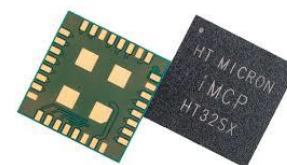
	APPLICATION NOTE 06, iMCP – HT32SX		iMCP, app note
	Application Note	11/05/2020	V 1.0

AT Commands for HT32SX



1 INTRODUCTION

The iMCP is a Multicomponent Integrated Circuit (MCO) built for the Internet of Things, it provides a ready-to-use connectivity solution for the SigFox™ network.

The system combines an ARM Cortex M0+ 32bit, a S2-LP high performance, ultra-low power RF transceiver and a RF Power Amplifier with all of the advantages, integration and convenience found in a SiP (System In Package), which is one of the most advanced semiconductor packaging technology.

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 About this document

This application note will explore a way to use the HT32SX as a slave through AT commands that will be controlled by a Hierarchical/Concurrent Finite State Machine.


1.2 Application description

After configuring every peripheral, the application starts running the HFSM that is going to wait for a command until the DMA detects the end of a string. The trigger that will makes the state machine change its state, is the detection of the ';' character. Then, a couple of tests are made in order to verify if that string is an available command. At the end of execution, the state machine calls the right API to process that command.


These are the current available commands which can be used in this application:

Table 1. Available Commands.

Command	Arguments	Description
AT+SEND	<p>DOWNLINK_FLAG: Downlink flag is set to 1 in order to wait for a downlink.</p> <p>PAYLOAD: Payload that will be sent to the SigFox Network. It must be less or equal than 12 bytes.</p>	<p>Send a payload to the SigFox Network. Before calling this function, it is necessary to call the AT+CFGRCZ first.</p> <p>Example 1: <code>AT+SEND=1:AAAAAAA;</code> (Wait for a downlink).</p> <p>Example 2: <code>A+SEND=0:AAAAAAA;</code> (Do not wait for a downlink).</p>
AT+CFGRCZ	<p>RCZ: RC value corresponding to the SigFox region where the device is going to operate in.</p>	<p>Open SigFox library according to the region. Returns 0 if ok.</p> <p>Example 1: <code>AT+CFG=2;</code></p>

	APPLICATION NOTE 06, iMCP – HT32SX		iMCP, app note
	Application Note	11/05/2020	V 1.0

AT+MONARCH	RCZ: RC beacon expected (in order to scan every region available, the RCZ value should be 127). Minutes: Timeout. (It is recommended to use at least 6 minutes).	Scan a Monarch Beacon and returns by a serial terminal, the region founded. The library must be closed before use this command (command AT+CLOSE). Example 1: AT+MONARCH=2:6; (scan only RC2 beacons). Example 2: AT+MONARCH=127:6;
AT+STPMONARCH	None	Stops an already running Monarch Scan. Returns 0 if ok. Example: AT+STPMONARCH;
AT+CLOSE	None	This command closes the SigFox library (Free the allocated memory of SIGFOX_API_open and close RF). Returns 0 if success. Example: AT+CLOSE;
AT+RESET	None	Resets the MCU. Example: AT+RESET;
AT+STOP	None	Makes the MCU enters in stop mode. Example: AT+STOP;
AT+DEEPSLEEP	None	Makes the MCU enters in deep sleep mode (calls stop mode and turn off most of peripherals, keeping on only the USART). Example: AT+DEEPSLEEP;
AT+WKP	None	Wake up the MCU after a stop or deep sleep mode. It will only work in these cases. Example: AT+WKP;

	APPLICATION NOTE 06, iMCP – HT32SX		iMCP, app note
	Application Note	11/05/2020	V 1.0

- **Payload data type:**

The default payload data consists of hexadecimal values. An example of this is the following command, which will send only hexadecimal values to the Sigfox network (24 hex numbers that means 12 bytes):.

AT+SEND=0:0123456789ABCDEF01234567;

If there is a need to send ASCII data, the user should set a new define symbol before compile the code again:

1. Click “Project -> Properties”:

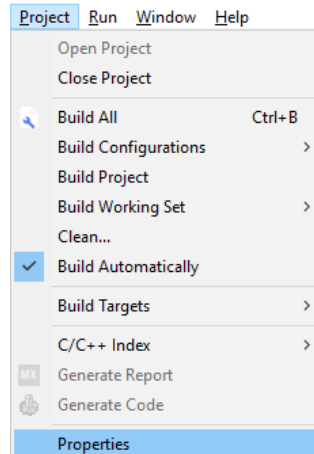


Image 1 – Project tab.

2. Click “C/C++ Build -> Settings”, select “Tool Settings” tab and click on the icon to add a new define symbol:

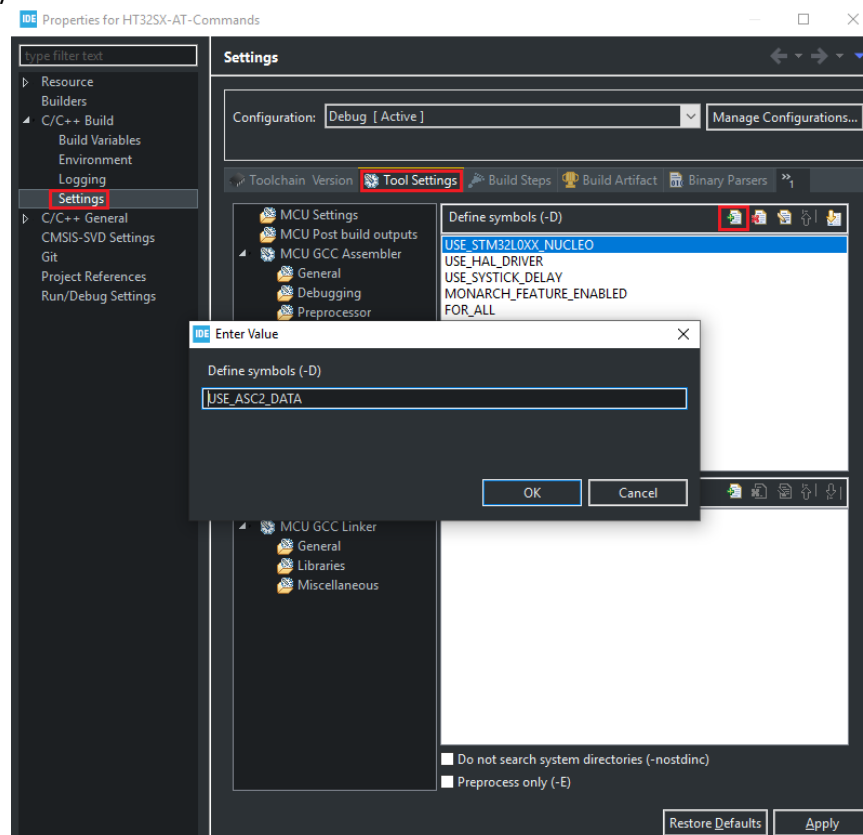



Image 2 – Project properties.

3. Write *USE_ASC2_DATA* and click on *Apply* to save all changes.

	APPLICATION NOTE 06, iMCP – HT32SX		iMCP, app note
	Application Note	11/05/2020	V 1.0

2 DEVELOPMENT SETUP

This section describes all necessary steps needed to use this application with the **iMCP – HT32SX**.

Also, these programs are recommended:

- GIT (for Windows, *git-scm.com* is recommended).
- STM32 ST-LINK (www.st.com/en/development-tools/stsw-link004.html).
- STM32CubeIDE (<https://www.st.com/en/development-tools/stm32cubeide.html>)
- RS232 terminal (Termite is recommended). (https://www.compuphase.com/software_termite.htm).

3 EXECUTION SETUP

- There are two ways to run this application correctly:
 1. Using the HT32SX as a slave of an extern MCU, sending all commands to its USART1, with 115200 of bound rate,
 2. Using a serial terminal. (It is recommended to use **Termite**), setting up the bound rate to 115200. In this case, your serial terminal must be configured like this:

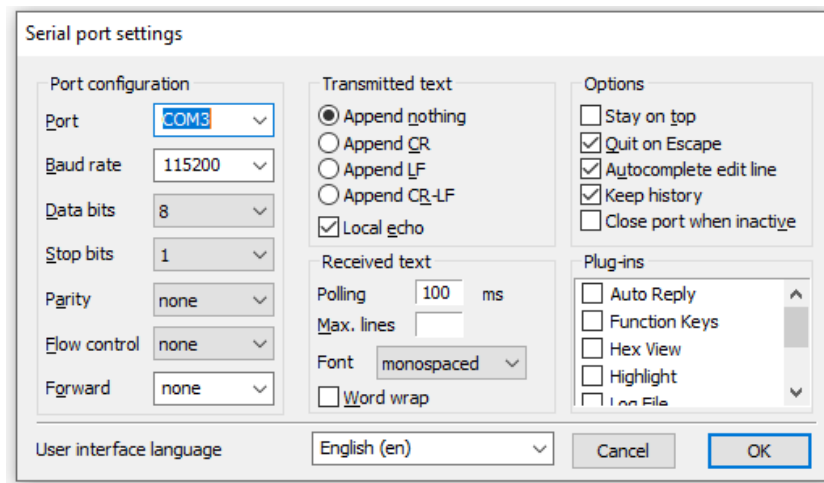


Image 3 – Termite configuration.

4 APPLICATION CODE

4.1 Software implementation

The project was developed on **STM32CubeIDE**, following the STM32CubeMX patterns. All the SigFox Libraries are also included.


For organizational purpose, the project was divided by MCU peripherals. The RTC implementation, for example, will be at the “rtc.c” file. The same was made for GPIOs, timers, SPI, and USART.

The whole logic was modulated with a Hierarchical/Concurrent Finite State Machine. The state machine will manage all DMA interruptions while the end of a command is not detected.

To handle with different string sizes, the DMA works with multiples IDLE interruptions, which will warn it that a new character was received.

4.2 STM32CubeMX or STM32CubeID generate code

If it is necessary to generate a new code using STM32CubeMX or even the STM32CubeIDE, the user must change the DMA/USART interruption handler again. The images below show how it should be made:

	APPLICATION NOTE 06, iMCP – HT32SX		iMCP, app note
	Application Note	11/05/2020	V 1.0

```

void DMA1_Channel4_5_6_7_IRQHandler(void)
{
    /* USER CODE BEGIN DMA1_Channel4_5_6_7_IRQn 0 */

    /* USER CODE END DMA1_Channel4_5_6_7_IRQn 0 */
    // HAL_DMA_IRQHandler(&hdma_usart1_tx);
    // HAL_DMA_IRQHandler(&hdma_usart1_rx);
    /* USER CODE BEGIN DMA1_Channel4_5_6_7_IRQn 1 */
        DMA_IrqHandler(&hdma_usart1_rx);
    /* USER CODE END DMA1_Channel4_5_6_7_IRQn 1 */
}

```

Image 4 – DMA1 IRQ Handler.

- Commenting lines `HAL_DMA_IRQHandler(&hdma_usart1_tx)` and `HAL_DMA_IRQHandler(&hdma_usart1_rx)`.

```

void USART1_IRQHandler(void)
{
    /* USER CODE BEGIN USART1_IRQn 0 */


    /* USER CODE END USART1_IRQn 0 */
    //HAL_UART_IRQHandler(&huart1);
    /* USER CODE BEGIN USART1_IRQn 1 */
        if(HT_McuApi_getDeepSleepModeFlag()) {
            HT_McuApi_configPeripherals();
            AT_updateFsm(0, 0);
            HT_McuApi_setDeepSleepModeFlag(0);
        }
        USART_IrqHandler(&huart1, &hdma_usart1_rx);

    /* USER CODE END USART1_IRQn 1 */
}

```

Image 5 – USART1 IRQ Handler.

- Commenting line `HAL_UART_IRQHandler(&huart1)`.

	APPLICATION NOTE 06, iMCP – HT32SX		iMCP, app note
	Application Note	11/05/2020	V 1.0

4.3 Code description

Table 2. HT MCU API functions.

Name	Arguments	Description
HT_McuApi_enterSlopMode	None	Starts MCU stop mode.
HT_McuApi_enterGpioLowPower	None	Set up all GPIOs to analog in order to reduce the current consumption.
HT_McuApi_enterDeepSleepMode	None	Enters in deep sleep mode (calls stop mode and turn off most of peripherals, keeping on only the USART).
HT_McuApi_configPeripherals	None	Reconfigure all peripherals again. It is called after a wake-up event.
HT_McuApi_softwareReset	None	Resets the MCU.
HT_McuApi_getDeepSleepModeFlag	None	Get the <i>deepSleepMode</i> flag in order to controlling the state machine.
HT_McuApi_setDeepSleepModeFlag	None	Sets the <i>deepSleepMode</i> flag in order to controlling the state machine.
HT_McuApi_enableUsartWkp	None	Configures USART to wake up from Stop or Deep Sleep Mode.

In order to wake-up from an USART interruption, this application configured the HSI clock to work with USART1. The image below shows how it was made:

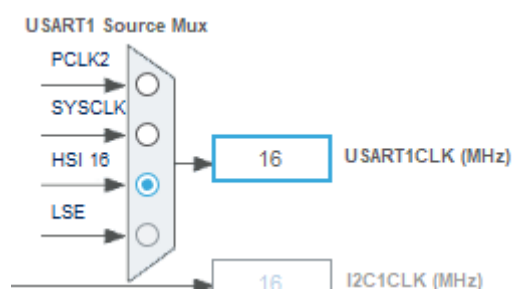



Image 6 – USART1 clock configuration.

Using STM32CubeIDE, open the .ioc file, go to *Clock Configuration* and search for the *USART1 Source Mux*.

Table 3. HT Monarch API functions.


Name	Arguments	Description
HT_MonarchApi_monarchScan	rc_capabilities_bit_mask: Bit mas of the RCx which must be executed. Timer: Monarch Scan timeout (in minutes).	Executes a scan of the air to detect a Sigfox Beacon. It will return 0, if success and the RC enum value corresponding to

	APPLICATION NOTE 06, iMCP – HT32SX		iMCP, app note
	Application Note	11/05/2020	V 1.0

		the beacon found and its RSSI level. The scan is executed during the specific timer/unit time.
HT_MonarchApi_callback	rc_capabilities_bit_mask: Bit mas of the RCx which must be executed. Rssi: RSSI signal.	Monarch callback called after finding a Monarch Beacon.
HT_MonarchApi_getRcBitMask	RCZ: Integer corresponding to the region wanted.	Get the corresponding rcz bit mask.

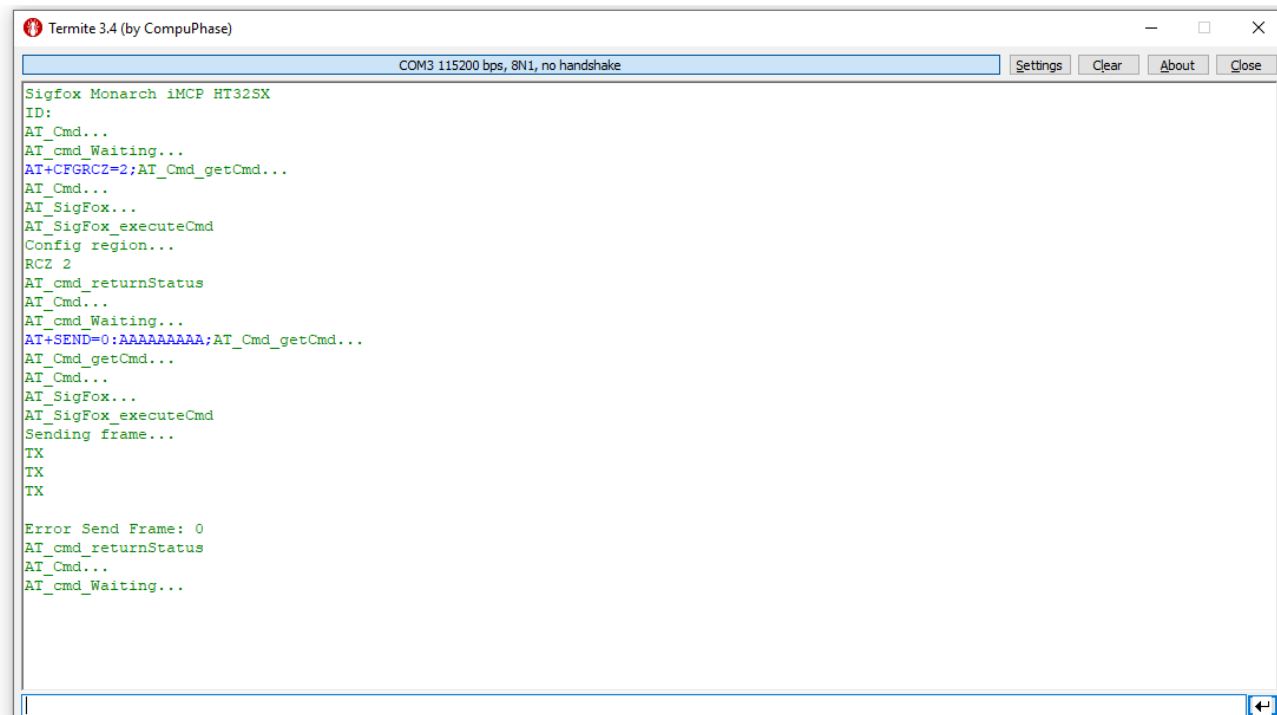
Table 4. HT SigFox API functions.

Name	Arguments	Description
HT_SigfoxApi_sendFrame	Customer_data: Buffer that is going to be sent. Customer_response: Buffer that will receive the downlink. Initiate_downlink_flag: Downlink flag. If it is 1, the device will wait for a downlink after send a payload. Len: Payload data length.	Send a frame to the SigFox Network.
HT_SigfoxApi_configRegion	RCZ: RC of the desired region	Configures the device with the region specified by the user.
HT_MonarchApi_closeSigfoxLib	None	This function closes the library (Free the allocated memory of SIGFOX_API_open and close RF).

	APPLICATION NOTE 06, iMCP – HT32SX		iMCP, app note
	Application Note	11/05/2020	V 1.0

5 RESULTS

Some results are expected running this application. It is recommended to use a serial terminal to check if the commands are being executed (Termite was used in these examples).



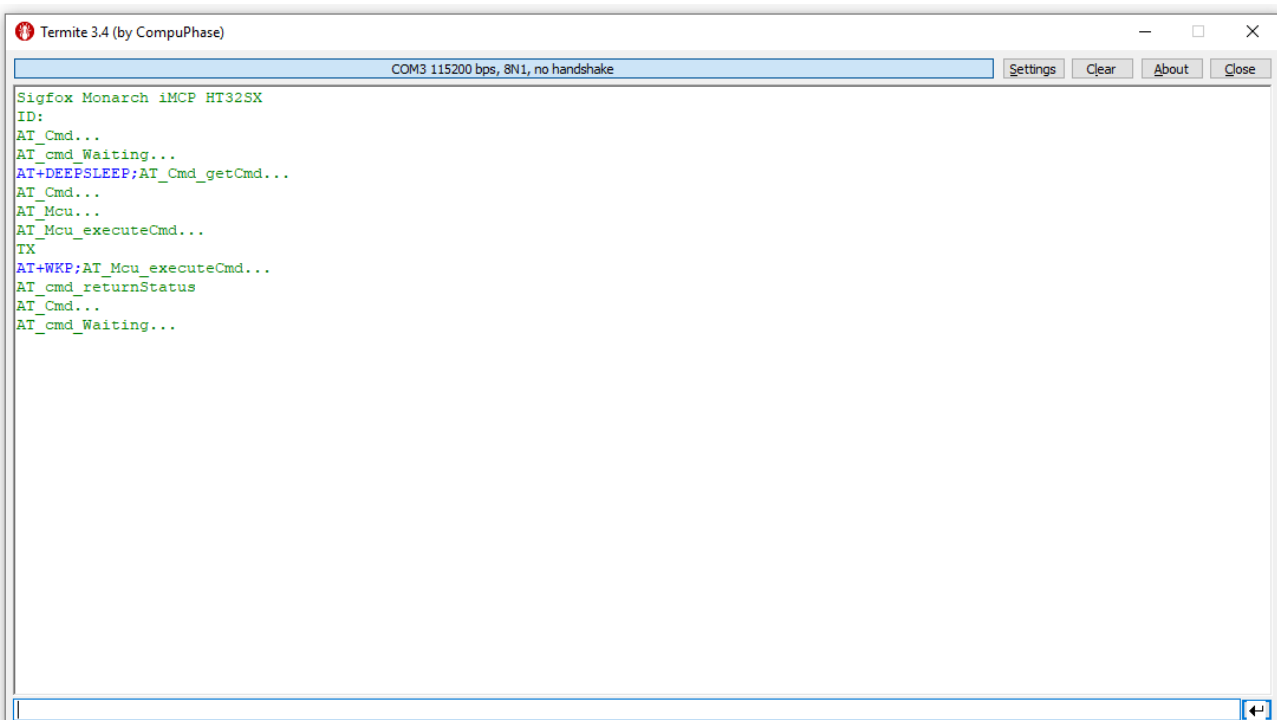
```

Termite 3.4 (by CompuPhase)
COM3 115200 bps, 8N1, no handshake
Sigfox Monarch iMCP HT32SX
ID:
AT_Cmd...
AT_cmd_Waiting...
AT+CFGRCZ=2;AT_Cmd_getCmd...
AT_Cmd...
AT_SigFox...
AT_SigFox_executeCmd
Config region...
RCZ 2
AT_cmd_returnStatus
AT_Cmd...
AT_cmd_Waiting...
AT+SEND=0:AAAAAAAA;AT_Cmd_getCmd...
AT_Cmd_getCmd...
AT_Cmd...
AT_SigFox...
AT_SigFox_executeCmd
Sending frame...
TX
TX
TX

Error Send Frame: 0
AT_cmd_returnStatus
AT_Cmd...
AT_cmd_Waiting...

```

Image 7: RS232 terminal with the expected results after the send frame command.




```

Termite 3.4 (by CompuPhase)
COM3 115200 bps, 8N1, no handshake
Sigfox Monarch iMCP HT32SX
ID:
AT_Cmd...
AT_cmd_Waiting...
AT+DEEPSLEEP;AT_Cmd_getCmd...
AT_Cmd...
AT_Mcu...
AT_Mcu_executeCmd...
TX
AT+WKP;AT_Mcu_executeCmd...
AT_cmd_returnStatus
AT_Cmd...
AT_cmd_Waiting...

```

Image 8: RS232 terminal with the expected results after the deep sleep command.

	APPLICATION NOTE 06, iMCP – HT32SX		iMCP, app note
	Application Note	11/05/2020	V 1.0

6 EXTRA DOCUMENTATION

Datasheets and application notes can be found at the HT32SX Repository (<https://github.com/htmicron/ht32sx>).


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).

8 CONTACT INFORMATION

Head Office – São Leopoldo, RS

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

	APPLICATION NOTE 06, iMCP – HT32SX		iMCP, app note
	Application Note	11/05/2020	V 1.0

Version Control:

1.0 – Aimed for the engineering samples of iMCP

Legal note: Copyright 2020 HT Micron Semicondutores S.A.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.