Embedding it better...

µTasker Document

Test-driving the µTasker i.MX RT Loader Concept

Table of Contents

1. Introduction

This document is designed to allow anyone with a supported i.MX RT reference board to easily and efficiently load and evaluate the uTasker loader on that board. It shows the completeness of the solution which is immediately available for product usage without the usual weeks or months of development needed when based on disparate examples.

Firmware image loading is discussed, followed by how to simply test various functions of the concept such as

- Update the application via OTA application
- Update the serial loader via OTA application
- Update the application via a serial loading method (eg. UART or USB)
- Update the serial loader via serial “Fall-back” loader method (eg. UART or USB)

Furthermore features of the loader concept can be experimented with, such as:
- commanding resets and watching reset cause counters (with watchdog and software reset counters)
- commanding the serial loader to be started from the application
- commanding the serial “Fall-back” loader to be started from the application
- commanding loaders to start the application
- recovery via serial “Fall-back” loader when a faulty application or a faulty serial loader is installed

The µTasker boot loader concept is discussed and demonstrated in the video:
https://youtu.be/2XfgZq19XDw
and its flow diagram is show at https://www.utasker.com/docs/iMX/Loader.pdf

For details serial loader methods and configuration see
https://www.utasker.com/docs/uTasker/uTaskerSerialLoader.pdf

2. Loading the Reference Solution

There is a reference binary available for each supported reference board on the µTasker web site – go to the i.MX RT landing page and then click on the link to the board in question: https://www.utasker.com/iMX.html

*- Note that these reference versions **don't program eFUSEs** and so **can't change or harm the boards in anyway**. Not using eFUSEs (which the production versions programs automatically according to their requirements) means that there is not complete security activated but the general operation still accurates emulates the overall operation.*

- The references may restrict the size of application that can run – such restrictions are removed or are configurable to different dimensions in production versions.

Loading to NXP reference boards can be simply accomplished by setting the ROM LOADER mode of the board and then dragging and dropping the binary image on to the external hard drive that appears. (After loading the normal XiP mode should be reconfigured – DIP switches to contol this are detailed in the guides supplied with the reference boards).

Loading to Teeny 4.x boards can be performed using the .HEX version of the image using its teensy.exe application via USB HID.

Other boards that do not have these loading methods will probably support the i.MX RT's ISP loader (usually via USB device) and so the NXP MCUBootUtility can be used as described in chapter 4 of the document: https://www.utasker.com/docs/iMX/uTaskerV1.4_iMX.pdf

The binary, or HEX, image is a complete image containing the following code:
- µTasker “Bare-Minimum” loader
- µTasker Serial “Fall-back” Loader
- µTasker Serial Loader
- µTasker Reference application demonstrating various project and peripheral functionality, including OTA (Over-The-Air) updating application

Once loaded it is possible to start testing various capabilities of the concept as described in the following sections. This basic image can be used to load new loaders and applications and so usually needs only to ever be loaded a single time.

3. Test-Drive of Loader Features

Due to the fact that the firmware image contains a valid application the board will initially start this application each time it boots. The included application allows communication with the board via UART, USB and Ethernet Telnet (when available) in form of a command line interface. This interface has a number of menus that allowsvarious operations to be performed,including allowing us to control re-entry into the loaders themselves.

When the application starts it will output initial data which looks something like the following: *This was recorded from an i.MX RT 1010 EVK on its debugger's VCOM USB on LPUART (115kBaud), with the processor's HS USB connected connected to the PC so that it enumerates as device too. It is recommended to work with a LPUART connection where possible because its output is immediately available after each reset.*

```
Hello, world... MIMXRT1010 [Power-on]
Static memory = 0x00001000
OS Heap use = 0x02b9 from 0x6ff4
Initial stack margin 0x00007e88
FlexRAM:
2 Code banks [0x00000000..0x0000ffff]
2 Data banks [0x20000000..0x2000ffff]
SPI Flash: adesto 16MByte AT25SF128A
uFileSystem integrity
Start: 0x600a0000
Granularity: 0x0001c000
End: 0x607d3fff
OK
Enumerated 0 (1)
```

When the enter key is hit the command line menu appears:

```
Serial number: 00
Software version V1.4.012
Device identification: uTasker Number 1

     Main menu
==================
1               Configure LAN interface
2               Configure serial interface
3               Go to I/O menu
4               Go to administration menu
5               Go to overview/statistics menu
6               Go to USB menu
7               Go to I2C menu
8               Go to utFAT disk interface
9               FTP/TELNET commands
a               CAN commands
help            Display menu specific help
quit            Leave command mode
```

Generally the same menu appears for all boards, whereby in the case of the i.MX RT 1010 EVK, with neither Ethernet nor CAN, some of the sub-menus are empty.

We enter the administration menu by entering 4 followed by the Enter key and then see:

```
   Admin. menu
====================
up                go to main menu
show_config       Show configuration
save              Save configuration to FLASH
reject            Reset non-saved changes
restore           Restore factory settings
show_time         Display date/time
set_time          Set time hh:mm:ss
set_date          Set Date dd:mm:yyyy
show_alarm        Display alarm d/t
set_alarm         Set alarm (date)(+)[time]
del_alarm         Delete alarm
bmode             Display boot mode
eFUSE             Display eFUSEs
show_lp           Show low power mode and options
set_lp            [option] Set low power mode
wdog              Watchdog
boot              Reset to boot loader
fboot             Reset to fall-back loader
reset             Reset device
last_rst          Reset cause
help              Display menu specific help
quit              Leave command mode

#
```

Command “**reset**” to perform a reset of the board, in which case the application immediately starts again.

It will be noticed that the reset cause is reported as a software reset and the reset count value is now at 1. If the command is repeated a number of times the reset counter increments each time:

```
Hello, world... MIMXRT1010 [Software 1]
Static memory = 0x00001000
OS Heap use = 0x02b9 from 0x6ff4
Initial stack margin 0x00007e88
FlexRAM:
2 Code banks [0x00000000..0x0000ffff]
2 Data banks [0x20000000..0x2000ffff]
SPI Flash: adesto 16MByte AT25SF128A
uFileSystem integrity
Start: 0x600a0000
Granularity: 0x0001c000
End: 0x607d3fff
OK
Enumerated 0 (1)
```

Now command a watchdog reset with the command “**wdog**”:

```
#wdog
Forever loop..

Hello, world... MIMXRT1010 [WDOG3 1 (1)]
Static memory = 0x00001000
OS Heap use = 0x02b9 from 0x6ff4
Initial stack margin 0x00007e88
FlexRAM:
2 Code banks [0x00000000..0x0000ffff]
2 Data banks [0x20000000..0x2000ffff]
SPI Flash: adesto 16MByte AT25SF128A
uFileSystem integrity
Start: 0x600a0000
Granularity: 0x0001c000
End: 0x607d3fff
OK
Enumerated 0 (1)
```

The code sets itself into a forever loop and stops re-triggering the watchdog 3 so that it fires after about 2s and the board restarts. This time the last reset is reported as a watchdog reset and the watchdog reset counter (in brackets) increments.

Now command the bootloader to start using the command "**boot**" in the administration menu, which causes the board to reset but this time the serial loader to be executed.

```
#boot

 uTasker Serial Loader V2.0
============================
[0x60020100/0x6009ffff]
bc = blank check
dc = delete code
ld = start load
bt = start boot loader
go = start application
> Enumerated
```

This menu allows deleting the the present application and loading a new one via SREC or iHEX files but the original application can also be restarted by commanding "**go**". Note that the USB-MSD loader will appear (when the USB device is connected) and the original application can also be started by ejecting the external "**UPLOAD_DISK**" disk that appears.

If no activity on the LPUART interface is detected the serial loader will automatically re-start the original application after a period of 60s.

Notice that the serial "Fall-back" loader can also be commanded from this menu (see next section).

The serial “Fall-back” loader can also be commanded from the administration menu by using the command “**fboot**”:

```
 uTasker Fall-back Loader V2.0
==============================
[0x60010100/0x6001ffff]
bc = blank check
dc = delete code
ld = start load
go = start serial loader
Enumerated
```

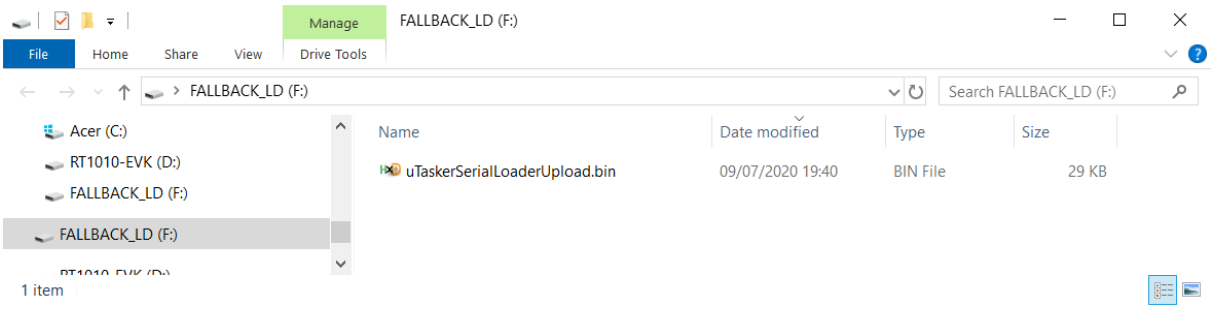
The serial “Fall-back” loader can have the same features as the serial loader but may also have a reduced set. Its function is to be able to load a new serial loader and is generally built with LPUART and USB-MSD loading support since these tend to be the most useful. The serial “Fall-Back loader itself can't be updated.

In a similar fashion to the serial loader starting the application, a “**go**” will command the serial loader to be started. When the serial “Fall-back” loader is supports USB-MSD the same occurs if its “**FALLBACK_LD**” disk is ejected. The serial “Fall-back” loader has a shorter timeout of 10s when there is no activity on the serial interface. Therefore the serial “Fall-back“ loader will start the serial loader, which will then timeout to stat the application again.

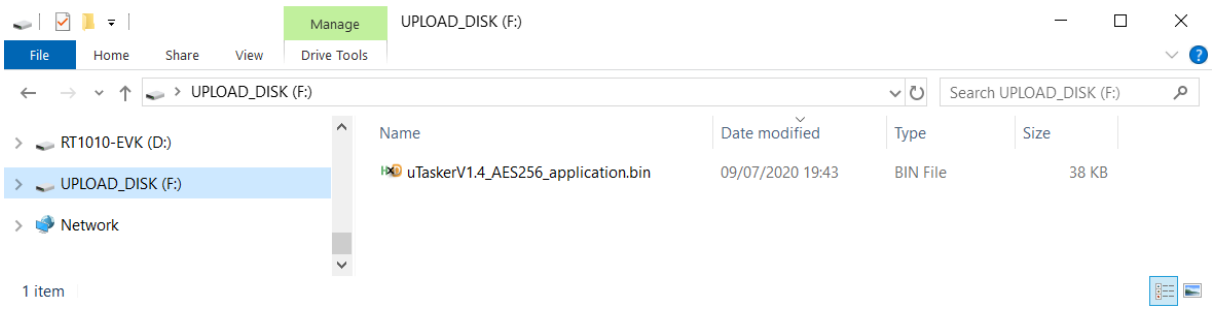
Of course, if either the serial loader or the application has been deleted the loaders will not be able to start the next firmware and will remain in their modes until new firmware is loaded.

When the USB-MSD loader mode is enabled in each of the loaders (serial “Fall-back” and serial) and the HS USB device cables are connected they appear as external disk drives to the host. As shown by the following images, the great advantage of the USB-MSD operation is that the file's name, date, time and size are displayed, which is useful for exact version management with no additional effort.

The first image shows the serial “Fall-back” loader's disk appearing as “**FALLBACK_LD**” showing the serial loader that is presently installed:



The second image shows the serial loader's disk appearing as “**UPLOAD_DISK**” showing the application that is presently installed.


Manage
UPLOAD_DISK (F:)
File
Home
Share
View
Drive Tools
UPLOAD_DISK (F:)
Search UPLOAD_DISK (F:)
RT1010-EVK (D:)
UPLOAD_DISK (F:)
Network
Name
Date modified
Type
Size
uTaskerV1.4_AES256_application.bin
09/07/2020 19:43
BIN File
38 KB
1 item

4. Conclusion

This document has guided and explained test-driving the µTasker loader using reference images that are available for a number of reference boards. The concept remains compatible across the various parts and allows flexible, secure loading operation based on various serial and OTA techniques.

The loader concept can be used with applications from any source as long as they are linked accordingly and prepared to be in the correct format and respect a few simple rules, which allows an important reduction in work (in comparison to custom developments based on disparate examples) when generally used for all product development.

Although the focus in this document is on the µTasker loader the standard µTasker application is also present and works compatibly with the concept, whereby it is to be noted that the loader is only a small part of the overall µTasker project and also the application – with OS, USB and TCP/IP stack and file systems, and many optimised drivers and features – allows great development time reductions and cost savings when used as foundation for overall product developments.

Modifications:
V0.01 09.07.2020: Initial version in development

Appendix A

a) Space for first Appendix