STM32-AI RELOCATABLE with ThreadX Module Manager and related Module application

**TABLE OF CONTENTS**

[1. PURPOSE & SCOPE 2](#_Toc107578573)

[1.1. Purpose 2](#_Toc107578574)

[1.2. Scope 2](#_Toc107578575)

[2. Getting started 2](#_Toc107578576)

[2.1. What this demo does 2](#_Toc107578577)

[2.2. The Demo architecture 2](#_Toc107578578)

[3. demo compilation 4](#_Toc107578579)

[3.1. Demo Unzip and build 4](#_Toc107578580)

[4. demo execution 6](#_Toc107578581)

[4.1. Resident Module Manager application flashing 6](#_Toc107578582)

[4.2. FTP Server and Tera Term setup 6](#_Toc107578583)

[5. demo CUSTOMIZATIONs 8](#_Toc107578584)

[5.1. WiFi customization 8](#_Toc107578585)

[5.2. Module “in place or from file execution” 9](#_Toc107578586)

[5.3. File System backend customization 9](#_Toc107578587)

[6. Conclusions 10](#_Toc107578588)

[6.1. The Modules advantages 10](#_Toc107578589)

[7. REFERENCES & DEFINITIONS 10](#_Toc107578590)

[7.1. References 10](#_Toc107578591)

[7.2. Acronyms & Definitions 10](#_Toc107578592)

[Appendix: Troubleshuttings and hints 11](#_Toc107578593)

# PURPOSE & SCOPE

## Purpose

This Application Note demonstrates how to build and run a sample application exploiting the STM32-AI RELOCATABLE running in a ThreadX thread together with the AZURE RTOS ThreadX Module Manager functionality.

To add the connectivity and file system capabilities the NetXDuo TCP/IP network stack and the FileX embedded file system middleware was also included.

This application was developed starting from the original application part of the STM32 U5 FW official delivery that can be found in \STM32Cube\_FW\_U5\_V1.1.0\Projects\NUCLEO-U575ZI-Q\Applications\ThreadX\Tx\_MPU.

The application shows how to download from and FTP server, and run within a thread, an STM32-AI RELOCATABLE network and weights generated as binary files. This allows to update “on the flight” the AI network and/or the related weights.

Is also shown the Module Manager capability to download from an FTP server, as a binary file, the Module and to execute it from memory and from file, other than simple “execution in place”.

## Scope

This Application Note is specifically built for the STM32 U585 processor mounted on the evaluation board MB1551-U585AI-C02 for more information please visit <https://www.st.com/en/evaluation-tools/b-u585i-iot02a.html>.

The IDE and compile used is IAR EWARM for more information please visit https://www.iar.com/products/architectures/arm/iar-embedded-workbench-for-arm/

The current application doesn’t demonstrate any security/encryption aspect of the U5 MCU.

## Background

The STM32Cube.AI is a set of tools capable to generate a pre-trained relocatable (PIC & PID) AI network with its related weights both generated in the form of relocatable binary files.

The AI binary files (network and weights) are generated thanks to the stm32ai Command Line Interface (CLI) starting from an .h5 network description file.

The ThreadX Module is a Position Independent Code and Data library (aka PIC/PID) built as an independent single binary file. It has the capability to communicate and exchange information with the main resident application running the ThreadX Module Manager. The Module can be un/loaded at run time by the resident application.

For more information about STM32Cube.AI please visit <https://stm32ai.st.com/stm32-cube-ai/>

For more information about Microsoft ThreadX RTOS please visit <https://docs.microsoft.com/en-us/azure/rtos/threadx/>

For more information about Microsoft ThredX Module feature please visit https://docs.microsoft.com/en-us/azure/rtos/threadx-modules/

# Getting started

## What this demo does

The demo, after start first

* try connecting to the WiFi network with the provided credentials (network SSID and password)
* than after having acquired from the router its IP address through the DHCP protocol, it tries to connect to SNTP server in order to acquire the current GMT time and set the correct system time
* after that it format the NOR Flash File System (if not already done previously)
* than it contacts the indicated FTP server IP address and connect it with the user provided credentials to download the files containing:
  + The AI Network
  + The AI Weights
  + The Module library
* If the AI network and weights files are downloaded successfully both are stored in the file system overriding the former, if already existing, then are flashed. After that AI starts executing in its own thread.
* If the FTP server cannot be reached the application look for the AI Network and weights in the local file system, if found, both the files are flashed in internal flash memory and executed.
* if the Module file is downloaded successfully, the Module file already present on NOR Flash File System (if any) is overridden by the just downloaded file
* if for any reason the Module file download fails (typically because of wrong FTP credential/IP or missing file on the FTP server) the demo runs automatically the Module file already present on Flash File System, if any was previously downloaded.
* The Module sends through a queue some messages to the resident application and finally it deliberately executes a memory access violation triggering an exception in the resident application.
* After having completed the Module execution and having released the related Module resources the demo prints periodically every 10Sec the current system time (GMT) and the green user led blinks every 500mSec
* The AI continues to run every 1 Second processing a fixed pattern of data and checking the output. Is also possible to compile the application to process a random input pattern.

## The Demo architecture

The Module Manager resident application is split over 4 threads plus 1 thread for the Module:

1. The NetX\_Thread\_Entry () receiving the IP address change notification and acting as a trampoline for the other application threads.
2. The App\_SNTP\_Thread\_Entry() connecting the Network Time Protocol server getting the GMT time and setting the system time to RTC. The thread, after having accomplished the SNTP synchronization signal the thread ModuleManager\_Entry() and enter in a loop displaying periodically the current GMT date and time every 10 Sec.
3. The ModuleManager\_Entry(), this thread after having allocated the resources needed by the Module Manager and the related Module, wait for the semaphore signaled by the App\_SNTP\_Thread\_Entry(). After the semaphore is signaled, meaning IP address and NTP time acquired, it initializes the file system (formatting the disk if needed), contact the FTP server, download the Module and execute it.
4. Module MainThread\_Entry() is the thread executed within the Module memory space when the Module is started by the Module Manager resident application thread ModuleManager\_Entry().
5. AIThreadEntry is the thread executing the AI Network, it is started by the ModuleManager\_Entry -acting as a “trampoline” thread- after having downloaded the Network and the Weights binary files if available on FTP server.

IP Address

Start SNTP thread and self-terminate

print time

NTP time

Signal semaphore

txm\_module\_manager\_start

Toggle Green LED

Queue

tx\_thread\_resume

AI Endless execution

Self-terminate

**Required Hardware**

* [B-U585I-IOT02A](https://www.st.com/en/evaluation-tools/b-u585i-iot02a.html) Rev C (code MB1551-U585AI-C02)
* Usb 2.0 male to Micro USB male cable

**Required software**

* Tera Term terminal
* [STM32CubeProgrammer](https://www.st.com/en/development-tools/stm32cubeprog.html) (V2.8..0 or later)
* IAR Embedded Workbench for ARM (V.8.50.6 or later)
* Patch for U5 support (v.8.50.8 or later) for IAR EWARM
* [ST Link USB Driver](https://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-utilities/stsw-link009.html)
* [Universal FTP Server](https://apps.microsoft.com/store/detail/universal-ftp-server/9NQKQ104HB9R?hl=en-us&gl=US)

# demo compilation

## Demo Unzip and build

After having unzipped the files, (preferably on C: to avoid long path issue) execute the below steps:

1. Double click on the master EWARM project

Graphical user interface, application, Teams

Description automatically generated

**In order to connect to the available WiFi network please enter the SSID and password in the file “mx\_wifi\_conf.h”**

#define WIFI\_SSID "MyWiFiSSID"

#define WIFI\_PASSWORD "MyWiFiPassword"

**In order to connect to the FTP server please take note from the FTP server application the hosting machine IP address and set the FTP username and password. The same IP address, username and password must be set in file “app\_netxduo.c”**

#define FTP\_SERVER\_ADDRESS IP\_ADDRESS(192,168,1,10)

#define FTP\_ACCOUNT "MyFTPAccount"

#define FTP\_PASSW "MyFTPPassword"

In case of need the module file name can be changed in file “app\_threadx.h” as below:

#define MODULE\_FILE\_NAME "Module.bin"

The AI Network and weights files name can be changed in the same file as below:

#define AI\_NETWORK\_FILE\_NAME "network\_rel.bin"

#define AI\_WEIGHTS\_FILE\_NAME "network\_data.bin"

The pre-generated AI Network files< network\_rel.bin > and < network\_data.bin >are in:

<….\DEMO\_AI\_RELOCATABLE\_THREADX\X-CUBE-AI>

After build the generated Module binary files is in:

<DEMO\_AI\_RELOCATABLE\_THREADX\Projects\B-U585I-IOT02A\Applications\ThreadX\Tx\_MPU\_LevelX\_NetX\_Module\EWARM\Tx\_Module\Exe>

After having set the above parameters in the src files, from the EWARM project press F8 key or select from textual menu Project->BatchBuild. This will build both the projects “Tx\_Module\_Manager” and “Tx\_Module”

Graphical user interface

Description automatically generated

After build two binary files will be produced:

* Tx\_Module\_Manager.out containing the resident application to be flashed on the board

Graphical user interface, application

Description automatically generated

* Tx\_Module.bin containing the Module code in form of PIC & PID (aka Position Independent Code/Data) library. This file needs to be copied on the external FTP server.

A picture containing timeline

Description automatically generated

In case of need the relocatable network files can be re-generated thanks to stm32ai CLI command tool with the command below launched from the power shell (please notice that the STM32AI tools should be installed and the PATH of <stm32ai> should be set)

<stm32ai generate -m .\vibration\_model\_mfcc.h5 -c none --series stm32u5 --binary --relocatable --no-c-files --allocate-inputs --allocate-outputs -o .\prv\_no\_src -n network>

# demo execution

## Overall, Hardware setup

A picture containing electronics, router

Description automatically generated

A close-up of a circuit board

Description automatically generated with medium confidence A picture containing text, computer, electronics, indoor

Description automatically generated

* The U5 board
* a laptop running FTP server, IAR EWARM and TeraTerm
* A router (possibly) with internet access

## Resident Module Manager application flashing

After having selected the project Tx\_Module\_Manager press Ctr+D the debugger will start, and the resident application will be flashed.

Graphical user interface, application

Description automatically generated

## FTP Server and Tera Term setup

Before to run the Module Manager resident application, it needs to setup the FTP server setting the FTP account and password and coping the Tx\_Module.bin file in the FTP server directory.

It needs also to connect the TeraTerm to the board in order see the application execution log. The serial speed 115200 bps

Graphical user interface, application

Description automatically generated

Graphical user interface

Description automatically generated

After having setup the FTP server and the TeraTerm the Module Manager application can be run and the below execution log should be shown on TeraTerm

Please notice that the COMn port number is self-assigned by widows USB enumerator so the port number can vary.

Text

Description automatically generated

Text

Description automatically generated

# demo CUSTOMIZATIONs

## WiFi customization

The demo offer two pre-defined wifi setup to connect to two different networks, typically the home WiFi and the work WiFi.

One of the two pre-defined networks can be chosen by means of definition of the compilation environmental variable HOME\_WIFI

Graphical user interface, application

Description automatically generated

## Module “in place or from file execution”

The demo by default is configured in order to download the file containing the module from an FTP server, than to execute it from SRAM. The demo can be also configured in order to execute the Module from SRAM, after having loaded it from a Flash, or to execute the Module directly from the Flash memory address where it was flashed (aka “execution in place”)

If the definition:

#define MODULE\_LOAD\_FROM\_FILE

In file app\_threadx.h is defined the Module is loaded from the file previously downloaded from FTP, otherwise the Module is loaded from the Flash address specified at

#define MODULE\_FLASH\_ADDRESS &module\_entry // from linker script

## File System backend customization

The demo can be customized to use as a File System backend driver the RAM driver running the FS on the internal SRAM. Or, by default, the demo run the FS on NOR Flash.

Default definition in app\_filex.h:

#define NOR\_FLASH\_FS

# Conclusions

## The Modules advantages

As seen above the Modules offer a variety of advantages in terms of application upgrade and update, possibility to add new features on the flight etc. This could be particularly useful for IoT connected applications.

In the current demo all the security aspects are not addressed, showing just a simple module memory isolation using MPU. The security aspects involve additional HW and FW and also third-parties for authentication, they can be the subject of a separate demo.

## The STM32AI Relocatable advantages

As demonstrated by the demo the STM32AI network compiled in form of self-contained binary files offer the below listed advantages in comparison with the traditional embedded AI approach:

* **Upgradability**: the AI network and weights can be updated on the flight
* **Memory optimization**: the AI network and weights can be placed or moved to any address in memory according to the application needs. In case of need the AI can be also offloaded from internal memory and stored in the file system for a later reuse

# REFERENCES & DEFINITIONS

## References

### External References

|  |  |
| --- | --- |
| Definition | Link |
| *ThreadX Module doc* | [*https://docs.microsoft.com/en-us/azure/rtos/threadx-modules/chapter1*](https://docs.microsoft.com/en-us/azure/rtos/threadx-modules/chapter1) |
| *Module Preamble ARM M33* | *https://docs.microsoft.com/en-us/azure/rtos/threadx-modules/appendix#cortex-m33-processor* |
| *IAR IDE and Compiler toolchain* | *IAR IDE and Compiler toolchain https://www.iar.com/products/architectures/arm/iar-embedded-workbench-for-arm/* |
| *B-U585I-IOT02A STM32 board* | *https://www.st.com/en/evaluation-tools/b-u585i-iot02a.html* |
| *STM32CubeProgrammer* | [*https://www.st.com/en/development-tools/stm32cubeprog.html*](https://www.st.com/en/development-tools/stm32cubeprog.html) |
| *Window FTP Server* | [*https://www.windowscentral.com/how-set-ftp-server-windows-10*](https://www.windowscentral.com/how-set-ftp-server-windows-10) |
| *Tera Term* | *https://tera-term.en.softonic.com/* |

### Internal References

## Acronyms & Definitions

### Acronyms

<List the acronyms used in your document in the table below. Refer users to the ST [Glossary](https://stmicroelectronics.sharepoint.com/sites/Glossary) for further common definitions. Another useful approach for readers is to define an acronym the first time it is used. >

|  |  |
| --- | --- |
| Acronym | Definition |
| *PIC* | *Position Independent Code* |
| *PID* | *Position Independent Data* |
| *EWARM* | *IAR IDE and Compiler toolchain* |
| *DHCP* | *Dynamic Host Configuration Protocol* |
| *DNS* | *Domain Name System* |
| *SNTP* | *Simple Network Time Protocol* |
| *FTP* | *File Transfer Protocol* |
| *ELF* | *Executable and Linkable Format* |

.

### Definitions

|  |  |
| --- | --- |
| **Term** | **Definition** |
|  |  |

Appendix: Troubleshuttings and hints

1. **WiFi connection and router IP address reservation**:

It is advisable to setup the router in order to reserve an IP address for the machine hosting the FTP server, doing so will prevent the DHCP server running on the router to change dynamically the host PC and board IP address.

1. **NOR Flash File System corruption**:

In case of abrupt interruption of the application (break or abort from IDE or HW reset) with opened files it can happen that the NOR flash file system gets corrupted, the application try to fix it automatically formatting the NOR if needed. In the unlikely event of FS corruption not detected by the application the FS formatting can be forced modifying the file app\_filex.c

1. **Module direct flashing using EWARM IDE**:

If, for any reason, the usage of the FTP server is not possible the Module can be flashed directly on the board by the EWARM IDE clicking on the check box “Download extra image” and specifying the **Module image ELF file** (not the binary) as shown below

Graphical user interface, application

Description automatically generated

The module will be flashed at the address 0x8080000 as specified in the Module linker script file “stm32u575xx\_flash\_module.icf”



So that the Module will be flashed by the EWARM IDE together with the resident Module Manager application. As shown below the Module signature pattern “55 44 4f 4d” indicating the first bytes of the Module preamble is placed at flash address 0x8080000.

Once the module is in flash memory it can be executed in place by means of calling the “txm\_module\_manager\_in\_place\_load()” commenting in file app\_threadx.h the

#define MODULE\_LOAD\_FROM\_FILE

Alternatively it can be loaded and executed in SRAM by means of calling “txm\_module\_manager\_memory\_load()” by means of commenting in file app\_threadx.h the

#define MODULE\_EXECUTE\_IN\_PLACE

Calendar

Description automatically generated

1. **Module Debugging and step by step execution**:

In order to enable module debugging the debugger needs the below listed informations:

* + module debugging symbols that can be found on the Module ELF file
  + module code offset (typically 0x34) from the module preamble start address (0x8080000)

Because of the above requirements the Module code must be flashed together with the Module Manager application as shown above at point 3, and it must be executed in place to allow Module code debugging (same load and execution address 0x8080000 despite its PIC/PID nature).

When compiling for debugging purposes don’t forget to disable all the Module Manager and the Module compiler optimizations:

Graphical user interface, application

Description automatically generated