

Atmosic SDK

User Guide

SUMMARY: This document provides instructions on how to install the Atmosic Software Development Kit (SDK) and use it with the Atmosic Evaluation Kit (EVK) based on ATM2/ATM3 Wireless SoC Devices.



Atmosic™

CONFIDENTIAL | Doc. No. ATM-UGCSDK-0010

Table of Contents

User Guide	1
Table of Contents	2
List of Figures	3
List of Tables	3
Acronyms and Abbreviations	4
1. Overview	5
2. EVK Setup	5
2.1 ATM2/ATM3 EVK	5
3. SDK Installation	6
3.1 Using Windows Installer	6
3.2 Install Atmosic SDK	6
3.3 Install Atmosic Reference Design Interface (RDI)	10
3.3.1 ATM2/ATM3 EVK	10
3.3.2 Remove RDI Interface	12
3.4 Connect to Debug Interface	13
3.5 Launch MSYS2 shell	14
3.6 Atmosic SDK Platform Working Directory	14
4. Platform Tools & Utilities	16
4.1 In-System Programming Tool	16
4.2 build_archive	17
4.3 Makefile Helpers	18
4.4 show_archive	19
4.5 burn_archive	20
4.6 ATM2/ATM3 Platform ROM and OTP Application	24
4.6.1 ROM and Custom Application	24
4.6.2 ROM and Flash Application	25
4.6.3 Flash Operating System	25
5. Make Options and Commands	27
6. Building Beacon Application Using Atmosic SDK	29
6.1 ATM2 and ATM3 Device	29
6.1.1 Board Configuration	29
6.1.2 Beacon	29
7. Building Energy Harvesting Beacon	32
7.1 ATM3 Devices	32
8. Uninstall Atmosic SDK	33
Reference Documents	34
Revision History	35

List of Figures

- Figure 1 - Extracted Release Package
- Figure 2 - Atmosic License Agreement
- Figure 3 - SDK Setup Directory
- Figure 4 - SDK Installation Path Message
- Figure 5 - Installing SDK
- Figure 6 - Installation Completed
- Figure 7 - SDK in Start Menu
- Figure 8 - USB COM Port with ATM2/ATM3 EVK
- Figure 9 - Install RDI USB Driver with ATM2/ATM3 EVK
- Figure 10 - RDI USB in Device Manager with ATM2/ATM3 EVK
- Figure 11 - Debug Interface Using PuTTY
- Figure 12 - Start Menu
- Figure 13 - Atmosic SDK Path
- Figure 14 - Atmosic Product Code on the EVB
- Figure 15 - Uninstall SDK

List of Tables

- Table 1 - Applicable ATM2/ATM3 EVKs
- Table 2 - Binary File Types
- Table 3 - build_archive Bundles for ATM2/ATM3 Platforms
- Table 4 - Field Name and Argument Description
- Table 5 - Make Options and Commands

Acronyms and Abbreviations

Acronyms	Definition
API	Application Programming Interface
ATM2	ATM2201 ATM2202 ATM2221 ATM2231
ATM3	ATM3201 ATM3202 ATM3221 ATM3231
CLI	Command Line Interface
EVB	Evaluation Board
EVK	Evaluation Kit
GADC	General purpose Analog to Digital Conversion
GPIO	General Purpose Input Output
HCI	Human Computer Interaction
HID	Human Interface Device
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
ISP	In-System Programming
NVDS	Non-Volatile Data Storage
OB	On-Board
OTA	Over-The-Air
OTP	One Time Programmable
RDI	Remote Debug Interface
ROM	Read Only Memory
SDK	Software Development Kit
SoC	System-on-Chip

1. Overview

This document provides instructions on how to install the Atmosic SDK to support EVK based on the ATM2/ATM3 Wireless SoC Devices. For more information about using the EVK, please refer to the EVK User Guides.

Related documents are listed in the [Reference Documents](#) section.

2. EVK Setup

2.1 ATM2/ATM3 EVK

There are multiple versions of the ATM2 or ATM3 EVKs based on the specific device and package configuration. For hardware setup and configuration instructions, consult the User Guide for the specific EVK being used.

[Table 1](#) shows the ATM2/ATM3 EVKs supported.

EVK	Kit Part Number
Evaluation Kit for ATM2202	ATMEVK-M2202-02
Evaluation Kit for ATM2221	ATMEVK-M2221-02
Evaluation Kit for ATM3202	ATMEVK-M3202-02
Evaluation Kit for ATM3221	ATMEVK-M3221-02

Table 1 - ATM2/ATM3 EVKs Supported

3. SDK Installation

3.1 Using Windows Installer

The SDK Windows Installer requires administrator privileges.

3.2 Install Atmosic SDK

The setup package of Windows Installer for Atmosic SDK is a compressed file named AtmosicSDK_Inst_<version_number>.zip. Please extract the file and double-click the AtmosicSDK_Inst_<version_number>.exe to start the Atmosic SDK Installer. [Figure 1](#) shows the release package source tree after the files are extracted.



 AtmosicSDK_Inst_x.y.z	8/15/2022 10:50 PM	Application	777,151 KB
 setup	8/15/2022 10:40 PM	Configuration settings	1 KB

Figure 1 - Extracted Release Package

Please review the license agreement at the Atmosic website. If you accept the terms of the agreement, click I Agree to continue.

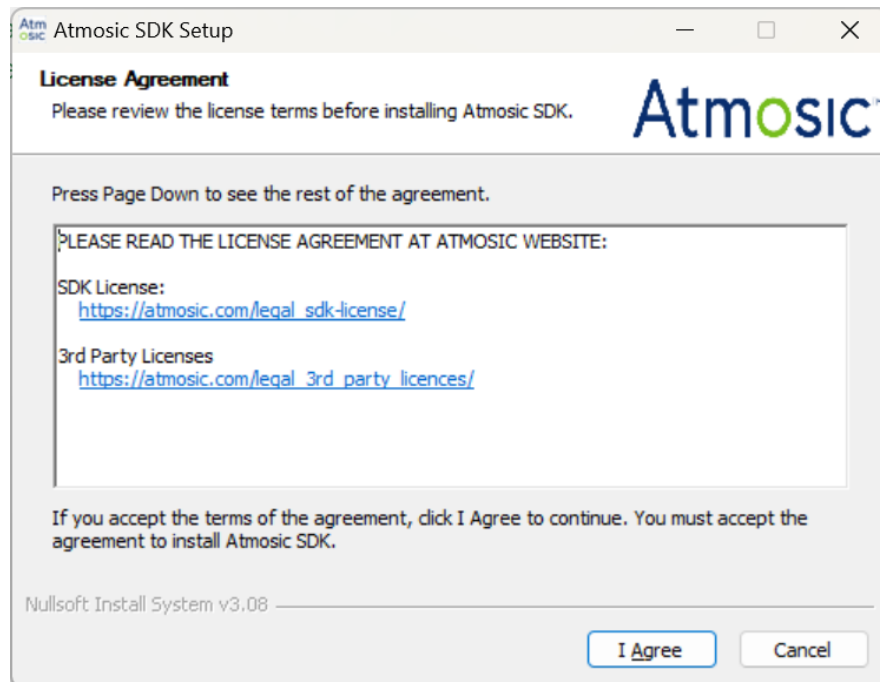


Figure 2 - Atmosic License Agreement

The Atmosic SDK will occupy 2.5 GB of disk space and will be located in the C:\AtmosicSDK\ folder, see [Figure 3](#). Please make sure the disk space is enough before installing the Atmosic SDK. Subsequently, click the Install button to continue.

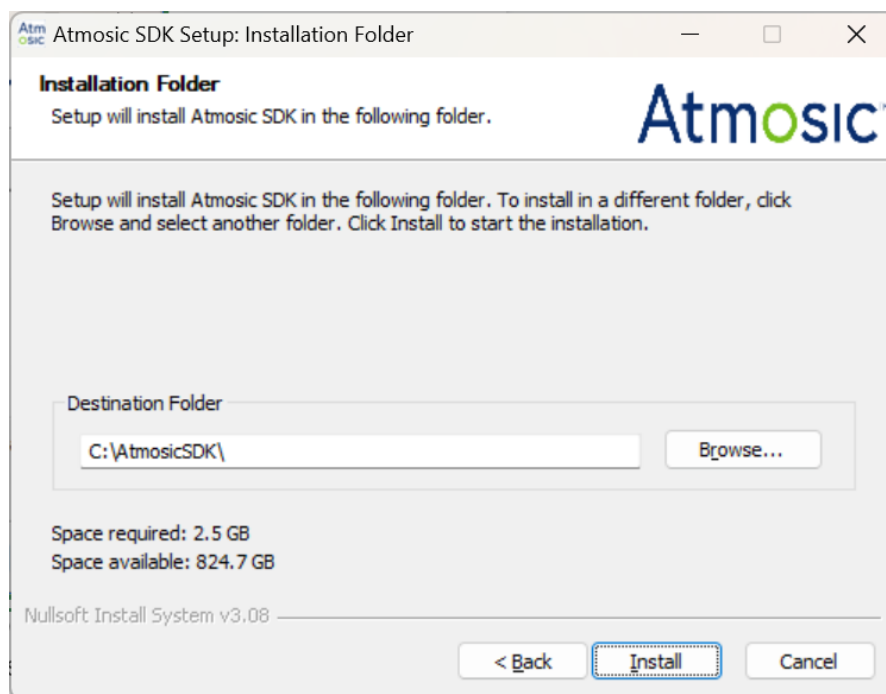


Figure 3 - SDK Installation Folder

Note: The installer cannot accept a path destination directory name with spaces. For example, *Atmosic SDK* is not acceptable while *AtmosicSDK* is acceptable. The Windows Installer will pop up a message if the user selects an installation path name that includes spaces. See [Figure 4](#).

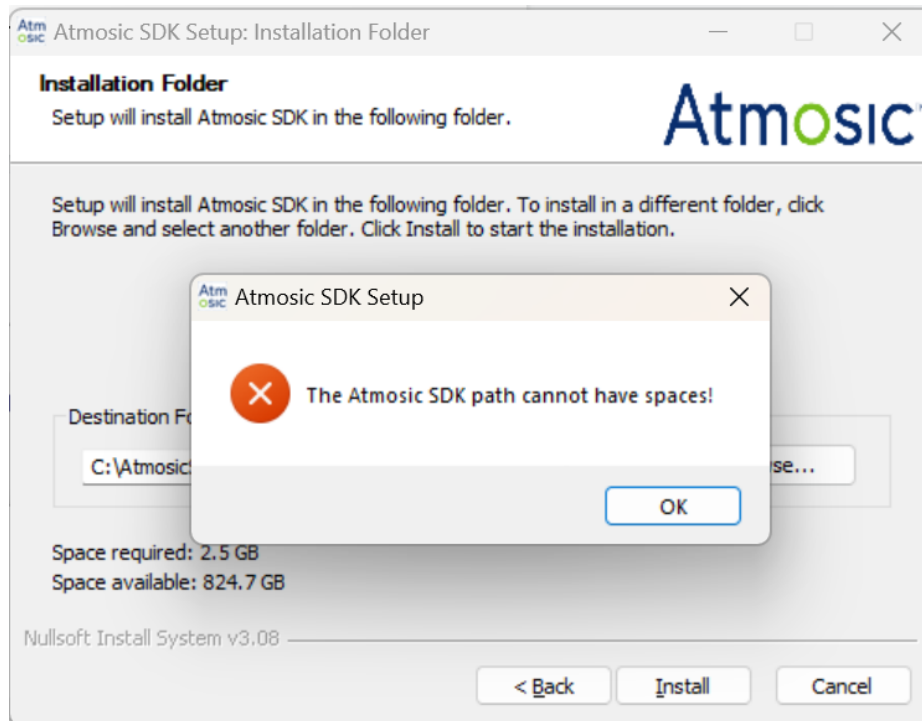


Figure 4 - SDK Installation Path Message

The installation process takes several minutes to complete. The Windows Installer is installing MSYS2, GNU Toolchain, and Atmosic SDK files to the target directory. See [Figure 5](#).

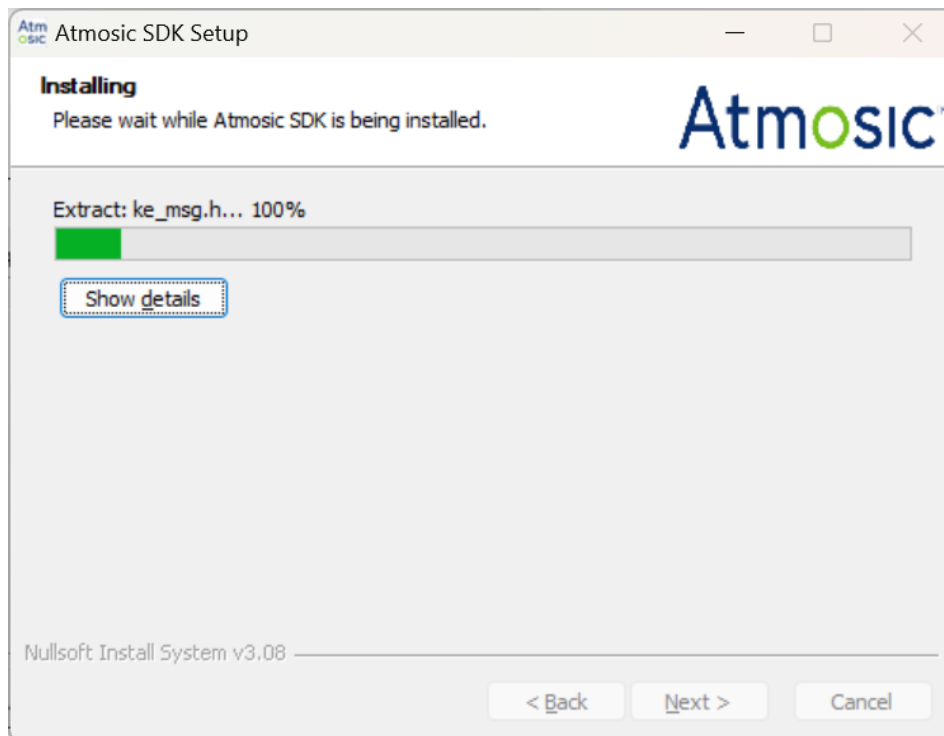


Figure 5 - Installing SDK

The Windows Installer will pop up a message to notify the installation is completed, click OK to close the Windows Installer. See [Figure 6](#).

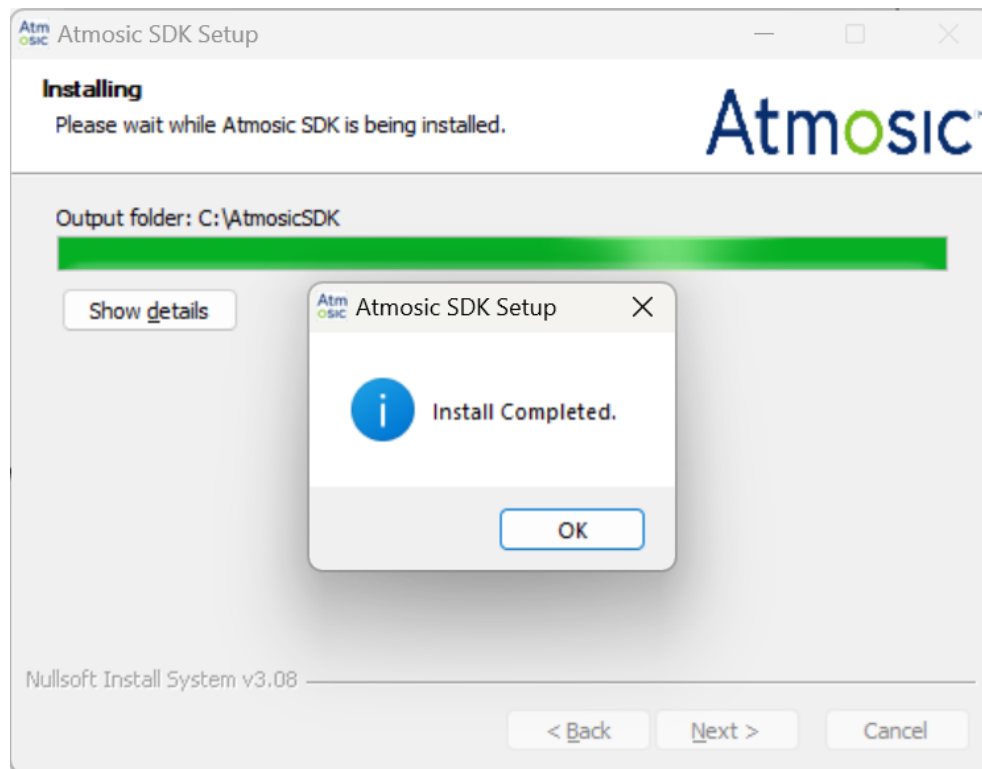


Figure 6 - Installation Completed

The Installer will create a folder named AtmosicSDK in the Start Menu as shown in [Figure 7](#) during the installation process.

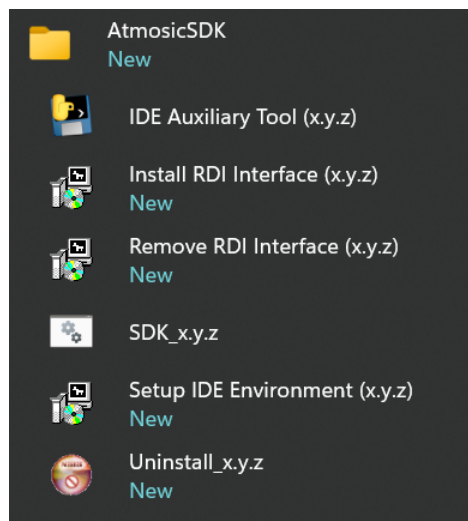


Figure 7 - SDK in Start Menu

3.3 Install Atmosic Reference Design Interface (RDI)

3.3.1 ATM2/ATM3 EVK

Please connect the Atmosic EVB (USB1) to the Interface board, and connect the USB cable to the laptop and wait for Windows to install the FTDI driver (~ 30 seconds).

Open the device manager (open the Run dialog box by pressing and holding the Windows key, then press the R key).

Enter `devmgmt.msc` then click the OK button to verify whether the RDI USB interface exists or not.

If there are 2 COM ports listed in the device manager after plugging-in the Atmosic EVB to the laptop, please click the Install RDI Interface shortcut in the Start Menu/Atmosic folder. See [Figure 8](#).

Note: *The COM port numbers are assigned by the Windows system.*

Note: *If the device manager shows Atmosic RDI USB driver is already installed, move to the [Connect To Debug Interface](#) section.*

Note: *Atmosic RDI USB driver installation is required when the new interface board is connected to the computer for the first time.*

The Device Manager shows COM port information before installing the Atmosic RDI USB driver.

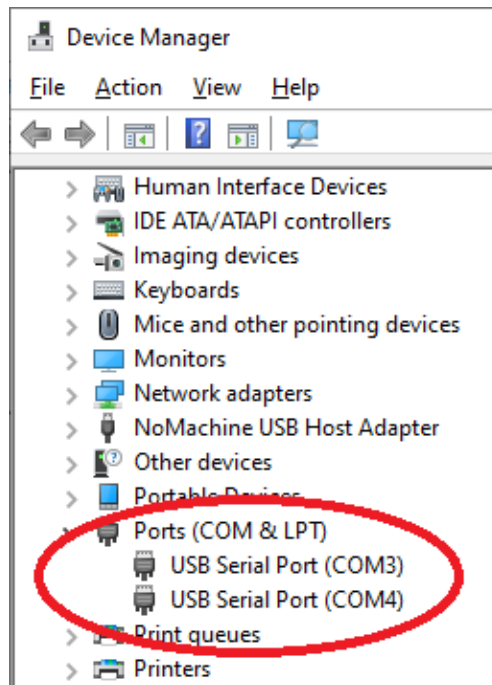


Figure 8 - USB COM Port with ATM2/ATM3 EVK

Please click on the Install button if the system pops up the following message.

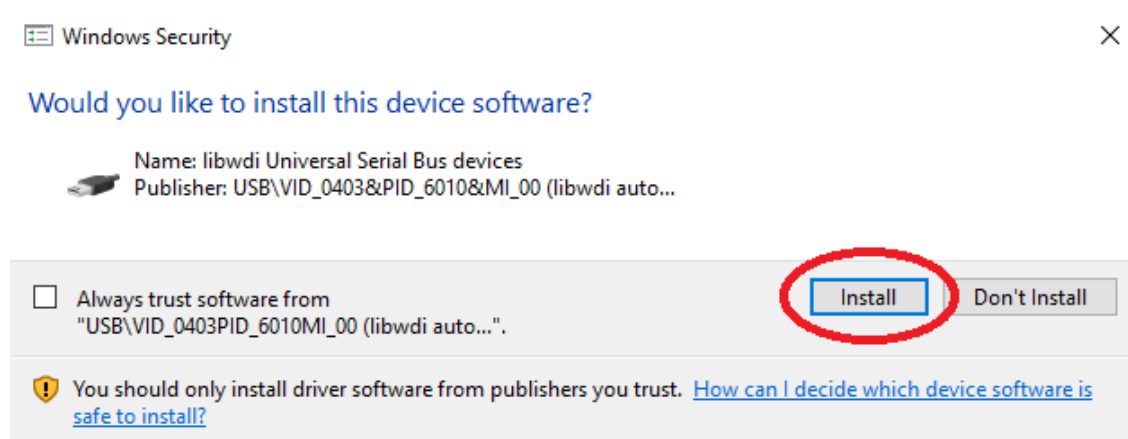


Figure 9 - Install RDI USB Driver with ATM2/ATM3 EVK

After installing the Atmosic RDI USB driver, the Atmosic RDI USB1 interface will display in the Device Manager. Moreover, the COM4 is a debug console that could be used to display debug messages through PuTTY.

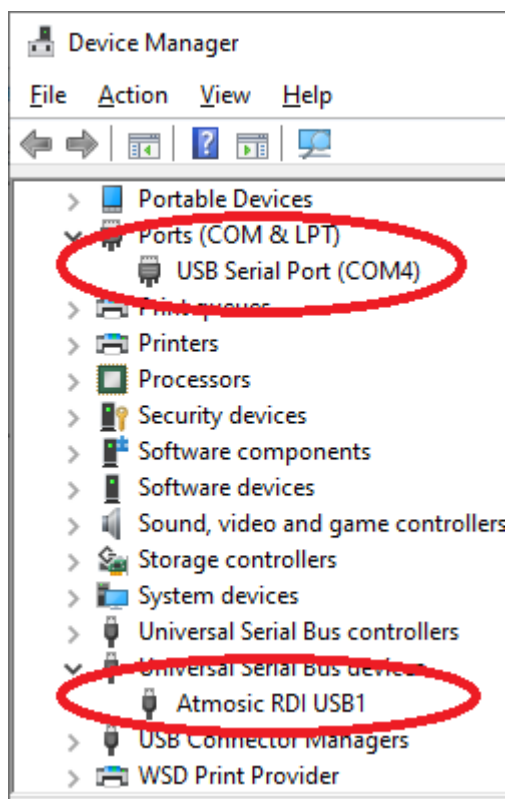
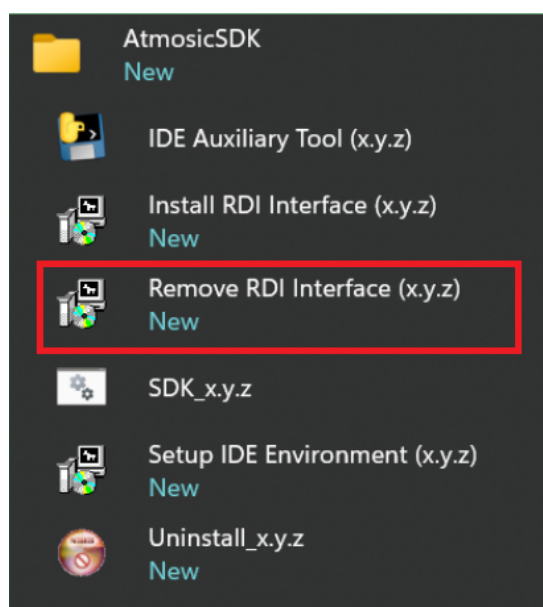


Figure 10 - RDI USB in Device Manager with ATM2/ATM3 EVK

3.3.2 Remove RDI Interface

If you want to remove the RDI driver or switch to JLink, please click the Remove RDI Interface shortcut in the Start Menu/Atmosic folder.



3.4 Connect to Debug Interface

The Atmosic EVK provides the log output through a debug interface (COM port). The developer could use utility to capture (i.e. PuTTY, Tera Term ...etc.) the log. This section describes how to capture logs through PuTTY.

When opening the PuTTY tool:

- 1) Select the Serial radio button
- 2) Enter the COM port information in the Serial line item (i.e. COM4)
- 3) Input 115200 in the Speed item
- 4) Click the Open button to open the COM port and wait for the debug message output

The debug console will only begin to display when an example is running; otherwise, the console will remain empty as shown in [Figure 11](#).

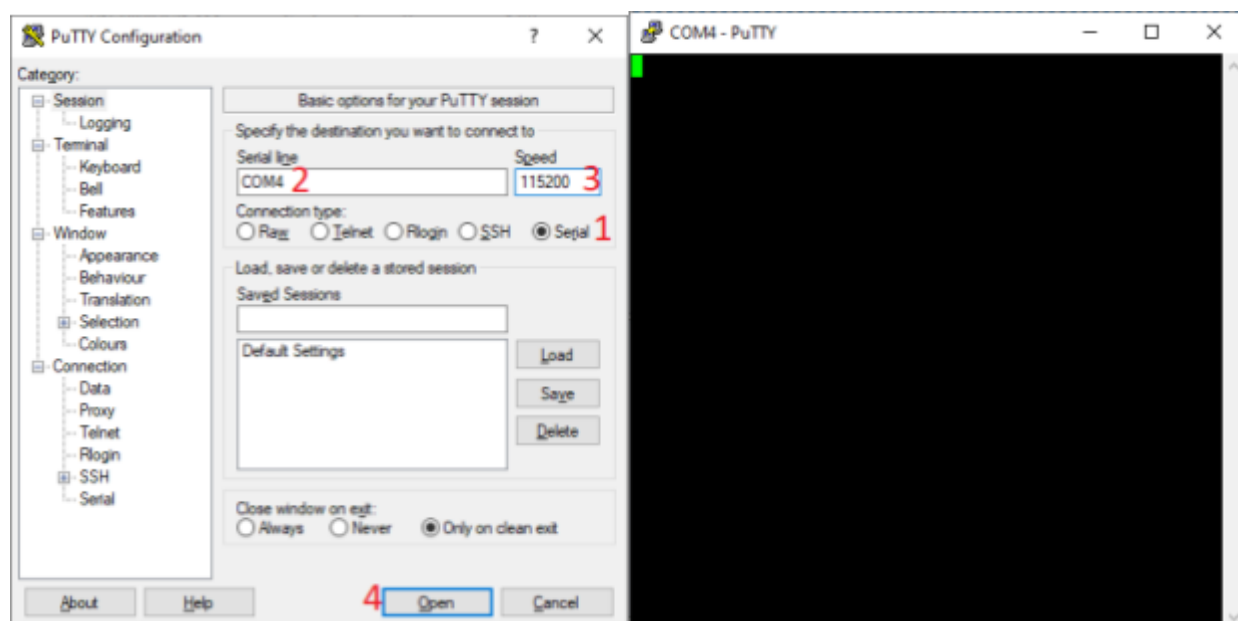


Figure 11 - Debug Interface Using PuTTY

3.5 Launch MSYS2 shell

Invoke the MSYS2 shell from the Windows menu, Start Menu/AtmosicSDK/SDK_<version_number>. [Figure 12](#) shows the Start Menu.

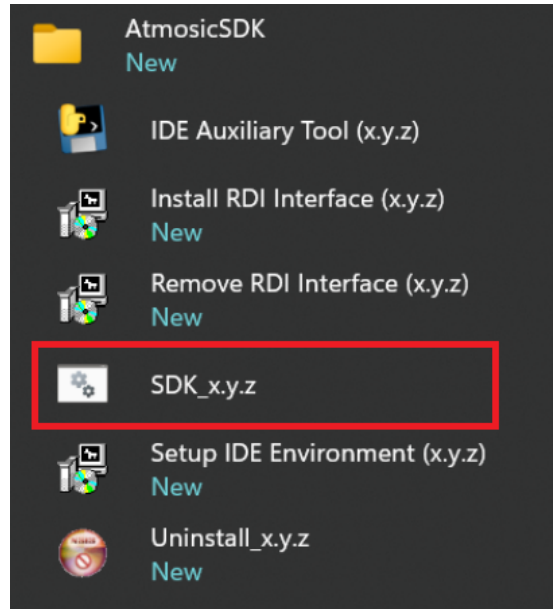


Figure 12 - Start Menu

The MSYS2 shell will be invoked and located in the AtmosicSDK working directory as shown in [Figure 13](#).

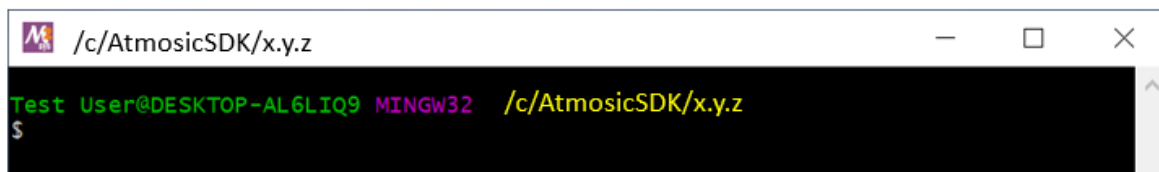


Figure 13 - Atmosic SDK Path

3.6 Atmosic SDK Platform Working Directory

SDK 5.x.0 and later supports ATM2/ATM3 devices

In order to build the correct application software for any ATM2 or ATM3, it is necessary to identify the device version and use the appropriate directory in the Atmosic SDK for the software build.

ATM2 and ATM3 EVB have a sticker that identifies the part number of the Atmosic device on the board. For example, the Atmosic EVB is labeled with ATM3221-010 as shown in [Figure 14](#). It is assembled with an ATM32xx-x1x chip revision and to compile for this revision, the platform path is `platform/atm3/ATM32xx-x1x`.

The ATM3 EVB has printed the part number of the Atmosic device on the board, please refer to [Figure 14](#).

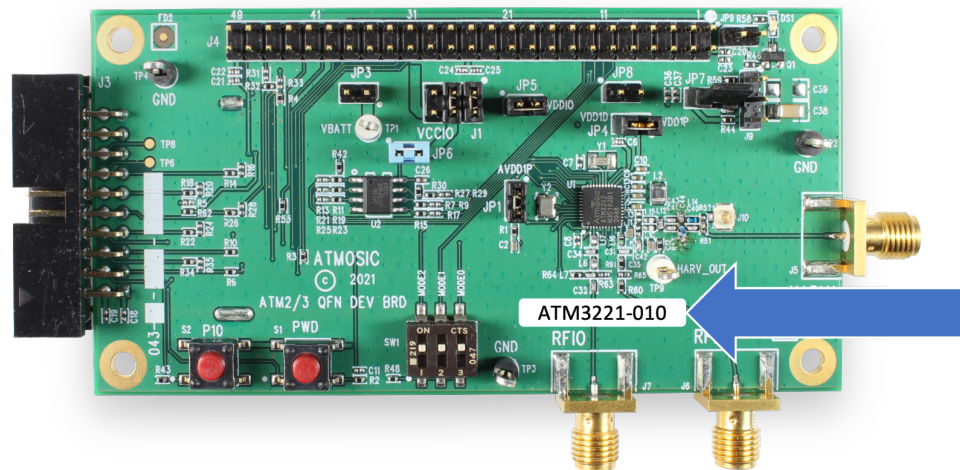


Figure 14 - Atmosic Product Code on the EVB

Following is a list of all Atmosic SDK platform working directories, please use the appropriate directory in the Atmosic SDK for the software build.

- `platform/atm2/ATM22xx-x1x`
- `platform/atm3/ATM32xx-x1x`

4. Platform Tools & Utilities

4.1 In-System Programming Tool

The SDK ships with a tool called Atmosic ISP Tool for bundling three types of binaries, flash, flash NVDS, and OTP NVDS into a single binary archive. See [Table 2](#).

Binary Type	Description
.bin	binary file contains flash or nvds data only.
.elf	elf file is a common standard file format, consists of elf headers and flash data.
.nvm	OTP NVDS file contains OTP nvds data.

Table 2 - Binary File Types

The ISP tool, which is also shipped as a stand-alone package, can then be used to unpack the components of the archive and download them on a device.

In every platform example, e.g. BLE_adv_scan, the makefiles have the following targets for building, examining, and programming an ISP archive, respectively.

```
make build_archive // requires appropriate nvds file, see below
make show_archive // display nvds and image details
make burn_archive // program files into storage
make clean_archive // remove generated binary archive
```


4.2 build_archive

build_archive bundles different binary files from ATM2/ATM3, and \$(APP).bin where \$(APP) is the name of the application directory. See [Table 3](#).

Platform	Non-OTA support	OTA support
ATM2/ATM3	<ul style="list-style-type: none"> • \$(APP).bin/.elf • flash_nvds.bin 	<ul style="list-style-type: none"> • \$(APP).bin/.elf • flash_nvds.bin • otp_nvds.nvm (optional)

Table 3 - build_archive Bundles for ATM2/ATM3 Platforms

In ATM2/ATM3, not all binaries are mandatory, build_archive can build selective binaries. For example, the target \$(APP).elf can be chosen instead of the \$(APP).bin by setting the make variable ARCH_FLASH_TYPE=elf.

Note: flash_nvds.bin and \$(APP).bin apply only to flash-based applications and otp_nvds.nvm applies to OTP storage, see OTP_beacon sample application. The Atmosic mobile app requires to load the generated *.atm from build_archive to perform OTA upgrade. The mobile app will resolve the bundled binaries by itself.

4.3 Makefile Helpers

Makefile helper allows various application configurations.

Key helpers for NVDS:

```
flash_nvds.data := List of NVDS tags required by application
otp_nvds.data: List of NVDS tags for application running in OTP
```

build_archive NVDS requirements:

For flash-based applications that define flash_nvds.data in makefile:

```
make build_flash_nvds // creates flash_nvds.bin
```

For OTP-based data that define otp_nvds.data in makefile:

```
make pull_otp_nvds // creates otp_nvds.nvm
make show_pretty_flash_nvds // shows contents of .nvm file
make rebuild_otp_nvds // creates new otp_nvds.nvm
```

Output of make build_archive

```
$(APP)_arch.atm // archived application, FW binary plus NVDS (OTP and/or
flash)
```

Output of make build_archive ARCH_FLASH_TYPE=elf

```
$(APP)_arch.atm // archived application, FW elf plus NVDS (OTP and/or flash)
```

4.4 show_archive

Example output of show_archive from WURX_ADV example application below. It shows platform and image size:

```

$ make show_archive
/c/AtmosicSDK/5.4.1/tools/atm_isp decode -i WURX_adv_arch.atm
MPR start: 0, size: 0, lock_size: 0
OTA support: 0
SDK version: 5.4.1
Platform: ATM32xx-x1x (atm3 family)
Board: ATMEVK_M3221
LoadFlashNvds image=(size=225,content=b"NVDS\x06\x06'\x00"... )
LoadFlash image=(size=24808,content=b'\x13H\x14I\x14J\x15K'...)
region_start=0x0 region_size=0x78000 address=0x10000000
CmdExtend image=(size=209,content=b'UFLASH_S'...) type=txt
extrainfo=LAYOUT_MAP extrainfo=LAYOUT_MAP

```

Field Name	Field Name Description	Argument	Argument Description
MPR	<ul style="list-style-type: none"> MPR meta data 	<ul style="list-style-type: none"> start size lock_size 	
OTA support	<ul style="list-style-type: none"> OTA support or not 		
SDK version	<ul style="list-style-type: none"> version of SDK that atm file generated 		
Platform	<ul style="list-style-type: none"> chipset and family name 		
Board	<ul style="list-style-type: none"> board name 		
LoadFlash	<ul style="list-style-type: none"> translate to openocd loadFlash command 	<ul style="list-style-type: none"> image size image content region_start region_size address 	<ul style="list-style-type: none"> length of binary file brief content of binary file start address of flash region to be loaded size of flash region to be loaded base address of flash

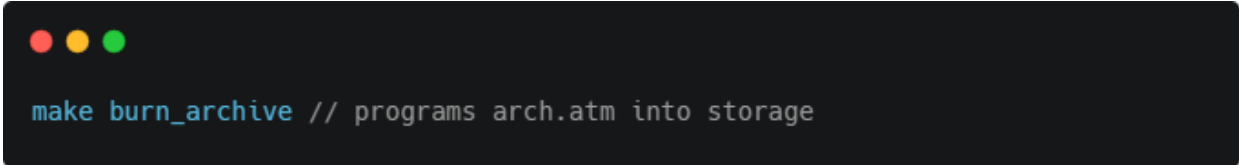
Field Name	Field Name Description	Argument	Argument Description
LoadFlashNvds	<ul style="list-style-type: none"> translate to openocd loadFlashNvds command 	<ul style="list-style-type: none"> image size image content region_start region_size 	<ul style="list-style-type: none"> length of binary file brief content of binary file start address of flash region to be loaded size of flash region to be loaded
LoadOtpNvds	<ul style="list-style-type: none"> translate to openocd loadOtpNvds command 	<ul style="list-style-type: none"> image size image content 	<ul style="list-style-type: none"> length of binary file brief content of binary file
EraseFlash	<ul style="list-style-type: none"> translate to openocd eraseFlash command 	<ul style="list-style-type: none"> region_size address base_address 	<ul style="list-style-type: none"> size of flash region to be erased address of flash region to be erased base address of flash
CmdExtend	<ul style="list-style-type: none"> extend with text or script or binary file 	<ul style="list-style-type: none"> image size image content type extrainfo 	<ul style="list-style-type: none"> length of binary file brief content of binary file image type description for this field

Table 4 - Field Name and Argument Description

4.5 burn_archive

The burn_archive argument programs the archived file into the target storage. The atm_isp tool is included with the SDK and is called when running the make burn_archive command. CLI can be used to program an archive file by directly calling the atm_isp tool from the tools directory (see below).

Also, the GUI-based atm_isp tool can be used to program archive files without the tools provided in SDK. See isp_tool_gui provided in the customer portal.



```
make burn_archive // programs arch.atm into storage
```

Defining the makefile variable BURN_ARCH_VERIFY tells the ISP tool to verify images after loading them on the device.

BURN_ARCH_DEBUG makes the tool more verbose.

If there is OTP data available, setting BURN_ARCH_ERASE_WORKAROUNDS will erase any unlocked workaround tags (0xfc, 0xfd, and 0xfe) from OTP before burning the new contents.

Finally, defining BURN_ARCH_PROGRAM_ONLY prevents reset-hard-on-exit; it can be used to get a behavior similar to make program_all as opposed to make run_all.

Using atm_isp tool CLI:

Create atm file:

```

$cd <SDK>/tools
../atm_isp init -h
usage: atm_isp init [-h] [-o NEW_ARCHIVE] [-t] family name board

```

positional arguments:

```

family          Platform family (e.g. atm2)
name            Full platform name (e.g. ATM2xxx-x0x)
board          Full board name (e.g. M2221, 3330e_QN, 3325_LQK)

```

optional arguments:

```

-h, --help          show this help message and exit
-o NEW_ARCHIVE, --output NEW_ARCHIVE
                    Output archive file (default: stdout)
-t, --ota           Support OTA

../atm_isp loadRram -h
usage: atm_isp loadRram [-h] [-i ARCHIVE] [-o NEW_ARCHIVE] [-v]
                    [-mpr_start MPR_START] [-mpr_size MPR_SIZE]
                    [-mpr_lock_size MPR_LOCK_SIZE] [-extrainfo
EXTRAINFO]
                    image [region_start] [region_size] [address]

```

positional arguments:

```

image          Path to image
region_start   Start address of flash region to erase
region_size    Size of flash region to erase
address        Address where image should be loaded

```

optional arguments:

```

-h, --help          show this help message and exit
-i ARCHIVE, --input ARCHIVE
                    Input archive file
-o NEW_ARCHIVE, --output NEW_ARCHIVE
                    Output archive file (default: stdout)
-v, --verbose       increase output verbosity
-mpr_start MPR_START, --mpr_start MPR_START
                    MPR_START
-mpr_size MPR_SIZE, --mpr_size MPR_SIZE
                    MPR_SIZE
-mpr_lock_size MPR_LOCK_SIZE, --mpr_lock_size MPR_LOCK_SIZE
                    MPR_LOCK_SIZE
-extrainfo EXTRAINFO, --extrainfo EXTRAINFO
                    extra infomation

```

Example:

```

../atm_isp init -o Demo.atm atm33 ATM33xx-5 ATMEVK_3330e_QN
../atm_isp loadRram Demo.bin 0x63000 0x1000 0x10000000 -extrainfo DEMO -i
Demo.atm -o Demo.atm

```

Burn atm file:

```

$cd <SDK>/tools
$ ./atm_isp burn -h
usage: atm_isp burn [-h] [-i ARCHIVE] [-r OPENOCD_PKG_ROOT] [-E] [-e] [-v]
                  [-c] [-t TCL_SCRIPT] [-d DST_DIR] [-p]

optional arguments:
  -h, --help                show this help message and exit
  -i ARCHIVE, --input ARCHIVE
                           Input archive file
  -r OPENOCD_PKG_ROOT, --openocd_pkg_root OPENOCD_PKG_ROOT
                           Path to directory where openocd and its scripts are
                           found
  -E, --openocd_script_only
                           Stop after preparing OpenOCD script
  -e, --erase_workarounds
                           Erase workaround tags in OTP before loading OTP
  -v, --verbose              Verbose mode
  -c, --check_image         Verify OTP/flash image after burning/loading
  -t TCL_SCRIPT, --tcl_script TCL_SCRIPT
                           Path to output Jim Tcl script for use by OpenOCD
                           (generates Jim Tcl script only; delays all
                           operations
                           post-unpacking of archive to Tcl/OpenOCD); implies
  -E
  -d DST_DIR, --dst_dir DST_DIR
                           Use this directory to dump openocd script in;
                           implies
                           -E
  -p, --program_only        Program the device only (no reset hard on exit)

```

Example:

```

$ ./atm_isp burn -c -v -i <archive.atm>

```

4.6 ATM2/ATM3 Platform ROM and OTP Application

The OTP_beacon directory provides an example of this mode of operation. A number of pre-configured beacon examples are provided in:

platform/atm[2,3]/ATM<part>/examples/OTP_beacon/reference_beacons.mk

One can be selected by setting the REF_BCN makefile variable when building NVDS.

For example:

```
make REF_BCN:=LR_coded rebuild_otp_nvds -> rebuild OTP based NVDS +  
application  
make push_otp_nvds -> store into OTP
```

4.6.1 ROM and Custom Application

The OTP is relatively small (4 KB), but that can be enough for certain custom solutions that want to avoid using FLASH. A special NVDS tag can be programmed into OTP that the ROM will copy into RAM and execute.

The tmp1075_sensor_adv directory shows how small applications can be rapidly developed using FLASH and then deployed to OTP.

Workflow for debug application and NVDS in flash:

```
make run_all
```

Deployment of optimized application and NVDS to OTP:

```
make pull_otp_nvds  
make rebuild_otp_nvds  
make push_otp_nvds
```


4.6.2 ROM and Flash Application

In this mode, the ROM and an application on FLASH cooperate to share the chip resources. This allows extremely compact yet full Bluetooth applications to be developed using APIs in ROM.

BLE_adv, BLE_scan, and HCI all use this mode of operation. See [Make Options and Commands](#) section for the list of make commands.


4.6.3 Flash Operating System

In this mode, the ROM acts as a first-stage bootloader for a fully formed operating system on the FLASH. The ROM will configure enough modules to discover the FLASH operating system and map it over the top of the ROM at address 0x00000000.

The image on FLASH begins with the ARM stack pointer, reset vector, and other exception vectors. When the ROM hands execution over to this reset vector, the OS is free to use all chip resources.

By default, the above goals build for and program to the flash on the target. The application will access read-only code and data in place from the flash. For certain applications, this can result in undesirable power consumption and performance.

The AVOID_XIP:=1 option can be specified to locate said code and data in RAM instead and to have it copied from flash at boot time. Be sure to clean the workspace when switching this option:



```
make clean
make AVOID_XIP:=1 run
```

The makefiles, linker scripts, and openocd scripts default to utilize an entire 4 Mb flash device with the last 32 KB reserved for NVDS.

The FLASH_SIZE and NVDS_SIZE makefile variables can be set on the command line or at the top of the example makefile to accommodate different layouts.

Be sure to clean the workspace when switching either of these options. For example, to build an image suitable for OTP partition programming on a 4 Mb device:

```
make clean
make FLASH_SIZE:=0x40000
```

To avoid the use of flash entirely, the `RUN_IN_RAM:=1` option can be specified. This is particularly useful for the HCI example in order to perform qualification tests on flashless designs.

As above, be sure to clean the workspace when switching between flash and RAM workflows:

```
make clean
make RUN_IN_RAM:=1 run
```

To run the system entirely out of flash, `FLASHROM=ble-full-all` option can be specified. When using this option, the ROM acts only as a boot loader.

Advantages when using FLASHROM build:




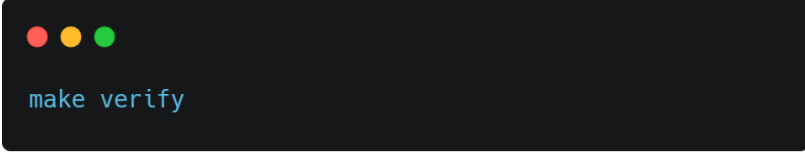
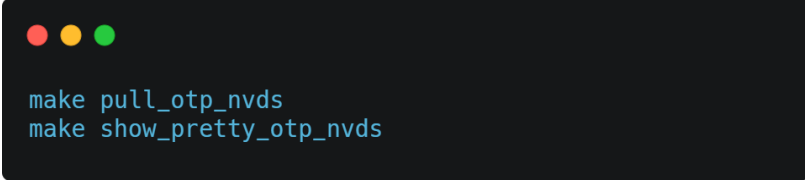
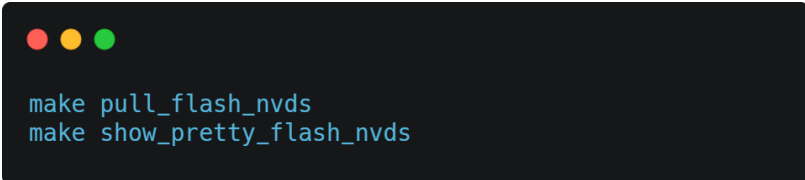
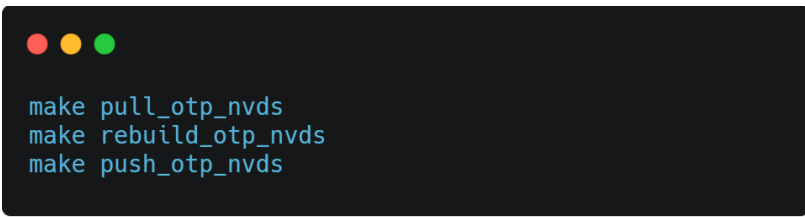
- Most up-to-date Bluetooth LE stack and HW workarounds.
- Bluetooth LE stack can be optimized for specific applications.
- RAM partitioning is not needed.

Disadvantages when using FLASHROM build:

- Increased application binary size.
- Increased power consumption due to more flash accesses.
- Some RAM is used for timing critical code sections.

```
make clean
make FLASHROM:=ble-full-all run
```

5. Make Options and Commands

Make Option	Command
Clean build artifacts	 <pre>make clean</pre>
Build flash application	 <pre>make</pre>
Build flash application, program hardware, but leave ARM M0 CPU halted	 <pre>make program</pre>
Build flash application and compare against flash on target	 <pre>make verify</pre>
Inspect OTP NVDS from device	 <pre>make pull_otp_nvds make show_pretty_otp_nvds</pre>
Inspect flash-based NVDS	 <pre>make pull_flash_nvds make show_pretty_flash_nvds</pre>
Update OTP NVDS on device	 <pre>make pull_otp_nvds make rebuild_otp_nvds make push_otp_nvds</pre>

Program new flash-based NVDS

```
● ● ●  
make build_flash_nvds  
make push_flash_nvds
```

Update flash-based NVDS

```
● ● ●  
make pull_flash_nvds  
make rebuild_flash_nvds  
make push_flash_nvds
```

Build flash NVDS, push, build app, and program

```
● ● ●  
make program_all
```

Build flash NVDS, push, build app, and run

```
● ● ●  
make run_all
```

Collect core dump for offline analysis

```
● ● ●  
make pull_core
```

Build archive with protected MP region

```
● ● ●  
make build_archive MPR_SIZE=0x1000 MPR_LOCK_SIZE=0x1000
```

Table 5 - Make Options and Commands

6. Building Beacon Application Using Atmosic SDK

6.1 ATM2 and ATM3 Device

At MSYS2 shell, go to the BLE_adv example folder to compile the project through the following command.

6.1.1 Board Configuration

The examples that depend on the pinmux driver directly or indirectly (via the GPIO, I2C, SPI, keyboard, PDM, GADC drivers) will need to specify the board for which the example is being built.


ATM22xx-x1x/ATM32xx-x1x device: ATMEVK_M2202, ATMEVK_M2221, ATMEVK_M3202, ATMEVK_M3221

6.1.2 Beacon

Beacon configuration: EddyStone, Non Connectable, Scannable, 100 ms Interval.

```
$ cd platform/atm2/ATM22xx-x1x/examples/BLE_adv  
$ make BOARD=<chip>
```

If everything is correct, the `make` should execute without error as shown below:



```
user@laptop MINGW32 /c/AtmosicSDK/x.y.z  
$ cd platform/atm2/ATM22xx-x1x/examples/BLE_adv  
  
user@laptop MINGW32 /c/AtmosicSDK/x.y.z/platform/atm2/ATM22xx-  
x1x/examples/BLE_adv  
$ make BOARD=ATMEVK_M2201
```

```

user@labtop MINGW32 /c/AtmosicSDK/x.y.z/platform/atm2/ATM22xx-x1x/examples/BLE_adv
$ ls
BLE_adv.asm      atm_gap.o      pinmux.o
BLE_adv.bin     atm_gap_param.d pmu.d
BLE_adv.c       atm_gap_param.o pmu.o
BLE_adv.d       atm_pm.d       porting_ble.d
BLE_adv.elf     atm_pm.o       porting_ble.o
BLE_adv.h       ble_gap.d      reference_beacons.mk
BLE_adv.map     ble_gap.o      reset.d
BLE_adv.o       ble_module.d   reset.o
README          ble_module.o   rf.d
ROM_errata_10.d ble_task.d     rf.o
ROM_errata_10.o ble_task.o     rom.elf
ROM_errata_14.d brwnout.d      rom.o
ROM_errata_14.o brwnout.o      sw_timer.d
ROM_errata_16.d config_adv_params.h sw_timer.o
ROM_errata_16.o ext_flash.d    swd_dbg.d
arm_traceback.d ext_flash.o    swd_dbg.o
arm_traceback.o flash_nvds.bin tag_data
atm_adv.d       hardfault_handler_armv6m.d timer.d
atm_adv.o       hardfault_handler_armv6m.o timer.o
atm_adv_param.d heap.d         user_debug.d
atm_adv_param.o heap.o        user_debug.o
atm_asm.d       hw_cfg.d      user_init.d
atm_asm.o       hw_cfg.o      user_init.o
atm_ble.d       interrupt.d    user_startup_CMSDK_CM0.o
atm_ble.o       interrupt.o    watchdog.d
atm_debug.d     makefile      watchdog.o
atm_debug.o     partition_info.map
atm_gap.d       pinmux.d

```

At MSYS2 shell, type `make BOARD=<chip> run_all` to build and load firmware into Atmosic EVK.

Shell and debug terminals after the `make BOARD=<chip> run_all` command is successfully executed.

```
@00000036 SDK Version: x.y.z
@0000008c APP Version: 0.0.0.9
6x6 EXT_FLASH: 4e 56 44 53 06 06 27 00 00 00 00 01 02 00 00 00 ...
@000001de boot_status = 1000000
@00000239 Cold boot
@00000274 [ BLE_adv][D]: user_main() done
@00000344 [ BLE_adv][D]: ble_adv_init: NVDS tag for adv timeout param not found. Using default
@000004f9 [ atm_adv][D]: Adv dur 0(unit:10ms) max_adv_evt 0 (timeout 0ms)
@00000608 [ BLE_adv][D]: adv_state = 2
@000006b6 [ BLE_adv][D]: adv_state = 4
@00000758 [ BLE_adv][D]: adv_state = 6
@000007e8 Optimize HW Scan: Active=1
@0000086a [ atm_adv][D]: Adv0: ON
@000008eb [ BLE_adv][D]: adv_state = 9
```


Detailed documentation for Beacon and other examples in SDK is available in `platform/atm2/ATM2xx-x1x/examples/<example_name>/README` file.

7. Building Energy Harvesting Beacon

7.1 ATM3 Devices

At MSYS2 shell, go to the BLE_harv_adv example folder to compile the project through the following command.

Beacon configuration: EddyStone, Non Connectable, Scannable, 1s Interval.

A terminal window with a black background and three colored window control buttons (red, yellow, green) in the top-left corner. The terminal displays two lines of text: a prompt followed by a directory change command and a make command with environment variables.

```
$ cd platform/atm3/ATM32xx-x1x/examples/BLE_harv_adv/  
$ make BOARD=<chip> DEBUG= run_all
```


8. Uninstall Atmosic SDK

Follow the steps to uninstall Atmosic SDK. Also, see [Figure 15](#).

- 1) Click the Start Menu and Settings.
- 2) In the Settings page, click the Apps item.
- 3) Find the Atmosic SDK (<version_number>) item in the Apps & features page.
- 4) Click the Uninstall button to start the Atmosic SDK uninstallation process.
- 5) The Atmosic RDI Interface will be removed during the uninstallation process. Please remove the Atmosic EVB before reinstalling the Atmosic SDK again.

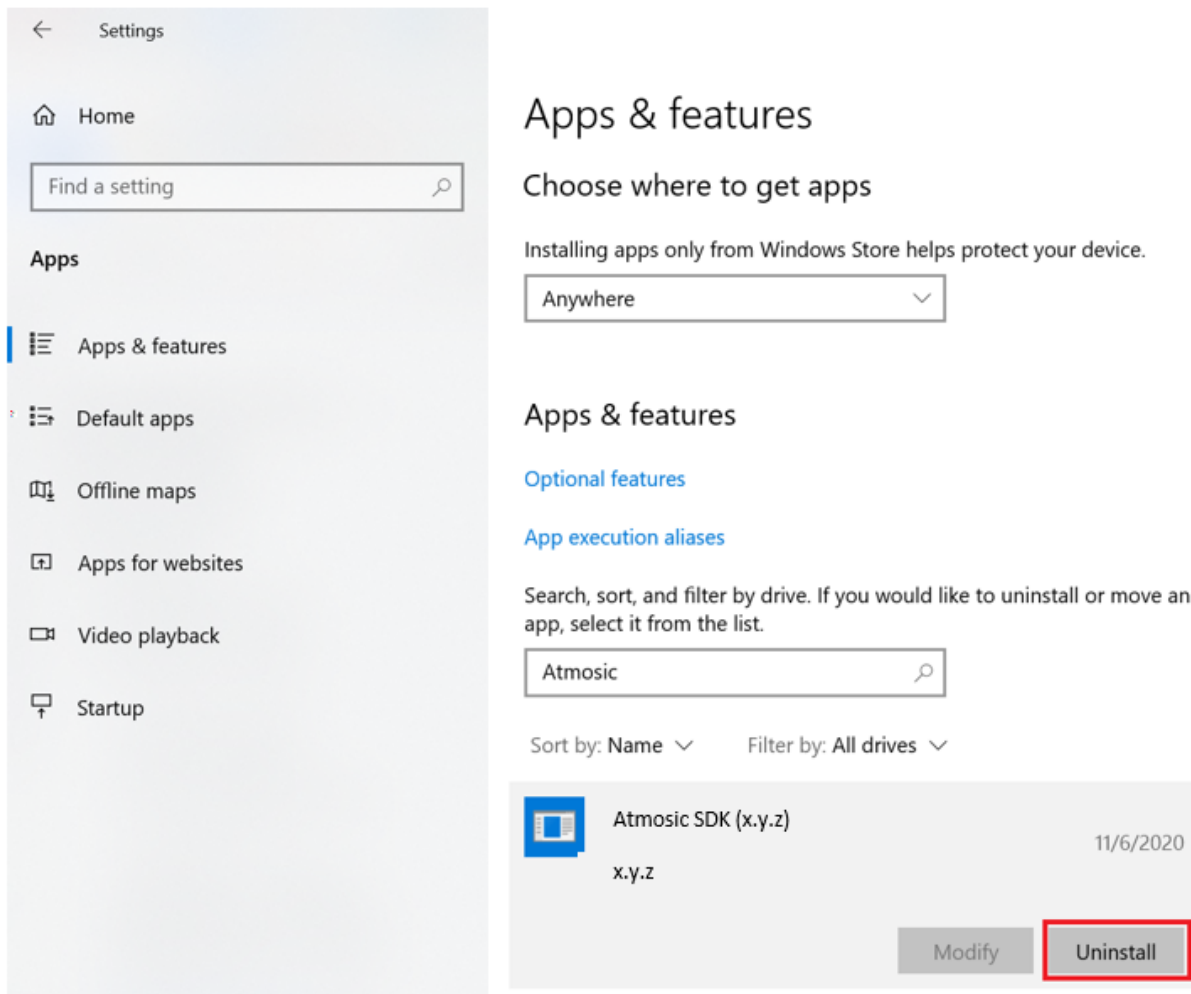


Figure 15 - Uninstall SDK

Reference Documents

Title	Document Number
ATM2/ATM3 EVK Power Consumption Evaluation User Guide	ATM2_ATM3-UGPCE
ATM2/ATM3 EVK User Guide	ATM2_ATM3-UGEVK
ATM32xx Energy Harvesting Quick Start Guide	ATM32xx-QSGEHV

Revision History

Date	Version	Description
October 27, 2023	0.1	Initial release



ATMOSIC TECHNOLOGIES – DISCLAIMER

This product document is intended to be a general informational aid and not a substitute for any literature or labeling accompanying your purchase of the Atmosic product. Atmosic reserves the right to amend its product literature at any time without notice and for any reason, including to improve product design or function. While Atmosic strives to make its documents accurate and current, Atmosic makes no warranty or representation that the information contained in this document is completely accurate, and Atmosic hereby disclaims (i) any and all liability for any errors or inaccuracies contained in any document or in any other product literature and any damages or lost profits resulting therefrom; (ii) any and all liability and responsibility for any action you take or fail to take based on the information contained in this document; and (iii) any and all implied warranties which may attach to this document, including warranties of fitness for particular purpose, non-infringement and merchantability. Consequently, you assume all risk in your use of this document, the Atmosic product, and in any action you take or fail to take based upon the information in this document. Any statements in this document in regard to the suitability of an Atmosic product for certain types of applications are based on Atmosic's general knowledge of typical requirements in generic applications and are not binding statements about the suitability of Atmosic products for any particular application. It is your responsibility as the customer to validate that a particular Atmosic product is suitable for use in a particular application. All content in this document is proprietary, copyrighted, and owned or licensed by Atmosic, and any unauthorized use of content or trademarks contained herein is strictly prohibited.

Copyright ©2022-2023 by Atmosic Technologies. All rights reserved. Atmosic logo is a registered trademark of Atmosic Technologies Inc. All other trademarks are the properties of their respective holders.

Atmosic™

Atmosic Technologies | 2105 S. Bascom Ave. | Campbell CA, 95008

www.atmosic.com