AN1340: Migrating to BT122 from iWRAP/BT121

Due to the end of WTXX module application support, driven by iWRAP firmware, Silicon Laboratories proposes the BT122 module as a hardware and firmware replacement. The new devices are dedicated to both new and ongoing projects in the maintenance phase. To help customers migrate from WTXX to new modules, we are offering the “BT122 - IWRAP Migration BGscript Efficient Realizator” application as a helper developer tool. This program is the first step in transfering the existing iWRAP application to BT122 modules.

KEY FEATURES

- Reading, parsing, and checking WTXX module configuration from Persistent Storage dump.
- Generating BT122 firmware project base on the WTXX module configuration.
- Providing tips and remarks for BT122 project to achieve expected application results.
- The easy first step for migration from a WTXX application to the new one dedicated for BT122 hardware.

The “BT122 - IWRAP Migration BGScript Efficient Realizator” (BIMBER) application helps to run BT122 firmware projects from a sample model project, which has been applicable with WTXX modules (such as WT11-A, WT11-E, WT11I-A, WT11I-E, WT11U-A, WT11U-E, WT12-A, WT32-A, WT32-E, WT32I-A, W32I-E, WT41-A, WT41-E, WT41-N, WT41U-A, WT41U-E, WT41U-N) so far.

The basic concept of migration is retrieving WTXX module configuration from a dump of Persistent Storage (of this module) and preparing an example BT122 firmware project file as a replacement proposal. WTXX modules are controlled by iWRAP firmware, which is based on the ASCI command console. Some typed commands can change configuration stored in Persistent Storage. A similar configuration and functionalities can be achieved on the BT122 module using API commands and other setup included in the firmware project files. API commands can be sent directly to the module (with API interface) or compiled in the standalone custom script (BGScript). Additionally, the results of the automatic migration process are ready to build the firmware project with BGScript script. The script source code is an example BT122 module application, which demonstrates the use of API commands and interactions with API events. Moreover, after adaptation and a detailed configuration, the module application example can be used as the final application.

This guide covers the basic aspects of using the BIMBER application (such as migration preparation, migration and results analysis). For a full introduction to the new BT122 module application (such as API commands description, firmware project structure, project files structure), refer to other user manuals (*BT122 API Reference Manual* [1], *BT122 Project Configuration User's Guide* [2], *BGScript Scripting Language Developer Guide* [3]).

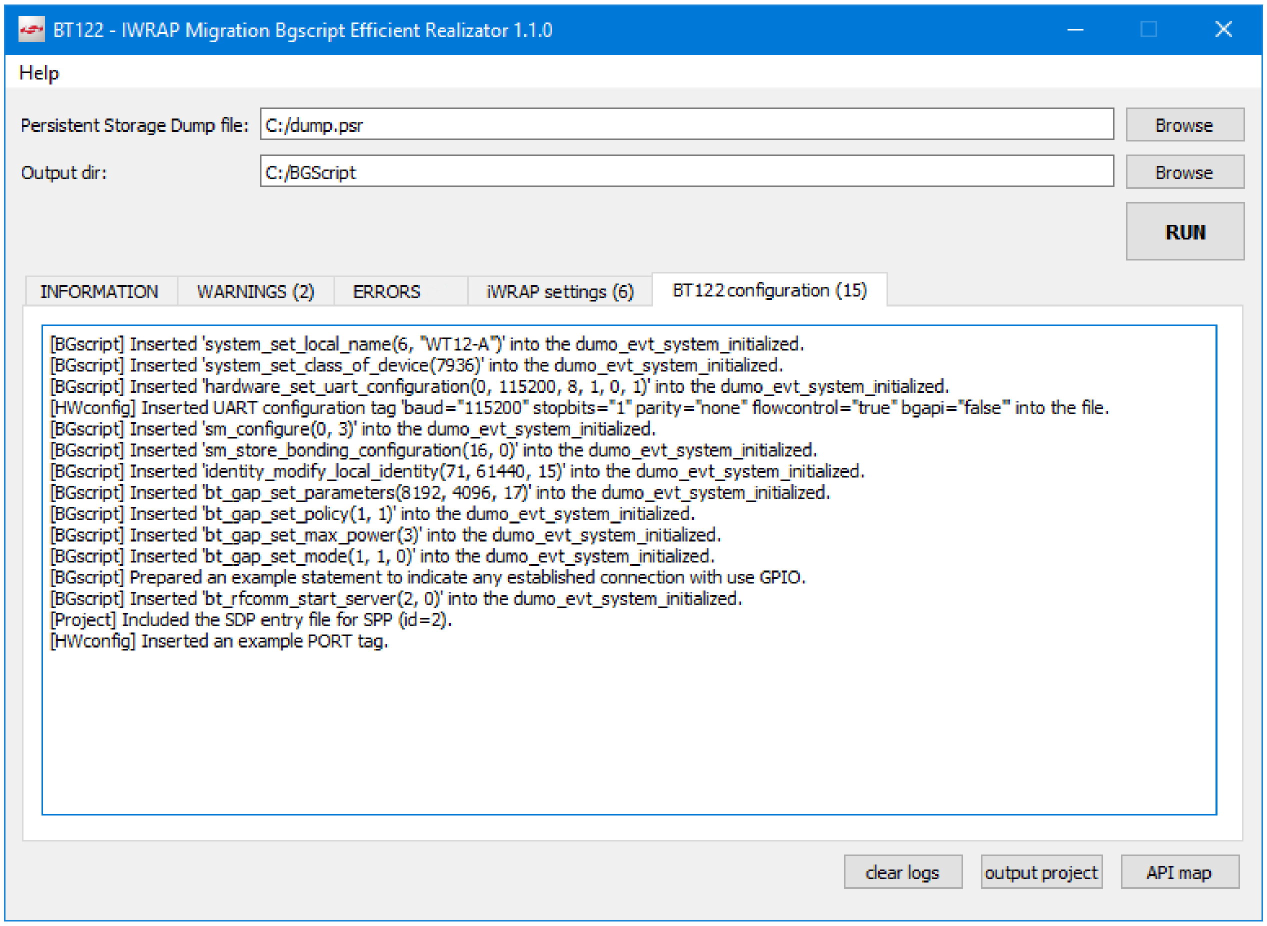
Table of Contents

1 Version History

Version	Comments
1.0	Initial version

2 Main Interfaces of the Application

BT122 - IWRAP Migration BGScript Efficient Realizator application has one main window that provides all the interfaces for choosing the correct input and output paths, run the migration process, and analyze the results. The following is a detailed description of each interface.



```
[BGscript] Inserted 'system_set_local_name(6, "WT12-A")' into the dumo_evt_system_initialized.
[BGscript] Inserted 'system_set_class_of_device(7936)' into the dumo_evt_system_initialized.
[BGscript] Inserted 'hardware_set_uart_configuration(0, 115200, 8, 1, 0, 1)' into the dumo_evt_system_initialized.
[HWconfig] Inserted UART configuration tag 'baud="115200" stopbits="1" parity="none" flowcontrol="true" bgapi="false" into the file.
[BGscript] Inserted 'sm_configure(0, 3)' into the dumo_evt_system_initialized.
[BGscript] Inserted 'sm_store_bonding_configuration(16, 0)' into the dumo_evt_system_initialized.
[BGscript] Inserted 'identity_modify_local_identity(71, 61440, 15)' into the dumo_evt_system_initialized.
[BGscript] Inserted 'bt_gap_set_parameters(8192, 4096, 17)' into the dumo_evt_system_initialized.
[BGscript] Inserted 'bt_gap_set_policy(1, 1)' into the dumo_evt_system_initialized.
[BGscript] Inserted 'bt_gap_set_max_power(3)' into the dumo_evt_system_initialized.
[BGscript] Inserted 'bt_gap_set_mode(1, 1, 0)' into the dumo_evt_system_initialized.
[BGscript] Prepared an example statement to indicate any established connection with use GPIO.
[BGscript] Inserted 'bt_rfcomm_start_server(2, 0)' into the dumo_evt_system_initialized.
[Project] Included the SDP entry file for SPP (id=2).
[HWconfig] Inserted an example PORT tag.
```

Figure 1. Main Application Window View

2.1 Persistent Storage Dump File

Specifies the absolute path to the WTXX module's persistent storage dump file. The path can be typed directly or selected in the file browser (opened by the Browse button). Methods that perform dump of Persistent Storage are described in paragraph 3.1.

2.2 Output Directory

Specifies the absolute path to the directory prepared for the output files. The path can be typed directly or selected in the file browser (opened by the Browse button). If the directory does not exist, it is created automatically (as long as the parent path exists and is writable).

2.3 Output Logs

To trace the migration process, messages are generated into the log outputs. There are five types of messages that are printed on the dedicated tabs of the log. Each of them is described in the following paragraphs.

2.3.1 INFORMATION

INFORMATION log messages are simple helper information, related to the migration steps, such as:

- creating new output files of the final project,
- starting/finishing following parts of the migration process,
- pushing the contents of the output files,
- providing other technical information.

2.3.2 WARNINGS

WARNINGS are important messages that should be kept in mind after the migration process is finished. They are related to:

- detecting non-default values of the configuration parameters,
- missing non-mandatory values of configuration parameters of a WTXX module,
- detecting non-standard or non-compatible firmware versions of a WTXX module,
- needed actions that must be taken after the migration process (such as parameters to be checked, part of the project to be replaced, etc.),
- detecting hardware and software limitations (e.g., non-supported profiles, missing required hardware, etc.).

2.3.3 ERRORS

ERROR messages indicate specific problems and failures in the migration process, such as:

- error of reading Persistent Storage keys database from the dump,
- no mandatory values for the WTXX module configuration parameters of WTXX module (needed for correct configuration of a BT122 module),
- out of range values of read configuration parameters,
- detecting firmware/software version that cannot be migrated to the BT122 project.

2.3.4 iWRAP Settings

iWRAP setting messages trace the process of reading and parsing the WTXX module configuration stored in the Persistent Storage memory dump file. These messages are related to the following:

- parsing values of stored configuration parameters (hardware configuration, Bluetooth configuration, software configuration),
- detecting specific Bluetooth and software features used,
- detecting user-defined parameters and their values.

2.3.5 BT122 Configuration

BT122 configuration messages follow the preparation and creation of a new configuration for the BT122 module (base on WTXX configuration). These messages are related to the following:

- inserting commands and blocks of code into the BGScript script code,
- inserting needed XML tags into other project files,
- including files in the project compilation.

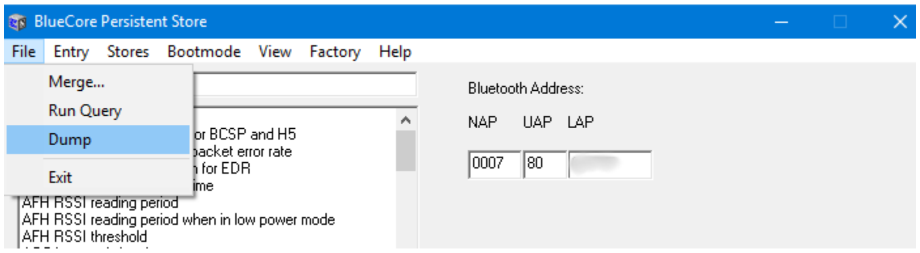
3 Executing the Migration

The following sections of this document describe the full process of preparing, executing, and analyzing the migration from a WTXX module setup to the BT122 firmware project.

3.1 Dumping the Persistent Storage Keys Database

The first step of the preparation is to make the dump of Persistent Storage of WTXX module. There are two ways to do it. First, using dedicated software and a programmer (e.g., BlueCore Persistent Store (PSTool) with DEV-SYS-1808-1A) delivered by the WTXX chip manufacturer (https://www.qualcomm.com). If you do not have these tools available, contact Qualcomm or Silicon Laboratories support (https://www.silabs.com/support).

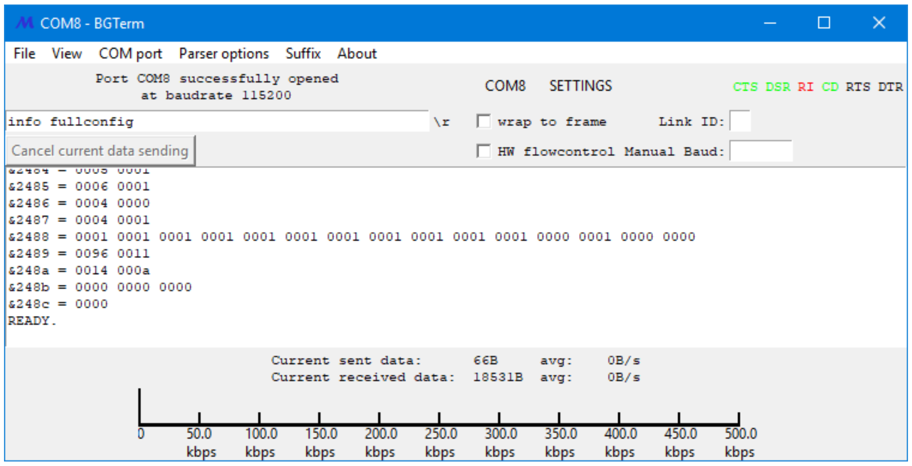
Run the Dump from File menu and save the values of all keys to the output file (see Figure 2).



or BCSP and H5
packet error rate
n for EDR
ime
AFH RSSI reading period
AFH RSSI reading period when in low power mode
AFH RSSI threshold

Figure 2. Using PSTool Application to Perform of the Persistent Storage Dump of the WTXX Module

If it is not possible to connect to the dedicated programmer, make a dump with the iWRAP console. To perform this task, connect to WTXX with the USART terminal (with any serial terminal program), make sure that the module is responsive and then type “info fullconfig” command. As a result, the key values should be printed in the console output in the expected format (Figure 3).



```
Port COM8 successfully opened
   at baudrate 115200
```

```
info fullconfig
```

```
&2485 = 0006 0001
&2486 = 0004 0000
&2487 = 0004 0001
&2488 = 0001 0001 0001 0001 0001 0001 0001 0001 0001 0001 0001 0000 0001 0000 0000
&2489 = 0096 0011
&248a = 0014 000a
&248b = 0000 0000 0000
&248c = 0000
READY.
```

```
Current sent data:      66B     avg:   0B/s
Current received data:  18531B  avg:   0B/s
```

Figure 3. Serial Port Console Output with Persistent Storage Keys Values after Invoking “info fullconfig” Command

A sample part of the Persistent Storage dump file is available in the list below. Make sure all readable storage keys have been written to the file. A typical dump can contain around 350 unique dumped keys (depending on the WTXX hardware and software version).

```
// First of the dumped Persistent Storage key.
&0001 = 00ff ffff ffff ffff// Key value (stored to 0x0001 address).
// Description of the Persistent Storage key.
&0003 = 0000 0000
...
Next dumped Persistent Storage keys.
...
&25be = 0000
// Last of the dumped Persistent Storage key.
&25c0 = 0000
```

3.2 Running the BIMBER Application

The BIMBER application is delivered with the BT122 SDK environment. The latest available version of the software for download/installation is available at https://www.silabs.com. After the installation is complete, search BIMBER in the Windows Start Menu and run it.

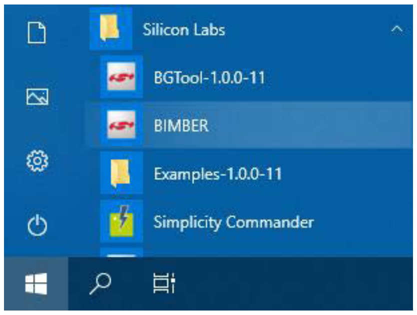


Figure 4. BIMBER Application in Windows Start Menu

3.3 Running the Migration

If the BIMBER application is running already, type/select the path to the file with Persistent Storage dump, and type (or choose) the path to the output directory for new BT122 project files. See Figure 1 and the description of interfaces in paragraphs 2.1 and 2.2. Next, click the RUN button in the main window of the BIMBER application (Figure 1). If everything is ok, the message "Migration process successful. Project files have been generated." will be printed (as long as the paths are correct and the reading of the persistent storage file is consistent).

3.4 Analysis of Output Messages and Project Output Files

After the migration process is complete, double-check the output messages and output files. The following steps are significant for controlling results:

- Check the iWRAP settings log tab. Ensure that the correct WTXX configuration has been parsed. Pay attention to the values of the main parameters such as: WTXX module type, module name, Bluetooth configuration, Bluetooth profiles used, hardware interfaces configuration, etc. Make sure that this configuration corresponds to your current WTXX module application.
- Check the WARNINGS log tab and try to resolve all problematic cases. Note that not all features can be ported automatically and not all features have an identical replacement (see paragraph 4 about migration limitations).
- Ensure that no errors have been reported in the ERRORS log tab.
- Check the INFORMATION log tab and trace the migration process flow. See which files have been generated as the BT122 output project. Also, pay attention to other printed information.
- Check the BT122 configuration log tab. Look for which commands, code blocks, and tags have been inserted into the output files. See the BT122 API documentation for command definitions and the passed parameter values.
- Click the "output project" button to open the project directory and investigate the output files. Pay attention to the project parts which that require manual modification, adaptation, and review (see paragraph 4).

4 Limitations of the Automatic Migration Process

Due to firmware and hardware differences between the BT122 and WTXX modules, automatic migration is only a helper tool that starts porting your Bluetooth device application. However, there are a few cases where additional efforts must be made to achieve the desired application:

- You must design the use of I/O pins, depending on expected communication interfaces and other GPIO. If any I/O lines have been assigned in your old application, they will need to be reassigned in the BT122 application. You must consider the possible use of pins and choose the optimal module pinout for your application.
- The new basic feature (unlike WTXX modules) is the use of runtime script written in BGScript language. It's a replacement for many features that have been delivered in WTXX firmware so far. For a better overview, read about BGScript usage in [3] and try to investigate examples delivered with BT122 SDK package.
- If you need some custom initialization of the module, it must be implemented in the expected event handling subroutine (dumo_msg_system_boot_evt or dumo_msg_system_initialized_evt). The difference is that on WTXX modules it was possible to set the initialization command with the "SET CONTROL INIT".
- If you would like to trigger some command or code block with a GPIO line, you must implement it in the expected dumo_msg_hardware_interrupt_evt event branches. Note that, during the migration, a sample template is included. Specific of implementation and hardware configuration details depend on user requirements and experience.

- If you would like to trigger GPIO lines to indicate some special cases (e.g., establishing a connection, breaking a connection), you must implement it in the expected events and code blocks (e.g., dumo_msg_bt_connection_opened_evt, event dumo_msg_bt_connection_closed_evt, etc.).
- Some commands used, such as CALL or CLOSE, leave no residue in Persistent Storage. One must use equivalent commands in the correct way (either by inserting into BGScript or by calling it via API). Click the “API map” button to open the guide, which binds iWRAP command and their replacements in BT122 API. Uses depend on application requirements in run time.
- The output BT122 firmware project, which is based on the BGScript scripts, is only an application proposal, one that shows the correct way to prepare the project files and use the API commands in code blocks. The same results can be achieved in many different ways (like connecting the BT122 module to the external MCU, connecting the MCU module to the PC, driving BT122 module by BGTool GUI application, etc). The choice of the final solution depends on user project requirements.

5 Reference

[1] Silicon Laboratories, *BT122 API Reference,* 2021.

[2] Silicon Laboratories, *BT122 Configuration Guide,* 2021.

[3] Silicon Laboratories, *UG209: BGScript Scripting Language Developer Guide,* 2017.

IoT Portfolio
www.silabs.com/products

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com