	AN-SXBootloader		HT32SX
	Application Note	24/01/2020	V B0

IMCP BOOTLOADER



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. This is possible because 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 b0)
- The programmer software (Python 3.8 script)
- Push-button application example adapted to the bootloader requirements


Please note the current version of the bootloader and associated materials are **preliminary**, thus future versions may vary drastically. Also, the current version is **not supported**.

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
- Set the bootloader to skip at the start-up time directly to the firmware uploaded
- Disable the bootloader skip at the start up time

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.

	AN-SXBootloader		HT32SX
	Application Note	24/01/2020	V B0

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 iMPC Evaluation Board
- 868MHz – 928MHz standard Antenna
- Micro-USB cable for power supply


Use the ST-Link software to upload the iMCP-bootloader.bin file to the iMCP Evaluation Board.

3 SOFTWARE SETUP

This section describes all necessary steps needed to use the SigFox™ with the **HT32SX**.

Also, 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)

	AN-SXBootloader		HT32SX
	Application Note	24/01/2020	V B0

4 PYTHON SCRIPT PROGRAMMER

Together with the bootloader binary file, we are making available a Python 3.8 script capable of 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 the function:
 - To store a new program, type STORE
 - The third argument is the path to the bin file to be programmed.
 - Example: `python3.8 programmer.py3 COM6 STORE thisfile.bin`
 - To boot, type BOOT
 - Example: `python3.8 programmer.py3 COM6 BOOT`
 - To enable autoboot at start-up time, type AUTOBOOTon
 - Example: `python3.8 programmer.py3 COM6 AUTOBOOTon`
 - To disable autoboot at start-up time, type AUTOBOOToff
 - Example: `python3.8 programmer.py3 COM6 AUTOBOOToff`

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:

```
$ python3.8 programmer.py3

Welcome to iMCP Bootloader Programmer Version 0.5

#####


Here are the serial ports available in your computer,
please select the one connected to the iMCP board:

['COM6']

#####

How to use this program:
The first argument is the serial port.
The second argument is the function:
    To store a new program, type STORE
        The third argument is the path to the bin file to be programmed.
        Example: python3.8 programmer.py3 COM6 STORE thisfile.bin
    To boot, type BOOT
        Example: python3.8 programmer.py3 COM6 BOOT
    To enable autoboot at startup time, type AUTOBOOTon
        Example: python3.8 programmer.py3 COM6 AUTOBOOTon
    To disable autoboot at startup time, type AUTOBOOToff
        Example: python3.8 programmer.py3 COM6 AUTOBOOToff

Note: mostly any reset but the external reset pin will put the system in the bootloader mode, even with the autoboot feature on. You may use it to force-start the bootloader through your software, using a software-generate reset (in Arm Keil compiler with the command "NVIC_SystemReset();").
```

	AN-SXBootloader		HT32SX
	Application Note	24/01/2020	V B0

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.

	AN-SXBootloader		HT32SX
	Application Note	24/01/2020	V B0

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 may be helpful.

	Command (binary data)	Functionality	Expected return (always 10 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	Stores a new firmware	storeSTART\n
	Bit order: lower to high (4 bytes).	Size of the new firmware	store,XXXX\n where XXXX is the size received (if it is more than 60k, the bootloader will assume the max size)
	64 bytes of data	Firmware data	store,XXXX\n the 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	Firmware data, last bytes of data	store,END \n
Other commands	0x05, 0x00, 0x00, 0xff	Boots the user firmware	No return
	0x84, 0x00, 0x00, 0xfa	Enables the auto-boot feature	autoboot1 \n
	0x2C, 0x00, 0x00, 0xfa	Disables the auto-boot feature	autoboot0 \n
	0x01, 0x00, 0x00, 0xff	Return the SiP and the bootloader version (useful to check the connection)	iMCP2.1 B0\n

	AN-SXBootloader		HT32SX
	Application Note	24/01/2020	V B0

7 REFERENCES

For additional information about SigFox™ libraries designed by ST Microelectronics, please refer to the UM2173 document (note: the function names still the same, but the code was adapted to use in the iMCP design, so it is different from the one distributed by ST).

	AN-SXBootloader		HT32SX
	Application Note	24/01/2020	V B0

8 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 Enquiries) htmicron@htmicron.com.br

Copyright 2020 HT Micron Semicondutores S.A.

This document is a property of HT Micron and can not be reproduced without its consent.

HT Micron does not assume any responsibility for the use what is described.

This document is subject to change without notice.

The binary file of the iMCP-Bootloader provided with this Application Note can be distributed free of charge. HT Micron does not assume any responsibility for the use of this piece of software. This program is subject to change without notice.

The Script to Program the iMCP-Bootloader (written in Python) provided with this Application Note is licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and limitations under the License.
