no application loaded the boot loader always runs.
6) Browse to the board at the defined IP address to see the following web page:

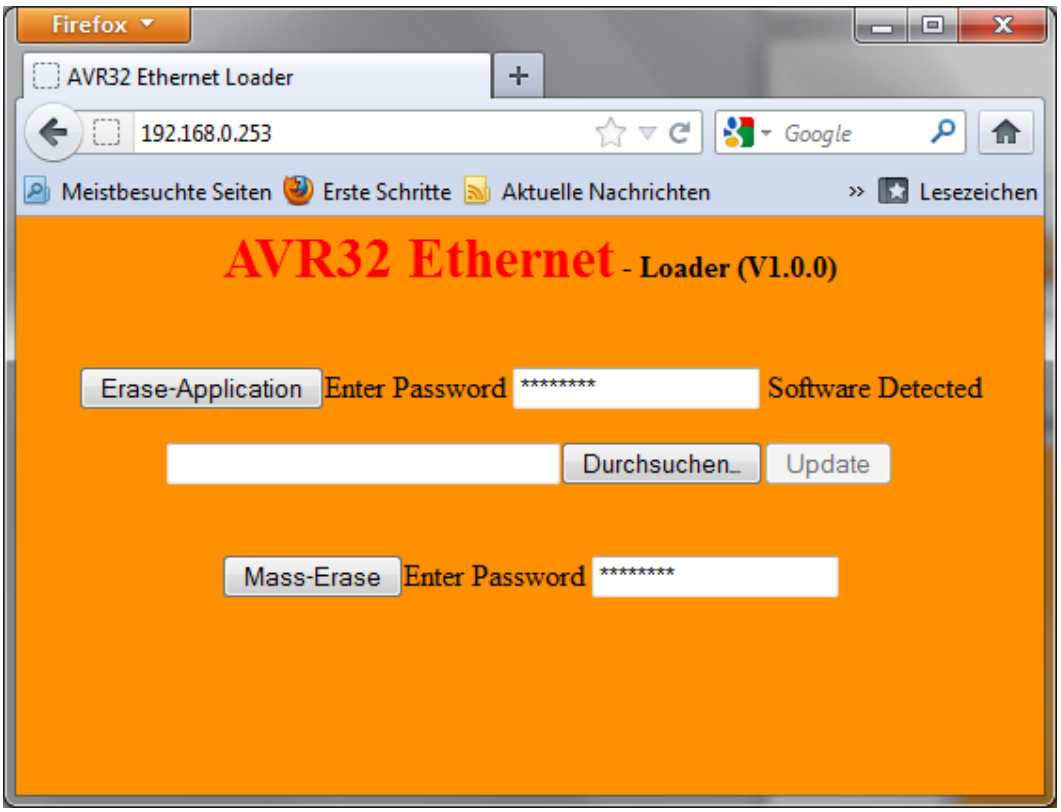


7) Since there is no application loaded an upload can be performed.
   Build therefore the uTasker main application, using the target uTaskerV1.4. This has been configured to link at the starting address 0x80008004 (the offset of 4 bytes is due to the upload format and 0x80008000 would not be correct!) – see "\Applications\uTaskerV1.4\GNU_AVR32\uTaskerV1.4_AVR32_BM_8004.ld"
8) Select the file "uTaskerV1.4_BM.bin", which was created when the project was built, for the upload and then press the "Upload" button.
9) After about 2s the following screen should show that the upload was successful:



The new application should run almost immediately – the uTasker application flashes the LED at 2.5Hz and so it is clear that it is running rather than the Ethernet loader (5Hz).
The uTasker application has a default IP address at 192.168.0.3 so it can be browsed to there.

10) In order to return to the Ethernet loader the switch PB2 (on the EVK1100) can be held when the board is reset and the web browser contacted again on its default IP address. This time the page shows that SW has been detected and the upload formula is disabled



11) In order to perform an upload the software must be deleted with the "Erase-Application" button. This is however only accepted when the password is correct. The default password is
66ED-90dk7WW65ss
and can be changed in webInterface.c
12) Once the application deletion is complete – this may take 4 or 5 seconds, depending on the size of the area being deleted, the image is the same as in 3) and so the software upload can be executed again as described there.

General set up details:

A) The start of the application is defined by APPLICATION_START_ADDRESS. This is 0x80008000 in the AVR32 project since the Ethernet Loader occupies about the first 27k of Flash. The link address of the application is however 0x80008004.
B) When the application is deleted an area of size FILE_SYSTEM_SIZE is deleted from the start of the application (APPLICATION_START_ADDRESS). If the system has parameters (often at the end of the Flash or sometimes between the end of the Ethernet boot-loader and the start of the application the size of FILE_SYSTEM_SIZE should be set to only delete the area for the new application and not the parameter area.

General Application requirements:

A) The application must be linked to start at the correct address (eg. 0x80008004 rather than 0). The start address starts with the initial stack pointer value and then the first program counter value, as when the code were to be used for booting.

B) Since the application cannot use interrupt vectors in the boot area it must configure the interrupt vector table to be in SRAM or to be located in an area of Flash corresponding to the requirements of the AVR32 interrupt controller. If using applications from other projects it is recommended to consider using the interrupt technique from the µTasker project since it allows interrupts to be defined in SRAM without the typical overhead from trampoline functions and also allows seamless operation together with the µTasker boot loader due to removal of interrupt controller restrictions of only being able to access interrupt handlers in a defined area of memory. Details of the technique can be found here: http://www.utasker.com/forum/index.php?topic=679.0 and all corresponding initialisation routines for it in AVR32.c

C) Note that the Ethernet Loader powers down peripherals by default (see fnInitHW() in AVR32.c). This can be modified if required or else the application can power up each one that it uses as needed. Application software from the µTasker project always automatically powers up peripherals based on use and powers them down once temporary use is terminated.

D) The Ethernet loader always configures the PLL to its defined speed (also when it doesn't stay in the Ethernet loader mode). The default AVR32UC3A speed is, for example, 66MHz. The application doesn't need to do this but can also overwrite it as required.