iMCP BOOTLOADER REFRESH

1 INTRODUCTION

The HT32SX is a System-in-Package device build for the Internet of Things providing a **ready-to-use** connectivity solution. The system provides an ARM Cortex M0+ 32bit, the STM S2-LP low power transceiver and an RF Front-End module combining all the advantages, integration, energy efficiency and convenience of advanced semiconductor packaging technology into a single chip with a 50Ω RF TX/RX interface. As a SigFox™ Monarch enabled device, it can operate in all regions covered by SigFox™ Network without need of reconfiguration, as the device can detect the region of operation and rearrange its setup automatically.

1.1 Application Description

In order to enable features such as online update through a USART connection, or to program the device without the need of an external hardware, we are making available the iMCP Bootloader.

This application note is composed by:

- This document
- Bootloader binary `iMCP-bootloader.bin (version c0)`
- The easy-programmer software (Python 3.8 script)

1.2 Bootloader Features

The following features are enabled in the current version:

- Upload a new user firmware to the iMCP through USART
- Boot to the firmware
- Autoboot:
  - Set the bootloader to skip at the start-up time directly to the firmware uploaded
  - Disable the bootloader skip at the start up time
- Credentials:
  - Recover the credentials pre-recorded (only if the bootloader was pre-loaded in the iMCP)
- Reset to bootloader:
  - By default, in the standard iMCP board, if the user button is pressed at boot time, the bootloader is loaded instead of the user program

Also, the device reverses to the bootloader at (almost) any reset (the reset pin is not included intentionally). So, at a software generated reset, the bootloader will be back and ready to use.

2 HARDWARE SETUP

The pin connections of the HT32SX_DEV_KiT can be found in the "*Getting Started with HT32SX"* document at https://github.com/htmicron/ht32sx. The required hardware to perform the steps described in this document consists of:

- HT Micron iMCP Evaluation Board
- 868MHz – 928MHz standard Antenna
- Micro-USB cable for power supply

If the bootloader is not preloaded, you need to use the ST-Link software to upload the iMCP-bootloader.bin file to the iMCP Evaluation Board.

3 SOFTWARE SETUP

These programs are recommended:

- GIT (for Windows, *git-scm.com* is recommended)
- HT Credential Generator (github.com/htmicron/imcp-credential)
- STM32 ST-LINK (www.st.com/en/development-tools/stsw-link004.html)

4 PYTHON SCRIPT PROGRAMMER

Together with the bootloader binary file, we are making available a Python 3.8 script capable of basic controlling the bootloader and uploading the firmware through USART port.

To run the script, the `pyserial` package is required (in the Linux terminal, usually, type `pip3.8 install pyserial`). The script is licensed with the APACHE 2.0 license, so the user can adapt it at her or his own discretion.

To use:

- The first argument is the serial port.
- The second argument is binary file (in the bin format) file:
- Example: `python3.8 easy_programmer.py COM6 myFirmware.bin`

Autoboot and skip-bootloader when pressing the user button are enabled by default.

To use the script, type in your terminal `python3.8 programmer.py` and the program will list the serial ports available and return its options.

5 USER PROGRAM REQUIRED CHANGES

The bootloader uses the flash storage space from the address 0x08000000 to 0x08000FFF, and the user program must start at the address 0x08001000 in order to work properly. To do so, the following changes are required to be made in the Arm Keil project:

- at file Drivers/CMSIS/Device/ST/STM32L0xx/Include/stm32l052xx.h
  - replace:

```
#define FLASH_BASE            (0x08000000UL) /*!< FLASH base address in the alias region */
```

  - by:

```
#define FLASH_BASE            (0x08001000UL) /*!< FLASH base address in the alias region */
```

- at file MDK-ARM/Sigfox/Sigfox.sct:
  - replace:

```
LR_IROM1 0x08000000 0x00010000  {    ; load region size_region
  ER_IROM1 0x08000000 0x00010000  {  ; load address = execution address
```

  - by:

```
LR_IROM1 0x08001000 0x0000F000  {    ; load region size_region
  ER_IROM1 0x08001000 0x0000F000  {  ; load address = execution address
```

- at file Src/system_stm32l0xx.c
  - replace:

```
#define VECT_TAB_OFFSET  0x00U /*!< Vector Table base offset field.
```

  - by

```
#define VECT_TAB_OFFSET  0x1000U /*0x00U /*!< Vector Table base offset field.
```

- at file MDK-ARM/Sigfox.uvprojx
  - replace:

```
<StartAddress>0x8000000</StartAddress>
```

  - by

```
<StartAddress>0x8001000</StartAddress>
```

Additionally, as the programmer take **as input binary (.bin) files and not hexadecimal files (.hex)**, follow these instructions: http://www.keil.com/support/docs/3213.htm.

We modified the push-button application note to fit these requirements. Also, we modified the functionality in order to show the user how to boot back to the bootloader. The functionality is:

- The firmware boots and send some useful information to the user (such as the ID)
- The user presses the development board button:
- A frame is sent through the SigFox network
- A small delay is executed
- A software reset is done (and thus the program is changed to the bootloader)
- Bootloader boots and start waiting commands from the serial port USART.

This example is hosted at the old-bootloader version folder (https://github.com/htmicron/ht32sx/tree/master/firmware_applications/bootloader/pushButton).

6 SERIAL COMMANDS DEFINITION

In this chapter we will show the *raw* commands sent through the serial to control the bootloader. In typical cases it is not necessary to know it, but if the user wants to implement its own programmer, it might be helpful.

	Command (binary data)	Functionality	Expected return (always 7 chars + \n)
Program a new firmware	To program a new firmware, send first the new firmware command, after the size, and after packages of 64 bytes of data, as follows:		
	0x02, 0x00, 0x00, 0xff	1. Stores a new firmware	STORE__\n
	Byte order: low to high (4 bytes).	2. Size of the new firmware	STOxxxx\n where XXXX is the size received (if it is more than 60k, the bootloader will assume the max size)
	64 bytes of data	3. Firmware data	STOxxxx\n where XXXX is the number of the 64 bytes package (starting from zero)
	Last bytes of data, padded with any data at the end, until completing 64 bytes	4. Firmware data, after the last 64 bytes of data	STOend_\n
Other commands	0x05, 0x00, 0x00, 0xff	Boots the user firmware	No return
	0x84, 0x00, 0x00, 0xfa	Enables the auto-boot feature	BOOTon_\n
	0x2C, 0x00, 0x00, 0xfa	Disables the auto-boot feature	BOOToff\n
	0x01, 0x00, 0x00, 0xff	Return the SiP and the bootloader version (useful to check the connection)	iMCP_C0\n
	0x02, 0x00, 0x00, 0xA7	Restore the credentials (only if the iMCP was pre-loaded with a bootloader)	RESTORE\n
	0x04, 0x00, 0x00, 0xCF	Locks the boot-firmware against changes	LOCKon_\n
	0xF1, 0x00, 0x00, 0xCF	Unlocks the boot-firmware against changes	LOCKoff\n
	0xB1, 0x00, 0x00, 0x14	Enables the skip-to-bootloader at reset when pressing the user button	RESTon_\n
	0x3F, 0x00, 0x00, 0x14	Disables the skip-to-bootloader at reset when pressing the user button	RESToff\n
	0xDD, 0x00, 0x00, 0xFF	Read 64 bytes of data stored in the iMCP memory. First send this command, after send 04 bytes of the first address (byte order: low to high)	READ___\n
	[redacted]	[redacted]	[redacted]

7 CONTACT INFORMATION

Head Office – São Leopoldo, RS

HT Micron Semiconductors
Unisinos Avenue, 1550
São Leopoldo - RS
ZIP 93022-750
Brazil
Tel: +55 51 3081.8650
E-mail (Support) support_iot@htmicron.com.br
E-mail (General Inquiries) htmicron@htmicron.com.br