Embedding it better...

µTasker Document

µTasker – Serial Loader Updating using the “Bare-Minimum” Boot Loader

Table of Contents

1. Introduction

This document explains how to use the ***µTasker Serial Loader*** together with the ***µTasker "Bare-Minimum" Bootloader*** in order to enable the serial loader to be able to update itself with new versions of the serial loader. It represents a very efficient technique that adds only about 2k Bytes to the complete code size.

2. Building the µTasker “Bare-Minimum” Bootloader

The full details about building the µTasker “Bare-Minumum” Bootloader can be found in the document:

http://www.utasker.com/docs/uTasker/uTasker_BM_Loader.pdf

In this role the µTasker Serial Loader can be considered to be the “Application” for the “BM” loader and the intermediate storage location for the new “Application is the location of the µTasker Serial Loader's Application. This means that the µTasker “Bare-Minumum” Loader is configured as follows, if we assume that the µTasker “BM” Loader itself occupies 4k of memory, the µTasker Serial Loader follows it and has a maximum size of 28k, so the µTasker Serial Loader's application is located at 32k.

Configuration defines:

```
#define UTASKER_APP_START    (4 * 1024)      // 0x1000
#define UPLOAD_FILE_LOCATION (32 * 1024)     // 0x8000
#define UTASK_APP_LENGTH     (UPLOAD_FILE_LOCATION - UTASKER_APP_START) // 28k (0x7000)

static const unsigned char ucKey[] = {0xf4, 0x08, 0xb2, 0x78, 0x01, 0x7a};

#define VALID_VERSION_MAGIC_NUMBER  0x1240   // magic number for serial loader updater
```

3. Building the µTasker Serial Loader

The full details about building the µTasker Serial Loader can be found in the document:

http://www.utasker.com/docs/uTasker/uTaskerSerialLoader.pdf

In order to operate with the µTasker "BM" Loader the µTasker Serial Loader needs to be linked to the address that the µTasker "BM" Loader is configured to find 'its' application (`0x1000` in our example).

Following building the µTasker "BM" Loader and the µTasker Serial Loader the two are combined together into a single file that can be loaded to the board in question. The following command shows how the serial loader binary can be combined:

```
uTaskerCombine "../../uTaskerBoot/GNU_Kinetis/uTaskerBoot.bin"
uTaskerSerialBoot_BM.bin 0x1000 uTaskerSerialBM.bin uTaskerSerialBM.hex
```

to generate a single one called `uTaskerSerialBM.bin/hex`

In order to prepare a version of the µTasker Serial Loader that can be loaded by the installed firmware a second step is required to add an authentication header that matches with the µTasker "BM" Loader settings (*see example in previous chapter*). The following command shows how the authentication header is added:

```
uTaskerConvert.exe uTaskerSerialBoot_BM.bin uTaskerSerialLoad.bin -0x1240
-f408b278017a
```

If the new µTasker Serial Loader is to be loaded in the form of an SREC or iHex a subsequent step is required to prepare it. This involves modifying the format to load to the intermediate address rather than its final execution address. The following command shows how this is done for the intermediate address `0x8000`, as used by our example:

```
uTaskerCombine.exe uTaskerSerialLoad.bin dummy.bin 0 dummy_out.bin
uTaskerSerialLoad.srec 0x8000
```

The file `dummy.bin` is an emply file that needs to be present in the directory to allow this utility to perform the desired operation. The output file `dummy_out.bin` is genereated but not further used. The file `uTaskerSerialLoad.srec` can be used to load the new µTasker Serial Loader version.

4. Using the µTasker Serial Loader to Load an Application

The full details about the use of the µTasker Serial Loader to load an application can be found in the document:

http://www.utasker.com/docs/uTasker/uTaskerSerialLoader.pdf

5. Using the µTasker Serial Loader to Load a new Serial Loader

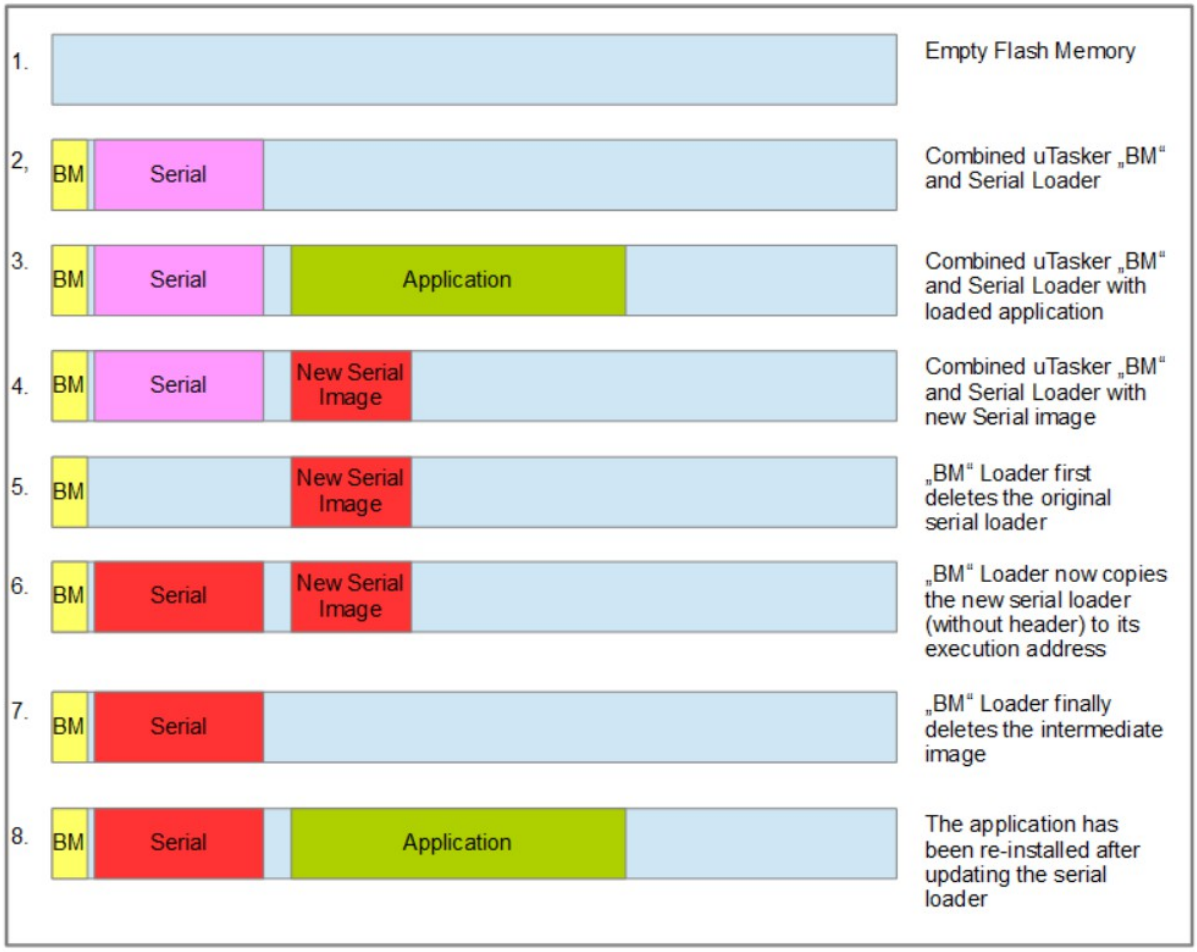
Loading the new version of the µTasker Serial Loader is performed by exactly the same procedure as loading an application.

The difference in operation is however that the loading saves the new code with its authentication header to the location of a normal application. A reset is performed after this loading has completed and the µTasker “BM” Loader now detects that there is a new µTasker Serial Loader version located at the place of its application. It thus performs a transfer of this new code to the area of the µTasker Serial Loader (`0x1000..0x7fff` in the case of the example) [*that is the update of µTasker Serial Loader*] and it deletes the image at the location `0x8000`.

Following the update the new µTasker Serial Loader is started.

Due to the fact that new µTasker Serial Loader image is first loaded to the application area the original application is deleted in the process. The application can subsequently be loaded again using the newly installed µTasker Serial Loader.

The following diagram illustrates the content of memory during the various phases of using the µTasker Serial Loader and this updating process:



6. Conclusion

This document has described the method of using the µTasker “Bare-Minimum” Loader and µTasker Serial Loader together to allow µTasker Serial Loader updating to be possible. The solution is very efficient in code size since it usually requires less that 2k of additional code space.