Embedding it better...

µTasker Document

µTasker – i.MX RT Production Programmer

Table of Contents

1. Introduction

Once an i.MX RT based product has been developed and moves from the development environment to a production environment the requirements for programming often change dramatically. Instead of a highly qualified developer using often complicated and manual programming steps the need changes to requiring a more fool-proof method that can be easily and reliably performed by almost anyone without needing deep technical knowledge of the process.

General purpose development tools may be flexibly but are unlikely to be optimally efficient in the factory and dedicated programming tools may be unnecessarily expensive.

This is where the **µTasker i.MX RT Production Programmer** comes in – it is designed for production programming with a single action and with clear indication of success or failure. And it allows reuse of existing HW as a programming tool meaning that it doesn't necessarily require the purchase of HW but instead allows recycling existing HW (development board or product prototypes) to easily ramp up programming capacity if and when needed.

1.1 Principles

All i.MX RT parts include a boot loader which is started automatically when there is no valid program in memory and this allows a programmer, connected via HS USB as a host, to communicate with the bot loader based on a standard protocol. The host (programmer) can load code to the internal RAM of the processor and command its execution. Finally the programmer can communicate with this injected program to perform additional, more complicated operations such as loading code the external memory, programming eFUSE values and even performing self test, calibration processes.

1.2 What does the µTasker project supply?

A framework for performing the tasks needed by the i.MX RT programmer is included in the µTasker project so that it can be used on various HW platforms. A complete general purpose configuration can be simply built which allows the typically needed programming operations.

This project can further be customised as needed for specific situations to ensure that the optimal overall solution can be achieved.

1.3 Test Vehicle

This document uses a standard i.MX RT 1020 EVK as reference and shows how it is used to program other i.MX RT EVKs. This board has limited HMI capabilities and so programming is configured via its UART and firmware to be loaded is placed on an SD card. Once configured (non-volatile setting) programming operations are started by pushing a button and the progress and success/failure shown on an LED.

Boards with TFT display and touch screen could of course use these to improve the user experience.

2. Using the i.MX RT Production Programmer

To be added!

3. Building the i.MX RT Production Programmer

For technical details of the operation see appendix A.

To be added!

4. Conclusion

This document and development are in progress and a release is expected in early 2021.

Modifications:

V0.01 30.12.2020 First preliminary version

Appendix A - Expert View under the Bonnet

This section contains technical details of the internal workings so that developers can quickly understand the operation and adapt behaviour for custom purposes.

Since all i.MX RT parts have a USB controller it is assumed that USB programming will be used – this is also faster than UART programming.

a) Phase 1

When an i.MX RT part start in ISP (In System Programming) mode its boot loader will enumerate as USB device with VID/PID of 0x1fc9 (NXP-Semiconductors)/PID where PID is:

Part Type	PID
i.MX RT 5xx	0x0020
i.MX RT 6xx	0x0020
i.MX RT 1011	0x0145
i.MX RT 1015	0x0130
i.MX RT 1021	0x0130
i.MX RT 105x	0x0130
i.MX RT 106x	0x0135
i.MX RT 117x	0x013d

which means that the USB host can identify the i.MX RT family on connection.

The device enumerates as an HID (Human Interface Device) with a single interrupt endpoint of 64 bytes to be polled at 1ms rate.

The host requests information by sending class requests, which are answered by the device on its interrupt endpoint.

The host sends data blocks (for programming firmware to RAM) via class requests too.

The protocol used is described in the i.MX RT user's manuals under “Serial Download Protocol (SDP)” which lists the 16-byte SDP commands that can be sent using the HID report 1, where the command types relevant for USB loading are:

- 0x0101 READ_REGISTER
- 0x0202 WRITE_REGISTER
- 0x0404 WRITE_FILE
- 0x0505 ERROR_STATUS
- 0x0A0A DCD_WRITE
- 0x0B0B JUMP_ADDRESS

The final command, after writing a file of firmware to be executed using a number of WRITE_FILE commands, is always the JUMP_ADDRESS command, which starts the loaded program. When the new program starts the original USB connection will be reset and the new program can then communicate with a similar or different USB class to do the next phase of operation.

b) Phase 2

During phase 1 a µTasker programming utility is programmed which enumerates as a USB-CDC device with the VID/PID 0x15a2/0x0044 which is an official combination allocated by Freescale to the µTasker USB-CDC application when running on Freescale/NXP silicon.

The USB host (programmer) detects this and can then run a dedicated protocol to perform further programming to QSPI memory and optionally eFUSEs. It also allows reading various information from the board to be programmed and can be used to customise additional in-system tests as needed.

c) Implementation of Firmware

There are two firmware projects involved, which can both be build from the uTasker application project:

- USB Host Programmer

This is installed on the i.MX RT Programmer board, which recognises the target enumerating in the ISP mode, programs the USB device programmer firmware to it and then recognises the target enumerating as USB device Programmer.

This program is build as standard application and loaded to the board using the µTasker secure loading concept.

- USB device Programmer

This is copied to the target by the USB Host Programmer during Phase 1 operation and runs on the target during Phase 2. This program is built as standard application and so can also be tested on any board using the µTasker secure loading concept before being loaded via the host programmer.