AppZone m2mb Sample Apps

80000NT11840A Rev. 0 - 2020-05-05

TELIT
TECHNICAL
DOCUMENTATION

1 AppZone m2mb Sample Apps

Package Version: **1.1.0-C1**

Minimum Firmware Version: **30.00.XX9**

1.1 Features

This package goal is to provide sample source code for common activities kick-start.

2 Quick start

2.1 Deployment Instructions

To manually deploy the Sample application on the devices perform the following steps:

1. Have **30.00.XX9** FW version flashed (`AT#SWPKGV` will give you the FW version)
2. Copy m2mapz.bin to /mod/

   `AT#M2MWRITE="/mod/m2mapz.bin",<size>,1`

   where <size> is in bytes
3. Configure the module to run the downloaded binary as default app: `AT#M2MRUN=2,m2mapz.bin`
4. Restart the module and if no AT commands are sent within **10** seconds, start the app: `AT+M2M=4,10`

2.2 References

More info on

- Getting started with ME910C1 (doc ID 80529NT11661A)
- How to run applications with AppZone

2.3 Known Issues

None

2.4 Contact Information, Support

For general contact, technical support services, technical questions and report documentation errors contact Telit Technical Support at: TS-EMEA@telit.com.

For detailed information about where you can buy the Telit modules or for recommendations on accessories and components visit:

http://www.telit.com

Our aim is to make this guide as helpful as possible. Keep us informed of your comments and suggestions for improvements.

Telit appreciates feedback from the users of our information.

2.5 Troubleshooting

- Application does not work/start:
  - Delete application binary and retry

    ```
    AT#M2MDEL="/mod/m2mapz.bin"
    ```

  - Delete everything, reflash and retry

    ```
    AT#M2MDEL="/mod/m2mapz.bin"
    AT#M2MDEL="/mod/appcfg.ini"
    ```

- Application project does not compile
  - Right click on project name
  - Select Properties
  - Select AppZone tab
  - Select the right plugin (firmware) version
  - Press “Restore Defaults”, then “Apply”, then “OK”
  - Build project again
- Application project shows missing symbols on IDE
  - Right click on project name
  - Select Index
  - Select Rebuild. This will regenerate the symbols index.

2.6 Making source code changes

2.6.1 Folder structure

The applications code follow the structure below:

- `hdr`: header files used by the application
  - `app_cfg.h`: the main configuration file for the application
- `src`: source code specific to the application
- `azx`: helpful utilities used by the application (for GPIOs, LOGGING etc)
  - `hdr`: generic utilities' header files
  - `src`: generic utilities' source files
- `Makefile.in`: customization of the Make process

2.7 Import a Sample App into an IDE project

Consider that the app HelloWorld that prints on Main UART is a good starting point. To import it in a project, please follow the steps below:

On IDE, create a new project: "File"-> "New" -> "Telit Project"

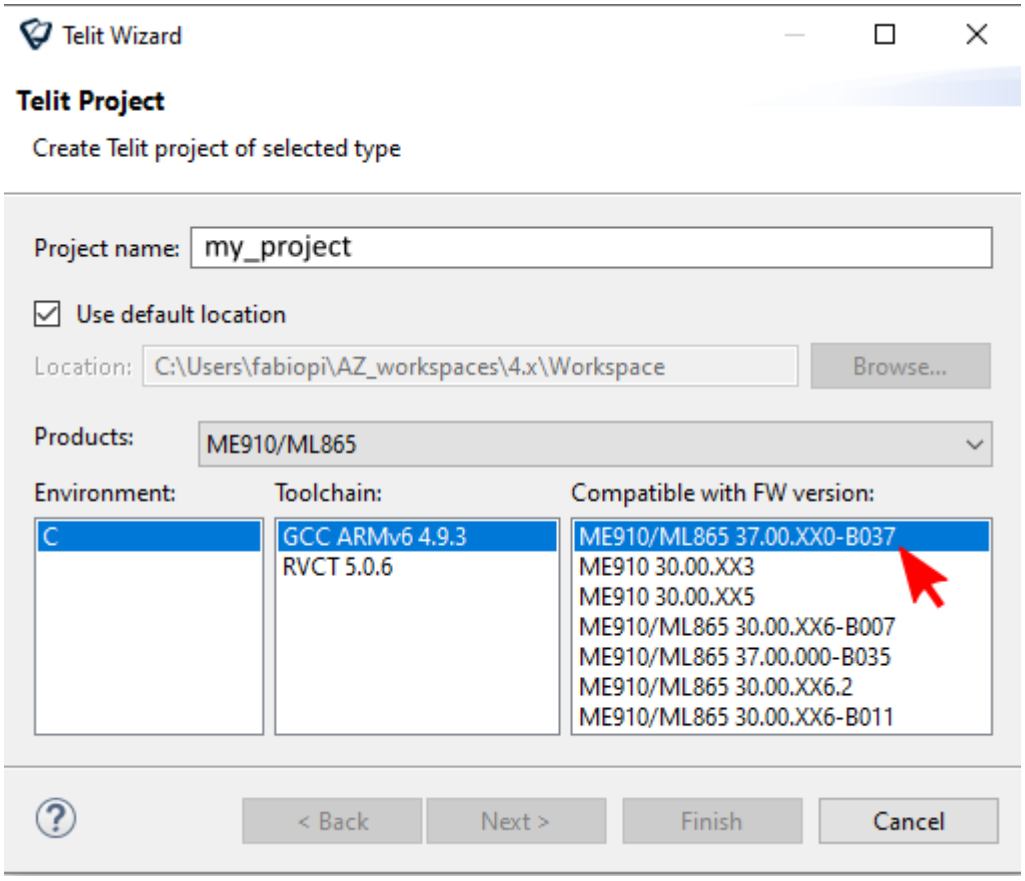

Figure 1

Select the preferred firmware version (e.g. 30.00.xx7) and create an empty project.

in the samples package, go in the HelloWorld folder (e.g. `AppZoneSampleApps-MAIN_UART\HelloWorld`), copy all the files and folders in it (as `src`, `hdr`, `azx`) and paste them in the root of the newly created IDE project. You are now ready tyo build and try the sample app on your device.

Contents

3 Applications

3.1 AUX UART

Applications that provide usage examples for various functionalities, log output on Auxiliary UART

3.1.1 ATI (AT Instance)

Sample application showing how to use AT Instance functionality (sending AT commands from code). The example supports both sync and async (using a callback) modes. Debug prints on **AUX UART**

Features

- How to open an AT interface from the application
- How to send AT commands and receive responses on the AT interface

Application workflow, sync mode

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Init AT0 (first AT instance)
- Send AT+CGMR command
- Print response.
- Release AT0

`at_sync.c`

- Init ati functionality and take AT0
- Send AT+CGMR command, then read response after 2 seconds, then return it
- Deinit ati, releasing AT0

```
Starting AT demo app. This is v1.0.7 built on Apr  1 2020 15:12:58.
[DEBUG] 17.15  at_sync.c:53 - at_cmd_sync_init{M2M_DamsStart}$ m2mb_ati_init() on instance 0
Sending command AT+CGMR in sync mode
[DEBUG] 17.16  at_sync.c:79 - send_sync_at_command{M2M_DamsStart}$ Sending AT Command: AT+CGMR
Command response: <AT+CGMR
M0B.950004-B008

OK
>

[DEBUG] 19.21  at_sync.c:61 - at_cmd_sync_deinit{M2M_DamsStart}$ m2mb_ati_deinit() on instance 0
Application end
```

Figure 2

Application workflow, async mode

M2MB_main.c

- Open USB/UART/UART_AUX
- Init AT0 (first AT instance)
- Send AT+CGMR command
- Print response.
- Release AT0

at_async.c

- Init ati functionality and take AT0, register AT events callback
- Send AT+CGMR command, wait for response semaphore (released in callback), then read it and return it
- Deinit ati, releasing AT0

```
Starting AT demo app. This is v1.0.7 built on Apr  1 2020 15:07:45.
[DEBUG] 17.13  at_async.c:116 - at_cmd_async_init{M2M_DamsStart}$ m2mb_ati_init() on instance 0
Sending command AT+CGMR in async mode
[DEBUG] 17.15  at_async.c:153 - send_async_at_command{M2M_DamsStart}$ Sending AT Command: AT+CGMR
[DEBUG] 17.15  at_async.c:169 - send_async_at_command{M2M_DamsStart}$ waiting command response...
[DEBUG] 17.17  at_async.c:88 - at_cmd_async_callback{pubTspt_0}$ Callback - available bytes: 25
[DEBUG] 17.18  at_async.c:181 - send_async_at_command{M2M_DamsStart}$ Receive response...
Command response: <AT+CGMR
M0B.950004-B008

OK
>

[DEBUG] 17.19  at_async.c:136 - at_cmd_async_deinit{M2M_DamsStart}$ m2mb_ati_deinit() on instance 0
Application end
```

Figure 3

3.1.2 App Manager

Sample application showing how to manage AppZone apps from m2mb code. Debug prints on **AUX UART**

Features

- How to get how many configured apps are available
- How to get the handle to manage the running app (change start delay, enable/disable)
- How to create the handle for a new binary app, enable it and set its parameters
- How to start the new app without rebooting the device, then stop it after a while.

3.1.2.1 Prerequisites

This app will try to manage another app called “second.bin”, which already exists in the module filesystem and can be anything (e.g. another sample app as GPIO toggle). the app must be built using the flag ROM_START=

in the Makefile to set a different starting address than the main app (by default, 0x40000000). For example, 0x41000000.

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- get a non existing app handle and verify it is NULL
- get the current app handle, then get the start delay **set in the INI file (so persistent)**
- change the current app delay value **in the INI file**
- verify that the change has been stored
- get current app state
- create an handle for a second application binary.
- add it to the INI file
- set its execution flag to 0
- get the delay time and the state from INI file for the new app
- get the current set address for the new app
- set the app delay **in RAM, INI will not be affected**.
- start the new app without reboot, using the right set delay
- wait some time, then get the app state and the used RAM amount
- wait 10 seconds, then stop the second app.
- set its execution flag to 1 so it will run at next boot.

```
Starting App Manager demo app. This is v1.0.14-C1 built on Sep 24 2020 12:33:25.
There are 2 configured apps.
Not existing app handle test (should be 0): 0x0
Manager app handle: 0x809e20e0
Manager app delay from nv memory: 5 seconds

Changing Manager app delay time (on non volatile configuration) to 5 seconds..
Manager app delay from nv memory is now 5 seconds
Manager app state is M2MB_APPMNG_STATE_RUN

Trying to get Second app handle...
Second app handle is valid
2nd app delay from nv memory is 1
2nd app current state is M2MB_APPMNG_STATE_READY
Second app current address is 0x41000000
Setting volatile Second app delay (not stored in nvm) to 0 seconds...
Starting Second app on the fly (without reboot)...
Waiting 2 seconds...
2nd app current state is M2MB_APPMNG_STATE_RUN
Second app is running!
Second App is using 475136 bytes of RAM
Stopping Second app now...
wait 10 seconds...
2nd app current state is M2MB_APPMNG_STATE_STOP
Set permanent run permission for Second app.
Done. Second App will also run from next boot-up
```

Figure 4

3.1.3 App update OTA via FTP

Sample application showcasing Application OTA over FTP with AZX FTP. Debug prints on **AUX UART**

Features

- How to check module registration and activate PDP context
- How to connect to a FTP server
- How to download an application binary and update the local version

The app uses a predefined set of parameters. To load custom parameters, upload the `ota_config.txt` file (provided in project's `/src` folder) in module's `/mod` folder, for example with

```
AT#M2MWRITE="/mod/ota_config.txt",<filesize>
```

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage app OTA and start it

`ftp_utils.c`

- Set parameters to default
- Try to load parameters from `ota_config.txt` file
- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Initialize FTP client
- Connect to FTP server and log in
- Get new App binary file size on remote server
- Download the file in `/mod` folder, with the provided name
- Close FTP connection
- Disable PDP context
- Update applications configuration in **app_utils.c**

`app_utils.c`

- Set new application as default

- Delete old app binary
- Restart module

```
Starting FTP APP OTA demo app. This is v1.0.7 built on Apr  7 2020 17:04:05.
[DEBUG] 21.23  ftp_utils.c:447 - msgFTPTask{FTPOTA_TASK}$ INIT
[DEBUG] 21.25  ftp_utils.c:152 - readConfigFromFile{FTPOTA_TASK}$ Reading parameters from file
[DEBUG] 21.26  ftp_utils.c:154 - readConfigFromFile{FTPOTA_TASK}$ Opening /mod/ota_config.txt in read mode..
Set APN to: <<web.omnitel.it>>
Set FTP URL to: <<ftp.telit.com>>
Set FTP PORT to: 21
Set FTP USER to: <<[illegible]>>
Set FTP PASS to: <<[illegible]>>
Set FTP FILE URI to: <</samples/APP_OTA/helloworld.bin>>
Set LOCAL FINAL APP NAME to: <<helloworld.bin>>
Set LOCAL ORIGINAL APP NAME to: <<m2mapz.bin>>
[DEBUG] 23.53  ftp_utils.c:464 - msgFTPTask{FTPOTA_TASK}$ m2mb_os_ev_init success
[DEBUG] 23.54  ftp_utils.c:470 - msgFTPTask{FTPOTA_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.55  ftp_utils.c:478 - msgFTPTask{FTPOTA_TASK}$ Waiting for registration...
[DEBUG] 23.56  ftp_utils.c:371 - NetCallback{pubTspt_0}$ Module is registered to network
[DEBUG] 23.56  ftp_utils.c:491 - msgFTPTask{FTPOTA_TASK}$ Pdp context activation
[DEBUG] 23.57  ftp_utils.c:495 - msgFTPTask{FTPOTA_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 25.61  ftp_utils.c:504 - msgFTPTask{FTPOTA_TASK}$ Activate PDP with APN web.omnitel.it on cid 3....
[DEBUG] 26.30  ftp_utils.c:398 - PdpCallback{pubTspt_0}$ Context active
[DEBUG] 26.30  ftp_utils.c:401 - PdpCallback{pubTspt_0}$ IP address: 176.246.110.148
Start ftp client...
[DEBUG] 27.36  ftp_utils.c:533 - msgFTPTask{FTPOTA_TASK}$ Connected.
[DEBUG] 28.87  ftp_utils.c:546 - msgFTPTask{FTPOTA_TASK}$ FTP login successful.
Get remote file /samples/APP_OTA/helloworld.bin size
[DEBUG] 29.31  ftp_utils.c:568 - msgFTPTask{FTPOTA_TASK}$ Done. File size: 116224.
Starting download of remote file /samples/APP_OTA/helloworld.bin into local /mod/helloworld.bin
/samples/APP_OTA/helloworld.bin  4.68% 5440
/samples/APP_OTA/helloworld.bin  9.36% 10880
/samples/APP_OTA/helloworld.bin 14.04% 16320
/samples/APP_OTA/helloworld.bin 18.72% 21760
/samples/APP_OTA/helloworld.bin 23.40% 27200
/samples/APP_OTA/helloworld.bin 28.08% 32640
/samples/APP_OTA/helloworld.bin 32.76% 38080
/samples/APP_OTA/helloworld.bin 37.44% 43520
/samples/APP_OTA/helloworld.bin 42.13% 48960
/samples/APP_OTA/helloworld.bin 46.81% 54400
/samples/APP_OTA/helloworld.bin 51.49% 59840
/samples/APP_OTA/helloworld.bin 56.17% 65280
/samples/APP_OTA/helloworld.bin 60.85% 70720
/samples/APP_OTA/helloworld.bin 65.53% 76160
/samples/APP_OTA/helloworld.bin 70.21% 81600
/samples/APP_OTA/helloworld.bin 74.89% 87040
/samples/APP_OTA/helloworld.bin 79.57% 92480
/samples/APP_OTA/helloworld.bin 84.25% 97920
/samples/APP_OTA/helloworld.bin 88.93% 103360
/samples/APP_OTA/helloworld.bin 93.61% 108800
/samples/APP_OTA/helloworld.bin 97.42% 113220
[DEBUG] 43.54  ftp_utils.c:608 - msgFTPTask{FTPOTA_TASK}$ download successful.
FTP quit...
[DEBUG] 43.77  ftp_utils.c:632 - msgFTPTask{FTPOTA_TASK}$ Deactivating PDP
[DEBUG] 43.77  ftp_utils.c:642 - msgFTPTask{FTPOTA_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 44.20  ftp_utils.c:407 - PdpCallback{pubTspt_0}$ Context deactive
[DEBUG] 45.44  app_utils.c:76 - update_app{FTPOTA_TASK}$ Application successfully configured.
[DEBUG] 45.45  app_utils.c:82 - update_app{FTPOTA_TASK}$ Deleting old application /mod/m2mapz.bin
€ÿStarting. This is v1.0.7 built on Apr  7 2020 17:02:52. LEVEL: 2

 Start Hello world Application  [ version: 2.000000 ]

 Hello world 2.0 [ 000001 ]
 Hello world 2.0 [ 000002 ]
 Hello world 2.0 [ 000003 ]
```

Figure 5

3.1.4 CJSON example:

Sample application showcasing how to manage JSON objects. Debug prints on **AUX UART**

Features

- How to read a JSON using cJSON library
- How to write a JSON
- How to manipulate JSON objects

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Parse an example string into a JSON object and print the result in a formatted string
- Print some test outcomes (e.g. non existing item correctly not found)
- Retrieve single elements from the parsed JSON object and use them to format a descriptive string
- Delete the JSON object
- Create a new JSON object appending elements to it
- Print the result JSON string from the object

```
Starting Logging demo app. This is v1.0.7 built on Apr  7 2020 08:33:03.
And here is what we got:
{
        "name":         "Atlantic Ocean",
        "format":       {
                "type":         "salt",
                "volume":       310410900,
                "depth":        -8486,
                "volume_percent":       23.300000,
                "tide":         -3.500000,
                "calm":         false,
                "life":         ["plankton", "corals", "fish", "mammals"]
        }
}
inexistent key not found
name found: Atlantic Ocean
format found (null)
Our JSON string contains info about an ocean named Atlantic Ocean,
has a volume of 310410900 km^3 of salt water with -8486 meters max depth,
represents 23.3% of total oceans volume,
has an average low tide of -3.5 meters,
hosts a huge number of living creatures such as plankton, corals, fish, mammals,
and is not always calm.

Let's build a TR50 command with a proprety.publish and an alarm.publish for MQTT (no auth).
And here is what we got:
{
        "1":    {
                "command":      "property.publish",
                "params":       {
                        "thingKey":     "mything",
                        "key":  "mykey",
                        "value":        123.144000
                }
        },
        "2":    {
                "command":      "alarm.publish",
                "params":       {
                        "thingKey":     "mything",
                        "key":  "mykey",
                        "state":        3,
                        "msg":  "Message."
                }
        }
}
END.
```

Figure 6

3.1.5 Crypto Elliptic Curve Cryptography (ECC) example

Sample application showcasing how to manage Elliptic Curve Cryptography functionalities. Debug prints on **AUX UART**

Features

- How to initialize ECC contexts A (Alice) and B (Bob). Alice is emulating a remote host, from which a public key is known.
- How to generate keypairs for contexts and export public keys
- how to export keyblobs from a context (a keyblob is encrypted with hw specific keys, and can only be used on the module where it was created)
- How to save a keyblob in secured TrustZone.
- How to reload a keyblob from the TrustZone into an initialized context
- How to sign a message with ECDSA from context B (Bob) and verify it from another context A (Alice) with the signature and public key of Bob.
- How to make Bob and Alice derive a shared session keys using each other's public key.
- How to make Bob and Alice create an AES context with the newly created shared keys, encode data and decode it on the other side

Application workflow

`M2MB_main.c`

- Create Bob ECC context, create a keypair and export it in a keyblob
- Open a file in secured Trust Zone, then store the keyblob in it.
- Destroy Bob ECC context
- Recreate Bob ECC context, open the file from Trust Zone and read the keyblob.
- Import the keyblob in Bob context.
- Export Bob public key
- Create Alice ECC context, to simulate an external host. Generate a keypair and export the public key.
- Sign a message with Bob context, generating a signature.
- Use Alice to verify the signed message using Bob's signature and public key
- Derive a shared key for Bob, using Alice's public key
- Create an AES context for Bob
- Import the shared key into the AES context
- Encrypt a message using Bob's AES context.
- Derive a shared key for Alice, using Bob's public key

- Create an AES context for Alice
- Import the shared key into the AES context
- Decrypt the message using Alice’s AES context.
- Check the decrypted message and the original one match
- Clear all resources

```
Starting Crypto ECC demo app. This is v1.0.9-C1 built on May 11 2020 16:30:23.

Bob (local) and Alice (remote) scenario
Bob's keypair generated
Bob's keyblob length is 224
Bob exported the keyblob to be securely stored.

Bob already had an item in Secure Data Area, it was removed to create a new one
Bob securely saved the keyblob in Secure Data Area
Releasing resources

Close Bob's context...
Done. Now Bob context does not exist anymore.

Re-initialize Bob Context and load the keyblob from the secure zone
Bob securely loaded the keyblob from the SDA
Import keyblob in Bob's context..
Done. Now export Bob's public key...
Bob's public key successfully exported

Alice's keypair generated
Alice's public key successfully exported

Bob's message signed with ECDSA!
Alice verified bob's message with his pubkey and signature!

------------------------------------------------------------
Bob and Alice will now exchange a message with AES encrypt
------------------------------------------------------------

Bob retrieved the generated shared key size
Bob's shared keyblob length is: 32. Allocate the required memory to store it.
Bob created a shared key using Alice's public key!

Bob created an AEX context to exchange encrypted data with Alice
Bob's AES context imported the shared keyblob
Bob Encrypted the message using AES and the shared key!
Encrypted data:
 94EE531E3B84B2A4EF05502186BFF5DA

Alice retrieved the generated shared key size
Alice's shared keyblob length is: 32. Allocate the required memory to store it.
Alice created a shared key using Bob's public key!

Alice created an AEX context to exchange encrypted data with Bob
Alice's AES context imported the shared keyblob
Alice decrypted the message using AES and the shared key!
Decrypted:
 414094941E8942A4445548035BFAE943

Original, plain message:
 414094941E8942A4445548035BFAE943

Plain and decrypted messages match!
```

Figure 7

3.1.6 EEPROM 24AA256

Sample application showing how to communicate with a MicroChip 24AA256T I2C EEPROM chip using azx eeprom utility APIs. Debug prints on **AUX UART**

Features

- Initialize the logs on the output channel
- configure the EEPROM utility, setting the slave address and the memory parameters (page size, memory size)
- Write single bytes on a random address
- Read written bytes as a page
- Write data using pages
- Read the new data using pages
- Read again using sequential reading
- Read a single byte from a specific address
- Read next byte using read from current address
- Erase the EEPROM
- Deinit EEPROM utility

Application workflow

`M2MB_main.c`

- call azx_eeprom_init() to set the utility parameters (SDA and SCL pins, page and memory sizes)
- call azx_eeprom_writeByte() to store a single byte with value “5” at the address 0x0213
- call azx_eeprom_writeByte() to store a single byte with value “6” at the address 0x0214
- call azx_eeprom_readPages() from address 0x0213 to retrieve the 2 bytes from the EEPROM
- call azx_eeprom_writePages to write 1024 bytes from a buffer, starting from address 0x00
- call azx_eeprom_readPages() again, to read 256 bytes from address 0x00
- call azx_eeprom_readSequentially() to read 256 bytes from 0x00 by without pages (less overhead on I2C protocol)
- call azx_eeprom_readByte() to get a single byte from address 0x00
- call azx_eeprom_readByteFromCurrentAddress() to get a byte from next address (0x01)
- call azx_eeprom_eraseAll() to completely erase the EEPROM memory (this writes 0xFF in each byte)
- call azx_eeprom_readPages from address 0x0213 to get 2 bytes and verify the values have been written to 0xFF

- call azx_eeprom_deinit to close the eeprom handler and the I2C channel

```
Starting I2C EEPROM 24AA256T demo app. This is v1.0.13-C1 built on Nov  3 2020 16:28:23.

Configuring the I2C device...
Opening I2C channel /dev/I2C-160 ( device address is 0xA0 )
Writing 1 byte at address 0x0213...
Done.
Writing 1 byte at address 0x0214...
Done.
Reading the 2 bytes from address 0x0213...
Done. Data: [0xFF 0xFF]

Writing 1024 bytes at address 0x0000..
Done.

Reading 256 bytes from address 0x0000...
Done. Data:
<<ABCDEFGHIJKLMNOPQRSTUVWXYZ.....................................................abcdefghijklmnopqrstuvwxyz.................

Reading 256 bytes sequentially from address 0x0000...
Done. Data:
<<ABCDEFGHIJKLMNOPQRSTUVWXYZ.....................................................abcdefghijklmnopqrstuvwxyz.................


Reading 1 byte from address 0x0000...
Done. Data: 'A'

Reading 1 byte from current address (should be 0x0001)...
Done. Data: 'B'


[DEBUG] 17.47  M2MB_main:177 - run_I2C_EEPROM_Demo{M2M_DamsStart}$ Erasing all the eeprom...
[DEBUG] 28.05  M2MB_main:185 - run_I2C_EEPROM_Demo{M2M_DamsStart}$ Done

Reading the 2 bytes from address 0x0213...
Done. Data: [0xFF 0xFF]

Deinit EEPROM...
Done
```

Figure 8

3.1.7 Easy AT example

Sample application showcasing Easy AT functionalities. Debug prints on **AUX UART**

Features

- Shows how to register custom commands

3.1.8 Events

Sample application showcasing events setup and usage. Debug prints on **AUX UART**

Features

- How to setup OS events with a custom bitmask
- How to wait for events and generate them in callback functions to synchronize blocks of code

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create an event handler
- Create a timer to generate an event, with a 2 seconds expiration time
- Wait for a specific event bit on the event handler
- At timer expiration, set the same event bit and verify that the code flow went through after the event.

```
Starting Events demo app. This is v1.0.7 built on Apr  7 2020 08:44:29.
[DEBUG] 20.55  M2MB_main.c:171 - M2MB_main{M2M_DamsStart}$ m2mb_os_ev_init success
Set the timer attributes structure success.
Timer successfully created
[DEBUG] 20.57  M2MB_main.c:125 - setup_timer{M2M_DamsStart}$ Start the timer, success.
[DEBUG] 22.60  M2MB_main.c:60 - hwTimerCb{pubTspt_0}$ Timer Callback, generate event!
[DEBUG] 22.61  M2MB_main.c:183 - M2MB_main{M2M_DamsStart}$ event occurred!
```

Figure 9

3.1.9 Events - Barrier (multi events)

Sample application showcasing how to setup and use multiple events to create a barrier. Debug prints on **AUX UART**

Features

- How to setup OS events to be used as a barrier
- How to wait for multiple events in the same point, and generate them in call-back functions to synchronize blocks of code

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create an event handler
- Create a timer to generate an event, with a 3 seconds expiration time
- Create another timer to generate an event, with a 6 seconds expiration time
- Start both timers
- Wait for both event bits on the event handler (each one will be set by one of the timers)
- At first timer expiration, set the first event bit and verify that the code flow does not procede.
- At second timer expiration, set the second event bit and verify that the code flow went through after the event (implementing a barrier).

```
Starting Barrier demo app. This is v1.0.7 built on Apr  7 2020 08:48:30.
[DEBUG] 20.01  M2MB_main.c:179 - M2MB_main{M2M_DamsStart}$ m2mb_os_ev_init success
Set the timer attributes structure success.
Timer successfully created with 3000 timeout (ms)
Set the timer attributes structure success.
Timer successfully created with 6000 timeout (ms)
[DEBUG] 23.08  M2MB_main.c:66 - hwTimerCb1{pubTspt_0}$ Timer Callback, generate event 1!
[DEBUG] 26.12  M2MB_main.c:75 - hwTimerCb2{pubTspt_0}$ Timer Callback, generate event 2!
[DEBUG] 26.13  M2MB_main.c:214 - M2MB_main{M2M_DamsStart}$ BOTH events occurred!
```

Figure 10

3.1.10 FOTA example

Sample application showcasing FOTA usage with M2MB API. Debug prints on **AUX UART**

Features

- How download a delta file from a remote server
- How to apply the delta and update the module firmware

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a main task to manage connectivity.
- create a fota task to manage FOTA and start it with INIT option

`fota.c`

fotaTask()

- Initialize FOTA system then reset parameters.
- Check current FOTA state, if not in IDLE, return error.
- Send a message to mainTask so networking is initialized.
- after PdPCallback() notifies the correct context activation, configure the fota client parameters such as FTP server URL, username and password
- get delta file from server. when it is completed, FOTADownloadCallback is called.
- If delta download went fine, check it.
- If delta file is correct, apply it. Once complete, restart the module.

mainTask()

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context. Event will be received on **PdP-Callback** function
- Disable PDP context when required to stop the app

PdpCallback()

- When PDP context is enabled, send a message to fotaTask to start the download

```
Starting FOTA demo app. This is v1.0.7 built on Apr  7 2020 16:24:29.
[DEBUG] 27.68  fota.c:185 - fotaTask{FOTA_TASK}$ Init FOTA...
Session file not present, procede with FOTA...
[DEBUG] 27.69  fota.c:229 - fotaTask{FOTA_TASK}$ m2mb_fota_reset PASS
[DEBUG] 27.70  fota.c:253 - fotaTask{FOTA_TASK}$ m2mb_fota_state_get  M2MB_FOTA_STATE_IDLE
[DEBUG] 27.71  fota.c:369 - mainTask{MAIN_TASK}$ Case INIT
[DEBUG] 27.71  fota.c:374 - mainTask{MAIN_TASK}$ Case WAIT_FOR_REGISTRATION
[DEBUG] 27.72  fota.c:378 - mainTask{MAIN_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 27.72  fota.c:385 - mainTask{MAIN_TASK}$ Waiting for registration...
[DEBUG] 27.73  fota.c:130 - NetCallback{pubTspt_0}$ Module is registered to cell 0xC4CF!
[DEBUG] 27.74  fota.c:395 - mainTask{MAIN_TASK}$ REGISTERED
[DEBUG] 27.74  fota.c:400 - mainTask{MAIN_TASK}$ Pdp context activation
[DEBUG] 27.75  fota.c:404 - mainTask{MAIN_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 29.71  fota.c:419 - mainTask{MAIN_TASK}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 30.44  fota.c:151 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 30.45  fota.c:154 - PdpCallback{pubTspt_0}$ IP address: 37.118.165.97

[DEBUG] 30.45  fota.c:278 - fotaTask{FOTA_TASK}$
Trying to download "samples/FOTA/30.00.006.2_to_30.00.006.2_ME910C1_NANVWW.bin" delta file...
[DEBUG] 30.47  fota.c:288 - fotaTask{FOTA_TASK}$ m2mb_fota_get_delta OK - Waiting for the completion callback
[DEBUG] 39.03  fota.c:95 - FOTADownloadCallBack{pubTspt_0}$ FOTA download Success - performing packet validation...
[DEBUG] 39.05  fota.c:294 - fotaTask{FOTA_TASK}$ Validating delta file...
[DEBUG] 122.34  fota.c:310 - fotaTask{FOTA_TASK}$ Packet is valid, start update...
[DEBUG] 122.38  fota.c:322 - fotaTask{FOTA_TASK}$ m2mb_fota_start PASS
[DEBUG] 124.39  fota.c:335 - fotaTask{FOTA_TASK}$
Rebooting...After reboot there will be the new FW running on module!

#OTAEV: Module Upgraded To New Fw

Starting FOTA demo app. This is v1.0.7 built on Apr  7 2020 16:24:29.
[DEBUG] 33.31  fota.c:185 - fotaTask{FOTA_TASK}$ Init FOTA...

Session file is already present, stop.
```

Figure 11

3.1.11 FTP

Sample application showcasing FTP client demo with AZX FTP. Debug prints on **AUX UART**

Features

- How to check module registration and activate PDP context
- How to connect to a FTP server
- How to exchange data with the server

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage FTP client and start it

`ftp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Init FTP client and set the debug function for it
- Connect to the server
- Perform log in
- Check remote file size and last modification time
- Download file from server to local filesystem. A data callback is set to report periodic info about the download status
- Upload the same file to the server with a different name. A data callback is set to report periodic info about the upload status
- Download another file content in a buffer instead of a file. A data callback is set to report periodic info about the download status
- Close the connection with FTP server
- Disable PDP context

```
Starting FTP demo app. This is v1.0.7 built on Apr  7 2020 11:17:36.
[DEBUG] 21.23  ftp_test.c:290 - msgFTPTask{FTP_TASK}$ INIT
[DEBUG] 21.23  ftp_test.c:304 - msgFTPTask{FTP_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.23  ftp_test.c:310 - msgFTPTask{FTP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.23  ftp_test.c:318 - msgFTPTask{FTP_TASK}$ Waiting for registration...
[DEBUG] 21.25  ftp_test.c:214 - NetCallback{pubTspt_0}$ Module is registered to network
[DEBUG] 21.26  ftp_test.c:331 - msgFTPTask{FTP_TASK}$ Pdp context activation
[DEBUG] 21.27  ftp_test.c:335 - msgFTPTask{FTP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.31  ftp_test.c:344 - msgFTPTask{FTP_TASK}$ Activate PDP with APN web.omnitel.it on cid 3....
[DEBUG] 24.09  ftp_test.c:241 - PdpCallback{pubTspt_0}$ Context active
[DEBUG] 24.10  ftp_test.c:244 - PdpCallback{pubTspt_0}$ IP address: 176.244.166.181
Start ftp client...
[DEBUG] 24.82  ftp_test.c:373 - msgFTPTask{FTP_TASK}$ Connected.
[DEBUG] 26.32  ftp_test.c:386 - msgFTPTask{FTP_TASK}$ FTP login successful.
Get remote file /samples/pattern_big.txt size
[DEBUG] 26.69  ftp_test.c:428 - msgFTPTask{FTP_TASK}$ Done. File size: 20026.
Get remote file /samples/pattern_big.txt last modification date
[DEBUG] 26.89  ftp_test.c:450 - msgFTPTask{FTP_TASK}$ Done. File last mod date: 20200407090654
.

Starting download of remote file /samples/pattern_big.txt into local /mod/_pattern_big.txt
/samples/pattern_big.txt 47.54% 9520
/samples/pattern_big.txt 100.00% 20026
[DEBUG] 29.75  ftp_test.c:488 - msgFTPTask{FTP_TASK}$ download successful.
[DEBUG] 29.76  ftp_test.c:522 - msgFTPTask{FTP_TASK}$
Local file /mod/_pattern_big.txt size: 20026

Starting upload of local file /mod/_pattern_big.txt
/mod/_pattern_big.txt 81.81% 16384
Upload successful.

Starting download of remote file /samples/pattern.txt into local buffer
Getting remote file /samples/pattern.txt size..
[DEBUG] 32.97  ftp_test.c:583 - msgFTPTask{FTP_TASK}$ Done. File size: 988.
Starting download of remote file /samples/pattern.txt to buffer
[DEBUG] 34.08  ftp_test.c:145 - buf_data_cb{FTP_TASK}$ Received START event
[DEBUG] 34.09  ftp_test.c:149 - buf_data_cb{FTP_TASK}$ Received DATA: 988 bytes on buffer 0x400399e0
[DEBUG] 34.26  ftp_test.c:153 - buf_data_cb{FTP_TASK}$ Received END event
[DEBUG] 34.26  ftp_test.c:623 - msgFTPTask{FTP_TASK}$ Download successful. Received 988 bytes<<<
0    |---------|    |---------|    |---------|    |---------|    |---------|   *
1    |    A    |    |    A    |    |    A    |    |    A    |    |    A    |    *
2    |   AAA   |    |   AAA   |    |   AAA   |    |   AAA   |    |   AAA   |     *
3    |  AAAAA  |    |  AAAAA  |    |  AAAAA  |    |  AAAAA  |    |  AAAAA  |      *
4    | AAAAAAA |    | AAAAAAA |    | AAAAAAA |    | AAAAAAA |    | AAAAAAA |       *
5    |AAAAAAAAA|    |AAAAAAAAA|    |AAAAAAAAA|    |AAAAAAAAA|    |AAAAAAAAA|        *
6    | AAAAAAA |    | AAAAAAA |    | AAAAAAA |    | AAAAAAA |    | AAAAAAA |       *
7    |  AAAAA  |    |  AAAAA  |    |  AAAAA  |    |  AAAAA  |    |  AAAAA  |      *
8    |   AAA   |    |   AAA   |    |   AAA   |    |   AAA   |    |   AAA   |     *
9    |    A    |    |    A    |    |    A    |    |    A    |    |    A    |    *
10   |---------|    |---------|    |---------|    |---------|    |---------|   *
11                                                                             *
12   |---------|    |---------|    |---------|    |---------|    |--->>>
```

Figure 12

3.1.12 File System example

Sample application showcasing M2MB File system API usage. Debug prints on **AUX UART**

Features

- How to open a file in write mode and write data in it
- How to reopen the file in read mode and read data from it

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Open file in write mode
- Write data in file
- Close file
- Reopen file in read mode
- Read data from file and print it
- Close file and delete it

```
Starting FileSystem demo app. This is v1.0.7 build on Mar 26 2020 09:50:19. LEVEL: 2
Opening /my_text_file.txt in write mode..
Buffer written successfully into file. 15 bytes were written.
Closing file.
Opening /my_text_file.txt in read only mode..
Received 15 bytes from file:
<Hello from file>
Closing file.
Deleting File
File deleted
App Completed
```

Figure 13

3.1.13 GNSS example

Sample application showing how to use GNSS functionality. Debug prints on **AUX UART**

Features

- How to enable GNSS receiver on module
- How to collect location information from receiver

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Init gnss, enable position report and start it.
- When a fix is available, a message will be printed by the GNSS callback function

```
START GNSS TEST APP
m2mb_gnss_enable OK
m2mb_gnss_start OK
latitude_valid: 1 - latitude: 39.228245
longitude_valid: 1 - longitude: 9.069106
altitude_valid: 1 - altitude: 12.000000
uncertainty_valid: 1 - uncertainty: 30.000000
velocity_valid: 1 - codingType: 0
speed_horizontal: 0.000000
bearing: 0.000000
timestamp_valid: 1 -timestamp: 1563376148000
speed_valid: 1 - speed: 0.060000
```

Figure 14

3.1.14 GPIO interrupt example

Sample application showing how to use GPIOs and interrupts. Debug prints on **AUX UART**

Features

- How to open a GPIO in input mode with interrupt
- How to open a second GPIO in output mode to trigger the first one

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Open GPIO 4 as output
- Open GPIO 3 as input and set interrupt for any edge (rising and falling). **A jumper must be used to short GPIO 3 and 4 pins.**
- Toggle GPIO 4 status high and low every second
- An interrupt is generated on GPIO 3

```
Starting GPIO interrupt demo app. This is v1.0.7 built on Mar 26 2020 16:33:01.
Setting gpio 3 interrupt...
Setting GPIO 4 HIGH
CALLBACK->Interrupt on GPIO 3! Value: 1
Setting GPIO 4 LOW
CALLBACK->Interrupt on GPIO 3! Value: 0
Setting GPIO 4 HIGH
CALLBACK->Interrupt on GPIO 3! Value: 1
Setting GPIO 4 LOW
CALLBACK->Interrupt on GPIO 3! Value: 0
Setting GPIO 4 HIGH
CALLBACK->Interrupt on GPIO 3! Value: 1
Setting GPIO 4 LOW
CALLBACK->Interrupt on GPIO 3! Value: 0
```

Figure 15

3.1.15 HTTP Client

Sample application showing how to use HTTPs client functionalities. Debug prints on **AUX UART**

Features

- How to check module registration and activate PDP context
- How to initialize the http client, set the debug hook function and the data call-back to manage incoming data
- How to perform GET, HEAD or POST operations

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage HTTP client and start it

`httpTaskCB`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create HTTP client options and initialize its functionality
- Create HTTP SSL config and initialize the SSL options
- Configure data management options for HTTP client
- Appy all configurations to HTTP client
- Perform a GET request to a server
- Disable PDP context

`DATA_CB`

- Print incoming data
- Set the abort flag to 0 to keep going.

```
Starting HTTP(s) client demo app. This is v1.0.13-C1 built on Aug 11 2020 16:56:28.
[DEBUG] 15.19  M2MB_main:259 - activatePdP{HttpClient}$ m2mb_os_ev_init success
[DEBUG] 15.20  M2MB_main:265 - activatePdP{HttpClient}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 15.20  M2MB_main:273 - activatePdP{HttpClient}$ Waiting for registration...
[DEBUG] 15.21  M2MB_main:119 - NetCallback{pubTspt_0}$ Module is registered to cell 0xC4CF!
[DEBUG] 15.22  M2MB_main:287 - activatePdP{HttpClient}$ Pdp context initialization
[DEBUG] 17.26  M2MB_main:297 - activatePdP{HttpClient}$ Activate PDP with APN NXT17.NET....
[DEBUG] 17.97  M2MB_main:146 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 17.97  M2MB_main:149 - PdpCallback{pubTspt_0}$ IP address: 100.77.54.97
Performing a GET request...

Host Address: linux-ip.net Port 80

Socket connected! |
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="author" content="Martin A. Brown" />
    <meta name="robots" content="index, follow"/>

    <meta property="og:title" content="http://linux-ip.net/"/>
    <meta property="og:url" content="http://linux-ip.net"/>
    <meta property="og:site_name" content="http://linux-ip.net/"/>
    <meta property="og:type" content="website"/>

    <link rel="canonical" href="http://linux-ip.net" />

    <title>http://linux-ip.net/</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" type="text/css" href="//netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-awesome.css" />
    <link rel="stylesheet" type="text/css" href="//netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css" />
    <link rel="stylesheet" type="text/css" href="http://linux-ip.net/theme/css/main.css" />

</head>

   ...
    <footer id="site-footer">
        <div class="row-fluid">
            <div class="span10 offset1">
                <address>
                    <p>
                        Powered by <a href="http://getpelican.com/">Pelican</a>
                        and <a href="http://python.org">Python</a>.
                        Theme based on <a href="http://github.com/jsliang/pelican-fresh/">Fresh</a>
                        by <a href="http://jsliang.com/">jsliang</a>
                    </p>
                </address>
            </div>
        </div>
    </footer>
</body>
</html>
Done.
[DEBUG] 22.44  M2MB_main:155 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 16

3.1.16 HW Timer (Hardware Timer)

The sample application shows how to use HW Timers M2MB API. Debug prints on **AUX UART**

Features

- How to open configure a HW timer
- How to use the timer to manage recurring events

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create hw timer structure
- Configure it with 100 ms timeout, periodic timer (auto fires when expires) and autostart
- Init the timer with the parameters
- Wait 10 seconds
- Stop the timer

`TimerCb`

- Print a message with an increasing counter

```
Starting HW Timers demo app. This is v1.0.7 built on Mar 26 2020 13:04:14.
[DEBUG] 14.06  M2MB_main.c:114 - M2MB_main{M2M_DamsStart}$ Set the timer attributes structure: success.
Timer successfully created
Start the timer, success.
[DEBUG] 14.18  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [0]
[DEBUG] 14.28  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [1]
[DEBUG] 14.38  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [2]
[DEBUG] 14.48  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [3]
[DEBUG] 14.58  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [4]
[DEBUG] 14.69  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [5]
[DEBUG] 14.79  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [6]
[DEBUG] 14.88  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [7]
[DEBUG] 14.98  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [8]
[DEBUG] 15.08  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [9]


[DEBUG] 23.90  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [96]
[DEBUG] 24.01  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [97]
[DEBUG] 24.11  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [98]
Stop a running timer: success
Application end
```

Figure 17

3.1.17 Hello World

The application prints “Hello World!” over selected output every two seconds. Debug prints on **AUX UART**, using AZX log example functions

Features

- How to open an output channel using AZX LOG sample functions
- How to print logging information on the channel using AZX LOG sample functions

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print “Hello World!” every 2 seconds in a while loop

```
Starting. This is v1.0.7 built on Mar 26 2020 09:34:16. LEVEL: 2

 Start Hello world Application  [ version: 2.000000 ]

 Hello world 2.0 [ 000001 ]
 Hello world 2.0 [ 000002 ]
 Hello world 2.0 [ 000003 ]
 Hello world 2.0 [ 000004 ]
 Hello world 2.0 [ 000005 ]
 Hello world 2.0 [ 000006 ]
 Hello world 2.0 [ 000007 ]
 Hello world 2.0 [ 000008 ]
 Hello world 2.0 [ 000009 ]
```

Figure 18

3.1.18 I2C example

Sample application showing how to communicate with an I2C slave device. Debug prints on **AUX UART**

Features

- How to open a communication channel with an I2C slave device
- How to send and receive data to/from the slave device

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Open I2C bus, setting SDA an SCL pins as 2 and 3 respectively
- Set registers to configure accelerometer -Read in a loop the 6 registers carrying the 3 axes values and show the g value for each of them

```
Starting I2C demo app. This is v1.0.7 built on Mar 26 2020 16:50:40.

Configuring the Kionix device...
opening channel /dev/I2C-30
[DEBUG] 20.18  M2MB_main.c:218 - test_I2C{M2M_DamsStart}$ -
WHOAMI content: 0x01
Configuring I2C Registers - Writing 0x4D into 0x1D register (CTRL_REG3)...
Write: success

I2C reading data from 0x1D register (CTRL_REG3)...
Read: success.
Accelerometer Enabled. ODR tilt: 12.5Hz, ODR directional tap: 400Hz, ORD Motion Wakeup: 50Hz
Configuring I2C Registers - Writing 0xC0 into 0x1B register (CTRL_REG1)...
Write: success

I2C reading data from 0x1B register (CTRL_REG1)...
Read: success.
Accelerometer Enabled. Operative mode, 12bit resolution
I2C read axes registers
------------
Reading Success.

X: -0.050 g
Y: -0.046 g
Z: 1.006 g
Reading Success.

X: -0.049 g
Y: -0.044 g
Z: 1.004 g
Reading Success.

X: -0.052 g
Y: -0.044 g
Z: 1.007 g
Reading Success.

X: -0.048 g
Y: -0.045 g
Z: 1.005 g
```

Figure 19

3.1.19 I2C Combined

Sample application showing how to communicate with an I2C slave device with I2C raw mode. Debug prints on **AUX UART**

Features

- How to open a communication channel with an I2C slave device
- How to send and receive data to/from the slave device using raw mode API

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Open I2C bus, setting SDA an SCL pins as 2 and 3 respectively
- Set registers to configure accelerometer -Read in a loop the 6 registers carrying the 3 axes values and show the g value for each of them

```
Starting I2C raw demo app. This is v1.0.13-C1 built on Jul 30 2020 11:28:18.

Configuring the I2C device...
Opening I2C channel /dev/I2C-30 ( device address is 0x0F << 1 )
Accelerometer Enabled. ODR tilt: 12.5Hz, ODR directional tap: 400Hz, ORD Motion Wakeup: 50Hz
Accelerometer Enabled. Operative mode, 12bit resolution
I2C read axes registers
------------

X: 0.000 g
Y: 0.000 g
Z: 0.000 g

X: -0.270 g
Y: 0.016 g
Z: 0.917 g

X: -0.268 g
Y: 0.013 g
Z: 0.925 g

X: -0.271 g
Y: 0.015 g
Z: 0.922 g

X: -0.267 g
Y: 0.016 g
Z: 0.918 g

X: -0.274 g
Y: 0.019 g
Z: 0.915 g
```

Figure 20

3.1.20 LWM2M

Sample application showcasing LWM2M client usage with M2MB API. Debug prints on **AUX UART**

Features

- Configure LWM2M client and enable it
- Create an instance of a custom object
- Set an integer value on a read only resource
- Set two integer values on a multi-instance read only resource
- write a string on a read/write resource
- Manage exec requests from the portal
- Manage write, read and monitoring resources changed from the portal

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create a task to manage the LWM2M client and start it

`lwm2m_demo.c`

`msgLWM2MTask` - Check registration status

- Configure APN to the correct one for CID 1
- Initialize LWM2M client,
- Check for XML file fo custom object
- Enable unsolicited messages from client
- Create a task (lwm2m_taskCB is its callback function)to manage events from Portal
- Enable LwM2M client
- Create a new instance for the custom object
- Wait for client to register to Portal
- Send integer and string values
- Wait for events from server

`lwm2mIndicationCB`

- Manage events arriving from client (operations completion status and unsolicited events)
- Run lwm2m_taskCB when a monitored resource changes, to manage the action to be done

3.1.20.1 Custom Object configuration

The XML file content must be loaded on the Telit IoT Portal for the demo application to be fully executed.

First, enter Developer section from the top menu

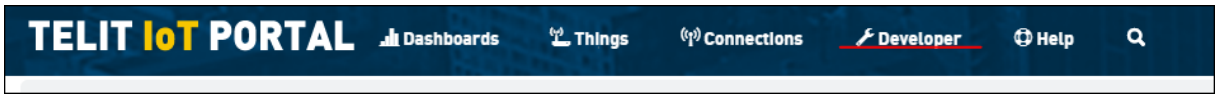

Figure 21

Choose Object Registry

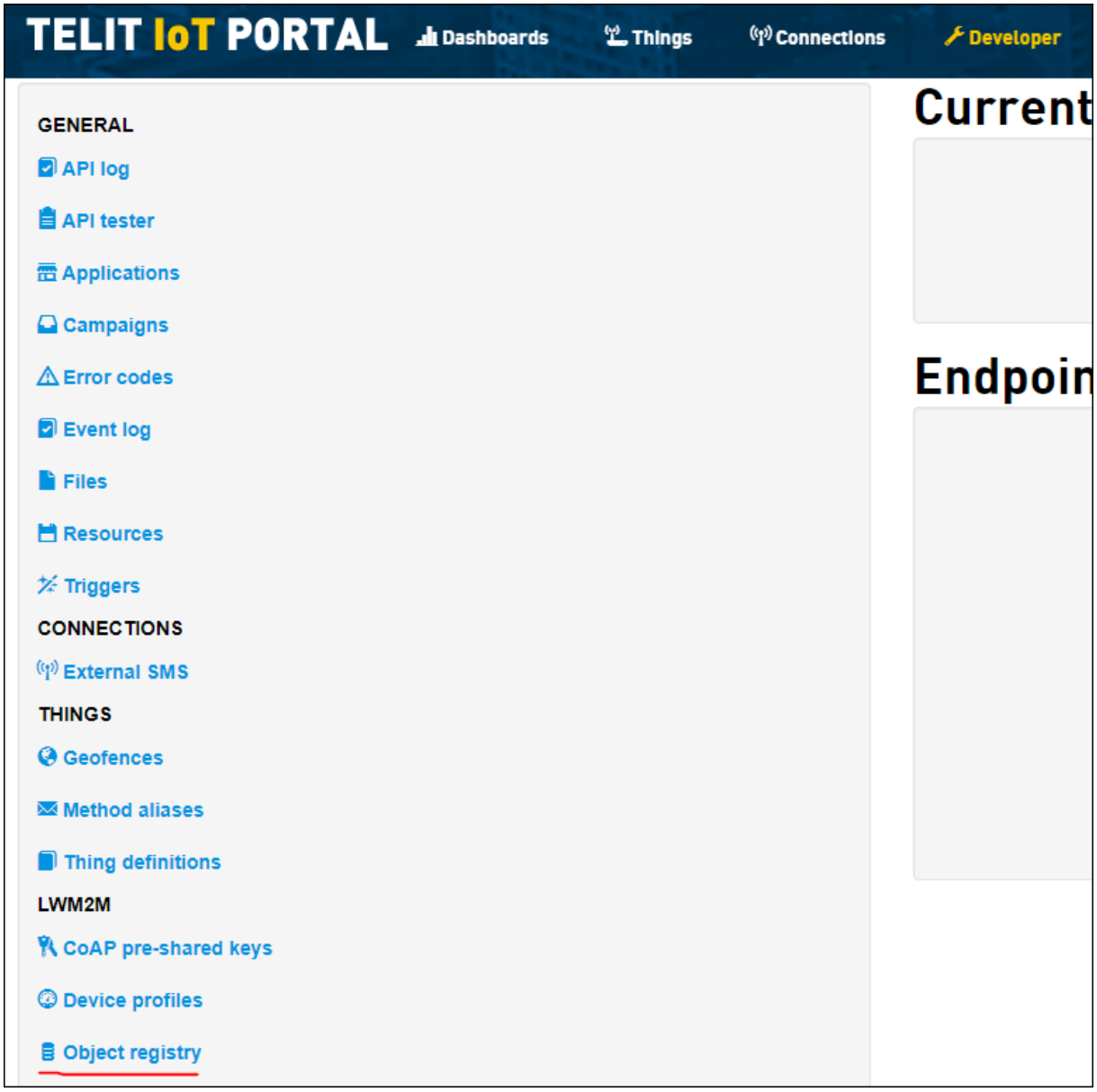

Figure 22

Create a New Object

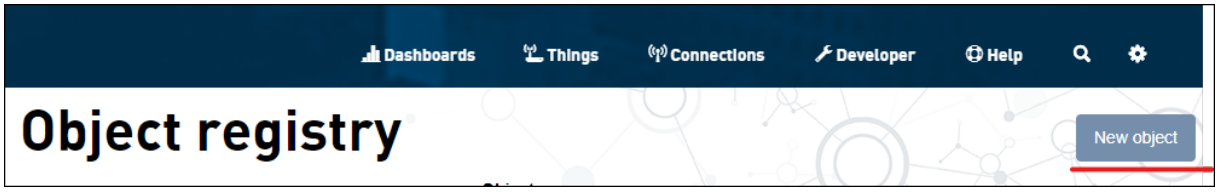

Figure 23

Copy the xml file content and paste it in the new Object form

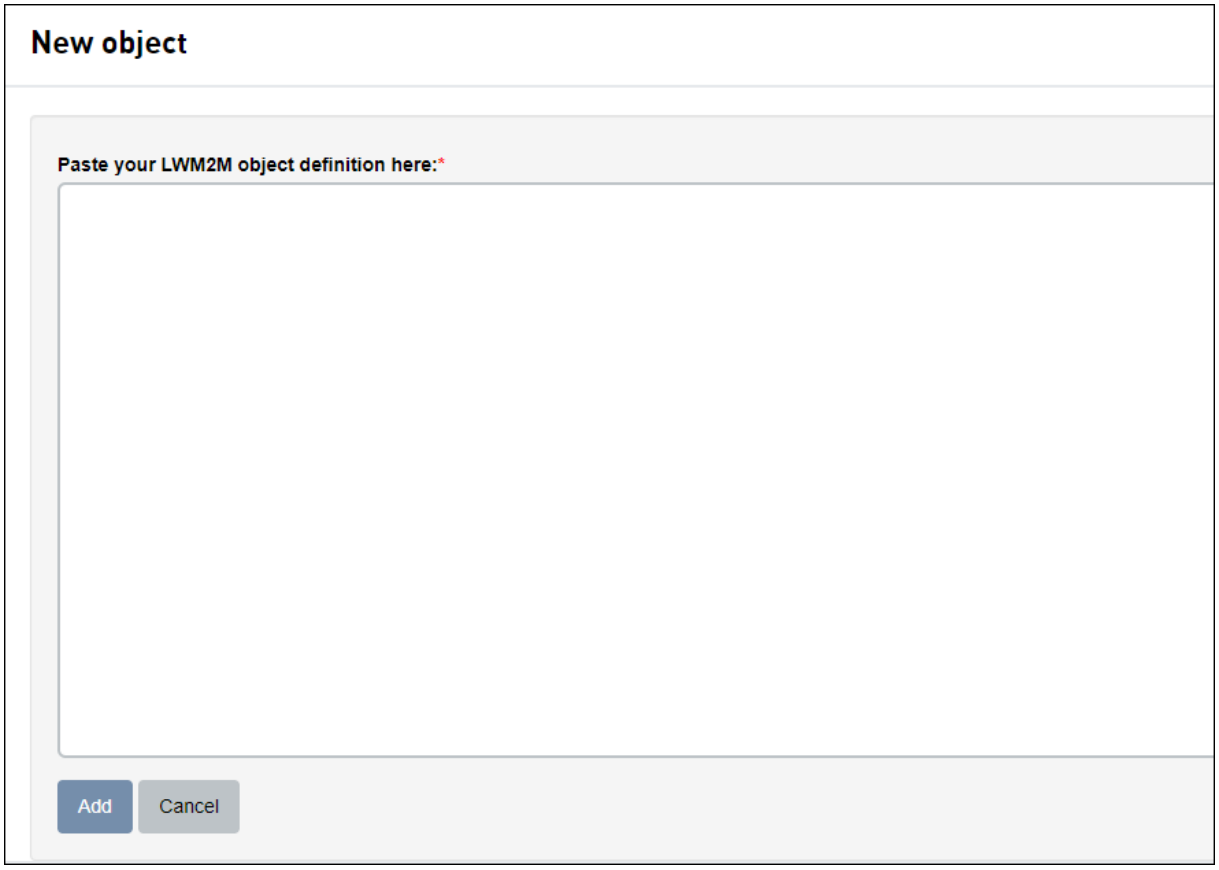

Figure 24

Also, the application requires the XML file `/xml/object_35000.xml` (provided with the sample files) to be stored in module's `/XML/` folder. It can be done with

`AT#M2MWRITE=/XML/object_35000.xml,<size_in_bytes>`

To load the XML file in the module, Telit AT Controller (TATC) can be used. Once the command above is issued, press the load content button:

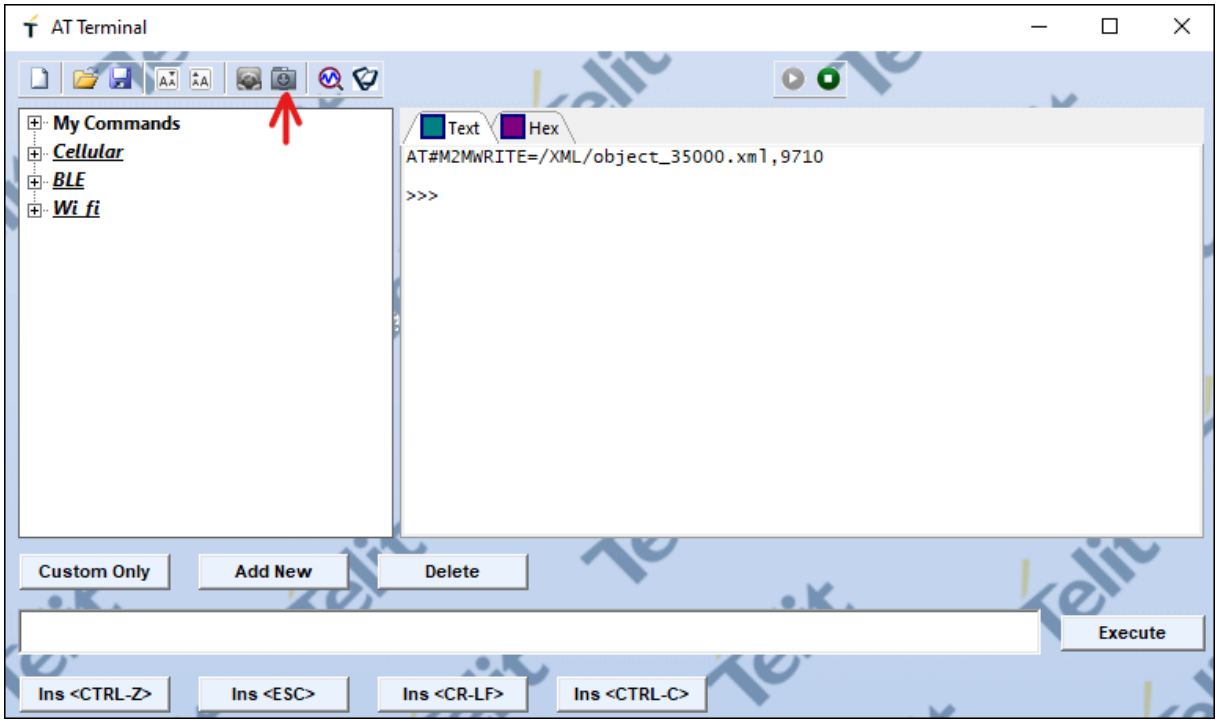

Figure 25

Select the file from your computer

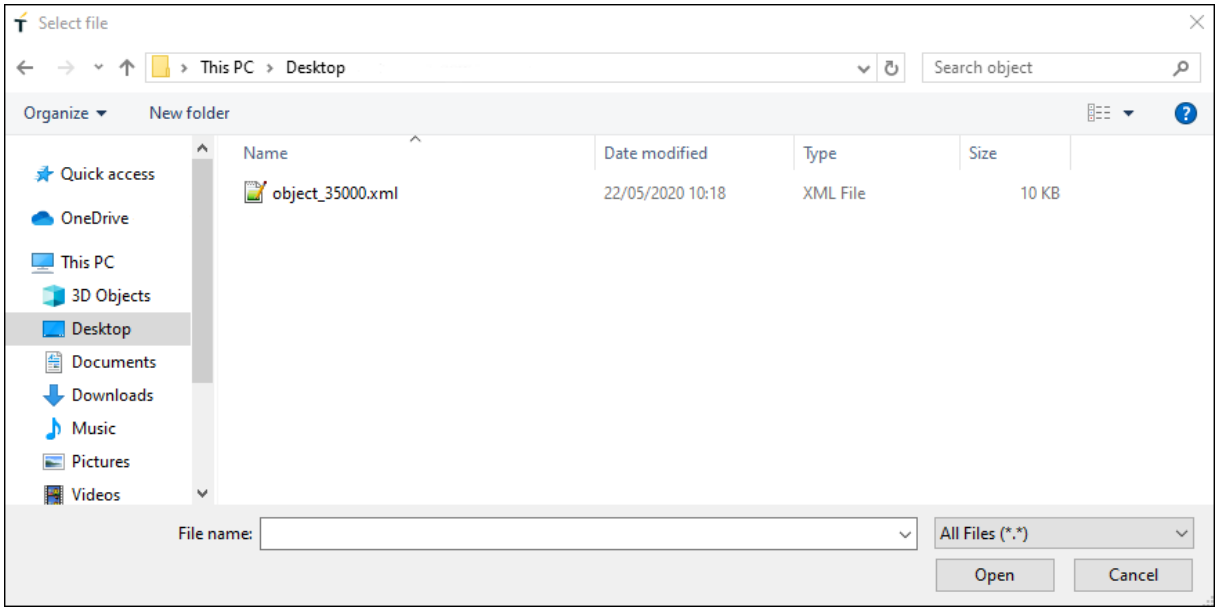

Figure 26

The file is successfully loaded on the module

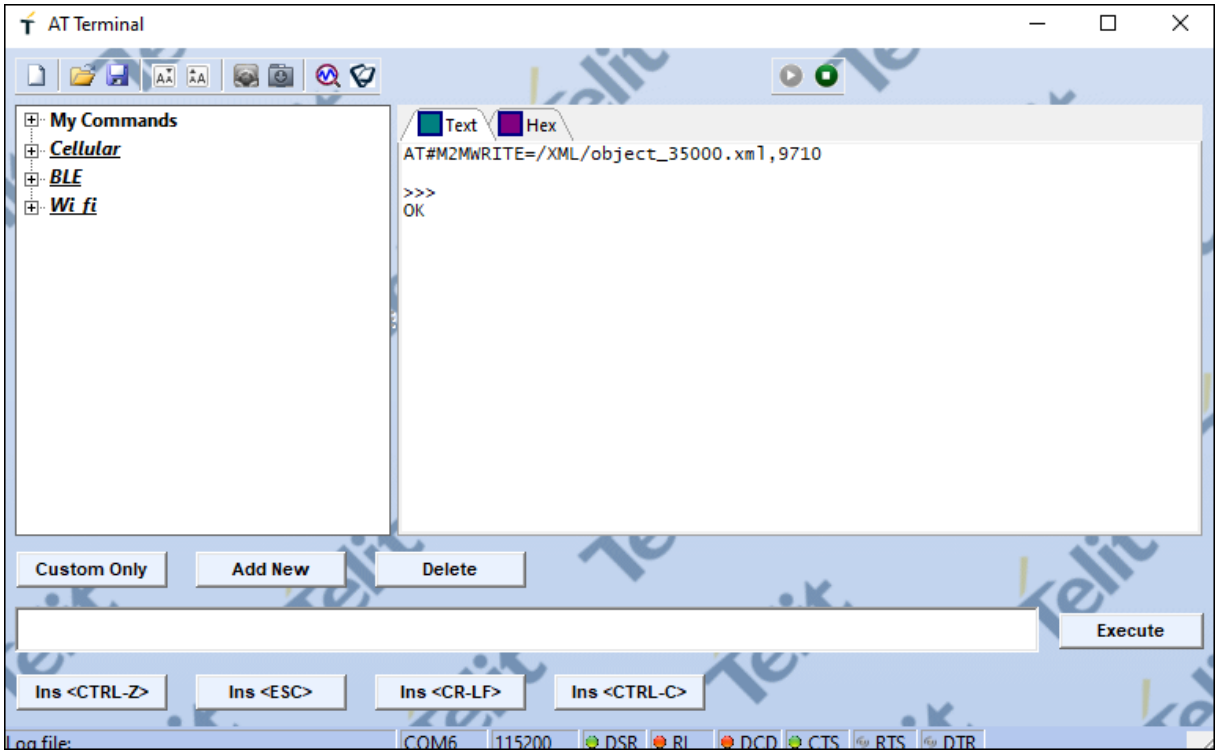

Figure 27

3.1.20.2 Application execution example

```
Starting lwm2m demo. This is v1.0.13-C1 built on Jul  6 2020 06:54:58.
On OneEdge portal, be sure that observations are enabled for the following object resources:
{35000/0/01} {35000/0/02} {35000/0/03} {35000/0/04} {35000/0/05} {35000/0/06} {35000/0/07}
{35000/0/11} {35000/0/12} {35000/0/13} {35000/0/14} {35000/0/15} {35000/0/16} {35000/0/17}
{35000/0/21} {35000/0/22} {35000/0/23} {35000/0/24} {35000/0/25} {35000/0/26} {35000/0/27}
{35000/0/31} {35000/0/32} {35000/0/33} {35000/0/34} {35000/0/35} {35000/0/36} {35000/0/37}

Initializing resources...
LWM2M enable result OK
[DEBUG] 32.39  lwm2m_demo:637 - lwm2mIndicationCB{pubTspt_0}$ Monitoring enabled.

Waiting LWM2M Registering (120 seconds timeout)...
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPING
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPED
resp->info == M2MB_LWM2M_CL_STATE_REGISTERING
resp->info == M2MB_LWM2M_CL_STATE_REGISTERED

Waiting for events from portal. Write on monitored resource or call an exec

GET STATUS.
IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_REGISTERING
```

Figure 28

```
Setting integer resource {35000/0/2} value to 50 on LWM2M client.

Setting integer resource {35000/0/22/0} value to 10 on LWM2M client.

Resource /35000/0/2/0 changed!

Setting integer resource {35000/0/22/1} value to 11 on LWM2M client.
[DEBUG] 52.05  lwm2m_demo:470 - lwm2mIndicationCB{pubTspt_0}$ Read type: 3; size: 4

Resource /35000/0/22/0 changed!

Integer data in {35000/0/2/0} resource was updated to new value: 50

Writing string resource {35000/0/11} value to demo_string on LWM2M client.
[DEBUG] 52.57  lwm2m_demo:470 - lwm2mIndicationCB{pubTspt_0}$ Read type: 3; size: 4

Resource /35000/0/22/1 changed!

Integer data in {35000/0/22/0} resource was updated to new value: 10
[DEBUG] 54.19  lwm2m_demo:470 - lwm2mIndicationCB{pubTspt_0}$ Read type: 3; size: 4

Resource /35000/0/11/0 changed!

Integer data in {35000/0/22/1} resource was updated to new value: 11
[DEBUG] 54.71  lwm2m_demo:470 - lwm2mIndicationCB{pubTspt_0}$ Read type: 2; size: 11

String data in {35000/0/11/0} resource was updated to new content: <demo_string>
```

Figure 29

After the Demo completes the initialization, it is possible to access the object resources from the Portal Object Browser

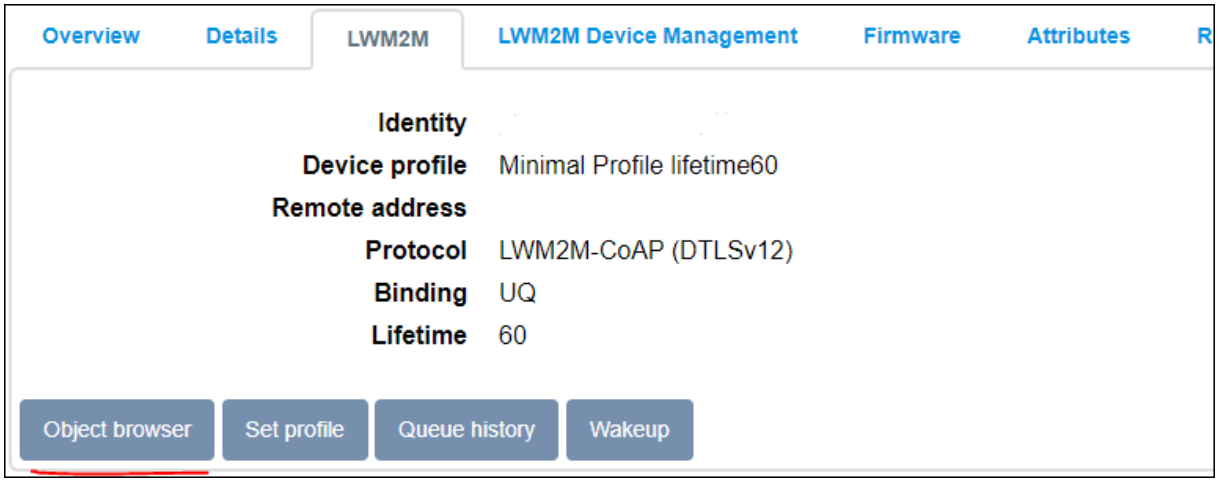

Figure 30

An instance of the object will be present and the resources can be modified.

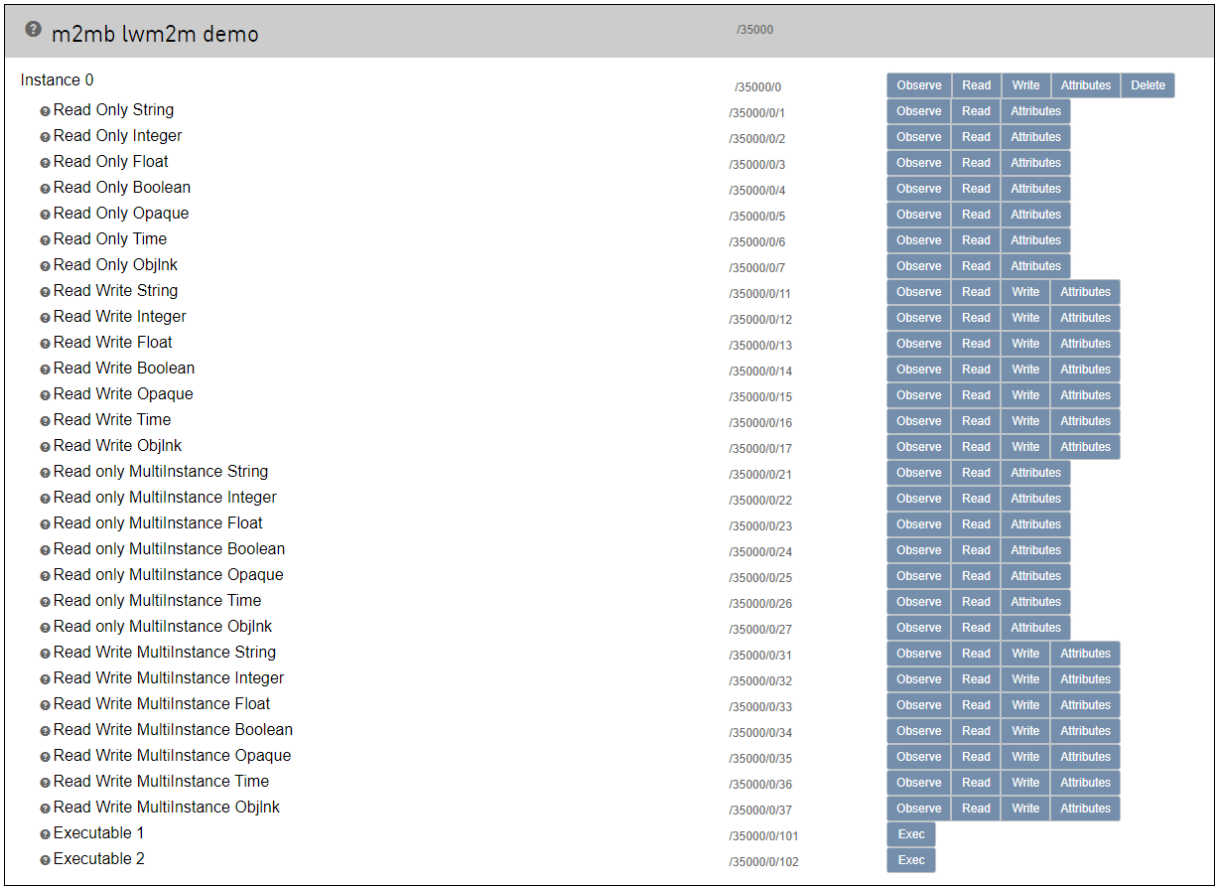

Figure 31

For example, executing the two Exec Resources at the bottom of the list, the application will react accordingly:

```
Info Exec Ind: 35000/0/101/0
Asked to execute resource 101

Info Exec Ind: 35000/0/102/0
Asked to execute resource 102
```

Figure 32

Writing a string resource (id /35000/0/11), the application will notify the change

```
Resource /35000/0/11/0 changed!
[DEBUG] 1249.86  lwm2m_demo:470 - lwm2mIndicationCB{pubTspt_0}$ Read type: 2; size: 5

String data in {35000/0/11/0} resource was updated to new content: <hello>
```

Figure 33

3.1.21 Logging Demo

Sample application showing how to print on one of the available output interfaces. Debug prints on **AUX UART**

Features

- How to open a logging channel
- How to set a logging level
- How to use different logging macros

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Print a message with every log level

```
Starting Logging demo app. This is v1.0.7 built on Mar 26 2020 13:57:06.
[WARN ] 20.17  M2MB_main.c:74 - M2MB_main{M2M_DamsStart}$ This is a WARNING MESSAGE
[ERROR] 20.18  M2MB_main.c:76 - M2MB_main{M2M_DamsStart}$ THIS IS AN ERROR MESSAGE
[CRITICAL] 20.19  M2MB_main.c:78 - M2MB_main{M2M_DamsStart}$ THIS IS AN CRITICAL MESSAGE
[DEBUG] 20.19  M2MB_main.c:80 - M2MB_main{M2M_DamsStart}$ This is a DEBUG message
[TRACE] 20.20  M2MB_main.c:82 - M2MB_main{M2M_DamsStart}$ This is a TRACE message
END.
```

Figure 34

3.1.22 MD5 example

Sample application showing how to compute MD5 hashes using m2mb crypto. Debug prints on **AUX UART**

Features

- Compute MD5 hash of a file
- Compute MD5 hash of a string

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create a temporary file with the expected content
- Compute MD5 hash of the provided text file
- Compare the hash with the expected one
- Compute MD5 hash of a string
- Compare the hash with the expected one
- Delete test file

```
Starting MD5 demo app. This is v1.0.7 built on Apr  7 2020 10:19:54.
Buffer written successfully into file. 45 bytes were written.

Computing hash from file...
Computed hash: bb0fa6eff92c305f166803b6938dd33a
Expected hash: bb0fa6eff92c305f166803b6938dd33a
Hashes are the same!

Computing hash from string...
Computed hash: bb0fa6eff92c305f166803b6938dd33a
Expected hash: bb0fa6eff92c305f166803b6938dd33a
Hashes are the same!
```

Figure 35

3.1.23 MQTT Client

Sample application showcasing MQTT client functionalities (with SSL). Debug prints on **AUX UART**

Features

- How to check module registration and enable PDP context
- How to configure MQTT client parameters
- How to connect to a broker with SSL and exchange data over a subscribed topic

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage MQTT client and start it

`mqtt_demo.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Init MQTT client
- Configure it with all parameters (Client ID, username, password, PDP context ID, keepalive timeout...)
- Connect MQTT client to broker
- Subscribe to two topics
- Publish 10 messages with increasing counter. Even messages are sent to topic 1, odd messages on topic 2.
- Print received message in mqtt_topc_cb function
- Disconnect MQTT client and deinit it
- Disable PDP context

```
Starting MQTT demo app. This is v1.0.7 built on Apr  7 2020 10:34:08.
[DEBUG] 16.18  mqtt_demo.c:192 - MQTT_Task{MQTT_TASK}$ INIT
[DEBUG] 16.18  mqtt_demo.c:206 - MQTT_Task{MQTT_TASK}$ m2mb_os_ev_init success
[DEBUG] 16.19  mqtt_demo.c:214 - MQTT_Task{MQTT_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 16.19  mqtt_demo.c:221 - MQTT_Task{MQTT_TASK}$ Waiting for registration...
[DEBUG] 16.20  mqtt_demo.c:131 - NetCallback{pubTspt_0}$ Module is registered
[DEBUG] 16.21  mqtt_demo.c:232 - MQTT_Task{MQTT_TASK}$ Pdp context activation
[DEBUG] 18.26  mqtt_demo.c:246 - MQTT_Task{MQTT_TASK}$ Activate PDP with APN web.omnitel.it on CID 3....
[DEBUG] 18.95  mqtt_demo.c:155 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 18.96  mqtt_demo.c:159 - PdpCallback{pubTspt_0}$ IP address: 37.118.201.56
[DEBUG] 18.96  mqtt_demo.c:268 - MQTT_Task{MQTT_TASK}$ Init MQTT
[DEBUG] 18.97  mqtt_demo.c:278 - MQTT_Task{MQTT_TASK}$ m2mb_mqtt_init succeeded

Connecting to broker <api-dev.devicewise.com>:1883...
Done.
Subscribing to test_topic and test_topic2..
[DEBUG] 20.35  mqtt_demo.c:367 - MQTT_Task{MQTT_TASK}$ Done.

[DEBUG] 20.36  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 2> to topic test_topic
[DEBUG] 20.37  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 20.71  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 20.72  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 2>
[DEBUG] 23.37  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 3> to topic test_topic2
[DEBUG] 23.38  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 23.92  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 23.93  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 3>
[DEBUG] 26.40  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 4> to topic test_topic
[DEBUG] 26.41  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 26.93  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 26.93  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 4>
[DEBUG] 29.42  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 5> to topic test_topic2
[DEBUG] 29.43  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 29.99  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 30.00  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 5>
[DEBUG] 32.46  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 6> to topic test_topic
[DEBUG] 32.48  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 33.00  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 33.01  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 6>
[DEBUG] 35.47  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 7> to topic test_topic2
[DEBUG] 35.48  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 36.01  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 36.02  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 7>
[DEBUG] 38.50  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 8> to topic test_topic
[DEBUG] 38.51  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 39.15  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 39.16  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 8>
[DEBUG] 41.52  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 9> to topic test_topic2
[DEBUG] 41.53  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 42.10  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 42.12  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 9>
[DEBUG] 44.56  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 10> to topic test_topic
[DEBUG] 44.57  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 45.09  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 28
[DEBUG] 45.11  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 10>
[DEBUG] 47.58  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 11> to topic test_topic2
[DEBUG] 47.59  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 48.12  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 28
[DEBUG] 48.13  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 11>

Disconnecting from MQTT broker..
[DEBUG] 50.60  mqtt_demo.c:414 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 50.61  mqtt_demo.c:443 - MQTT_Task{MQTT_TASK}$ application exit
[DEBUG] 50.62  mqtt_demo.c:453 - MQTT_Task{MQTT_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 50.63  mqtt_demo.c:457 - MQTT_Task{MQTT_TASK}$ Application complete.
[DEBUG] 51.23  mqtt_demo.c:164 - PdpCallback{pubTspt_0}$ Context deactivated!
```

Figure 36

3.1.24 MultiTask

Sample application showcasing multi tasking functionalities with M2MB API. Debug prints on **AUX UART**

Features

- How to create tasks using azx utilities
- How to use send messages to tasks
- How to use a semaphore to synchronize two tasks

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create three tasks with the provided utility (this calls public m2mb APIs)
- Send a message to the task1, its callback function azx_msgTask1 will be called

`azx_msgTask1`

- Print received parameters from main
- Send modified parameters to task2 (its callback function azx_msgTask2 will be called)
- wait for an InterProcess Communication semaphore to be available (released by task3)
- Once the semaphore is available, print a message and return

`azx_msgTask2`

- Print received parameters from caller
- If first parameter is bigger than a certain value, Send modified parameters to task3
- Else, use the second parameter as a task handle and print the corresponding name plus the value of the first parameter

`azx_msgTask3`

- Print received parameters from task 2
- release IPC semaphore
- send message to task 2 with first parameter below the threshold and second parameter with task3 handle

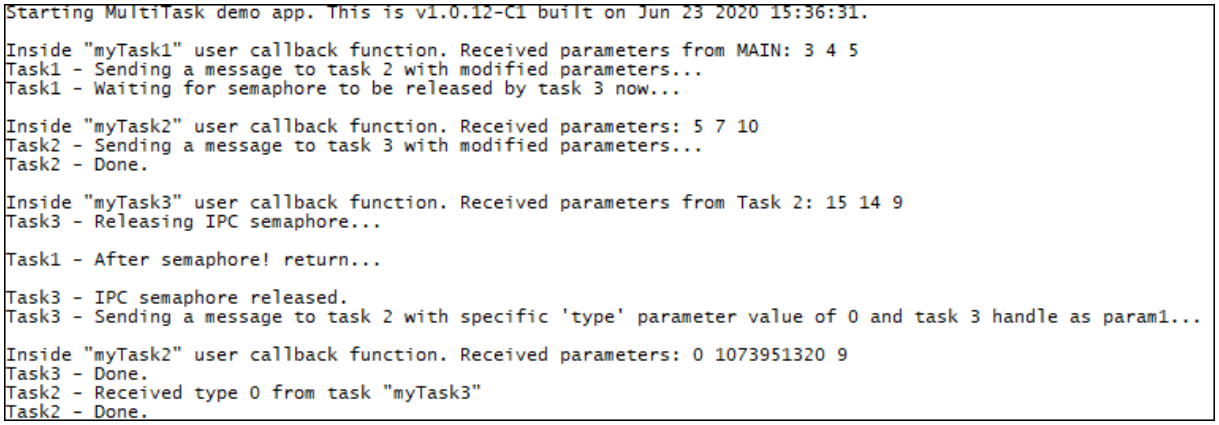
Starting MultiTask demo app. This is v1.0.12-C1 built on Jun 23 2020 15:36:31.
Inside "myTask1" user callback function. Received parameters from MAIN: 3 4 5
Task1 - Sending a message to task 2 with modified parameters...
Task1 - Waiting for semaphore to be released by task 3 now...
Inside "myTask2" user callback function. Received parameters: 5 7 10
Task2 - Sending a message to task 3 with modified parameters...
Task2 - Done.
Inside "myTask3" user callback function. Received parameters from Task 2: 15 14 9
Task3 - Releasing IPC semaphore...
Task1 - After semaphore! return...
Task3 - IPC semaphore released.
Task3 - Sending a message to task 2 with specific 'type' parameter value of 0 and task 3 handle as param1...
Inside "myTask2" user callback function. Received parameters: 0 1073951320 9
Task3 - Done.
Task2 - Received type 0 from task "myTask3"
Task2 - Done.

Figure 37

3.1.25 MutEx

Sample application showing mutex usage, with ownership and prioritization usage. Debug prints on **AUX UART**

Features

- How to create a mutex
- How to use the mutex with tasks having different priorities
- how to reorder the pending tasks queue for the mutex

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create four tasks with the provided utility (this calls public m2mb APIs). The first task is a “producer”, putting data on a shared buffer. The second is a “consumer” of said data, the other two are used for prioritization demo
- run producer and consumer tasks at the same pace. the shared buffer will stay empty, because the resource is consumed right after creation
- run producer twice as fast as consumer. The buffer is slowly filled
- run consumer twice as fast as publisher. The buffer is always empty.
- reserve the mutex in the main task and run producer, support and support2 tasks (in this order). Then release the mutex and check the execution order. It should be by arrival.
- reserve the mutex in the main task and run the same three task, but before releasing the mutex, call the prioritization API. the task with highest priority (producer) is put as first in the queue.

```
Starting MutEx app. This is v1.0.12-C1 built on Jul  1 2020 08:37:15.
[DEBUG] 14.50  M2MB_main:90 - mutex_init{M2M_DamsStart}$ [MUTEX] Mutex initialized

[CASE 1 ] Producer and consumer have same idle time

[DEBUG] 14.51  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 14.52  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 14.53  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 14.53  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 14.54  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 14.54  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 14.55  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 14.56  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 15.56  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 15.56  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 15.57  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 15.58  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 15.58  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 15.59  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 15.60  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 15.60  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 16.61  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 16.61  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 16.62  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 16.63  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 16.63  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 16.64  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 16.64  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 16.65  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
```

Figure 38

```
[CASE 2 ] Producer has double idle time

[DEBUG] 17.56  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 17.56  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 17.57  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 17.58  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 17.58  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 17.59  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 17.59  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 17.60  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 18.63  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 18.64  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 0 items
[DEBUG] 18.64  M2MB_main:268 - msgConsumer{CONSUMER}$ Can't consume anything, buffer size is 0
[DEBUG] 18.65  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 19.62  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 19.62  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 19.63  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 19.64  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 19.68  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 19.69  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 19.69  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 19.70  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 20.73  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 20.74  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 0 items
[DEBUG] 20.75  M2MB_main:268 - msgConsumer{CONSUMER}$ Can't consume anything, buffer size is 0
[DEBUG] 20.75  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 21.67  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 21.68  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 21.68  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 21.69  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 21.77  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 21.79  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 21.80  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 21.80  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
```

Figure 39

```
[CASE 3 ] Producer has half idle time

[DEBUG] 22.62  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 22.63  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 22.64  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 22.64  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 22.65  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 22.65  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 22.66  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 22.67  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 23.67  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 23.68  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 23.68  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 23.69  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 24.71  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 24.72  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 24.72  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 24.73  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 24.74  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 24.74  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 24.75  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 24.76  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 25.79  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 25.79  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 1 items
[DEBUG] 25.80  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 1
[DEBUG] 25.81  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 26.78  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 26.78  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 2 items
[DEBUG] 26.79  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 1
[DEBUG] 26.79  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 26.84  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 26.84  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 1 items
[DEBUG] 26.85  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 1
[DEBUG] 26.86  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
```

Figure 40

```
[CASE 4 ] NO HTPF

Reserve MUTEX so all tasks are enqueued
[DEBUG] 30.77  M2MB_main:387 - msgSupport{HPTF_SUPPORT}$ freepos = 0 | evaluate[freepos]= 3
[DEBUG] 30.78  M2MB_main:416 - msgSupport2{HPTF_SUPPORT2}$ freepos = 1 | evaluate[freepos]= 4
[DEBUG] 30.79  M2MB_main:223 - msgProducer{PRODUCER}$ producer: freepos = 2 | evaluate[freepos]= 1
[DEBUG] 35.85  M2MB_main:586 - M2MB_main{M2M_DamsStart}$ EVALUATE SEQUENCE IS 3 4 1. Expected: 3 4 1
NO HTPF OK

[CASE 4.1 ] HTPF USED

Reserve MUTEX so all tasks are enqueued
m2mb_os_mtx_hptf OK
[DEBUG] 41.98  M2MB_main:223 - msgProducer{PRODUCER}$ producer: freepos = 0 | evaluate[freepos]= 1
[DEBUG] 41.99  M2MB_main:387 - msgSupport{HPTF_SUPPORT}$ freepos = 1 | evaluate[freepos]= 3
[DEBUG] 42.00  M2MB_main:416 - msgSupport2{HPTF_SUPPORT2}$ freepos = 2 | evaluate[freepos]= 4
[DEBUG] 44.03  M2MB_main:650 - M2MB_main{M2M_DamsStart}$ EVALUATE SEQUENCE IS 1 3 4, expected 1 3 4
HTPF DEMO OK
The application has ended...
```

Figure 41

3.1.26 SMS PDU

Sample application showcasing how to create and decode PDUs to be used with m2mb_sms_* API set. A SIM card and antenna must be present. Debug prints on **AUX UART**

Features

- How to enable SMS functionality
- How to use encode an SMS PDU to be sent with m2mb_api
- How to decode a received SMS response from PDU to ASCII mode.

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Init sms functionality
- Create PDU from text message
- Send message to destination number
- Wait for response
- When SMS PDU response is received, decode it and print information about it, plus the message content

```
m2mb_sms_init() succeeded

Sending message <How are you?>...
 m2mb_sms_send() - succeeded
M2MB_SMS_SEND_RESP Callback
Send resp msg ID 10
SMS received!
SMS correctly received!

Reading SMS from memory...
 m2mb_sms_read() request succeeded

--- SMS read ---
SMS tag M2MB_SMS_TAG_MT_NOT_READ
SMS format M2MB_SMS_FORMAT_3GPP
Code type: 0
Sender type: 145
Msg len: 12
Msg bytes: 11
Msg date 19/7/17 16:7:58 (timezone: 2)
Received SMS, content: <<Fine thanks >>
Sender: +[redacted]
```

Figure 42

3.1.27 SW Timer (Software Timer)

The sample application shows how to use SW Timers M2MB API. Debug prints on **AUX UART**

Features

- How to open configure a SW timer
- How to use the timer to manage recurring events

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create sw timer structure
- Configure it with 4 seconds timeout, periodic timer (auto fires when expires)
- Init the timer with the parameters
- Start the timer
- Wait 10 seconds
- Stop the timer

`timerCb`

- Print a message with inside the callback

```
Starting SW Timers demo app. This is v1.0.7 built on Apr  7 2020 09:51:25.
timer expired!
[DEBUG] 21.41  M2MB_main.c:59 - timerCb{pubTspt_0}$ timer handle: 0x4002b004
timer expired!
[DEBUG] 25.47  M2MB_main.c:59 - timerCb{pubTspt_0}$ timer handle: 0x4002b004
stopping the timer
Stop a running timer: success
Application end
```

Figure 43

3.1.28 Secure MicroService

Sample application showcasing how to manage secure microservice functionalities. Debug prints on **AUX UART**

Features

- Write data in Secure Data Area (SDA), non protected
- Read the written data and compare with the original buffer
- Write a cripty key in Secure Data Area (SDA), non protected
- Perform a rotate of the written key data
- Perform MD5 sum of written data from TZ file
- Compare computed digest with expected one
- Write data in trust zone as a trusted object (it will not be possible to read it again but only use its content for crypto operations)
- Try to read the trusted object and verify it fails
- Rotate trusted item and verify retrieving the content fails
- compute MD5 sum of trusted item and compare with the expected one
- Try to pass data from a trusted item to a non trusted item using untrusted TZ buffers, and verify it fails

Application workflow

`M2MB_main.c`

- Write a buffer in a SDA item using `m2mb_secure_ms_write`
- Read the same item using `m2mb_secure_ms_read`
- Write a buffer containing some cripty key in a SDA item using `m2mb_secure_ms_write`
- Rotate the content of the key item
- Read it with `m2mb_secure_ms_read`
- Load the key content using `m2mb_secure_ms_crypto_alloc` and `m2mb_secure_crypto_add_it` in a SECURE_MS buffer
- Compute MD digest with `m2mb_secure_ms_crypto_md`
- Write a buffer containing some cripty key in a SDA item using `m2mb_secure_ms_write` but with **TRUSTED** option in `m2mb_secure_ms_open`
- Verify that `m2mb_secure_ms_read` on the trusted item fails
- Verify that `m2mb_secure_ms_crypto_rotate` fails for the trusted item
- Verify the MD5 digest
- Try to copy the trusted item data in a SECURE_MS buffer with `m2mb_secure_ms_crypto_alloc` and `m2mb_secure_crypto_add_item`, then load it in an untrusted object with `m2mb_secure_ms_crypto_write`, and verify it fails.

```
Starting secure ms demo app. This is v1.0.13-C1 built on Jul 30 2020 12:19:02.
Writing data in normal item
Stored input data in Secure Data Area
Reading data from normal item
Data length in SDA: 11 bytes
Securely loaded the data from the SDA
Read 11 bytes: <hello world>
original and retrieved strings are the same

Writing key in normal item
Stored input data in Secure Data Area

Rotate data in normal item
Original key: AA_THIS_IS_MY_SECRET_KEY_BB
Rotated key:

Compute MD5 of data in normal item
Data length in SDA: 27 bytes
MD5: 8EDAD26E26E1C74C7C02386C1C7F541D
hash is the expected one!

Writing data in trusted item
Stored input data in Secure Data Area
Reading data from trusted item (should fail!)
Data length in SDA: 27 bytes
m2mb_secure_ms_read() failed for trusted item, as expected!

Rotate data in trusted item
[ERROR] 17.01  M2MB_main:329 - read_rotate{M2M_DamsStart}$ Cannot read data from SECURE_MS_BUFFER to user buffer
Original key: AA_THIS_IS_MY_SECRET_KEY_BB
Rotated key:

Compute MD5 of data in trusted item
Data length in SDA: 27 bytes

MD5: 8EDAD26E26E1C74C7C02386C1C7F541D
Hash is the expected one!

Try to pass data from trusted to untrusted through TZ buffers
Cannot store data from SECURE_MS_BUFFER to SDA 'non-trusted', as expected
```

Figure 44

3.1.29 TCP IP

Sample application showcasing TCP echo demo with M2MB API. Debug prints on **AUX UART**

Features

- How to check module registration and activate PDP context
- How to open a TCP client socket
- How to communicate over the socket

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Connect to the server
- Send data and receive response
- Close socket
- Disable PDP context

```
Starting TCP-IP demo app. This is v1.0.7 built on Mar 26 2020 16:20:30.
[DEBUG] 21.23  m2m_tcp_test.c:201 - M2M_msgTCPTask{TCP_TASK}$ INIT
[DEBUG] 21.25  m2m_tcp_test.c:217 - M2M_msgTCPTask{TCP_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.26  m2m_tcp_test.c:223 - M2M_msgTCPTask{TCP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.26  m2m_tcp_test.c:231 - M2M_msgTCPTask{TCP_TASK}$ Waiting for registration...
[DEBUG] 21.28  m2m_tcp_test.c:128 - NetCallback{pubTspt_0}$ Module is registered to cell 0x816B!
[DEBUG] 21.29  m2m_tcp_test.c:244 - M2M_msgTCPTask{TCP_TASK}$ Pdp context activation
[DEBUG] 21.30  m2m_tcp_test.c:248 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.34  m2m_tcp_test.c:263 - M2M_msgTCPTask{TCP_TASK}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 24.52  m2m_tcp_test.c:155 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.52  m2m_tcp_test.c:158 - PdpCallback{pubTspt_0}$ IP address: 83.225.44.56
[DEBUG] 24.54  m2m_tcp_test.c:273 - M2M_msgTCPTask{TCP_TASK}$ Creating Socket...
[DEBUG] 24.54  m2m_tcp_test.c:284 - M2M_msgTCPTask{TCP_TASK}$ Socket created
[DEBUG] 24.55  m2m_tcp_test.c:294 - M2M_msgTCPTask{TCP_TASK}$ Socket ctx set to 3
[DEBUG] 24.95  m2m_tcp_test.c:307 - M2M_msgTCPTask{TCP_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 25.17  m2m_tcp_test.c:322 - M2M_msgTCPTask{TCP_TASK}$ Socket Connected!
[DEBUG] 25.18  m2m_tcp_test.c:329 - M2M_msgTCPTask{TCP_TASK}$ Sending data over socket..
[DEBUG] 25.19  m2m_tcp_test.c:342 - M2M_msgTCPTask{TCP_TASK}$ Data send successfully (16 bytes)
[DEBUG] 27.20  m2m_tcp_test.c:356 - M2M_msgTCPTask{TCP_TASK}$ trying to receive 16 bytes..
[DEBUG] 27.21  m2m_tcp_test.c:364 - M2M_msgTCPTask{TCP_TASK}$ Data received (16): <hello from m2mb!>
[DEBUG] 27.21  m2m_tcp_test.c:373 - M2M_msgTCPTask{TCP_TASK}$ application exit
[DEBUG] 27.22  m2m_tcp_test.c:385 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 27.24  m2m_tcp_test.c:388 - M2M_msgTCPTask{TCP_TASK}$ Application complete.
[DEBUG] 29.43  m2m_tcp_test.c:164 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 45

3.1.30 TCP Socket status

Sample application showcasing how to check a TPC connected socket current status. Debug prints on **AUX UART**

Features

- How to check module registration and activate PDP context
- How to open a TCP client socket
- How to check if the TCP socket is still valid

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Connect to the server
- Check in a loop the current socket status using the adv_select function with a 2 seconds timeout
- Close socket when the remote host closes it
- Disable PDP context

```
Starting TCP socket status check demo app. This is v1.0.14-C1 built on Sep  8 2020 14:59:25.
[DEBUG] 21.33  m2m_tcp_tes:324 - M2M_msgTCPTask{TCP_TASK}$ INIT
[DEBUG] 21.34  m2m_tcp_tes:338 - M2M_msgTCPTask{TCP_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.34  m2m_tcp_tes:344 - M2M_msgTCPTask{TCP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.35  m2m_tcp_tes:352 - M2M_msgTCPTask{TCP_TASK}$ Waiting for registration...
[DEBUG] 21.36  m2m_tcp_tes:365 - M2M_msgTCPTask{TCP_TASK}$ Pdp context activation
[DEBUG] 21.37  m2m_tcp_tes:369 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.41  m2m_tcp_tes:384 - M2M_msgTCPTask{TCP_TASK}$ Activate PDP with APN NXT17.NET....
[DEBUG] 24.09  m2m_tcp_tes:281 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.10  m2m_tcp_tes:284 - PdpCallback{pubTspt_0}$ IP address: 100.77.5.223
[DEBUG] 24.10  m2m_tcp_tes:394 - M2M_msgTCPTask{TCP_TASK}$ Creating Socket...
[DEBUG] 24.11  m2m_tcp_tes:405 - M2M_msgTCPTask{TCP_TASK}$ Socket created
[DEBUG] 24.11  m2m_tcp_tes:415 - M2M_msgTCPTask{TCP_TASK}$ Socket ctx set to 3
[DEBUG] 24.60  m2m_tcp_tes:428 - M2M_msgTCPTask{TCP_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 24.93  m2m_tcp_tes:443 - M2M_msgTCPTask{TCP_TASK}$ Socket Connected!
[DEBUG] 26.98  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 29.03  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...

...

[DEBUG] 82.18  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 84.23  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 86.28  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 88.31  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 88.90  m2m_tcp_tes:154 - adv_select{TCP_TASK}$ Data is available on socket <0x40032b3c>
[DEBUG] 88.92  m2m_tcp_tes:160 - adv_select{TCP_TASK}$ There are <0> pending bytes on socket
Socket was closed by remote!
[DEBUG] 88.92  m2m_tcp_tes:494 - M2M_msgTCPTask{TCP_TASK}$ application exit
[DEBUG] 88.94  m2m_tcp_tes:506 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 88.94  m2m_tcp_tes:509 - M2M_msgTCPTask{TCP_TASK}$ Application complete.
[DEBUG] 89.31  m2m_tcp_tes:290 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 46

3.1.31 TCP Server

Sample application showcasing TCP listening socket demo with M2MB API. Debug prints on **AUX UART**

Features

- How to check module registration and activate PDP context
- How to open a TCP listening socket
- How to manage external hosts connection and exchange data

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and set it in non-blocking mode
- Bind the socket to the listening port
- Start listening for incoming connection
- Check if a connection is incoming using m2mb_socket_bsd_select function
- If a client connects, perform accept on the child socket
- Send a “START” message to the client
- Send some data
- Wait for data from client and print it
- Close the child socket
- Start listening again, up to 3 times
- Close listening socket
- Disable PDP context

Debug Log

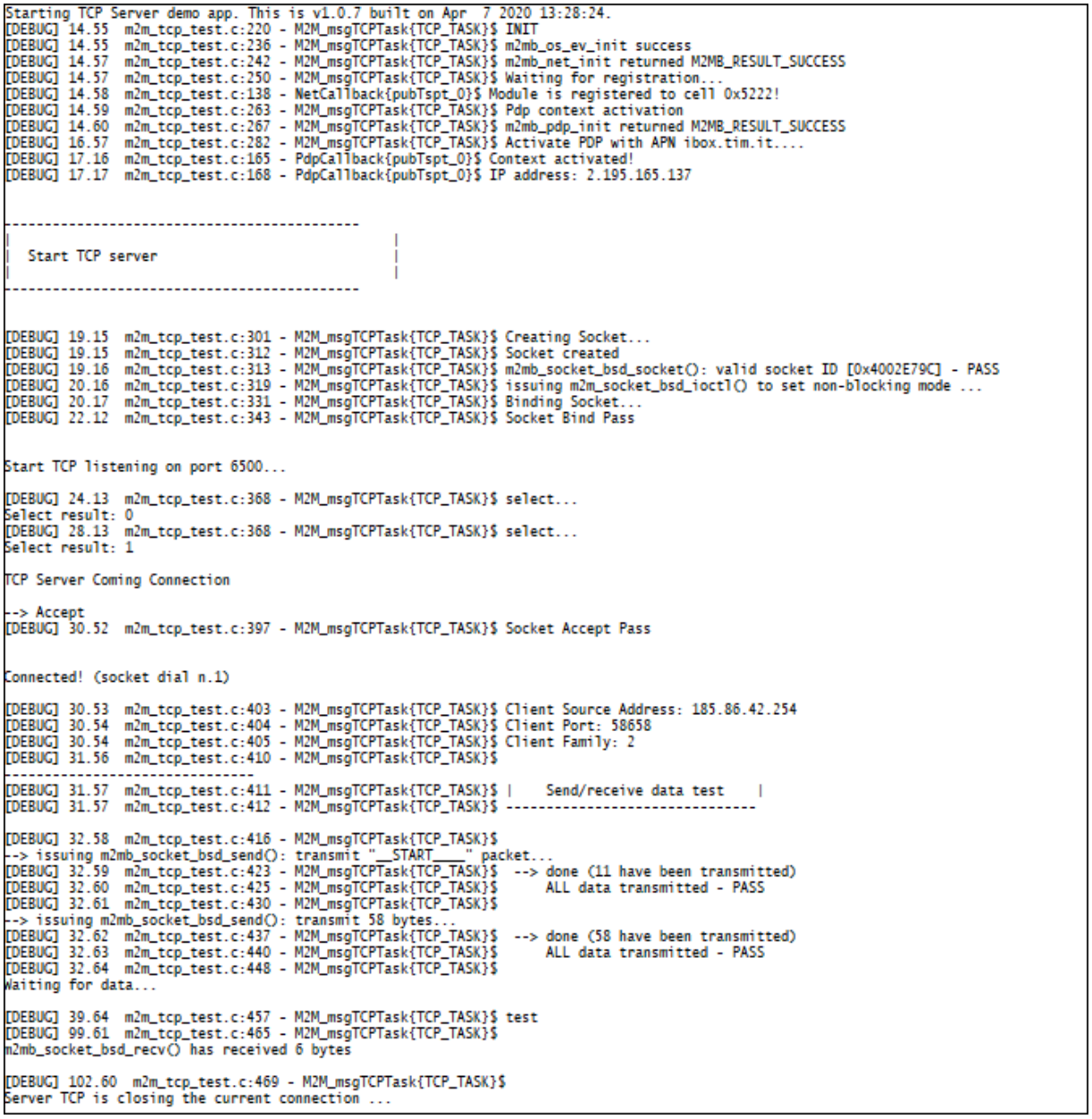

Figure 47

Data on a PuTTY terminal

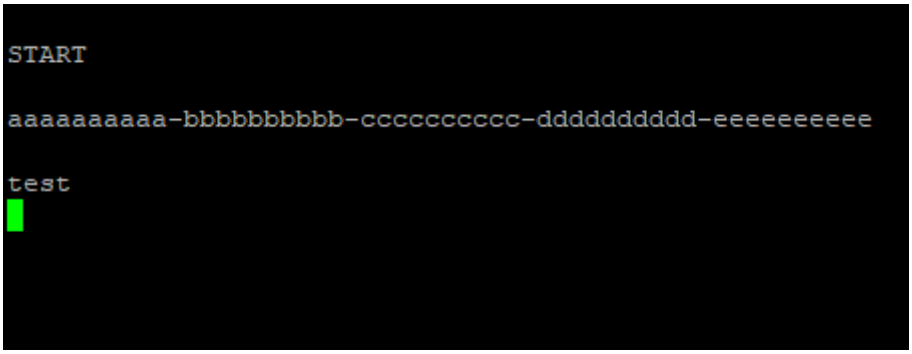
START
aaaaaaaaaa-bbbbbbbbbb-cccccccccc-dddddddddd-eeeeeeeeee
test

Figure 48

3.1.32 TLS SSL Client

Sample application showcasing TLS/SSL with client certificates usage with M2MB API. Debug prints on **AUX UART**

Features

- How to check module registration and enable PDP context
- How to open a SSL client socket
- How to communicate over SSL socket

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create a task to manage the connection and start it

`ssl_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Connect to the server over TCP socket
- Initialize the TLS parameters (TLS1.2) andh auth mode (server+client auth in the example)
- Create SSL context
- Read certificates files and store them
- Create secure socket and connect to the server using SSL
- Send data and receive response
- Close secure socket
- Close socket
- Delete SSL context
- Disable PDP context

The application requires the certificates to be stored in `/test_ssl_certs/` folder. It can be created with `AT#M2MMKDIR=/test_ssl_certs`

```
Starting TLS-SSL demo app. This is v1.0.7 built on Mar 26 2020 16:27:00.
[DEBUG] 21.27  ssl_test.c:253 - msgHTTPSTask{TLS_TASK}$ INIT
[DEBUG] 21.27  ssl_test.c:267 - msgHTTPSTask{TLS_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.28  ssl_test.c:271 - msgHTTPSTask{TLS_TASK}$ Init SSL session test app
[DEBUG] 21.28  ssl_test.c:286 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_config sslConfigHndl = 0x400330a8, sslRes= 0
[DEBUG] 21.29  ssl_test.c:295 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_config PASSED
[DEBUG] 21.30  ssl_test.c:307 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_ctxt PASSED
[DEBUG] 21.31  ssl_test.c:312 - msgHTTPSTask{TLS_TASK}$ loading CA CERT from file /test_ssl_certs/modulesCA.crt
[DEBUG] 21.32  ssl_test.c:316 - msgHTTPSTask{TLS_TASK}$ file size: 1740
[DEBUG] 21.32  ssl_test.c:329 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1740
Buffer successfully received from file. 1740 bytes were loaded.
Closing file.
[DEBUG] 21.34  ssl_test.c:458 - msgHTTPSTask{TLS_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.35  ssl_test.c:466 - msgHTTPSTask{TLS_TASK}$ Waiting for registration...
[DEBUG] 21.36  ssl_test.c:172 - NetCallback{pubTspt_0}$ Module is registered to cell 0xC4CF!
[DEBUG] 21.36  ssl_test.c:478 - msgHTTPSTask{TLS_TASK}$ Pdp context activation
[DEBUG] 21.37  ssl_test.c:482 - msgHTTPSTask{TLS_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.41  ssl_test.c:497 - msgHTTPSTask{TLS_TASK}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 24.24  ssl_test.c:198 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.24  ssl_test.c:201 - PdpCallback{pubTspt_0}$ IP address: 37.118.158.27
[DEBUG] 24.25  ssl_test.c:515 - msgHTTPSTask{TLS_TASK}$ Creating Socket...
[DEBUG] 24.26  ssl_test.c:526 - msgHTTPSTask{TLS_TASK}$ Socket created
[DEBUG] 24.26  ssl_test.c:536 - msgHTTPSTask{TLS_TASK}$ Socket ctx set to 3
[DEBUG] 24.61  ssl_test.c:549 - msgHTTPSTask{TLS_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 24.87  ssl_test.c:563 - msgHTTPSTask{TLS_TASK}$ Socket Connected!
[DEBUG] 26.14  ssl_test.c:588 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_connect ret 0
[DEBUG] 28.17  ssl_test.c:594 - msgHTTPSTask{TLS_TASK}$ Sending bytes..
[DEBUG] 28.17  ssl_test.c:597 - msgHTTPSTask{TLS_TASK}$ SSL write result = 44
[DEBUG] 32.18  ssl_test.c:609 - msgHTTPSTask{TLS_TASK}$ pending bytes: 1087
[DEBUG] 32.19  ssl_test.c:612 - msgHTTPSTask{TLS_TASK}$ trying to receive bytes..
[DEBUG] 32.19  ssl_test.c:618 - msgHTTPSTask{TLS_TASK}$ Server response: (269)<HTTP/1.1 200 OK
Date: Thu, 26 Mar 2020 15:29:43 GMT
Server: Apache/2.2.15 (CentOS)
Last-Modified: Mon, 22 Jan 2018 10:57:39 GMT
ETag: "1fffc-27f-5635b4c6f12b3"
Accept-Ranges: bytes
Content-Length: 639
Connection: close
Content-Type: text/html; charset=UTF-8

>
[DEBUG] 32.23  ssl_test.c:635 - msgHTTPSTask{TLS_TASK}$ application exit
[DEBUG] 32.23  ssl_test.c:653 - msgHTTPSTask{TLS_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 32.26  ssl_test.c:656 - msgHTTPSTask{TLS_TASK}$ Application complete.
[DEBUG] 32.89  ssl_test.c:207 - PdpCallback{pubTspt_0}$ Context deactivated!
```

Figure 49

3.1.33 UDP client

Sample application showcasing UDP echo demo with M2MB API. Debug prints on **AUX UART**

Features

- How to check module registration and activate PDP context
- How to open a UDP client socket
- How to communicate over the socket

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task and start it

`m2m_udp_test.c` - Initialize Network structure and check registration - Initialize PDP structure and start PDP context - Create socket and link it to the PDP context id - Send data and receive response - Close socket - Disable PDP context

```
Starting UDP client demo app. This is v1.0.7 built on Apr  1 2020 14:57:13.
INIT
[DEBUG] 21.23  m2m_udp_test.c:223 - M2M_msgUDPTask{UDP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
Waiting for registration...
[DEBUG] 21.25  m2m_udp_test.c:131 - NetCallback{pubTspt_0}$ Module is registered to cell 0xC4CF!
[DEBUG] 21.26  m2m_udp_test.c:241 - M2M_msgUDPTask{UDP_TASK}$ Pdp context initialization
[DEBUG] 21.26  m2m_udp_test.c:245 - M2M_msgUDPTask{UDP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
Activate PDP with APN web.omnitel.it....
[DEBUG] 24.11  m2m_udp_test.c:157 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.11  m2m_udp_test.c:160 - PdpCallback{pubTspt_0}$ IP address: 109.113.222.12
[DEBUG] 24.12  m2m_udp_test.c:268 - M2M_msgUDPTask{UDP_TASK}$ Creating Socket...
[DEBUG] 24.13  m2m_udp_test.c:280 - M2M_msgUDPTask{UDP_TASK}$ Socket created
Socket ctx set to 3
[DEBUG] 24.41  m2m_udp_test.c:306 - M2M_msgUDPTask{UDP_TASK}$ Retrieved IP: 185.86.42.218
Socket ready.
Data successfully sent (16 bytes)
Socket recv...
[DEBUG] 26.47  m2m_udp_test.c:352 - M2M_msgUDPTask{UDP_TASK}$ m2mb_socket_bsd_set_sock_opt() M2MB_SOCKET_BSD_SO_RCVTIMEO - success
trying to receive 16 bytes..
Data received (16): <hello from m2mb!>
[DEBUG] 26.48  m2m_udp_test.c:377 - M2M_msgUDPTask{UDP_TASK}$ application exit
Socket Closed
[DEBUG] 26.49  m2m_udp_test.c:399 - M2M_msgUDPTask{UDP_TASK}$ m2mb_pdp_deactivate returned success
Application complete.
[DEBUG] 27.04  m2m_udp_test.c:166 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 50

3.1.34 USB Cable Check

Sample application showing how to check if USB cable is plugged in or not. Debug prints on **AUX UART**

Features

- How to open an USB channel and configure it with a callback function
- How to manage USB cable events in the callback function

Application workflow

`M2MB_main.c`

- Open UART/UART_AUX for debug
- open usb channel and set the callback
- Print greeting message
- Print current usb status

`USB_Cb`

- if the event is a connection/disconnection, show the current status

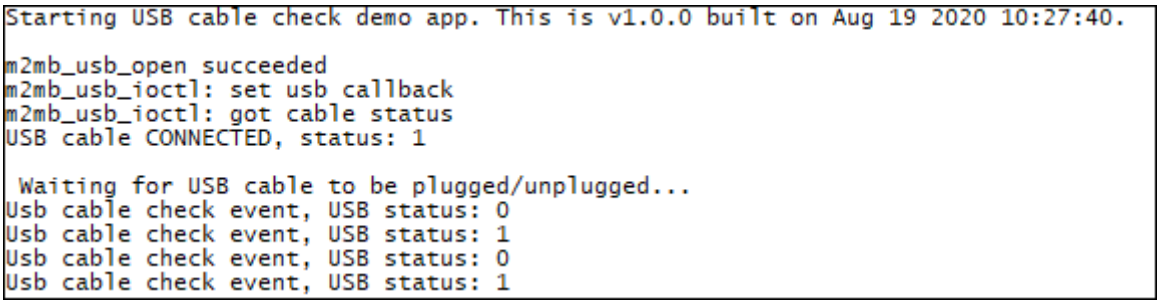

Figure 51

3.1.35 ZLIB example

Sample application showing how to compress/uncompress with ZLIB. Debug prints on **AUX UART**

Features

- How to compress a file
- How to uncompress a file

In order to execute the entire test, copy `test.gz` file into your module running the following AT command:

```
AT#M2MWRITE="/mod/test.gz",138
>>> here receive the prompt; then type or send the file, sized 138 bytes
```

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Test the compression and decompression of a data string
- Test the decompression of a .gz file (test.gz), expected to be in `/mod` folder, into its content `test.txt`. The file must be uploaded by the user (see steps above).

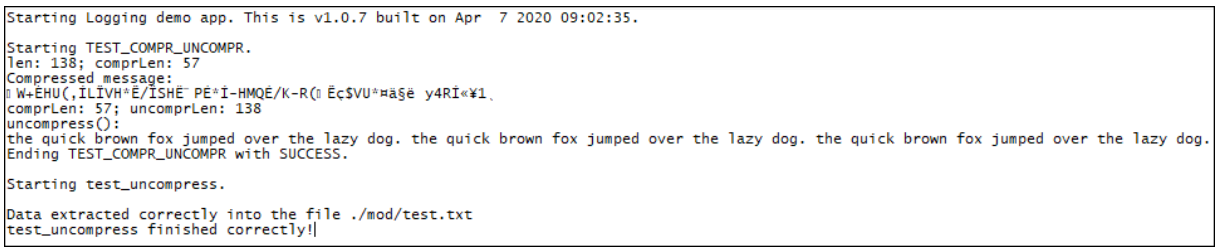

Figure 52

3.2 MISC

Applications that provide usage examples for various functionalities, without prints

3.2.1 GPIO toggle example

Sample application showcasing GPIO usage with M2MB API

Features

- How to open a gpio in output mode and change its status

3.3 MAIN UART

Applications that provide usage examples for various functionalities, log output on MAIN UART

3.3.1 ATI (AT Instance)

Sample application showing how to use AT Instance functionality (sending AT commands from code). The example supports both sync and async (using a callback) modes. Debug prints on **MAIN UART**

Features

- How to open an AT interface from the application
- How to send AT commands and receive responses on the AT interface

Application workflow, sync mode

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Init AT0 (first AT instance)
- Send AT+CGMR command
- Print response.
- Release AT0

`at_sync.c`

- Init ati functionality and take AT0
- Send AT+CGMR command, then read response after 2 seconds, then return it
- Deinit ati, releasing AT0

```
Starting AT demo app. This is v1.0.7 built on Apr  1 2020 15:12:58.
[DEBUG] 17.15  at_sync.c:53 - at_cmd_sync_init{M2M_DamsStart}$ m2mb_ati_init() on instance 0
Sending command AT+CGMR in sync mode
[DEBUG] 17.16  at_sync.c:79 - send_sync_at_command{M2M_DamsStart}$ Sending AT Command: AT+CGMR
Command response: <AT+CGMR
M0B.950004-B008

OK
>

[DEBUG] 19.21  at_sync.c:61 - at_cmd_sync_deinit{M2M_DamsStart}$ m2mb_ati_deinit() on instance 0
Application end
```

Figure 53

Application workflow, async mode

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Init AT0 (first AT instance)
- Send AT+CGMR command
- Print response.
- Release AT0

at_async.c

- Init ati functionality and take AT0, register AT events callback
- Send AT+CGMR command, wait for response semaphore (released in callback), then read it and return it
- Deinit ati, releasing AT0

```
Starting AT demo app. This is v1.0.7 built on Apr  1 2020 15:07:45.
[DEBUG] 17.13  at_async.c:116 - at_cmd_async_init{M2M_DamsStart}$ m2mb_ati_init() on instance 0
Sending command AT+CGMR in async mode
[DEBUG] 17.15  at_async.c:153 - send_async_at_command{M2M_DamsStart}$ Sending AT Command: AT+CGMR
[DEBUG] 17.15  at_async.c:169 - send_async_at_command{M2M_DamsStart}$ waiting command response...
[DEBUG] 17.17  at_async.c:88 - at_cmd_async_callback{pubTspt_0}$ Callback - available bytes: 25
[DEBUG] 17.18  at_async.c:181 - send_async_at_command{M2M_DamsStart}$ Receive response...
Command response: <AT+CGMR
M0B.950004-B008

OK
>

[DEBUG] 17.19  at_async.c:136 - at_cmd_async_deinit{M2M_DamsStart}$ m2mb_ati_deinit() on instance 0
Application end
```

Figure 54

3.3.2 AT Tunnel

Sample application showcasing how to perform an AT tunnel from Main UART to an AT instance. Debug prints on **USB1**.

Features

- How to open an AT interface from the application
- How to receive data from main UART and tunnel it to the AT interface, then report back to UART the AT response

Application workflow

`M2MB_main.c`

- Open USB1 for debug
- Initialize UART with callback function to manage input data
- Initialize AT system to manage AT commands from UART
- wait 5 minutes then deinit AT system

Main UART:

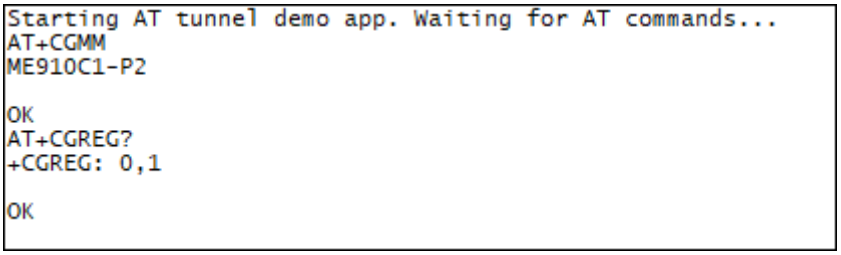

Figure 55

USB1 debug log:

```
Starting AT tunnel demo app. This is v1.0.7 built on Apr  7 2020 08:21:41.
Uart opened, setting callback for data..
[DEBUG] 17.21  M2MB_main.c:183 - at_cmd_async_init{M2M_DamsStart}$ m2mb_ati_init() on instance 0
[DEBUG] 20.43  M2MB_main.c:144 - UART_Cb{pubTspt_0}$ Received 8 bytes
[DEBUG] 20.43  M2MB_main.c:84 - msgUARTTask{uart_task}$ Received data on uart, read it and send on ATI
UART IN: <AT+CGMM
>. Sending to ATI...
[DEBUG] 20.43  M2MB_main.c:171 - at_cmd_async_callback{pubTspt_0}$ Callback - available bytes: 8
[DEBUG] 20.43  M2MB_main.c:107 - msgUARTTask{uart_task}$ Received data on ATI, read it and send on UART
[DEBUG] 20.43  M2MB_main.c:116 - msgUARTTask{uart_task}$ Received: <AT+CGMM
>
[DEBUG] 20.43  M2MB_main.c:171 - at_cmd_async_callback{pubTspt_0}$ Callback - available bytes: 20
[DEBUG] 20.43  M2MB_main.c:107 - msgUARTTask{uart_task}$ Received data on ATI, read it and send on UART
[DEBUG] 20.43  M2MB_main.c:116 - msgUARTTask{uart_task}$ Received: <
ME910C1-P2

OK
>
[DEBUG] 32.82  M2MB_main.c:144 - UART_Cb{pubTspt_0}$ Received 10 bytes
[DEBUG] 32.82  M2MB_main.c:84 - msgUARTTask{uart_task}$ Received data on uart, read it and send on ATI
UART IN: <AT+CGREG?
>. Sending to ATI...
[DEBUG] 32.82  M2MB_main.c:171 - at_cmd_async_callback{pubTspt_0}$ Callback - available bytes: 10
[DEBUG] 32.82  M2MB_main.c:107 - msgUARTTask{uart_task}$ Received data on ATI, read it and send on UART
[DEBUG] 32.82  M2MB_main.c:116 - msgUARTTask{uart_task}$ Received: <AT+CGREG?
>
[DEBUG] 32.83  M2MB_main.c:171 - at_cmd_async_callback{pubTspt_0}$ Callback - available bytes: 21
[DEBUG] 32.83  M2MB_main.c:107 - msgUARTTask{uart_task}$ Received data on ATI, read it and send on UART
[DEBUG] 32.83  M2MB_main.c:116 - msgUARTTask{uart_task}$ Received: <
+CGREG: 0,1

OK
>
```

Figure 56

3.3.3 App Manager

Sample application showing how to manage AppZone apps from m2mb code. Debug prints on **MAIN UART**

Features

- How to get how many configured apps are available
- How to get the handle to manage the running app (change start delay, enable/disable)
- How to create the handle for a new binary app, enable it and set its parameters
- How to start the new app without rebooting the device, then stop it after a while.

3.3.3.1 Prerequisites

This app will try to manage another app called “second.bin”, which already exists in the module filesystem and can be anything (e.g. another sample app as GPIO toggle). the app must be built using the flag ROM_START=

in the Makefile to set a different starting address than the main app (by default, 0x40000000). For example, 0x41000000.

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- get a non existing app handle and verify it is NULL
- get the current app handle, then get the start delay **set in the INI file (so persistent)**
- change the current app delay value **in the INI file**
- verify that the change has been stored
- get current app state
- create an handle for a second application binary.
- add it to the INI file
- set its execution flag to 0
- get the delay time and the state from INI file for the new app
- get the current set address for the new app
- set the app delay **in RAM, INI will not be affected**.
- start the new app without reboot, using the right set delay
- wait some time, then get the app state and the used RAM amount
- wait 10 seconds, then stop the second app.
- set its execution flag to 1 so it will run at next boot.

```
Starting App Manager demo app. This is v1.0.14-C1 built on Sep 24 2020 12:33:25.
There are 2 configured apps.
Not existing app handle test (should be 0): 0x0
Manager app handle: 0x809e20e0
Manager app delay from nv memory: 5 seconds

Changing Manager app delay time (on non volatile configuration) to 5 seconds..
Manager app delay from nv memory is now 5 seconds
Manager app state is M2MB_APPMNG_STATE_RUN

Trying to get Second app handle...
Second app handle is valid
2nd app delay from nv memory is 1
2nd app current state is M2MB_APPMNG_STATE_READY
Second app current address is 0x41000000
Setting volatile Second app delay (not stored in nvm) to 0 seconds...
Starting Second app on the fly (without reboot)...
Waiting 2 seconds...
2nd app current state is M2MB_APPMNG_STATE_RUN
Second app is running!
Second App is using 475136 bytes of RAM
Stopping Second app now...
wait 10 seconds...
2nd app current state is M2MB_APPMNG_STATE_STOP
Set permanent run permission for Second app.
Done. Second App will also run from next boot-up
```

Figure 57

3.3.4 App update OTA via FTP

Sample application showcasing Application OTA over FTP with AZX FTP. Debug prints on **MAIN UART**

Features

- How to check module registration and activate PDP context
- How to connect to a FTP server
- How to download an application binary and update the local version

The app uses a predefined set of parameters. To load custom parameters, upload the `ota_config.txt` file (provided in project's `/src` folder) in module's `/mod` folder, for example with

```
AT#M2MWRITE="/mod/ota_config.txt",<filesize>
```

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage app OTA and start it

`ftp_utils.c`

- Set parameters to default
- Try to load parameters from `ota_config.txt` file
- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Initialize FTP client
- Connect to FTP server and log in
- Get new App binary file size on remote server
- Download the file in `/mod` folder, with the provided name
- Close FTP connection
- Disable PDP context
- Update applications configuration in **app_utils.c**

`app_utils.c`

- Set new application as default

- Delete old app binary
- Restart module

```
Starting FTP APP OTA demo app. This is v1.0.7 built on Apr  7 2020 17:04:05.
[DEBUG] 21.23  ftp_utils.c:447 - msgFTPTask{FTPOTA_TASK}$ INIT
[DEBUG] 21.25  ftp_utils.c:152 - readConfigFromFile{FTPOTA_TASK}$ Reading parameters from file
[DEBUG] 21.26  ftp_utils.c:154 - readConfigFromFile{FTPOTA_TASK}$ Opening /mod/ota_config.txt in read mode..
Set APN to: <<web.omnitel.it>>
Set FTP URL to: <<ftp.telit.com>>
Set FTP PORT to: 21
Set FTP USER to: <<[illegible]>>
Set FTP PASS to: <<[illegible]>>
Set FTP FILE URI to: <</samples/APP_OTA/helloworld.bin>>
Set LOCAL FINAL APP NAME to: <<helloworld.bin>>
Set LOCAL ORIGINAL APP NAME to: <<m2mapz.bin>>
[DEBUG] 23.53  ftp_utils.c:464 - msgFTPTask{FTPOTA_TASK}$ m2mb_os_ev_init success
[DEBUG] 23.54  ftp_utils.c:470 - msgFTPTask{FTPOTA_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.55  ftp_utils.c:478 - msgFTPTask{FTPOTA_TASK}$ Waiting for registration...
[DEBUG] 23.56  ftp_utils.c:371 - NetCallback{pubTspt_0}$ Module is registered to network
[DEBUG] 23.56  ftp_utils.c:491 - msgFTPTask{FTPOTA_TASK}$ Pdp context activation
[DEBUG] 23.57  ftp_utils.c:495 - msgFTPTask{FTPOTA_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 25.61  ftp_utils.c:504 - msgFTPTask{FTPOTA_TASK}$ Activate PDP with APN web.omnitel.it on cid 3....
[DEBUG] 26.30  ftp_utils.c:398 - PdpCallback{pubTspt_0}$ Context active
[DEBUG] 26.30  ftp_utils.c:401 - PdpCallback{pubTspt_0}$ IP address: 176.246.110.148
Start ftp client...
[DEBUG] 27.36  ftp_utils.c:533 - msgFTPTask{FTPOTA_TASK}$ Connected.
[DEBUG] 28.87  ftp_utils.c:546 - msgFTPTask{FTPOTA_TASK}$ FTP login successful.
Get remote file /samples/APP_OTA/helloworld.bin size
[DEBUG] 29.31  ftp_utils.c:568 - msgFTPTask{FTPOTA_TASK}$ Done. File size: 116224.
Starting download of remote file /samples/APP_OTA/helloworld.bin into local /mod/helloworld.bin
/samples/APP_OTA/helloworld.bin  4.68% 5440
/samples/APP_OTA/helloworld.bin  9.36% 10880
/samples/APP_OTA/helloworld.bin 14.04% 16320
/samples/APP_OTA/helloworld.bin 18.72% 21760
/samples/APP_OTA/helloworld.bin 23.40% 27200
/samples/APP_OTA/helloworld.bin 28.08% 32640
/samples/APP_OTA/helloworld.bin 32.76% 38080
/samples/APP_OTA/helloworld.bin 37.44% 43520
/samples/APP_OTA/helloworld.bin 42.13% 48960
/samples/APP_OTA/helloworld.bin 46.81% 54400
/samples/APP_OTA/helloworld.bin 51.49% 59840
/samples/APP_OTA/helloworld.bin 56.17% 65280
/samples/APP_OTA/helloworld.bin 60.85% 70720
/samples/APP_OTA/helloworld.bin 65.53% 76160
/samples/APP_OTA/helloworld.bin 70.21% 81600
/samples/APP_OTA/helloworld.bin 74.89% 87040
/samples/APP_OTA/helloworld.bin 79.57% 92480
/samples/APP_OTA/helloworld.bin 84.25% 97920
/samples/APP_OTA/helloworld.bin 88.93% 103360
/samples/APP_OTA/helloworld.bin 93.61% 108800
/samples/APP_OTA/helloworld.bin 97.42% 113220
[DEBUG] 43.54  ftp_utils.c:608 - msgFTPTask{FTPOTA_TASK}$ download successful.
FTP quit...
[DEBUG] 43.77  ftp_utils.c:632 - msgFTPTask{FTPOTA_TASK}$ Deactivating PDP
[DEBUG] 43.77  ftp_utils.c:642 - msgFTPTask{FTPOTA_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 44.20  ftp_utils.c:407 - PdpCallback{pubTspt_0}$ Context deactive
[DEBUG] 45.44  app_utils.c:76 - update_app{FTPOTA_TASK}$ Application successfully configured.
[DEBUG] 45.45  app_utils.c:82 - update_app{FTPOTA_TASK}$ Deleting old application /mod/m2mapz.bin
€ÿStarting. This is v1.0.7 built on Apr  7 2020 17:02:52. LEVEL: 2

 Start Hello world Application  [ version: 2.000000 ]

 Hello world 2.0 [ 000001 ]
 Hello world 2.0 [ 000002 ]
 Hello world 2.0 [ 000003 ]
```

Figure 58

3.3.5 CJSON example:

Sample application showcasing how to manage JSON objects. Debug prints on **MAIN UART**

Features

- How to read a JSON using cJSON library
- How to write a JSON
- How to manipulate JSON objects

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Parse an example string into a JSON object and print the result in a formatted string
- Print some test outcomes (e.g. non existing item correctly not found)
- Retrieve single elements from the parsed JSON object and use them to format a descriptive string
- Delete the JSON object
- Create a new JSON object appending elements to it
- Print the result JSON string from the object

```
Starting Logging demo app. This is v1.0.7 built on Apr  7 2020 08:33:03.
And here is what we got:
{
        "name":         "Atlantic Ocean",
        "format":       {
                "type":         "salt",
                "volume":       310410900,
                "depth":        -8486,
                "volume_percent":       23.300000,
                "tide":         -3.500000,
                "calm":         false,
                "life":         ["plankton", "corals", "fish", "mammals"]
        }
}
inexistent key not found
name found: Atlantic Ocean
format found (null)
Our JSON string contains info about an ocean named Atlantic Ocean,
has a volume of 310410900 km^3 of salt water with -8486 meters max depth,
represents 23.3% of total oceans volume,
has an average low tide of -3.5 meters,
hosts a huge number of living creatures such as plankton, corals, fish, mammals,
and is not always calm.

Let's build a TR50 command with a proprety.publish and an alarm.publish for MQTT (no auth).
And here is what we got:
{
        "1":    {
                "command":      "property.publish",
                "params":       {
                        "thingKey":     "mything",
                        "key":  "mykey",
                        "value":        123.144000
                }
        },
        "2":    {
                "command":      "alarm.publish",
                "params":       {
                        "thingKey":     "mything",
                        "key":  "mykey",
                        "state":        3,
                        "msg":  "Message."
                }
        }
}
END.
```

Figure 59

3.3.6 Crypto Elliptic Curve Cryptography (ECC) example

Sample application showcasing how to manage Elliptic Curve Cryptography functionalities. Debug prints on **MAIN UART**

Features

- How to initialize ECC contexts A (Alice) and B (Bob). Alice is emulating a remote host, from which a public key is known.
- How to generate keypairs for contexts and export public keys
- how to export keyblobs from a context (a keyblob is encrypted with hw specific keys, and can only be used on the module where it was created)
- How to save a keyblob in secured TrustZone.
- How to reload a keyblob from the TrustZone into an initialized context
- How to sign a message with ECDSA from context B (Bob) and verify it from another context A (Alice) with the signature and public key of Bob.
- How to make Bob and Alice derive a shared session keys using each other's public key.
- How to make Bob and Alice create an AES context with the newly created shared keys, encode data and decode it on the other side

Application workflow

`M2MB_main.c`

- Create Bob ECC context, create a keypair and export it in a keyblob
- Open a file in secured Trust Zone, then store the keyblob in it.
- Destroy Bob ECC context
- Recreate Bob ECC context, open the file from Trust Zone and read the keyblob.
- Import the keyblob in Bob context.
- Export Bob public key
- Create Alice ECC context, to simulate an external host. Generate a keypair and export the public key.
- Sign a message with Bob context, generating a signature.
- Use Alice to verify the signed message using Bob's signature and public key
- Derive a shared key for Bob, using Alice's public key
- Create an AES context for Bob
- Import the shared key into the AES context
- Encrypt a message using Bob's AES context.
- Derive a shared key for Alice, using Bob's public key

- Create an AES context for Alice
- Import the shared key into the AES context
- Decrypt the message using Alice’s AES context.
- Check the decrypted message and the original one match
- Clear all resources

```
Starting Crypto ECC demo app. This is v1.0.9-C1 built on May 11 2020 16:30:23.

Bob (local) and Alice (remote) scenario
Bob's keypair generated
Bob's keyblob length is 224
Bob exported the keyblob to be securely stored.

Bob already had an item in Secure Data Area, it was removed to create a new one
Bob securely saved the keyblob in Secure Data Area
Releasing resources

Close Bob's context...
Done. Now Bob context does not exist anymore.

Re-initialize Bob Context and load the keyblob from the secure zone
Bob securely loaded the keyblob from the SDA
Import keyblob in Bob's context..
Done. Now export Bob's public key...
Bob's public key successfully exported

Alice's keypair generated
Alice's public key successfully exported

Bob's message signed with ECDSA!
Alice verified bob's message with his pubkey and signature!

------------------------------------------------------------
Bob and Alice will now exchange a message with AES encrypt
------------------------------------------------------------

Bob retrieved the generated shared key size
Bob's shared keyblob length is: 32. Allocate the required memory to store it.
Bob created a shared key using Alice's public key!

Bob created an AEX context to exchange encrypted data with Alice
Bob's AES context imported the shared keyblob
Bob Encrypted the message using AES and the shared key!
Encrypted data:
 94EE531E3B84B2A4EF05502186BFF5DA

Alice retrieved the generated shared key size
Alice's shared keyblob length is: 32. Allocate the required memory to store it.
Alice created a shared key using Bob's public key!

Alice created an AEX context to exchange encrypted data with Bob
Alice's AES context imported the shared keyblob
Alice decrypted the message using AES and the shared key!
Decrypted:
 414094941E8942A4445548035BFAE943

Original, plain message:
 414094941E8942A4445548035BFAE943

Plain and decrypted messages match!
```

Figure 60

3.3.7 EEPROM 24AA256

Sample application showing how to communicate with a MicroChip 24AA256T I2C EEPROM chip using azx eeprom utility APIs. Debug prints on **MAIN UART**

Features

- Initialize the logs on the output channel
- configure the EEPROM utility, setting the slave address and the memory parameters (page size, memory size)
- Write single bytes on a random address
- Read written bytes as a page
- Write data using pages
- Read the new data using pages
- Read again using sequential reading
- Read a single byte from a specific address
- Read next byte using read from current address
- Erase the EEPROM
- Deinit EEPROM utility

Application workflow

`M2MB_main.c`

- call azx_eeprom_init() to set the utility parameters (SDA and SCL pins, page and memory sizes)
- call azx_eeprom_writeByte() to store a single byte with value “5” at the address 0x0213
- call azx_eeprom_writeByte() to store a single byte with value “6” at the address 0x0214
- call azx_eeprom_readPages() from address 0x0213 to retrieve the 2 bytes from the EEPROM
- call azx_eeprom_writePages to write 1024 bytes from a buffer, starting from address 0x00
- call azx_eeprom_readPages() again, to read 256 bytes from address 0x00
- call azx_eeprom_readSequentially() to read 256 bytes from 0x00 by without pages (less overhead on I2C protocol)
- call azx_eeprom_readByte() to get a single byte from address 0x00
- call azx_eeprom_readByteFromCurrentAddress() to get a byte from next address (0x01)
- call azx_eeprom_eraseAll() to completely erase the EEPROM memory (this writes 0xFF in each byte)
- call azx_eeprom_readPages from address 0x0213 to get 2 bytes and verify the values have been written to 0xFF

- call azx_eeprom_deinit to close the eeprom handler and the I2C channel

```
Starting I2C EEPROM 24AA256T demo app. This is v1.0.13-C1 built on Nov  3 2020 16:28:23.

Configuring the I2C device...
Opening I2C channel /dev/I2C-160 ( device address is 0xA0 )
Writing 1 byte at address 0x0213...
Done.
Writing 1 byte at address 0x0214...
Done.
Reading the 2 bytes from address 0x0213...
Done. Data: [0xFF 0xFF]

Writing 1024 bytes at address 0x0000..
Done.

Reading 256 bytes from address 0x0000...
Done. Data:
<<ABCDEFGHIJKLMNOPQRSTUVWXYZ.....................................................abcdefghijklmnopqrstuvwxyz.....................

Reading 256 bytes sequentially from address 0x0000...
Done. Data:
<<ABCDEFGHIJKLMNOPQRSTUVWXYZ.....................................................abcdefghijklmnopqrstuvwxyz.....................


Reading 1 byte from address 0x0000...
Done. Data: 'A'

Reading 1 byte from current address (should be 0x0001)...
Done. Data: 'B'


[DEBUG] 17.47  M2MB_main:177 - run_I2C_EEPROM_Demo{M2M_DamsStart}$ Erasing all the eeprom...
[DEBUG] 28.05  M2MB_main:185 - run_I2C_EEPROM_Demo{M2M_DamsStart}$ Done

Reading the 2 bytes from address 0x0213...
Done. Data: [0xFF 0xFF]

Deinit EEPROM...
Done
```

Figure 61

3.3.8 Easy AT example

Sample application showcasing Easy AT functionalities. Debug prints on **MAIN UART**

Features

- Shows how to register custom commands

3.3.9 Events

Sample application showcasing events setup and usage. Debug prints on **MAIN UART**

Features

- How to setup OS events with a custom bitmask
- How to wait for events and generate them in callback functions to synchronize blocks of code

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create an event handler
- Create a timer to generate an event, with a 2 seconds expiration time
- Wait for a specific event bit on the event handler
- At timer expiration, set the same event bit and verify that the code flow went through after the event.

```
Starting Events demo app. This is v1.0.7 built on Apr  7 2020 08:44:29.
[DEBUG] 20.55  M2MB_main.c:171 - M2MB_main{M2M_DamsStart}$ m2mb_os_ev_init success
Set the timer attributes structure success.
Timer successfully created
[DEBUG] 20.57  M2MB_main.c:125 - setup_timer{M2M_DamsStart}$ Start the timer, success.
[DEBUG] 22.60  M2MB_main.c:60 - hwTimerCb{pubTspt_0}$ Timer Callback, generate event!
[DEBUG] 22.61  M2MB_main.c:183 - M2MB_main{M2M_DamsStart}$ event occurred!
```

Figure 62

3.3.10 Events - Barrier (multi events)

Sample application showcasing how to setup and use multiple events to create a barrier. Debug prints on **MAIN UART**

Features

- How to setup OS events to be used as a barrier
- How to wait for multiple events in the same point, and generate them in call-back functions to synchronize blocks of code

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create an event handler
- Create a timer to generate an event, with a 3 seconds expiration time
- Create another timer to generate an event, with a 6 seconds expiration time
- Start both timers
- Wait for both event bits on the event handler (each one will be set by one of the timers)
- At first timer expiration, set the first event bit and verify that the code flow does not procede.
- At second timer expiration, set the second event bit and verify that the code flow went through after the event (implementing a barrier).

```
Starting Barrier demo app. This is v1.0.7 built on Apr  7 2020 08:48:30.
[DEBUG] 20.01  M2MB_main.c:179 - M2MB_main{M2M_DamsStart}$ m2mb_os_ev_init success
Set the timer attributes structure success.
Timer successfully created with 3000 timeout (ms)
Set the timer attributes structure success.
Timer successfully created with 6000 timeout (ms)
[DEBUG] 23.08  M2MB_main.c:66 - hwTimerCb1{pubTspt_0}$ Timer Callback, generate event 1!
[DEBUG] 26.12  M2MB_main.c:75 - hwTimerCb2{pubTspt_0}$ Timer Callback, generate event 2!
[DEBUG] 26.13  M2MB_main.c:214 - M2MB_main{M2M_DamsStart}$ BOTH events occurred!
```

Figure 63

3.3.11 FOTA example

Sample application showcasing FOTA usage with M2MB API. Debug prints on **MAIN UART**

Features

- How download a delta file from a remote server
- How to apply the delta and update the module firmware

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a main task to manage connectivity.
- create a fota task to manage FOTA and start it with INIT option

`fota.c`

fotaTask()

- Initialize FOTA system then reset parameters.
- Check current FOTA state, if not in IDLE, return error.
- Send a message to mainTask so networking is initialized.
- after PdPCallback() notifies the correct context activation, configure the fota client parameters such as FTP server URL, username and password
- get delta file from server. when it is completed, FOTADownloadCallback is called.
- If delta download went fine, check it.
- If delta file is correct, apply it. Once complete, restart the module.

mainTask()

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context. Event will be received on **PdP-Callback** function
- Disable PDP context when required to stop the app

PdpCallback()

- When PDP context is enabled, send a message to fotaTask to start the download

```
Starting FOTA demo app. This is v1.0.7 built on Apr  7 2020 16:24:29.
[DEBUG] 27.68  fota.c:185 - fotaTask{FOTA_TASK}$ Init FOTA...
Session file not present, procede with FOTA...
[DEBUG] 27.69  fota.c:229 - fotaTask{FOTA_TASK}$ m2mb_fota_reset PASS
[DEBUG] 27.70  fota.c:253 - fotaTask{FOTA_TASK}$ m2mb_fota_state_get  M2MB_FOTA_STATE_IDLE
[DEBUG] 27.71  fota.c:369 - mainTask{MAIN_TASK}$ Case INIT
[DEBUG] 27.71  fota.c:374 - mainTask{MAIN_TASK}$ Case WAIT_FOR_REGISTRATION
[DEBUG] 27.72  fota.c:378 - mainTask{MAIN_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 27.72  fota.c:385 - mainTask{MAIN_TASK}$ Waiting for registration...
[DEBUG] 27.73  fota.c:130 - NetCallback{pubTspt_0}$ Module is registered to cell 0xC4CF!
[DEBUG] 27.74  fota.c:395 - mainTask{MAIN_TASK}$ REGISTERED
[DEBUG] 27.74  fota.c:400 - mainTask{MAIN_TASK}$ Pdp context activation
[DEBUG] 27.75  fota.c:404 - mainTask{MAIN_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 29.71  fota.c:419 - mainTask{MAIN_TASK}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 30.44  fota.c:151 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 30.45  fota.c:154 - PdpCallback{pubTspt_0}$ IP address: 37.118.165.97

[DEBUG] 30.45  fota.c:278 - fotaTask{FOTA_TASK}$
Trying to download "samples/FOTA/30.00.006.2_to_30.00.006.2_ME910C1_NANVWW.bin" delta file...
[DEBUG] 30.47  fota.c:288 - fotaTask{FOTA_TASK}$ m2mb_fota_get_delta OK - Waiting for the completion callback
[DEBUG] 39.03  fota.c:95 - FOTADownloadCallBack{pubTspt_0}$ FOTA download Success - performing packet validation...
[DEBUG] 39.05  fota.c:294 - fotaTask{FOTA_TASK}$ Validating delta file...
[DEBUG] 122.34  fota.c:310 - fotaTask{FOTA_TASK}$ Packet is valid, start update...
[DEBUG] 122.38  fota.c:322 - fotaTask{FOTA_TASK}$ m2mb_fota_start PASS
[DEBUG] 124.39  fota.c:335 - fotaTask{FOTA_TASK}$
Rebooting...After reboot there will be the new FW running on module!

#OTAEV: Module Upgraded To New Fw

Starting FOTA demo app. This is v1.0.7 built on Apr  7 2020 16:24:29.
[DEBUG] 33.31  fota.c:185 - fotaTask{FOTA_TASK}$ Init FOTA...

Session file is already present, stop.
```

Figure 64

3.3.12 FTP

Sample application showcasing FTP client demo with AZX FTP. Debug prints on **MAIN UART**

Features

- How to check module registration and activate PDP context
- How to connect to a FTP server
- How to exchange data with the server

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage FTP client and start it

`ftp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Init FTP client and set the debug function for it
- Connect to the server
- Perform log in
- Check remote file size and last modification time
- Download file from server to local filesystem. A data callback is set to report periodic info about the download status
- Upload the same file to the server with a different name. A data callback is set to report periodic info about the upload status
- Download another file content in a buffer instead of a file. A data callback is set to report periodic info about the download status
- Close the connection with FTP server
- Disable PDP context

```
Starting FTP demo app. This is v1.0.7 built on Apr  7 2020 11:17:36.
[DEBUG] 21.23  ftp_test.c:290 - msgFTPTask{FTP_TASK}$ INIT
[DEBUG] 21.23  ftp_test.c:304 - msgFTPTask{FTP_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.23  ftp_test.c:310 - msgFTPTask{FTP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.23  ftp_test.c:318 - msgFTPTask{FTP_TASK}$ Waiting for registration...
[DEBUG] 21.25  ftp_test.c:214 - NetCallback{pubTspt_0}$ Module is registered to network
[DEBUG] 21.26  ftp_test.c:331 - msgFTPTask{FTP_TASK}$ Pdp context activation
[DEBUG] 21.27  ftp_test.c:335 - msgFTPTask{FTP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.31  ftp_test.c:344 - msgFTPTask{FTP_TASK}$ Activate PDP with APN web.omnitel.it on cid 3....
[DEBUG] 24.09  ftp_test.c:241 - PdpCallback{pubTspt_0}$ Context active
[DEBUG] 24.10  ftp_test.c:244 - PdpCallback{pubTspt_0}$ IP address: 176.244.166.181
Start ftp client...
[DEBUG] 24.82  ftp_test.c:373 - msgFTPTask{FTP_TASK}$ Connected.
[DEBUG] 26.32  ftp_test.c:386 - msgFTPTask{FTP_TASK}$ FTP login successful.
Get remote file /samples/pattern_big.txt size
[DEBUG] 26.69  ftp_test.c:428 - msgFTPTask{FTP_TASK}$ Done. File size: 20026.
Get remote file /samples/pattern_big.txt last modification date
[DEBUG] 26.89  ftp_test.c:450 - msgFTPTask{FTP_TASK}$ Done. File last mod date: 20200407090654
.

Starting download of remote file /samples/pattern_big.txt into local /mod/_pattern_big.txt
/samples/pattern_big.txt 47.54% 9520
/samples/pattern_big.txt 100.00% 20026
[DEBUG] 29.75  ftp_test.c:488 - msgFTPTask{FTP_TASK}$ download successful.
[DEBUG] 29.76  ftp_test.c:522 - msgFTPTask{FTP_TASK}$
Local file /mod/_pattern_big.txt size: 20026

Starting upload of local file /mod/_pattern_big.txt
/mod/_pattern_big.txt 81.81% 16384
Upload successful.

Starting download of remote file /samples/pattern.txt into local buffer
Getting remote file /samples/pattern.txt size..
[DEBUG] 32.97  ftp_test.c:583 - msgFTPTask{FTP_TASK}$ Done. File size: 988.
Starting download of remote file /samples/pattern.txt to buffer
[DEBUG] 34.08  ftp_test.c:145 - buf_data_cb{FTP_TASK}$ Received START event
[DEBUG] 34.09  ftp_test.c:149 - buf_data_cb{FTP_TASK}$ Received DATA: 988 bytes on buffer 0x400399e0
[DEBUG] 34.26  ftp_test.c:153 - buf_data_cb{FTP_TASK}$ Received END event
[DEBUG] 34.26  ftp_test.c:623 - msgFTPTask{FTP_TASK}$ Download successful. Received 988 bytes<<<
0    |---------|    |---------|    |---------|    |---------|    |---------|   *
1    |    A    |    |    A    |    |    A    |    |    A    |    |    A    |    *
2    |   AAA   |    |   AAA   |    |   AAA   |    |   AAA   |    |   AAA   |     *
3    |  AAAAA  |    |  AAAAA  |    |  AAAAA  |    |  AAAAA  |    |  AAAAA  |      *
4    | AAAAAAA |    | AAAAAAA |    | AAAAAAA |    | AAAAAAA |    | AAAAAAA |       *
5    |AAAAAAAAA|    |AAAAAAAAA|    |AAAAAAAAA|    |AAAAAAAAA|    |AAAAAAAAA|        *
6    | AAAAAAA |    | AAAAAAA |    | AAAAAAA |    | AAAAAAA |    | AAAAAAA |       *
7    |  AAAAA  |    |  AAAAA  |    |  AAAAA  |    |  AAAAA  |    |  AAAAA  |      *
8    |   AAA   |    |   AAA   |    |   AAA   |    |   AAA   |    |   AAA   |     *
9    |    A    |    |    A    |    |    A    |    |    A    |    |    A    |    *
10   |---------|    |---------|    |---------|    |---------|    |---------|   *
11                                                                             *
12   |---------|    |---------|    |---------|    |---------|    |--->>>
```

Figure 65

3.3.13 File System example

Sample application showcasing M2MB File system API usage. Debug prints on **MAIN UART**

Features

- How to open a file in write mode and write data in it
- How to reopen the file in read mode and read data from it

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Open file in write mode
- Write data in file
- Close file
- Reopen file in read mode
- Read data from file and print it
- Close file and delete it

```
Starting FileSystem demo app. This is v1.0.7 build on Mar 26 2020 09:50:19. LEVEL: 2
Opening /my_text_file.txt in write mode..
Buffer written successfully into file. 15 bytes were written.
Closing file.
Opening /my_text_file.txt in read only mode..
Received 15 bytes from file:
<Hello from file>
Closing file.
Deleting File
File deleted
App Completed
```

Figure 66

3.3.14 GNSS example

Sample application showing how to use GNSS functionality. Debug prints on **MAIN UART**

Features

- How to enable GNSS receiver on module
- How to collect location information from receiver

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Init gnss, enable position report and start it.
- When a fix is available, a message will be printed by the GNSS callback function

```
START GNSS TEST APP
m2mb_gnss_enable OK
m2mb_gnss_start OK
latitude_valid: 1 - latitude: 39.228245
longitude_valid: 1 - longitude: 9.069106
altitude_valid: 1 - altitude: 12.000000
uncertainty_valid: 1 - uncertainty: 30.000000
velocity_valid: 1 - codingType: 0
speed_horizontal: 0.000000
bearing: 0.000000
timestamp_valid: 1 -timestamp: 1563376148000
speed_valid: 1 - speed: 0.060000
```

Figure 67

3.3.15 GPIO interrupt example

Sample application showing how to use GPIOs and interrupts. Debug prints on **MAIN UART**

Features

- How to open a GPIO in input mode with interrupt
- How to open a second GPIO in output mode to trigger the first one

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Open GPIO 4 as output
- Open GPIO 3 as input and set interrupt for any edge (rising and falling). **A jumper must be used to short GPIO 3 and 4 pins.**
- Toggle GPIO 4 status high and low every second
- An interrupt is generated on GPIO 3

```
Starting GPIO interrupt demo app. This is v1.0.7 built on Mar 26 2020 16:33:01.
Setting gpio 3 interrupt...
Setting GPIO 4 HIGH
CALLBACK->Interrupt on GPIO 3! Value: 1
Setting GPIO 4 LOW
CALLBACK->Interrupt on GPIO 3! Value: 0
Setting GPIO 4 HIGH
CALLBACK->Interrupt on GPIO 3! Value: 1
Setting GPIO 4 LOW
CALLBACK->Interrupt on GPIO 3! Value: 0
Setting GPIO 4 HIGH
CALLBACK->Interrupt on GPIO 3! Value: 1
Setting GPIO 4 LOW
CALLBACK->Interrupt on GPIO 3! Value: 0
```

Figure 68

3.3.16 HTTP Client

Sample application showing how to use HTTPs client functionalities. Debug prints on **MAIN UART**

Features

- How to check module registration and activate PDP context
- How to initialize the http client, set the debug hook function and the data call-back to manage incoming data
- How to perform GET, HEAD or POST operations

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage HTTP client and start it

`httpTaskCB`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create HTTP client options and initialize its functionality
- Create HTTP SSL config and initialize the SSL options
- Configure data management options for HTTP client
- Appy all configurations to HTTP client
- Perform a GET request to a server
- Disable PDP context

`DATA_CB`

- Print incoming data
- Set the abort flag to 0 to keep going.

```
Starting HTTP(s) client demo app. This is v1.0.13-C1 built on Aug 11 2020 16:56:28.
[DEBUG] 15.19  M2MB_main:259 - activatePdP{HttpClient}$ m2mb_os_ev_init success
[DEBUG] 15.20  M2MB_main:265 - activatePdP{HttpClient}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 15.20  M2MB_main:273 - activatePdP{HttpClient}$ Waiting for registration...
[DEBUG] 15.21  M2MB_main:119 - NetCallback{pubTspt_0}$ Module is registered to cell 0xC4CF!
[DEBUG] 15.22  M2MB_main:287 - activatePdP{HttpClient}$ Pdp context initialization
[DEBUG] 17.26  M2MB_main:297 - activatePdP{HttpClient}$ Activate PDP with APN NXT17.NET....
[DEBUG] 17.97  M2MB_main:146 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 17.97  M2MB_main:149 - PdpCallback{pubTspt_0}$ IP address: 100.77.54.97
Performing a GET request...

Host Address: linux-ip.net Port 80

Socket connected! |
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="author" content="Martin A. Brown" />
    <meta name="robots" content="index, follow"/>

    <meta property="og:title" content="http://linux-ip.net/"/>
    <meta property="og:url" content="http://linux-ip.net"/>
    <meta property="og:site_name" content="http://linux-ip.net/"/>
    <meta property="og:type" content="website"/>

    <link rel="canonical" href="http://linux-ip.net" />

    <title>http://linux-ip.net/</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" type="text/css" href="//netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-awesome.css" />
    <link rel="stylesheet" type="text/css" href="//netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css" />
    <link rel="stylesheet" type="text/css" href="http://linux-ip.net/theme/css/main.css" />

</head>

   ...
    <footer id="site-footer">
        <div class="row-fluid">
            <div class="span10 offset1">
                <address>
                    <p>
                        Powered by <a href="http://getpelican.com/">Pelican</a>
                        and <a href="http://python.org">Python</a>.
                        Theme based on <a href="http://github.com/jsliang/pelican-fresh/">Fresh</a>
                        by <a href="http://jsliang.com/">jsliang</a>
                    </p>
                </address>
            </div>
        </div>
    </footer>
</body>
</html>
Done.
[DEBUG] 22.44  M2MB_main:155 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 69

3.3.17 HW Timer (Hardware Timer)

The sample application shows how to use HW Timers M2MB API. Debug prints on **MAIN UART**

Features

- How to open configure a HW timer
- How to use the timer to manage recurring events

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create hw timer structure
- Configure it with 100 ms timeout, periodic timer (auto fires when expires) and autostart
- Init the timer with the parameters
- Wait 10 seconds
- Stop the timer

`TimerCb`

- Print a message with an increasing counter

```
Starting HW Timers demo app. This is v1.0.7 built on Mar 26 2020 13:04:14.
[DEBUG] 14.06  M2MB_main.c:114 - M2MB_main{M2M_DamsStart}$ Set the timer attributes structure: success.
Timer successfully created
Start the timer, success.
[DEBUG] 14.18  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [0]
[DEBUG] 14.28  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [1]
[DEBUG] 14.38  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [2]
[DEBUG] 14.48  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [3]
[DEBUG] 14.58  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [4]
[DEBUG] 14.69  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [5]
[DEBUG] 14.79  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [6]
[DEBUG] 14.88  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [7]
[DEBUG] 14.98  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [8]
[DEBUG] 15.08  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [9]


[DEBUG] 23.90  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [96]
[DEBUG] 24.01  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [97]
[DEBUG] 24.11  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [98]
Stop a running timer: success
Application end
```

Figure 70

3.3.18 Hello World

The application prints “Hello World!” over selected output every two seconds. Debug prints on **MAIN UART**, using AZX log example functions

Features

- How to open an output channel using AZX LOG sample functions
- How to print logging information on the channel using AZX LOG sample functions

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print “Hello World!” every 2 seconds in a while loop

```
Starting. This is v1.0.7 built on Mar 26 2020 09:34:16. LEVEL: 2

 Start Hello world Application  [ version: 2.000000 ]

 Hello world 2.0 [ 000001 ]
 Hello world 2.0 [ 000002 ]
 Hello world 2.0 [ 000003 ]
 Hello world 2.0 [ 000004 ]
 Hello world 2.0 [ 000005 ]
 Hello world 2.0 [ 000006 ]
 Hello world 2.0 [ 000007 ]
 Hello world 2.0 [ 000008 ]
 Hello world 2.0 [ 000009 ]
```

Figure 71

3.3.19 I2C example

Sample application showing how to communicate with an I2C slave device. Debug prints on **MAIN UART**

Features

- How to open a communication channel with an I2C slave device
- How to send and receive data to/from the slave device

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Open I2C bus, setting SDA an SCL pins as 2 and 3 respectively
- Set registers to configure accelerometer -Read in a loop the 6 registers carrying the 3 axes values and show the g value for each of them

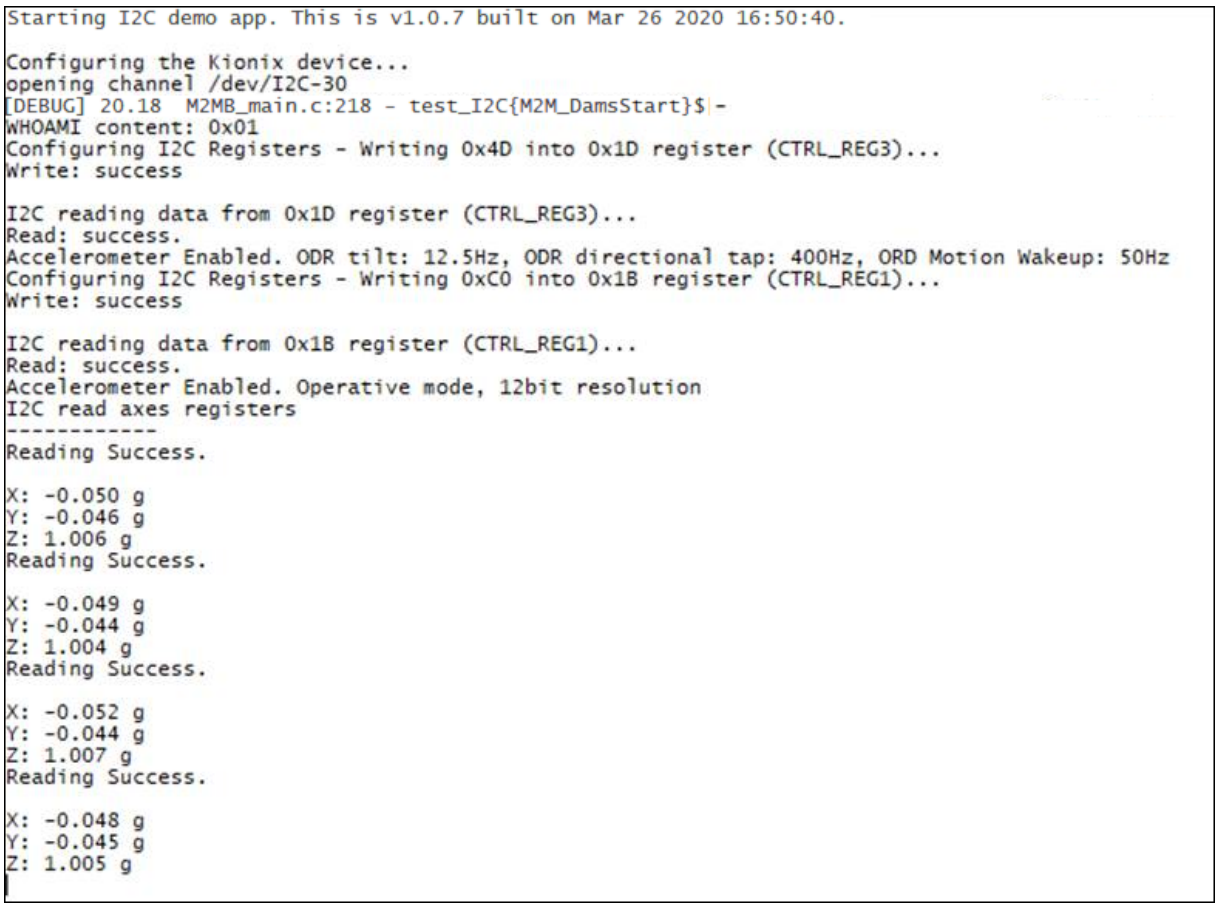

Figure 72

3.3.20 I2C Combined

Sample application showing how to communicate with an I2C slave device with I2C raw mode. Debug prints on **MAIN UART**

Features

- How to open a communication channel with an I2C slave device
- How to send and receive data to/from the slave device using raw mode API

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Open I2C bus, setting SDA an SCL pins as 2 and 3 respectively
- Set registers to configure accelerometer -Read in a loop the 6 registers carrying the 3 axes values and show the g value for each of them

```
Starting I2C raw demo app. This is v1.0.13-C1 built on Jul 30 2020 11:28:18.

Configuring the I2C device...
Opening I2C channel /dev/I2C-30 ( device address is 0x0F << 1 )
Accelerometer Enabled. ODR tilt: 12.5Hz, ODR directional tap: 400Hz, ORD Motion Wakeup: 50Hz
Accelerometer Enabled. Operative mode, 12bit resolution
I2C read axes registers
------------

X: 0.000 g
Y: 0.000 g
Z: 0.000 g

X: -0.270 g
Y: 0.016 g
Z: 0.917 g

X: -0.268 g
Y: 0.013 g
Z: 0.925 g

X: -0.271 g
Y: 0.015 g
Z: 0.922 g

X: -0.267 g
Y: 0.016 g
Z: 0.918 g

X: -0.274 g
Y: 0.019 g
Z: 0.915 g
```

Figure 73

3.3.21 LWM2M

Sample application showcasing LWM2M client usage with M2MB API. Debug prints on **MAIN UART**

Features

- Configure LWM2M client and enable it
- Create an instance of a custom object
- Set an integer value on a read only resource
- Set two integer values on a multi-instance read only resource
- write a string on a read/write resource
- Manage exec requests from the portal
- Manage write, read and monitoring resources changed from the portal

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create a task to manage the LWM2M client and start it

`lwm2m_demo.c`

`msgLWM2MTask` - Check registration status

- Configure APN to the correct one for CID 1
- Initialize LWM2M client,
- Check for XML file fo custom object
- Enable unsolicited messages from client
- Create a task (lwm2m_taskCB is its callback function)to manage events from Portal
- Enable LwM2M client
- Create a new instance for the custom object
- Wait for client to register to Portal
- Send integer and string values
- Wait for events from server

`lwm2mIndicationCB`

- Manage events arriving from client (operations completion status and unsolicited events)
- Run lwm2m_taskCB when a monitored resource changes, to manage the action to be done

3.3.21.1 Custom Object configuration

The XML file content must be loaded on the Telit IoT Portal for the demo application to be fully executed.

First, enter Developer section from the top menu

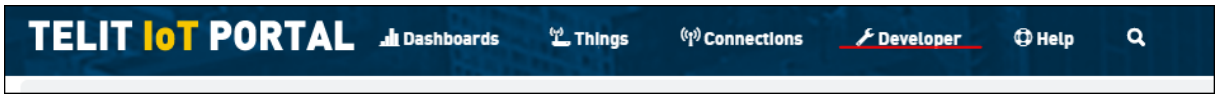

Figure 74

Choose Object Registry

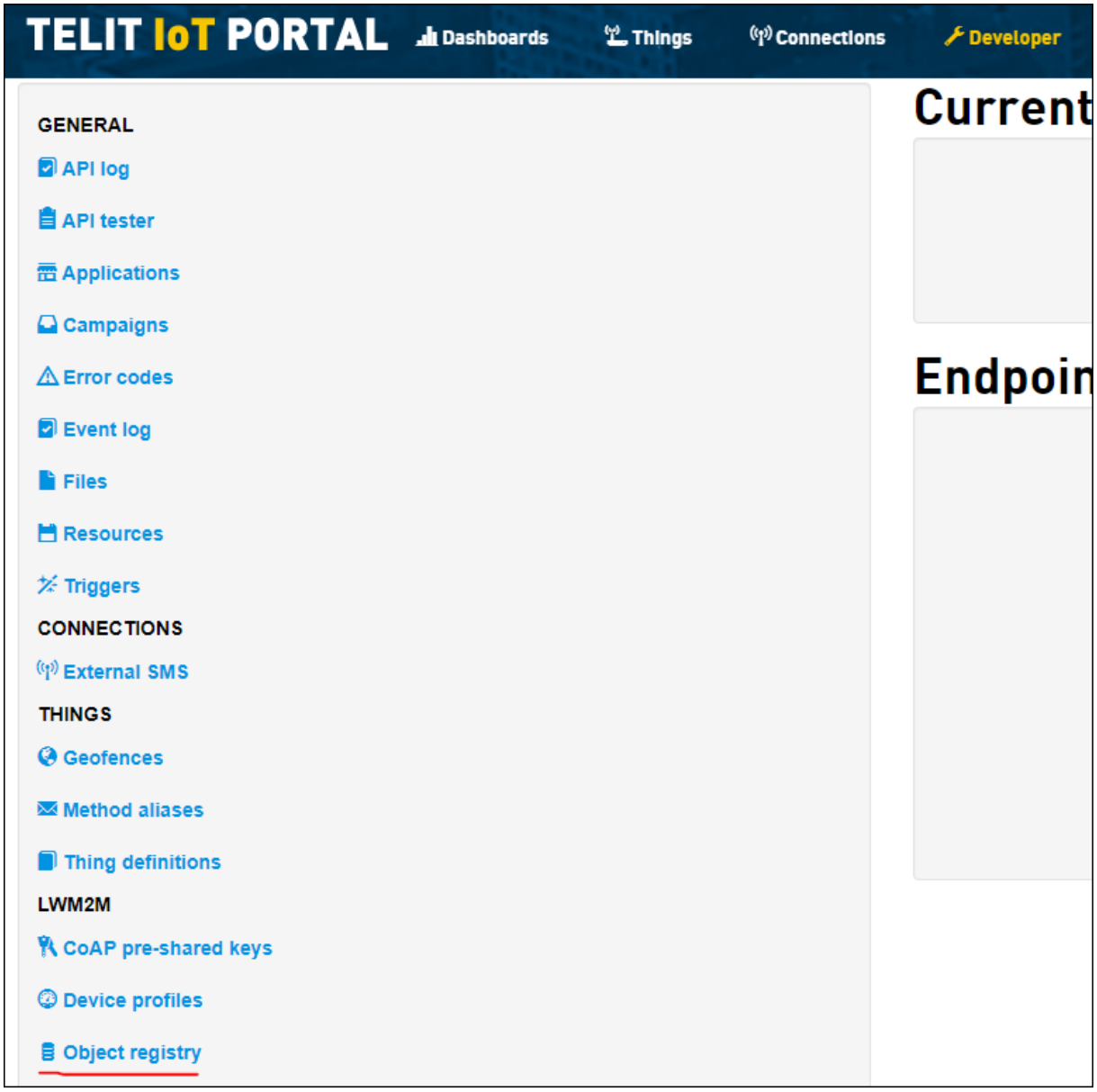

Figure 75

Create a New Object

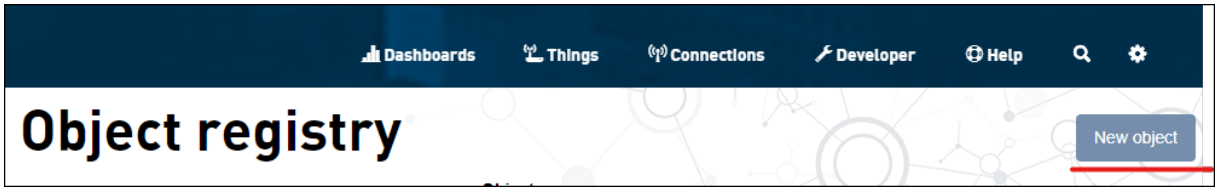

Figure 76

Copy the xml file content and paste it in the new Object form

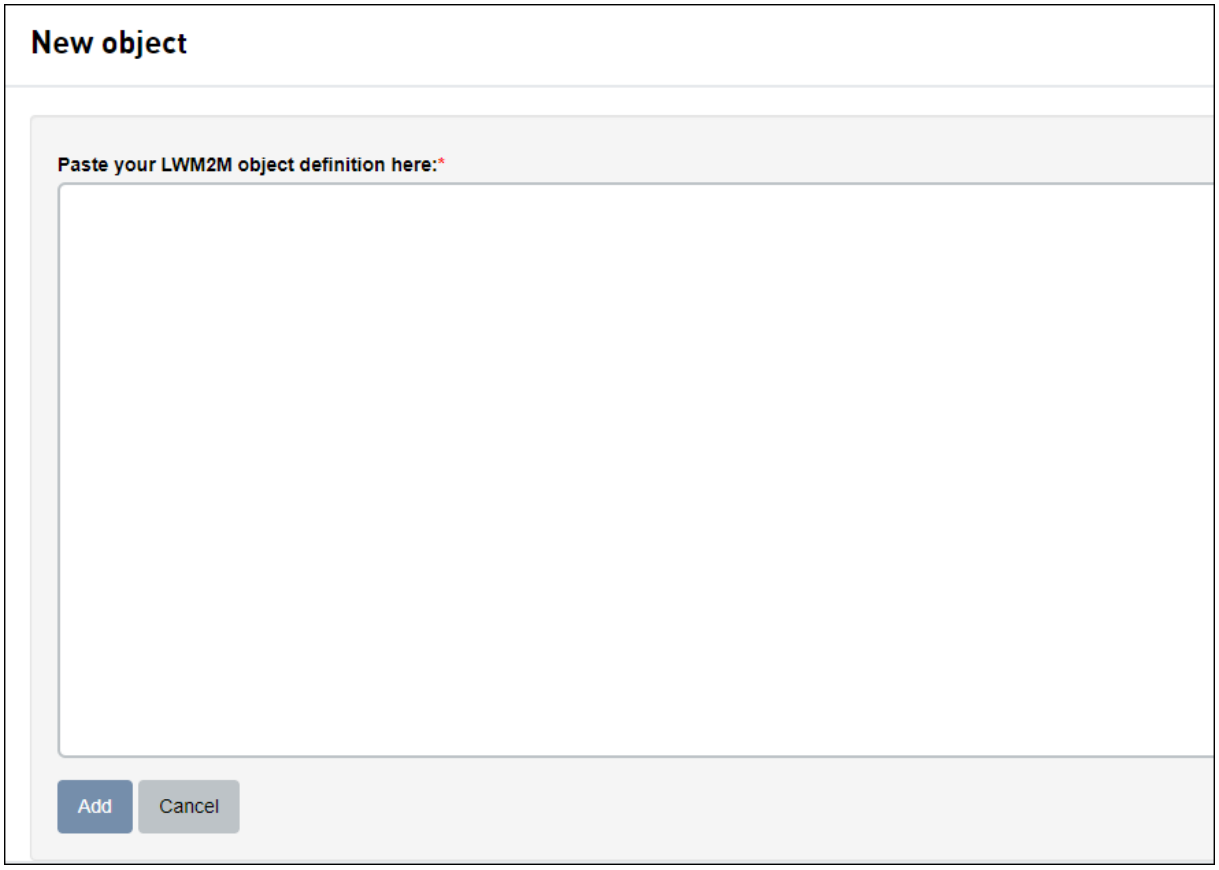

Figure 77

Also, the application requires the XML file `/xml/object_35000.xml` (provided with the sample files) to be stored in module's `/XML/` folder. It can be done with

`AT#M2MWRITE=/XML/object_35000.xml,<size_in_bytes>`

To load the XML file in the module, Telit AT Controller (TATC) can be used. Once the command above is issued, press the load content button:

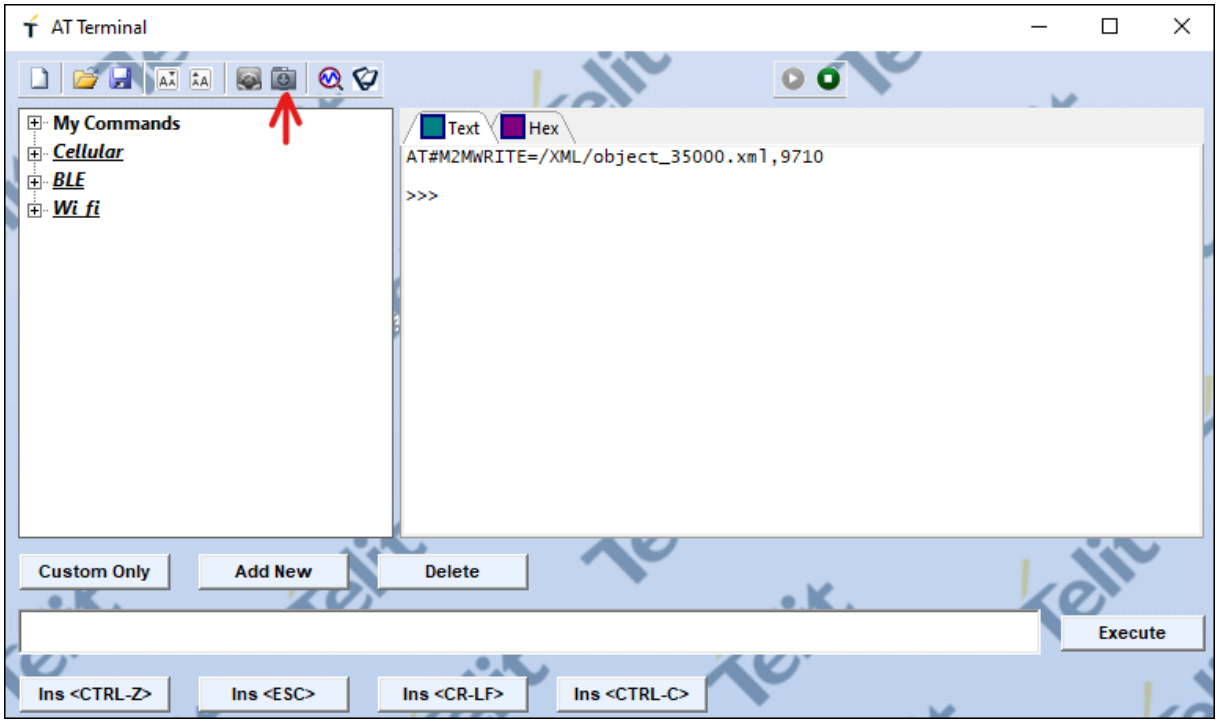

Figure 78

Select the file from your computer

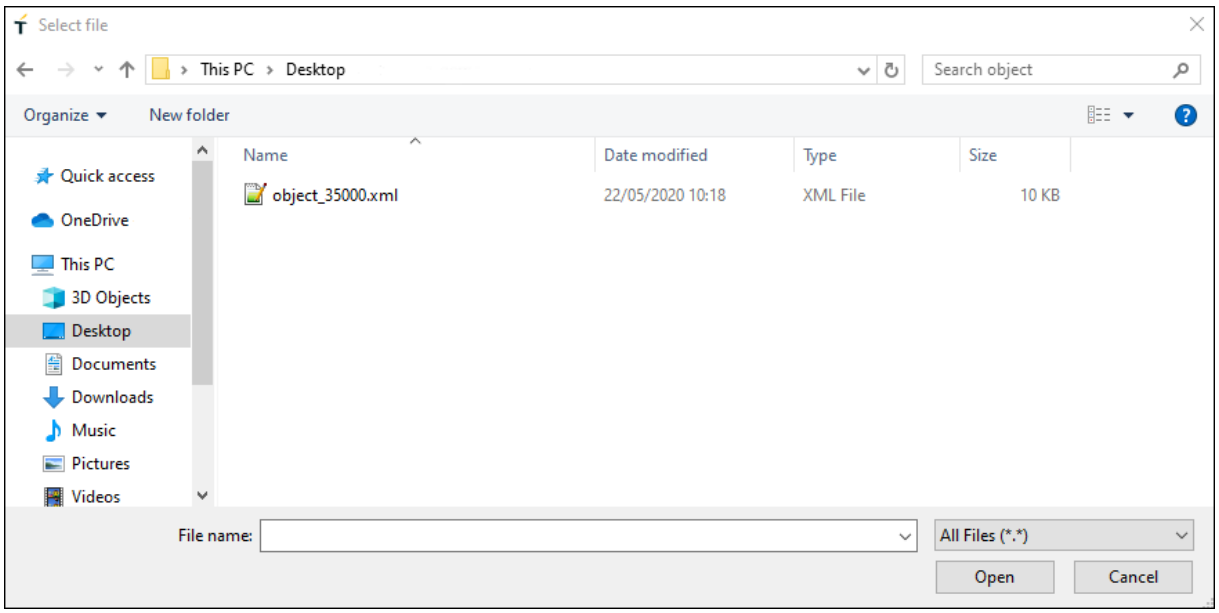

Figure 79

The file is successfully loaded on the module

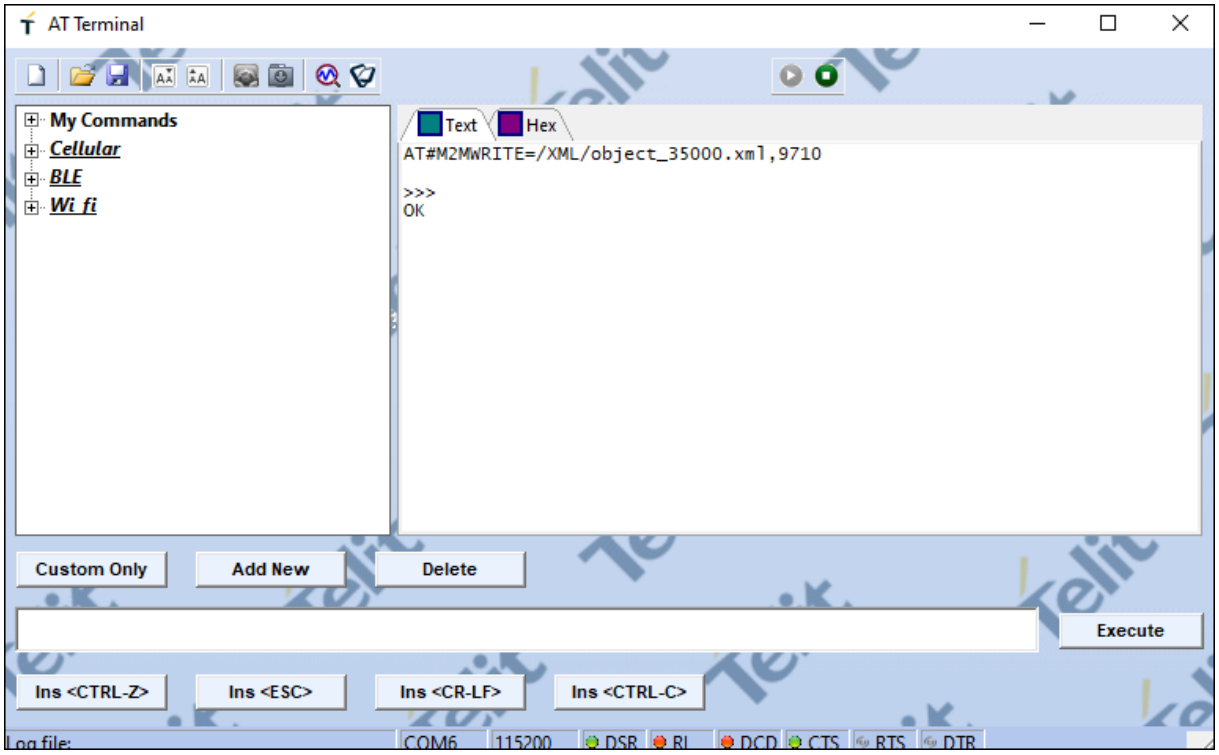

Figure 80

3.3.21.2 Application execution example

```
Starting lwm2m demo. This is v1.0.13-C1 built on Jul  6 2020 06:54:58.
On OneEdge portal, be sure that observations are enabled for the following object resources:
{35000/0/01} {35000/0/02} {35000/0/03} {35000/0/04} {35000/0/05} {35000/0/06} {35000/0/07}
{35000/0/11} {35000/0/12} {35000/0/13} {35000/0/14} {35000/0/15} {35000/0/16} {35000/0/17}
{35000/0/21} {35000/0/22} {35000/0/23} {35000/0/24} {35000/0/25} {35000/0/26} {35000/0/27}
{35000/0/31} {35000/0/32} {35000/0/33} {35000/0/34} {35000/0/35} {35000/0/36} {35000/0/37}

Initializing resources...
LWM2M enable result OK
[DEBUG] 32.39  lwm2m_demo:637 - lwm2mIndicationCB{pubTspt_0}$ Monitoring enabled.

Waiting LWM2M Registering (120 seconds timeout)...
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPING
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPED
resp->info == M2MB_LWM2M_CL_STATE_REGISTERING
resp->info == M2MB_LWM2M_CL_STATE_REGISTERED

Waiting for events from portal. Write on monitored resource or call an exec

GET STATUS.
IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_REGISTERING
```

Figure 81

```
Setting integer resource {35000/0/2} value to 50 on LWM2M client.

Setting integer resource {35000/0/22/0} value to 10 on LWM2M client.

Resource /35000/0/2/0 changed!

Setting integer resource {35000/0/22/1} value to 11 on LWM2M client.
[DEBUG] 52.05  lwm2m_demo:470 - lwm2mIndicationCB{pubTspt_0}$ Read type: 3; size: 4

Resource /35000/0/22/0 changed!

Integer data in {35000/0/2/0} resource was updated to new value: 50

Writing string resource {35000/0/11} value to demo_string on LWM2M client.
[DEBUG] 52.57  lwm2m_demo:470 - lwm2mIndicationCB{pubTspt_0}$ Read type: 3; size: 4

Resource /35000/0/22/1 changed!

Integer data in {35000/0/22/0} resource was updated to new value: 10
[DEBUG] 54.19  lwm2m_demo:470 - lwm2mIndicationCB{pubTspt_0}$ Read type: 3; size: 4

Resource /35000/0/11/0 changed!

Integer data in {35000/0/22/1} resource was updated to new value: 11
[DEBUG] 54.71  lwm2m_demo:470 - lwm2mIndicationCB{pubTspt_0}$ Read type: 2; size: 11

String data in {35000/0/11/0} resource was updated to new content: <demo_string>
```

Figure 82

After the Demo completes the initialization, it is possible to access the object resources from the Portal Object Browser

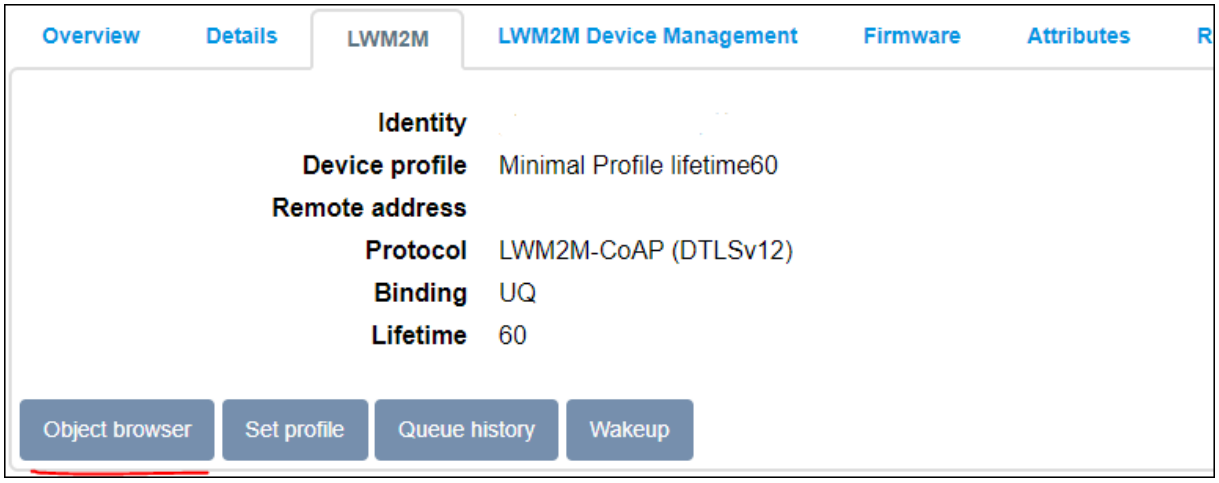

Figure 83

An instance of the object will be present and the resources can be modified.

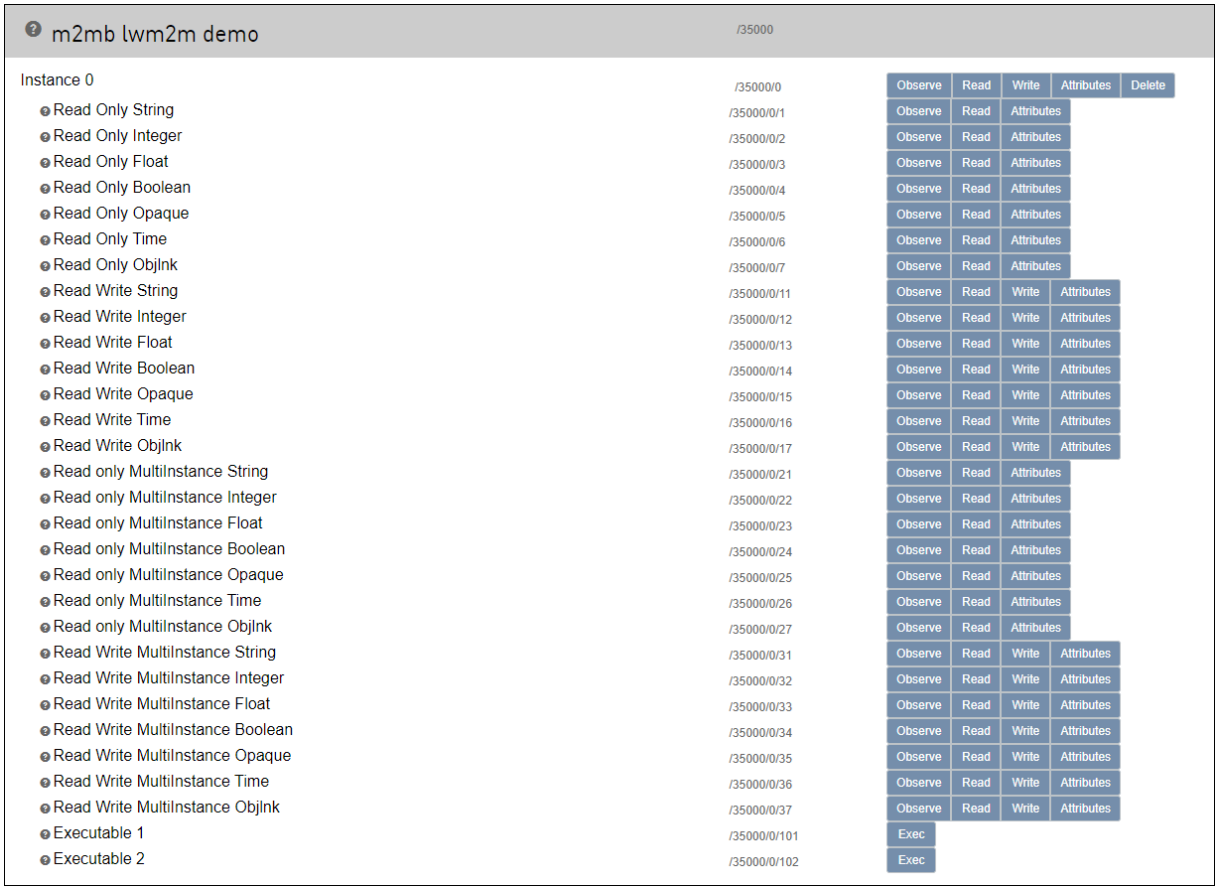

Figure 84

For example, executing the two Exec Resources at the bottom of the list, the application will react accordingly:

```
Info Exec Ind: 35000/0/101/0
Asked to execute resource 101

Info Exec Ind: 35000/0/102/0
Asked to execute resource 102
```

Figure 85

Writing a string resource (id /35000/0/11), the application will notify the change

```
Resource /35000/0/11/0 changed!
[DEBUG] 1249.86  lwm2m_demo:470 - lwm2mIndicationCB{pubTspt_0}$ Read type: 2; size: 5

String data in {35000/0/11/0} resource was updated to new content: <hello>
```

Figure 86

3.3.22 Logging Demo

Sample application showing how to print on one of the available output interfaces. Debug prints on **MAIN UART**

Features

- How to open a logging channel
- How to set a logging level
- How to use different logging macros

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Print a message with every log level

```
Starting Logging demo app. This is v1.0.7 built on Mar 26 2020 13:57:06.
[WARN ] 20.17  M2MB_main.c:74 - M2MB_main{M2M_DamsStart}$ This is a WARNING MESSAGE
[ERROR] 20.18  M2MB_main.c:76 - M2MB_main{M2M_DamsStart}$ THIS IS AN ERROR MESSAGE
[CRITICAL] 20.19  M2MB_main.c:78 - M2MB_main{M2M_DamsStart}$ THIS IS AN CRITICAL MESSAGE
[DEBUG] 20.19  M2MB_main.c:80 - M2MB_main{M2M_DamsStart}$ This is a DEBUG message
[TRACE] 20.20  M2MB_main.c:82 - M2MB_main{M2M_DamsStart}$ This is a TRACE message
END.
```

Figure 87

3.3.23 MD5 example

Sample application showing how to compute MD5 hashes using m2mb crypto. Debug prints on **MAIN UART**

Features

- Compute MD5 hash of a file
- Compute MD5 hash of a string

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create a temporary file with the expected content
- Compute MD5 hash of the provided text file
- Compare the hash with the expected one
- Compute MD5 hash of a string
- Compare the hash with the expected one
- Delete test file

```
Starting MD5 demo app. This is v1.0.7 built on Apr  7 2020 10:19:54.
Buffer written successfully into file. 45 bytes were written.

Computing hash from file...
Computed hash: bb0fa6eff92c305f166803b6938dd33a
Expected hash: bb0fa6eff92c305f166803b6938dd33a
Hashes are the same!

Computing hash from string...
Computed hash: bb0fa6eff92c305f166803b6938dd33a
Expected hash: bb0fa6eff92c305f166803b6938dd33a
Hashes are the same!
```

Figure 88

3.3.24 MQTT Client

Sample application showcasing MQTT client functionalities (with SSL). Debug prints on **MAIN UART**

Features

- How to check module registration and enable PDP context
- How to configure MQTT client parameters
- How to connect to a broker with SSL and exchange data over a subscribed topic

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage MQTT client and start it

`mqtt_demo.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Init MQTT client
- Configure it with all parameters (Client ID, username, password, PDP context ID, keepalive timeout...)
- Connect MQTT client to broker
- Subscribe to two topics
- Publish 10 messages with increasing counter. Even messages are sent to topic 1, odd messages on topic 2.
- Print received message in mqtt_topc_cb function
- Disconnect MQTT client and deinit it
- Disable PDP context

```
Starting MQTT demo app. This is v1.0.7 built on Apr  7 2020 10:34:08.
[DEBUG] 16.18  mqtt_demo.c:192 - MQTT_Task{MQTT_TASK}$ INIT
[DEBUG] 16.18  mqtt_demo.c:206 - MQTT_Task{MQTT_TASK}$ m2mb_os_ev_init success
[DEBUG] 16.19  mqtt_demo.c:214 - MQTT_Task{MQTT_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 16.19  mqtt_demo.c:221 - MQTT_Task{MQTT_TASK}$ Waiting for registration...
[DEBUG] 16.20  mqtt_demo.c:131 - NetCallback{pubTspt_0}$ Module is registered
[DEBUG] 16.21  mqtt_demo.c:232 - MQTT_Task{MQTT_TASK}$ Pdp context activation
[DEBUG] 18.26  mqtt_demo.c:246 - MQTT_Task{MQTT_TASK}$ Activate PDP with APN web.omnitel.it on CID 3....
[DEBUG] 18.95  mqtt_demo.c:155 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 18.96  mqtt_demo.c:159 - PdpCallback{pubTspt_0}$ IP address: 37.118.201.56
[DEBUG] 18.96  mqtt_demo.c:268 - MQTT_Task{MQTT_TASK}$ Init MQTT
[DEBUG] 18.97  mqtt_demo.c:278 - MQTT_Task{MQTT_TASK}$ m2mb_mqtt_init succeeded

Connecting to broker <api-dev.devicewise.com>:1883...
Done.
Subscribing to test_topic and test_topic2..
[DEBUG] 20.35  mqtt_demo.c:367 - MQTT_Task{MQTT_TASK}$ Done.

[DEBUG] 20.36  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 2> to topic test_topic
[DEBUG] 20.37  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 20.71  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 20.72  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 2>
[DEBUG] 23.37  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 3> to topic test_topic2
[DEBUG] 23.38  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 23.92  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 23.93  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 3>
[DEBUG] 26.40  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 4> to topic test_topic
[DEBUG] 26.41  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 26.93  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 26.93  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 4>
[DEBUG] 29.42  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 5> to topic test_topic2
[DEBUG] 29.43  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 29.99  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 30.00  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 5>
[DEBUG] 32.46  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 6> to topic test_topic
[DEBUG] 32.48  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 33.00  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 33.01  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 6>
[DEBUG] 35.47  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 7> to topic test_topic2
[DEBUG] 35.48  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 36.01  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 36.02  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 7>
[DEBUG] 38.50  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 8> to topic test_topic
[DEBUG] 38.51  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 39.15  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 39.16  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 8>
[DEBUG] 41.52  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 9> to topic test_topic2
[DEBUG] 41.53  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 42.10  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 42.12  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 9>
[DEBUG] 44.56  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 10> to topic test_topic
[DEBUG] 44.57  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 45.09  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 28
[DEBUG] 45.11  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 10>
[DEBUG] 47.58  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 11> to topic test_topic2
[DEBUG] 47.59  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 48.12  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 28
[DEBUG] 48.13  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 11>

Disconnecting from MQTT broker..
[DEBUG] 50.60  mqtt_demo.c:414 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 50.61  mqtt_demo.c:443 - MQTT_Task{MQTT_TASK}$ application exit
[DEBUG] 50.62  mqtt_demo.c:453 - MQTT_Task{MQTT_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 50.63  mqtt_demo.c:457 - MQTT_Task{MQTT_TASK}$ Application complete.
[DEBUG] 51.23  mqtt_demo.c:164 - PdpCallback{pubTspt_0}$ Context deactivated!
```

Figure 89

3.3.25 MultiTask

Sample application showcasing multi tasking functionalities with M2MB API. Debug prints on **MAIN UART**

Features

- How to create tasks using azx utilities
- How to use send messages to tasks
- How to use a semaphore to synchronize two tasks

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create three tasks with the provided utility (this calls public m2mb APIs)
- Send a message to the task1, its callback function azx_msgTask1 will be called

`azx_msgTask1`

- Print received parameters from main
- Send modified parameters to task2 (its callback function azx_msgTask2 will be called)
- wait for an InterProcess Communication semaphore to be available (released by task3)
- Once the semaphore is available, print a message and return

`azx_msgTask2`

- Print received parameters from caller
- If first parameter is bigger than a certain value, Send modified parameters to task3
- Else, use the second parameter as a task handle and print the corresponding name plus the value of the first parameter

`azx_msgTask3`

- Print received parameters from task 2
- release IPC semaphore
- send message to task 2 with first parameter below the threshold and second parameter with task3 handle

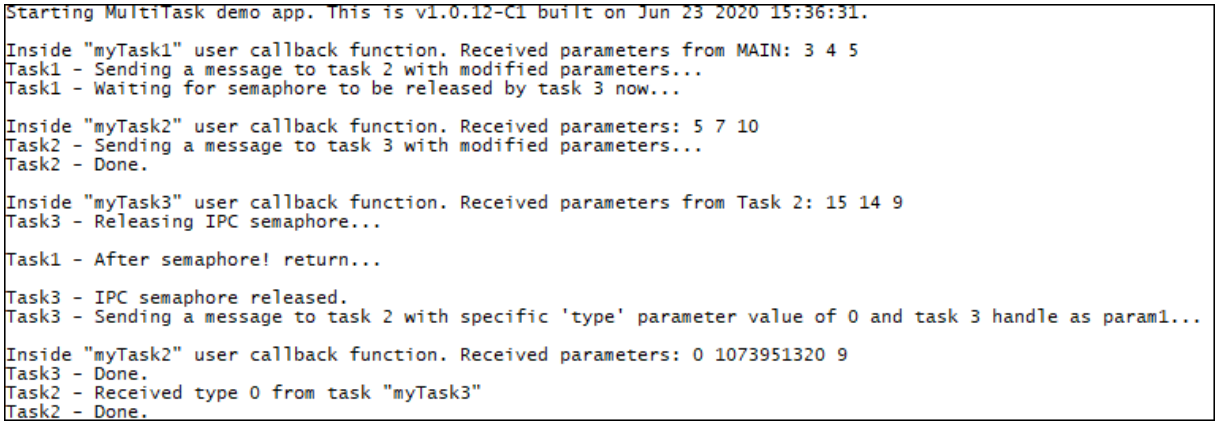
Starting MultiTask demo app. This is v1.0.12-C1 built on Jun 23 2020 15:36:31.
Inside "myTask1" user callback function. Received parameters from MAIN: 3 4 5
Task1 - Sending a message to task 2 with modified parameters...
Task1 - Waiting for semaphore to be released by task 3 now...
Inside "myTask2" user callback function. Received parameters: 5 7 10
Task2 - Sending a message to task 3 with modified parameters...
Task2 - Done.
Inside "myTask3" user callback function. Received parameters from Task 2: 15 14 9
Task3 - Releasing IPC semaphore...
Task1 - After semaphore! return...
Task3 - IPC semaphore released.
Task3 - Sending a message to task 2 with specific 'type' parameter value of 0 and task 3 handle as param1...
Inside "myTask2" user callback function. Received parameters: 0 1073951320 9
Task3 - Done.
Task2 - Received type 0 from task "myTask3"
Task2 - Done.

Figure 90

3.3.26 MutEx

Sample application showing mutex usage, with ownership and prioritization usage. Debug prints on **MAIN UART**

Features

- How to create a mutex
- How to use the mutex with tasks having different priorities
- how to reorder the pending tasks queue for the mutex

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create four tasks with the provided utility (this calls public m2mb APIs). The first task is a “producer”, putting data on a shared buffer. The second is a “consumer” of said data, the other two are used for prioritization demo
- run producer and consumer tasks at the same pace. the shared buffer will stay empty, because the resource is consumed right after creation
- run producer twice as fast as consumer. The buffer is slowly filled
- run consumer twice as fast as publisher. The buffer is always empty.
- reserve the mutex in the main task and run producer, support and support2 tasks (in this order). Then release the mutex and check the execution order. It should be by arrival.
- reserve the mutex in the main task and run the same three task, but before releasing the mutex, call the prioritization API. the task with highest priority (producer) is put as first in the queue.

```
Starting MutEx app. This is v1.0.12-C1 built on Jul  1 2020 08:37:15.
[DEBUG] 14.50  M2MB_main:90 - mutex_init{M2M_DamsStart}$ [MUTEX] Mutex initialized

[CASE 1 ] Producer and consumer have same idle time

[DEBUG] 14.51  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 14.52  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 14.53  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 14.53  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 14.54  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 14.54  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 14.55  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 14.56  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 15.56  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 15.56  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 15.57  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 15.58  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 15.58  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 15.59  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 15.60  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 15.60  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 16.61  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 16.61  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 16.62  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 16.63  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 16.63  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 16.64  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 16.64  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 16.65  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
```

Figure 91

```
[CASE 2 ] Producer has double idle time

[DEBUG] 17.56  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 17.56  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 17.57  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 17.58  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 17.58  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 17.59  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 17.59  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 17.60  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 18.63  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 18.64  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 0 items
[DEBUG] 18.64  M2MB_main:268 - msgConsumer{CONSUMER}$ Can't consume anything, buffer size is 0
[DEBUG] 18.65  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 19.62  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 19.62  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 19.63  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 19.64  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 19.68  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 19.69  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 19.69  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 19.70  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 20.73  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 20.74  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 0 items
[DEBUG] 20.75  M2MB_main:268 - msgConsumer{CONSUMER}$ Can't consume anything, buffer size is 0
[DEBUG] 20.75  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 21.67  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 21.68  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 21.68  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 21.69  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 21.77  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 21.79  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 21.80  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 21.80  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
```

Figure 92

```
[CASE 3 ] Producer has half idle time

[DEBUG] 22.62  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 22.63  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 22.64  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 22.64  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 22.65  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 22.65  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 22.66  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 22.67  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 23.67  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 23.68  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 23.68  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 23.69  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 24.71  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 24.72  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 24.72  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 24.73  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 24.74  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 24.74  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 24.75  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 24.76  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 25.79  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 25.79  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 1 items
[DEBUG] 25.80  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 1
[DEBUG] 25.81  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 26.78  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 26.78  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 2 items
[DEBUG] 26.79  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 1
[DEBUG] 26.79  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 26.84  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 26.84  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 1 items
[DEBUG] 26.85  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 1
[DEBUG] 26.86  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
```

Figure 93

```
[CASE 4 ] NO HTPF

Reserve MUTEX so all tasks are enqueued
[DEBUG] 30.77  M2MB_main:387 - msgSupport{HPTF_SUPPORT}$ freepos = 0 | evaluate[freepos]= 3
[DEBUG] 30.78  M2MB_main:416 - msgSupport2{HPTF_SUPPORT2}$ freepos = 1 | evaluate[freepos]= 4
[DEBUG] 30.79  M2MB_main:223 - msgProducer{PRODUCER}$ producer: freepos = 2 | evaluate[freepos]= 1
[DEBUG] 35.85  M2MB_main:586 - M2MB_main{M2M_DamsStart}$ EVALUATE SEQUENCE IS 3 4 1. Expected: 3 4 1
NO HTPF OK

[CASE 4.1 ] HTPF USED

Reserve MUTEX so all tasks are enqueued
m2mb_os_mtx_hptf OK
[DEBUG] 41.98  M2MB_main:223 - msgProducer{PRODUCER}$ producer: freepos = 0 | evaluate[freepos]= 1
[DEBUG] 41.99  M2MB_main:387 - msgSupport{HPTF_SUPPORT}$ freepos = 1 | evaluate[freepos]= 3
[DEBUG] 42.00  M2MB_main:416 - msgSupport2{HPTF_SUPPORT2}$ freepos = 2 | evaluate[freepos]= 4
[DEBUG] 44.03  M2MB_main:650 - M2MB_main{M2M_DamsStart}$ EVALUATE SEQUENCE IS 1 3 4, expected 1 3 4
HTPF DEMO OK
The application has ended...
```

Figure 94

3.3.27 SMS PDU

Sample application showcasing how to create and decode PDUs to be used with m2mb_sms_* API set. A SIM card and antenna must be present. Debug prints on **MAIN UART**

Features

- How to enable SMS functionality
- How to use encode an SMS PDU to be sent with m2mb_api
- How to decode a received SMS response from PDU to ASCII mode.

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Init sms functionality
- Create PDU from text message
- Send message to destination number
- Wait for response
- When SMS PDU response is received, decode it and print information about it, plus the message content

```
m2mb_sms_init() succeeded

Sending message <How are you?>...
 m2mb_sms_send() - succeeded
M2MB_SMS_SEND_RESP Callback
Send resp msg ID 10
SMS received!
SMS correctly received!

Reading SMS from memory...
 m2mb_sms_read() request succeeded

--- SMS read ---
SMS tag M2MB_SMS_TAG_MT_NOT_READ
SMS format M2MB_SMS_FORMAT_3GPP
Code type: 0
Sender type: 145
Msg len: 12
Msg bytes: 11
Msg date 19/7/17 16:7:58 (timezone: 2)
Received SMS, content: <<Fine thanks >>
Sender: +[redacted]
```

Figure 95

3.3.28 SPI Echo

Sample application showing how to communicate over SPI with m2mb API. Debug prints on **MAIN UART**

Features

- How to open an SPI bus. MOSI and MISO will be shorted, to have an echo.
- How to communicate over SPI bus

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Open SPI bus, set parameters
- Send data on MOSI and read the same in MISO

```
Starting SPI demo app. This is v1.0.7 built on Apr  1 2020 13:48:05.
Transfer successful. Received: hello from spi echo
```

Figure 96

3.3.29 SPI sensors

Sample application showing SPI usage, configuring two ST devices: a magnetometer (ST LIS3MDL) and a gyroscope (ST L3G4200D). The application will read values from both devices using GPIO4 and 3 (respectively) as magnetometer CS and gyro CS. Debug prints on **MAIN UART**

Features

- How to open an SPI bus with a slave device
- How to communicate with the device over the SPI bus

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Open SPI bus, set parameters
- Configure `GPIO 3` and `GPIO 4` as output, set them high (idle)
- Set registers to configure magnetometer
- Read in a loop (10 iterations) the registers carrying the 3 axes values and show the gauss value for each of them. A metal object is put close to the sensor to change the read values.
- Set registers to configure gyroscope
- Read in a loop (10 iterations) the registers carrying the 3 axes values and show the degrees per second value for each of them. The board is rotated to change the read values.

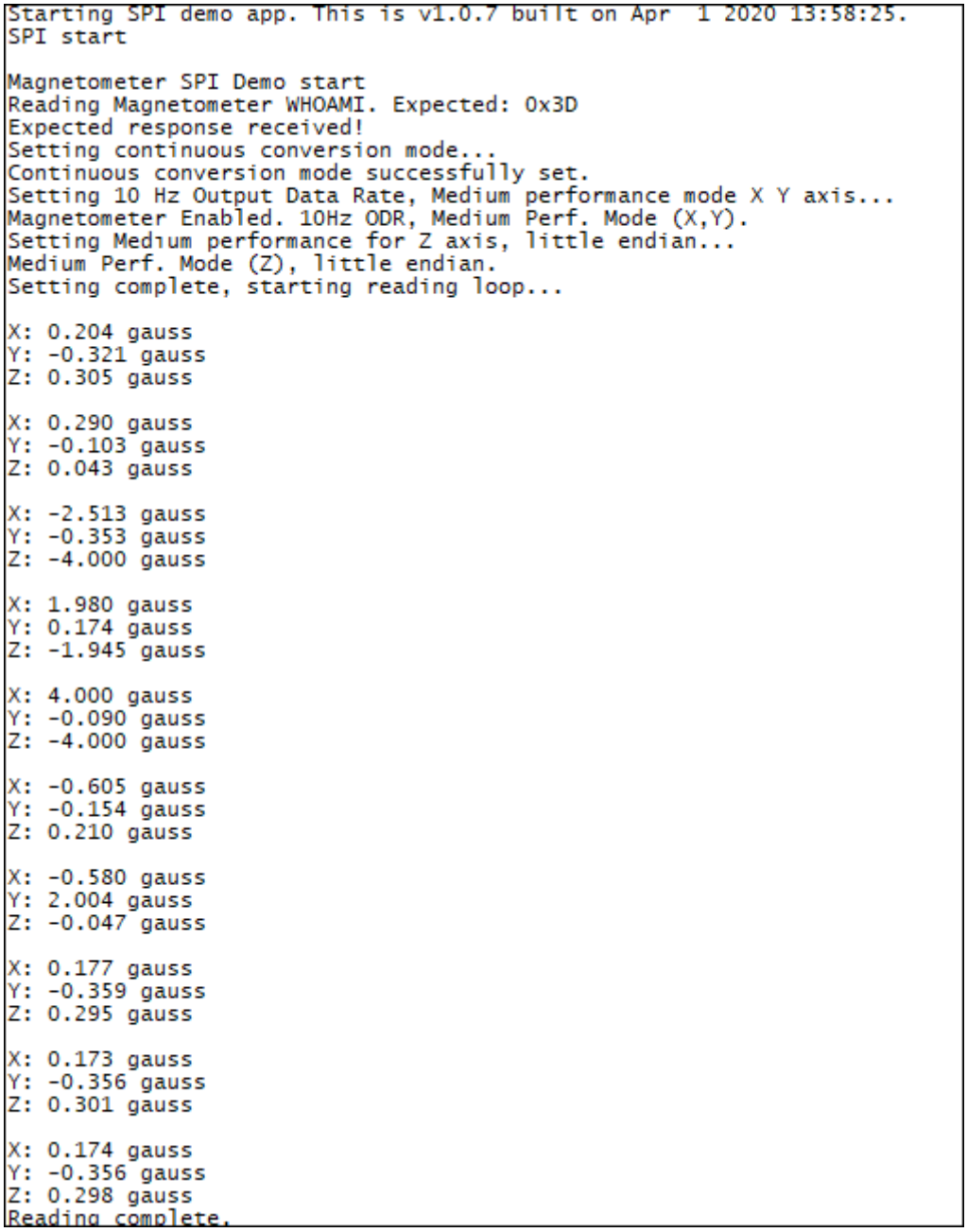
Starting SPI demo app. This is v1.0.7 built on Apr 1 2020 13:58:25.
SPI start
Magnetometer SPI Demo start
Reading Magnetometer WHOAMI. Expected: 0x3D
Expected response received!
Setting continuous conversion mode...
Continuous conversion mode successfully set.
Setting 10 Hz Output Data Rate, Medium performance mode X Y axis...
Magnetometer Enabled. 10Hz ODR, Medium Perf. Mode (X,Y).
Setting Medium performance for Z axis, little endian...
Medium Perf. Mode (Z), little endian.
Setting complete, starting reading loop...
X: 0.204 gauss
Y: -0.321 gauss
Z: 0.305 gauss
X: 0.290 gauss
Y: -0.103 gauss
Z: 0.043 gauss
X: -2.513 gauss
Y: -0.353 gauss
Z: -4.000 gauss
X: 1.980 gauss
Y: 0.174 gauss
Z: -1.945 gauss
X: 4.000 gauss
Y: -0.090 gauss
Z: -4.000 gauss
X: -0.605 gauss
Y: -0.154 gauss
Z: 0.210 gauss
X: -0.580 gauss
Y: 2.004 gauss
Z: -0.047 gauss
X: 0.177 gauss
Y: -0.359 gauss
Z: 0.295 gauss
X: 0.173 gauss
Y: -0.356 gauss
Z: 0.301 gauss
X: 0.174 gauss
Y: -0.356 gauss
Z: 0.298 gauss
Reading complete.

Figure 97

3.3.30 SW Timer (Software Timer)

The sample application shows how to use SW Timers M2MB API. Debug prints on **MAIN UART**

Features

- How to open configure a SW timer
- How to use the timer to manage recurring events

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create sw timer structure
- Configure it with 4 seconds timeout, periodic timer (auto fires when expires)
- Init the timer with the parameters
- Start the timer
- Wait 10 seconds
- Stop the timer

`timerCb`

- Print a message with inside the callback

```
Starting SW Timers demo app. This is v1.0.7 built on Apr  7 2020 09:51:25.
timer expired!
[DEBUG] 21.41  M2MB_main.c:59 - timerCb{pubTspt_0}$ timer handle: 0x4002b004
timer expired!
[DEBUG] 25.47  M2MB_main.c:59 - timerCb{pubTspt_0}$ timer handle: 0x4002b004
stopping the timer
Stop a running timer: success
Application end
```

Figure 98

3.3.31 Secure MicroService

Sample application showcasing how to manage secure microservice functionalities. Debug prints on **MAIN UART**

Features

- Write data in Secure Data Area (SDA), non protected
- Read the written data and compare with the original buffer
- Write a cripty key in Secure Data Area (SDA), non protected
- Perform a rotate of the written key data
- Perform MD5 sum of written data from TZ file
- Compare computed digest with expected one
- Write data in trust zone as a trusted object (it will not be possible to read it again but only use its content for crypto operations)
- Try to read the trusted object and verify it fails
- Rotate trusted item and verify retrieving the content fails
- compute MD5 sum of trusted item and compare with the expected one
- Try to pass data from a trusted item to a non trusted item using untrusted TZ buffers, and verify it fails

Application workflow

`M2MB_main.c`

- Write a buffer in a SDA item using `m2mb_secure_ms_write`
- Read the same item using `m2mb_secure_ms_read`
- Write a buffer containing some cripty key in a SDA item using `m2mb_secure_ms_write`
- Rotate the content of the key item
- Read it with `m2mb_secure_ms_read`
- Load the key content using `m2mb_secure_ms_crypto_alloc` and `m2mb_secure_crypto_add_it` in a SECURE_MS buffer
- Compute MD digest with `m2mb_secure_ms_crypto_md`
- Write a buffer containing some cripty key in a SDA item using `m2mb_secure_ms_write` but with **TRUSTED** option in `m2mb_secure_ms_open`
- Verify that `m2mb_secure_ms_read` on the trusted item fails
- Verify that `m2mb_secure_ms_crypto_rotate` fails for the trusted item
- Verify the MD5 digest
- Try to copy the trusted item data in a SECURE_MS buffer with `m2mb_secure_ms_crypto_alloc` and `m2mb_secure_crypto_add_item`, then load it in an untrusted object with `m2mb_secure_ms_crypto_write`, and verify it fails.

```
Starting secure ms demo app. This is v1.0.13-C1 built on Jul 30 2020 12:19:02.
Writing data in normal item
Stored input data in Secure Data Area
Reading data from normal item
Data length in SDA: 11 bytes
Securely loaded the data from the SDA
Read 11 bytes: <hello world>
original and retrieved strings are the same

Writing key in normal item
Stored input data in Secure Data Area

Rotate data in normal item
Original key: AA_THIS_IS_MY_SECRET_KEY_BB
Rotated key:

Compute MD5 of data in normal item
Data length in SDA: 27 bytes
MD5: 8EDAD26E26E1C74C7C02386C1C7F541D
hash is the expected one!

Writing data in trusted item
Stored input data in Secure Data Area
Reading data from trusted item (should fail!)
Data length in SDA: 27 bytes
m2mb_secure_ms_read() failed for trusted item, as expected!

Rotate data in trusted item
[ERROR] 17.01  M2MB_main:329 - read_rotate{M2M_DamsStart}$ Cannot read data from SECURE_MS_BUFFER to user buffer
Original key: AA_THIS_IS_MY_SECRET_KEY_BB
Rotated key:

Compute MD5 of data in trusted item
Data length in SDA: 27 bytes

MD5: 8EDAD26E26E1C74C7C02386C1C7F541D
Hash is the expected one!

Try to pass data from trusted to untrusted through TZ buffers
Cannot store data from SECURE_MS_BUFFER to SDA 'non-trusted', as expected
```

Figure 99

3.3.32 TCP IP

Sample application showcasing TCP echo demo with M2MB API. Debug prints on **MAIN UART**

Features

- How to check module registration and activate PDP context
- How to open a TCP client socket
- How to communicate over the socket

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Connect to the server
- Send data and receive response
- Close socket
- Disable PDP context

```
Starting TCP-IP demo app. This is v1.0.7 built on Mar 26 2020 16:20:30.
[DEBUG] 21.23  m2m_tcp_test.c:201 - M2M_msgTCPTask{TCP_TASK}$ INIT
[DEBUG] 21.25  m2m_tcp_test.c:217 - M2M_msgTCPTask{TCP_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.26  m2m_tcp_test.c:223 - M2M_msgTCPTask{TCP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.26  m2m_tcp_test.c:231 - M2M_msgTCPTask{TCP_TASK}$ Waiting for registration...
[DEBUG] 21.28  m2m_tcp_test.c:128 - NetCallback{pubTspt_0}$ Module is registered to cell 0x816B!
[DEBUG] 21.29  m2m_tcp_test.c:244 - M2M_msgTCPTask{TCP_TASK}$ Pdp context activation
[DEBUG] 21.30  m2m_tcp_test.c:248 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.34  m2m_tcp_test.c:263 - M2M_msgTCPTask{TCP_TASK}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 24.52  m2m_tcp_test.c:155 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.52  m2m_tcp_test.c:158 - PdpCallback{pubTspt_0}$ IP address: 83.225.44.56
[DEBUG] 24.54  m2m_tcp_test.c:273 - M2M_msgTCPTask{TCP_TASK}$ Creating Socket...
[DEBUG] 24.54  m2m_tcp_test.c:284 - M2M_msgTCPTask{TCP_TASK}$ Socket created
[DEBUG] 24.55  m2m_tcp_test.c:294 - M2M_msgTCPTask{TCP_TASK}$ Socket ctx set to 3
[DEBUG] 24.95  m2m_tcp_test.c:307 - M2M_msgTCPTask{TCP_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 25.17  m2m_tcp_test.c:322 - M2M_msgTCPTask{TCP_TASK}$ Socket Connected!
[DEBUG] 25.18  m2m_tcp_test.c:329 - M2M_msgTCPTask{TCP_TASK}$ Sending data over socket..
[DEBUG] 25.19  m2m_tcp_test.c:342 - M2M_msgTCPTask{TCP_TASK}$ Data send successfully (16 bytes)
[DEBUG] 27.20  m2m_tcp_test.c:356 - M2M_msgTCPTask{TCP_TASK}$ trying to receive 16 bytes..
[DEBUG] 27.21  m2m_tcp_test.c:364 - M2M_msgTCPTask{TCP_TASK}$ Data received (16): <hello from m2mb!>
[DEBUG] 27.21  m2m_tcp_test.c:373 - M2M_msgTCPTask{TCP_TASK}$ application exit
[DEBUG] 27.22  m2m_tcp_test.c:385 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 27.24  m2m_tcp_test.c:388 - M2M_msgTCPTask{TCP_TASK}$ Application complete.
[DEBUG] 29.43  m2m_tcp_test.c:164 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 100

3.3.33 TCP Socket status

Sample application showcasing how to check a TPC connected socket current status. Debug prints on **MAIN UART**

Features

- How to check module registration and activate PDP context
- How to open a TCP client socket
- How to check if the TCP socket is still valid

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Connect to the server
- Check in a loop the current socket status using the adv_select function with a 2 seconds timeout
- Close socket when the remote host closes it
- Disable PDP context

```
Starting TCP socket status check demo app. This is v1.0.14-C1 built on Sep  8 2020 14:59:25.
[DEBUG] 21.33  m2m_tcp_tes:324 - M2M_msgTCPTask{TCP_TASK}$ INIT
[DEBUG] 21.34  m2m_tcp_tes:338 - M2M_msgTCPTask{TCP_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.34  m2m_tcp_tes:344 - M2M_msgTCPTask{TCP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.35  m2m_tcp_tes:352 - M2M_msgTCPTask{TCP_TASK}$ Waiting for registration...
[DEBUG] 21.36  m2m_tcp_tes:365 - M2M_msgTCPTask{TCP_TASK}$ Pdp context activation
[DEBUG] 21.37  m2m_tcp_tes:369 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.41  m2m_tcp_tes:384 - M2M_msgTCPTask{TCP_TASK}$ Activate PDP with APN NXT17.NET....
[DEBUG] 24.09  m2m_tcp_tes:281 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.10  m2m_tcp_tes:284 - PdpCallback{pubTspt_0}$ IP address: 100.77.5.223
[DEBUG] 24.10  m2m_tcp_tes:394 - M2M_msgTCPTask{TCP_TASK}$ Creating Socket...
[DEBUG] 24.11  m2m_tcp_tes:405 - M2M_msgTCPTask{TCP_TASK}$ Socket created
[DEBUG] 24.11  m2m_tcp_tes:415 - M2M_msgTCPTask{TCP_TASK}$ Socket ctx set to 3
[DEBUG] 24.60  m2m_tcp_tes:428 - M2M_msgTCPTask{TCP_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 24.93  m2m_tcp_tes:443 - M2M_msgTCPTask{TCP_TASK}$ Socket Connected!
[DEBUG] 26.98  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 29.03  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...

...

[DEBUG] 82.18  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 84.23  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 86.28  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 88.31  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 88.90  m2m_tcp_tes:154 - adv_select{TCP_TASK}$ Data is available on socket <0x40032b3c>
[DEBUG] 88.92  m2m_tcp_tes:160 - adv_select{TCP_TASK}$ There are <0> pending bytes on socket
Socket was closed by remote!
[DEBUG] 88.92  m2m_tcp_tes:494 - M2M_msgTCPTask{TCP_TASK}$ application exit
[DEBUG] 88.94  m2m_tcp_tes:506 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 88.94  m2m_tcp_tes:509 - M2M_msgTCPTask{TCP_TASK}$ Application complete.
[DEBUG] 89.31  m2m_tcp_tes:290 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 101

3.3.34 TCP Server

Sample application showcasing TCP listening socket demo with M2MB API. Debug prints on **MAIN UART**

Features

- How to check module registration and activate PDP context
- How to open a TCP listening socket
- How to manage external hosts connection and exchange data

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and set it in non-blocking mode
- Bind the socket to the listening port
- Start listening for incoming connection
- Check if a connection is incoming using m2mb_socket_bsd_select function
- If a client connects, perform accept on the child socket
- Send a “START” message to the client
- Send some data
- Wait for data from client and print it
- Close the child socket
- Start listening again, up to 3 times
- Close listening socket
- Disable PDP context

Debug Log

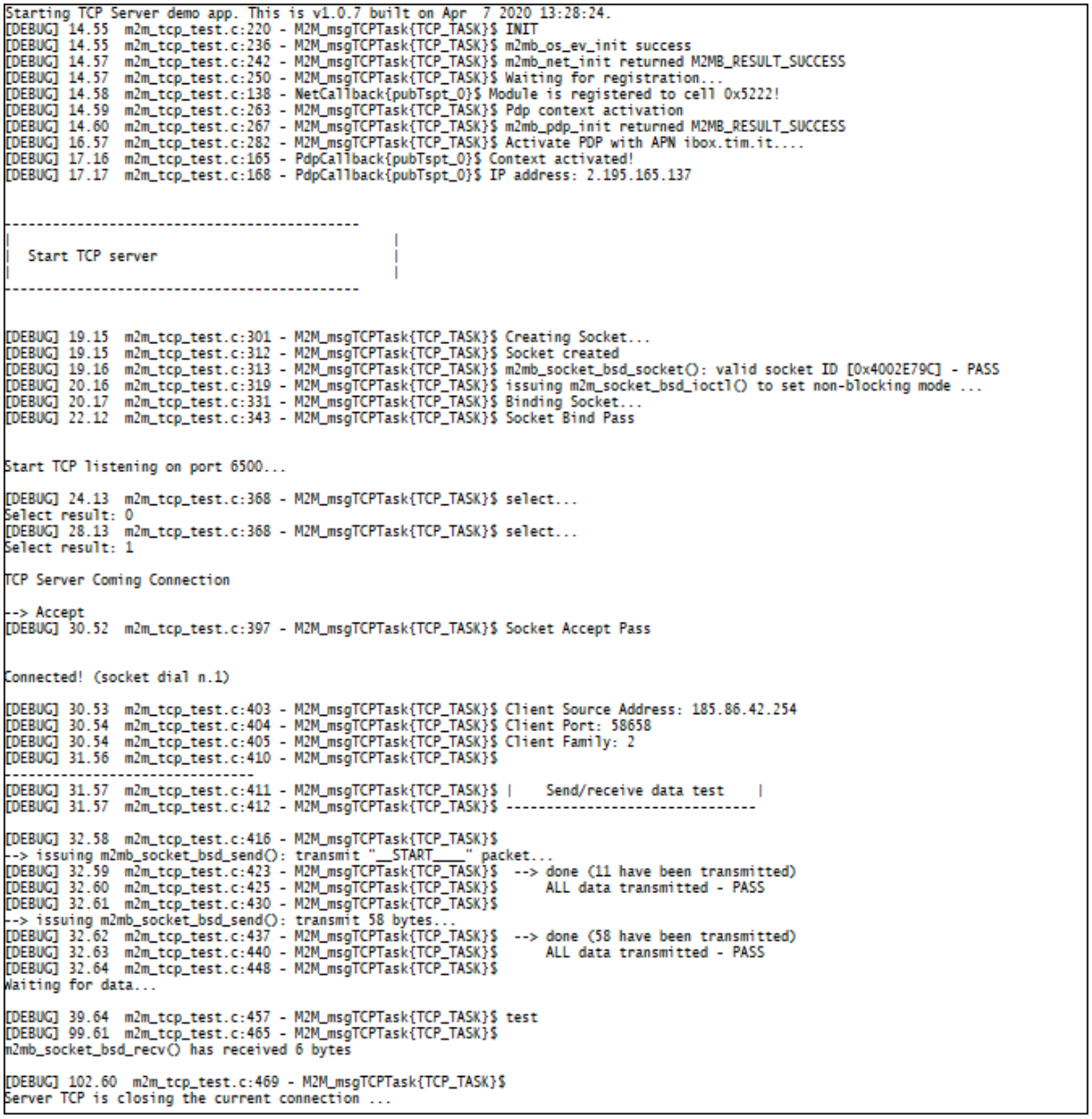

Figure 102

Data on a PuTTY terminal

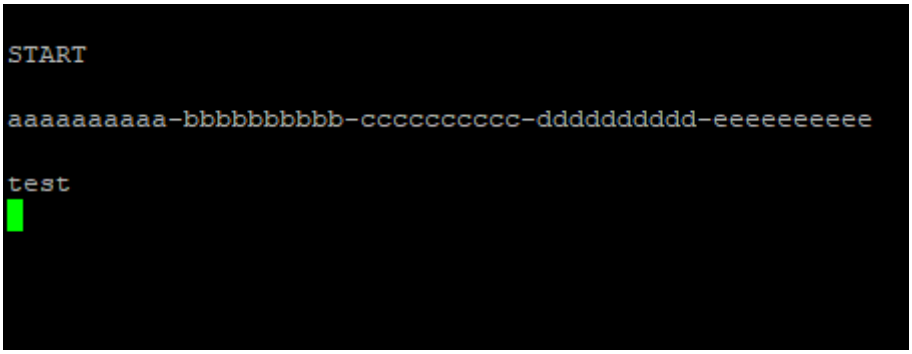
START
aaaaaaaaaa-bbbbbbbbbb-cccccccccc-dddddddddd-eeeeeeeeee
test

Figure 103

3.3.35 TLS SSL Client

Sample application showcasing TLS/SSL with client certificates usage with M2MB API. Debug prints on **MAIN UART**

Features

- How to check module registration and enable PDP context
- How to open a SSL client socket
- How to communicate over SSL socket

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create a task to manage the connection and start it

`ssl_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Connect to the server over TCP socket
- Initialize the TLS parameters (TLS1.2) andh auth mode (server+client auth in the example)
- Create SSL context
- Read certificates files and store them
- Create secure socket and connect to the server using SSL
- Send data and receive response
- Close secure socket
- Close socket
- Delete SSL context
- Disable PDP context

The application requires the certificates to be stored in `/test_ssl_certs/` folder. It can be created with `AT#M2MMKDIR=/test_ssl_certs`

```
Starting TLS-SSL demo app. This is v1.0.7 built on Mar 26 2020 16:27:00.
[DEBUG] 21.27  ssl_test.c:253 - msgHTTPSTask{TLS_TASK}$ INIT
[DEBUG] 21.27  ssl_test.c:267 - msgHTTPSTask{TLS_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.28  ssl_test.c:271 - msgHTTPSTask{TLS_TASK}$ Init SSL session test app
[DEBUG] 21.28  ssl_test.c:286 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_config sslConfigHndl = 0x400330a8, sslRes= 0
[DEBUG] 21.29  ssl_test.c:295 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_config PASSED
[DEBUG] 21.30  ssl_test.c:307 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_ctxt PASSED
[DEBUG] 21.31  ssl_test.c:312 - msgHTTPSTask{TLS_TASK}$ loading CA CERT from file /test_ssl_certs/modulesCA.crt
[DEBUG] 21.32  ssl_test.c:316 - msgHTTPSTask{TLS_TASK}$ file size: 1740
[DEBUG] 21.32  ssl_test.c:329 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1740
Buffer successfully received from file. 1740 bytes were loaded.
Closing file.
[DEBUG] 21.34  ssl_test.c:458 - msgHTTPSTask{TLS_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.35  ssl_test.c:466 - msgHTTPSTask{TLS_TASK}$ Waiting for registration...
[DEBUG] 21.36  ssl_test.c:172 - NetCallback{pubTspt_0}$ Module is registered to cell 0xC4CF!
[DEBUG] 21.36  ssl_test.c:478 - msgHTTPSTask{TLS_TASK}$ Pdp context activation
[DEBUG] 21.37  ssl_test.c:482 - msgHTTPSTask{TLS_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.41  ssl_test.c:497 - msgHTTPSTask{TLS_TASK}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 24.24  ssl_test.c:198 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.24  ssl_test.c:201 - PdpCallback{pubTspt_0}$ IP address: 37.118.158.27
[DEBUG] 24.25  ssl_test.c:515 - msgHTTPSTask{TLS_TASK}$ Creating Socket...
[DEBUG] 24.26  ssl_test.c:526 - msgHTTPSTask{TLS_TASK}$ Socket created
[DEBUG] 24.26  ssl_test.c:536 - msgHTTPSTask{TLS_TASK}$ Socket ctx set to 3
[DEBUG] 24.61  ssl_test.c:549 - msgHTTPSTask{TLS_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 24.87  ssl_test.c:563 - msgHTTPSTask{TLS_TASK}$ Socket Connected!
[DEBUG] 26.14  ssl_test.c:588 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_connect ret 0
[DEBUG] 28.17  ssl_test.c:594 - msgHTTPSTask{TLS_TASK}$ Sending bytes..
[DEBUG] 28.17  ssl_test.c:597 - msgHTTPSTask{TLS_TASK}$ SSL write result = 44
[DEBUG] 32.18  ssl_test.c:609 - msgHTTPSTask{TLS_TASK}$ pending bytes: 1087
[DEBUG] 32.19  ssl_test.c:612 - msgHTTPSTask{TLS_TASK}$ trying to receive bytes..
[DEBUG] 32.19  ssl_test.c:618 - msgHTTPSTask{TLS_TASK}$ Server response: (269)<HTTP/1.1 200 OK
Date: Thu, 26 Mar 2020 15:29:43 GMT
Server: Apache/2.2.15 (CentOS)
Last-Modified: Mon, 22 Jan 2018 10:57:39 GMT
ETag: "1fffc-27f-5635b4c6f12b3"
Accept-Ranges: bytes
Content-Length: 639
Connection: close
Content-Type: text/html; charset=UTF-8

>
[DEBUG] 32.23  ssl_test.c:635 - msgHTTPSTask{TLS_TASK}$ application exit
[DEBUG] 32.23  ssl_test.c:653 - msgHTTPSTask{TLS_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 32.26  ssl_test.c:656 - msgHTTPSTask{TLS_TASK}$ Application complete.
[DEBUG] 32.89  ssl_test.c:207 - PdpCallback{pubTspt_0}$ Context deactivated!
```

Figure 104

3.3.36 Uart To Server

Sample application showcasing how to send data from main UART to a connected TCP server. Debug messages are printed on AUX UART port.

Features

- How to open main UART to receive data
- How to connect to a server
- How to transmit received data from the UART to the server and viceversa

Application workflow

`M2MB_main.c`

- Open UART for data and UART_AUX for debug
- Init socket, activate PDP context and connect to server
- Init UART, set its callback function, create tasks to handle input from UART and response from server (optional)
- Send a confirmation on UART
- Wait for data, when it is received, send it to the server
- When a response is received, print it on UART.

Main UART:

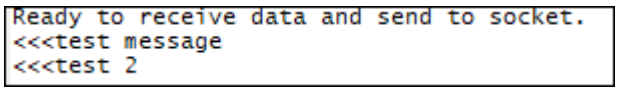

Figure 105

Debug log on AUX UART:

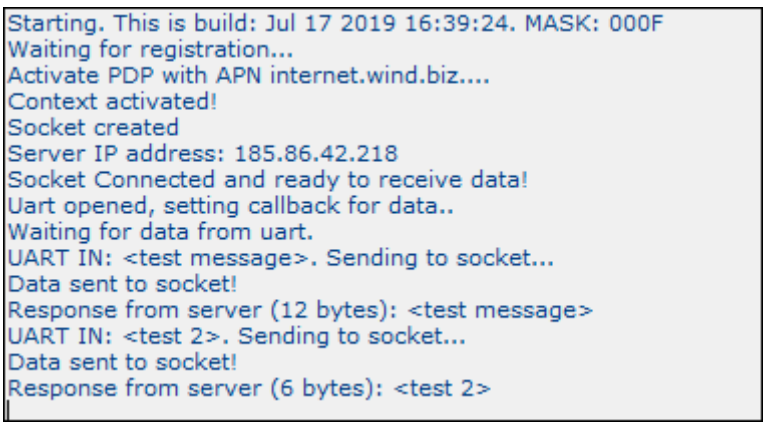

Figure 106

3.3.37 UDP client

Sample application showcasing UDP echo demo with M2MB API. Debug prints on **MAIN UART**

Features

- How to check module registration and activate PDP context
- How to open a UDP client socket
- How to communicate over the socket

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task and start it

`m2m_udp_test.c` - Initialize Network structure and check registration - Initialize PDP structure and start PDP context - Create socket and link it to the PDP context id - Send data and receive response - Close socket - Disable PDP context

```
Starting UDP client demo app. This is v1.0.7 built on Apr  1 2020 14:57:13.
INIT
[DEBUG] 21.23  m2m_udp_test.c:223 - M2M_msgUDPTask{UDP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
Waiting for registration...
[DEBUG] 21.25  m2m_udp_test.c:131 - NetCallback{pubTspt_0}$ Module is registered to cell 0xC4CF!
[DEBUG] 21.26  m2m_udp_test.c:241 - M2M_msgUDPTask{UDP_TASK}$ Pdp context initialization
[DEBUG] 21.26  m2m_udp_test.c:245 - M2M_msgUDPTask{UDP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
Activate PDP with APN web.omnitel.it....
[DEBUG] 24.11  m2m_udp_test.c:157 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.11  m2m_udp_test.c:160 - PdpCallback{pubTspt_0}$ IP address: 109.113.222.12
[DEBUG] 24.12  m2m_udp_test.c:268 - M2M_msgUDPTask{UDP_TASK}$ Creating Socket...
[DEBUG] 24.13  m2m_udp_test.c:280 - M2M_msgUDPTask{UDP_TASK}$ Socket created
Socket ctx set to 3
[DEBUG] 24.41  m2m_udp_test.c:306 - M2M_msgUDPTask{UDP_TASK}$ Retrieved IP: 185.86.42.218
Socket ready.
Data successfully sent (16 bytes)
Socket recv...
[DEBUG] 26.47  m2m_udp_test.c:352 - M2M_msgUDPTask{UDP_TASK}$ m2mb_socket_bsd_set_sock_opt() M2MB_SOCKET_BSD_SO_RCVTIMEO - success
trying to receive 16 bytes..
Data received (16): <hello from m2mb!>
[DEBUG] 26.48  m2m_udp_test.c:377 - M2M_msgUDPTask{UDP_TASK}$ application exit
Socket Closed
[DEBUG] 26.49  m2m_udp_test.c:399 - M2M_msgUDPTask{UDP_TASK}$ m2mb_pdp_deactivate returned success
Application complete.
[DEBUG] 27.04  m2m_udp_test.c:166 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 107

3.3.38 USB Cable Check

Sample application showing how to check if USB cable is plugged in or not. Debug prints on **MAIN UART**

Features

- How to open an USB channel and configure it with a callback function
- How to manage USB cable events in the callback function

Application workflow

`M2MB_main.c`

- Open UART/UART_AUX for debug
- open usb channel and set the callback
- Print greeting message
- Print current usb status

`USB_Cb`

- if the event is a connection/disconnection, show the current status

```
Starting USB cable check demo app. This is v1.0.0 built on Aug 19 2020 10:27:40.

m2mb_usb_open succeeded
m2mb_usb_ioctl: set usb callback
m2mb_usb_ioctl: got cable status
USB cable CONNECTED, status: 1

 Waiting for USB cable to be plugged/unplugged...
Usb cable check event, USB status: 0
Usb cable check event, USB status: 1
Usb cable check event, USB status: 0
Usb cable check event, USB status: 1
```

Figure 108

3.3.39 ZLIB example

Sample application showing how to compress/uncompress with ZLIB. Debug prints on **MAIN UART**

Features

- How to compress a file
- How to uncompress a file

In order to execute the entire test, copy `test.gz` file into your module running the following AT command:

```
AT#M2MWRITE="/mod/test.gz",138
>>> here receive the prompt; then type or send the file, sized 138 bytes
```

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Test the compression and decompression of a data string
- Test the decompression of a .gz file (test.gz), expected to be in `/mod` folder, into its content `test.txt`. The file must be uploaded by the user (see steps above).

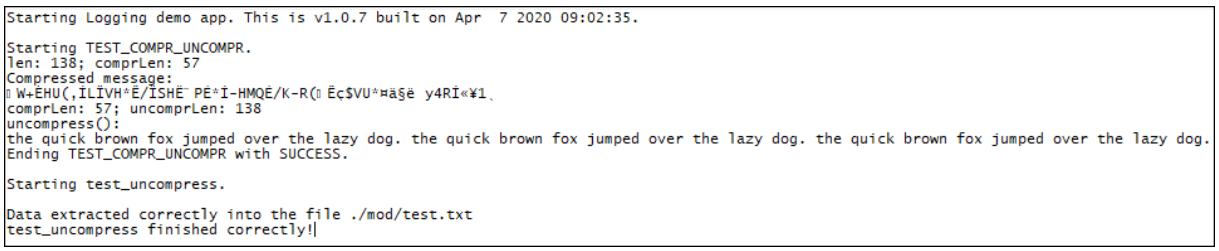

Figure 109

3.3.40 Little fs2

Sample application showing how use lfs2 porting with RAM disk and SPI data flash. Debug prints on **MAIN UART**

Features

- How to create and manage Ram Disk
- How to manage file-system in Ram disk partition
- How to create and manage SPI Flash memory partition
- How to manage file-system in SPI Flash memory partition

Application workflow

`M2MB_main.c`

- Init logging system
- Call Ram Disk tests
- Call Flash memory tests

`ram_utils_usage.c`

- Initialize Ram Disk
- Format and Mount partition
- List files
- Files creation and write content
- List files
- Read files
- Unmount and Release resources

`spi_utils_usage.c` - Initialize SPI Flash chip - Initialize SPI Flash Disk - Format and Mount partition - List files - Files creation and write content - List files - Read files - Delete files - Directories creation and deletion - Unmount and Release resources

Notes: For SPI Flash a JSC memory is used with chip select pin connected to module GPIO2 pin. For better performances, a 33kOhm pull-down resistor on SPI clock is suggested. Please refer to SPI_echo sample app for SPI connection details.

```
Starting lfs2 demo app. This is v1.0.14-C1 built on Oct 22 2020 09:43:08.

>>>>>>> Starting RAMDiskDemo ...
[DEBUG] 18.28  azx_lfs_uti:125 - azx_ram_initialize{M2M_DamsStart}$ Ram Memory allocated correctly from 0x40042228 to 0x40046228!!

Mounting partition...

Formatting...

Mounting...


Mounted partition...

<><><><>fileListUtils

List:
., 0, 2
.., 0, 2

file_name: file000.txt
size: 10
buffer: content000
mode: 0
RAM TYPE size: 10000


File created and closed: file000.txt


<><><><>fileListUtils
____INSIDE --->file000.txt, 10, 1

List:
., 0, 2
.., 0, 2
file000.txt, 10, 1

------>File reading

File: file000.txt, Size: 10, Buffer: content000

Nand released
Partition unmounted
[DEBUG] 20.31  azx_lfs_uti:165 - azx_ram_releaseResources{M2M_DamsStart}$ Ram Memory released correctly!!
```

```
>>>>>>> Starting FlashDiskDemo ...
Starting initialization...

table id[0] = 191
table id[1] = 1
table id[2] = 0

nandLFSCallback Callback event <1>
NAND Callback event: NAND_JSC_INITIALIZED <1>
nandLFSCallback Callback event <1>
NAND Callback event: NAND_JSC_INITIALIZED <1>

Mounting partition...

Formatting...
spiErase: address = 0, len = 131072

spiErase: address = 131072, len = 131072


Mounting...


Mounted partition...


<><><><>fileListUtils

List:
., 0, 2
.., 0, 2


Formatting...
spiErase: address = 0, len = 131072

spiErase: address = 131072, len = 131072


Mounting...


Mounted partition...


<><><><>fileListUtils

List:
., 0, 2
.., 0, 2

file_name: file000.txt
size: 10
buffer: content000
mode: 0


File created and closed: file000.txt
```

```
<><><><fileListUtils

List:
., 0, 2
.., 0, 2
file000.txt, 10, 1
file001.txt, 10, 1
file002.txt, 10, 1
file003.txt, 10, 1
file004.txt, 10, 1

------>File reading

File: file000.txt, Size: 10, Buffer: content000


File: file004.txt, Size: 10, Buffer: content004


File: file002.txt, Size: 10, Buffer: content002
------>File removing
file001.txt<<<<<<<


File removed: file001.txt
file000.txt<<<<<<<


File removed: file000.txt
file004.txt<<<<<<<


File removed: file004.txt


<><><><fileListUtils

List:
., 0, 2
.., 0, 2
file002.txt, 10, 1
file003.txt, 10, 1

spiErase: address = 59637760, len = 131072

[DEBUG] 58.61  azx_lfs_uti:648 - azx_lfsDirCreationByContext{M2M_DamsStart}$ Directory created: dir000!!
[DEBUG] 59.78  azx_lfs_uti:631 - azx_lfsDirCreationByContext{M2M_DamsStart}$ Directory already exists: dir000!!
spiErase: address = 59899904, len = 131072

[DEBUG] 61.70  azx_lfs_uti:648 - azx_lfsDirCreationByContext{M2M_DamsStart}$ Directory created: dir001!!
spiErase: address = 60162048, len = 131072

[DEBUG] 63.67  azx_lfs_uti:648 - azx_lfsDirCreationByContext{M2M_DamsStart}$ Directory created: dir002!!
```

```
<><><><fileListUtils

List:
., 0, 2
.., 0, 2
dir000, 0, 2
dir001, 0, 2
dir002, 0, 2
file002.txt, 10, 1
file003.txt, 10, 1



<><><><fileListUtils

List:
., 0, 2
.., 0, 2
dir001, 0, 2
dir002, 0, 2
file002.txt, 10, 1
file003.txt, 10, 1

Nand released
Partition unmounted

Unmounted process ended...

testAllInOneFunction ended...
```

3.4 C++

Applications that provide usage examples with C++

3.4.1 Logging C++

Sample application showcasing how to create a C++ OO code, providing a logging class (equivalent to the one in Logging demo)

Features

- how to define a class object
- how to instantiate and call the class from a C++ main
- how to configure makefile flags to build the application

Application workflow

`M2MB_main.c`

- Call C++ main function

`main.cpp`

- Create a Logger class instance and set it to USB/UART/UART_AUX
- Print one message for every log level

```
[TRACE] 15.20  main.cpp:68 - cpp_main{M2M_DamsStart}$ C++ Trace print example
[DEBUG] 15.22  main.cpp:69 - cpp_main{M2M_DamsStart}$ C++ Debug print example
C++ Info print example
[WARN ] 15.22  main.cpp:71 - cpp_main{M2M_DamsStart}$ C++ Warning print example
[ERROR] 15.23  main.cpp:72 - cpp_main{M2M_DamsStart}$ C++ Error print example
[CRITICAL] 15.24  main.cpp:73 - cpp_main{M2M_DamsStart}$ C++ Critical print example
```

Figure 110

3.4.2 C++ method to function pointer

Sample application showing how to manage class methods as function pointers. Debug prints on MAIN_UART

Features

- how to define a class object with a generic method with the same prototype as a m2mb callback function (in this case, a hw timer callback)
- how to use a single static function in the class workspace to call multiple class instances method by using “this” as argument in the timer creation
- how to configure the static function to convert the input parameter with a static cast and call the input class instance method

Application workflow

`M2MB_main.c`

- Call C++ main function

`main.cpp`

- Create two HwTimer class instance with different timeouts
- Start both timers.
- Each will expire at a different time, and both m2mb timers will call the static function, which will run the appropriate class instance method as callback.

```
Starting C++ method as function pointer example. This is v1.0.11-C1 built on Jul  9 2020 14:58:25.
Timer "first" created with 1000 ms timeout
Timer "second" created with 1500 ms timeout
[DEBUG] 18.70  hwtimer:167 - start{M2M_DamsStart}$ Starting "first" timer
[DEBUG] 18.71  hwtimer:167 - start{M2M_DamsStart}$ Starting "second" timer
In the static timer callback. Calling class method...
[DEBUG] 19.73  hwtimer:179 - timer_cb{pubTspt_0}$
Timer "first" class callback called. Class instance: 0x400212e0; handle: 0x4002b288
In the static timer callback. Calling class method...
[DEBUG] 20.25  hwtimer:179 - timer_cb{pubTspt_0}$
Timer "second" class callback called. Class instance: 0x400212c8; handle: 0x4002b30c
```

Figure 111

3.5 BASIC

Basic applications showing simple operations with minimum code overhead

3.5.1 Basic Hello World (Aux UART)

The application prints “Hello World!” on Auxiliary UART every 2 seconds using

Features

- How to open Auxiliary UART as an output channel
- How to print messages out of the channel

Application workflow

M2MB_main.c

- Open Auxiliary UART with **m2mb_uart_open** function
- write a welcome message using **m2mb_uart_write**
- write “Hello World!” every 2 seconds in a while loop, using **m2mb_uart_write**

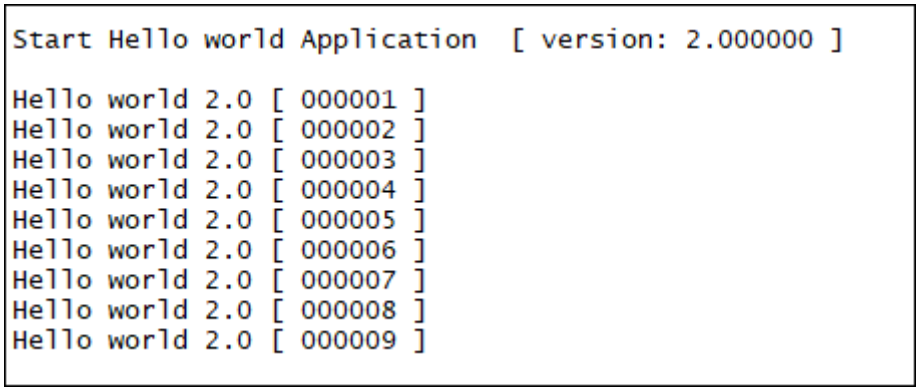

Figure 112

3.5.2 Basic Hello World (Main UART)

The application prints “Hello World!” on Main UART every 2 seconds using

Features

- How to open Main UART as an output channel
- How to print messages out of the channel

Application workflow

M2MB_main.c

- Open Main UART with **m2mb_uart_open** function
- write a welcome message using **m2mb_uart_write**
- write “Hello World!” every 2 seconds in a while loop, using **m2mb_uart_write**

```
Start Hello world Application  [ version: 2.000000 ]

Hello world 2.0 [ 000001 ]
Hello world 2.0 [ 000002 ]
Hello world 2.0 [ 000003 ]
Hello world 2.0 [ 000004 ]
Hello world 2.0 [ 000005 ]
Hello world 2.0 [ 000006 ]
Hello world 2.0 [ 000007 ]
Hello world 2.0 [ 000008 ]
Hello world 2.0 [ 000009 ]
```

Figure 113

3.5.3 Basic Hello World (USB0)

The application prints “Hello World!” on USB 0 every 2 seconds using

Features

- How to open USB 0 as an output channel
- How to print messages out of the channel

Application workflow

M2MB_main.c

- Open USB 0 with **m2mb_usb_open** function
- write a welcome message using **m2mb_usb_write**
- write “Hello World!” every 2 seconds in a while loop, using **m2mb_usb_write**

```
Start Hello world Application  [ version: 2.000000 ]

Hello world 2.0 [ 000001 ]
Hello world 2.0 [ 000002 ]
Hello world 2.0 [ 000003 ]
Hello world 2.0 [ 000004 ]
Hello world 2.0 [ 000005 ]
Hello world 2.0 [ 000006 ]
Hello world 2.0 [ 000007 ]
Hello world 2.0 [ 000008 ]
Hello world 2.0 [ 000009 ]
```

Figure 114

3.5.4 Basic Task

The application shows how to create and manage tasks with m2mb APIs. Debug prints on MAIN UART (can be changed in M2MB_Main function)

Features

- How to create a new task using m2mb APIs
- How to start the task and send messages to it
- how to destroy the task

Application workflow

`M2MB_main.c`

- Open UART
- Print welcome message
- Configure and create message queue for task
- Configure and create task
- Send 2 messages to the task queue

`task_entry_function`

- Receive messages from the task queue in a loop
- Print the message data when one arrives

```
Starting Basic Task demo app. This is v1.0.8 built on Apr 16 2020 06:40:40.
Successfully created a queue area buffer of 720 bytes.
Queue successfully created.
Creating the task...
Task created and ready to receive messages!
[DEBUG] 16.88  M2MB_main:411 - M2MB_main{M2M_DamsStart}$ Sending a message to the task...
[DEBUG] 16.88  M2MB_main:125 - task_entry_function{mytask}$ Received a message with a 5 bytes payload: <hello>
[DEBUG] 18.90  M2MB_main:420 - M2MB_main{M2M_DamsStart}$ Sending a second message to the task...
[DEBUG] 18.90  M2MB_main:430 - M2MB_main{M2M_DamsStart}$ Result code at the end: 0
[DEBUG] 18.91  M2MB_main:125 - task_entry_function{mytask}$ Received a message with a 5 bytes payload: <world>
Clearing resources...
Done. App complete
```

Figure 115

3.6 USB0

Applications that provide usage examples for various functionalities, log output on USB0

3.6.1 ATI (AT Instance)

Sample application showing how to use AT Instance functionality (sending AT commands from code). The example supports both sync and async (using a callback) modes. Debug prints on **USB0**

Features

- How to open an AT interface from the application
- How to send AT commands and receive responses on the AT interface

Application workflow, sync mode

M2MB_main.c

- Open USB/UART/UART_AUX
- Init AT0 (first AT instance)
- Send AT+CGMR command
- Print response.
- Release AT0

at_sync.c

- Init ati functionality and take AT0
- Send AT+CGMR command, then read response after 2 seconds, then return it
- Deinit ati, releasing AT0

```
Starting AT demo app. This is v1.0.7 built on Apr  1 2020 15:12:58.
[DEBUG] 17.15  at_sync.c:53 - at_cmd_sync_init{M2M_DamsStart}$ m2mb_ati_init() on instance 0
Sending command AT+CGMR in sync mode
[DEBUG] 17.16  at_sync.c:79 - send_sync_at_command{M2M_DamsStart}$ Sending AT Command: AT+CGMR
Command response: <AT+CGMR
M0B.950004-B008

OK
>

[DEBUG] 19.21  at_sync.c:61 - at_cmd_sync_deinit{M2M_DamsStart}$ m2mb_ati_deinit() on instance 0
Application end
```

Figure 116

Application workflow, async mode

M2MB_main.c

- Open USB/UART/UART_AUX
- Init AT0 (first AT instance)
- Send AT+CGMR command
- Print response.
- Release AT0

at_async.c

- Init ati functionality and take AT0, register AT events callback
- Send AT+CGMR command, wait for response semaphore (released in callback), then read it and return it
- Deinit ati, releasing AT0

```
Starting AT demo app. This is v1.0.7 built on Apr  1 2020 15:07:45.
[DEBUG] 17.13  at_async.c:116 - at_cmd_async_init{M2M_DamsStart}$ m2mb_ati_init() on instance 0
Sending command AT+CGMR in async mode
[DEBUG] 17.15  at_async.c:153 - send_async_at_command{M2M_DamsStart}$ Sending AT Command: AT+CGMR
[DEBUG] 17.15  at_async.c:169 - send_async_at_command{M2M_DamsStart}$ waiting command response...
[DEBUG] 17.17  at_async.c:88 - at_cmd_async_callback{pubTspt_0}$ Callback - available bytes: 25
[DEBUG] 17.18  at_async.c:181 - send_async_at_command{M2M_DamsStart}$ Receive response...
Command response: <AT+CGMR
M0B.950004-B008

OK
>

[DEBUG] 17.19  at_async.c:136 - at_cmd_async_deinit{M2M_DamsStart}$ m2mb_ati_deinit() on instance 0
Application end
```

Figure 117

3.6.2 App Manager

Sample application showing how to manage AppZone apps from m2mb code. Debug prints on **USB0**

Features

- How to get how many configured apps are available
- How to get the handle to manage the running app (change start delay, enable/disable)
- How to create the handle for a new binary app, enable it and set its parameters
- How to start the new app without rebooting the device, then stop it after a while.

3.6.2.1 Prerequisites

This app will try to manage another app called “second.bin”, which already exists in the module filesystem and can be anything (e.g. another sample app as GPIO toggle). the app must be built using the flag ROM_START=

in the Makefile to set a different starting address than the main app (by default, 0x40000000). For example, 0x41000000.

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- get a non existing app handle and verify it is NULL
- get the current app handle, then get the start delay **set in the INI file (so persistent)**
- change the current app delay value **in the INI file**
- verify that the change has been stored
- get current app state
- create an handle for a second application binary.
- add it to the INI file
- set its execution flag to 0
- get the delay time and the state from INI file for the new app
- get the current set address for the new app
- set the app delay **in RAM, INI will not be affected**.
- start the new app without reboot, using the right set delay
- wait some time, then get the app state and the used RAM amount
- wait 10 seconds, then stop the second app.
- set its execution flag to 1 so it will run at next boot.

```
Starting App Manager demo app. This is v1.0.14-C1 built on Sep 24 2020 12:33:25.
There are 2 configured apps.
Not existing app handle test (should be 0): 0x0
Manager app handle: 0x809e20e0
Manager app delay from nv memory: 5 seconds

Changing Manager app delay time (on non volatile configuration) to 5 seconds..
Manager app delay from nv memory is now 5 seconds
Manager app state is M2MB_APPMNG_STATE_RUN

Trying to get Second app handle...
Second app handle is valid
2nd app delay from nv memory is 1
2nd app current state is M2MB_APPMNG_STATE_READY
Second app current address is 0x41000000
Setting volatile Second app delay (not stored in nvm) to 0 seconds...
Starting Second app on the fly (without reboot)...
Waiting 2 seconds...
2nd app current state is M2MB_APPMNG_STATE_RUN
Second app is running!
Second App is using 475136 bytes of RAM
Stopping Second app now...
wait 10 seconds...
2nd app current state is M2MB_APPMNG_STATE_STOP
Set permanent run permission for Second app.
Done. Second App will also run from next boot-up
```

Figure 118

3.6.3 App update OTA via FTP

Sample application showcasing Application OTA over FTP with AZX FTP. Debug prints on **USB0**

Features

- How to check module registration and activate PDP context
- How to connect to a FTP server
- How to download an application binary and update the local version

The app uses a predefined set of parameters. To load custom parameters, upload the `ota_config.txt` file (provided in project's `/src` folder) in module's `/mod` folder, for example with

```
AT#M2MWRITE="/mod/ota_config.txt",<filesize>
```

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage app OTA and start it

`ftp_utils.c`

- Set parameters to default
- Try to load parameters from `ota_config.txt` file
- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Initialize FTP client
- Connect to FTP server and log in
- Get new App binary file size on remote server
- Download the file in `/mod` folder, with the provided name
- Close FTP connection
- Disable PDP context
- Update applications configuration in **app_utils.c**

`app_utils.c`

- Set new application as default

- Delete old app binary
- Restart module

```
Starting FTP APP OTA demo app. This is v1.0.7 built on Apr  7 2020 17:04:05.
[DEBUG] 21.23  ftp_utils.c:447 - msgFTPTask{FTPOTA_TASK}$ INIT
[DEBUG] 21.25  ftp_utils.c:152 - readConfigFromFile{FTPOTA_TASK}$ Reading parameters from file
[DEBUG] 21.26  ftp_utils.c:154 - readConfigFromFile{FTPOTA_TASK}$ Opening /mod/ota_config.txt in read mode..
Set APN to: <<web.omnitel.it>>
Set FTP URL to: <<ftp.telit.com>>
Set FTP PORT to: 21
Set FTP USER to: <<[illegible]>>
Set FTP PASS to: <<[illegible]>>
Set FTP FILE URI to: <</samples/APP_OTA/helloworld.bin>>
Set LOCAL FINAL APP NAME to: <<helloworld.bin>>
Set LOCAL ORIGINAL APP NAME to: <<m2mapz.bin>>
[DEBUG] 23.53  ftp_utils.c:464 - msgFTPTask{FTPOTA_TASK}$ m2mb_os_ev_init success
[DEBUG] 23.54  ftp_utils.c:470 - msgFTPTask{FTPOTA_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.55  ftp_utils.c:478 - msgFTPTask{FTPOTA_TASK}$ Waiting for registration...
[DEBUG] 23.56  ftp_utils.c:371 - NetCallback{pubTspt_0}$ Module is registered to network
[DEBUG] 23.56  ftp_utils.c:491 - msgFTPTask{FTPOTA_TASK}$ Pdp context activation
[DEBUG] 23.57  ftp_utils.c:495 - msgFTPTask{FTPOTA_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 25.61  ftp_utils.c:504 - msgFTPTask{FTPOTA_TASK}$ Activate PDP with APN web.omnitel.it on cid 3....
[DEBUG] 26.30  ftp_utils.c:398 - PdpCallback{pubTspt_0}$ Context active
[DEBUG] 26.30  ftp_utils.c:401 - PdpCallback{pubTspt_0}$ IP address: 176.246.110.148
Start ftp client...
[DEBUG] 27.36  ftp_utils.c:533 - msgFTPTask{FTPOTA_TASK}$ Connected.
[DEBUG] 28.87  ftp_utils.c:546 - msgFTPTask{FTPOTA_TASK}$ FTP login successful.
Get remote file /samples/APP_OTA/helloworld.bin size
[DEBUG] 29.31  ftp_utils.c:568 - msgFTPTask{FTPOTA_TASK}$ Done. File size: 116224.
Starting download of remote file /samples/APP_OTA/helloworld.bin into local /mod/helloworld.bin
/samples/APP_OTA/helloworld.bin  4.68% 5440
/samples/APP_OTA/helloworld.bin  9.36% 10880
/samples/APP_OTA/helloworld.bin 14.04% 16320
/samples/APP_OTA/helloworld.bin 18.72% 21760
/samples/APP_OTA/helloworld.bin 23.40% 27200
/samples/APP_OTA/helloworld.bin 28.08% 32640
/samples/APP_OTA/helloworld.bin 32.76% 38080
/samples/APP_OTA/helloworld.bin 37.44% 43520
/samples/APP_OTA/helloworld.bin 42.13% 48960
/samples/APP_OTA/helloworld.bin 46.81% 54400
/samples/APP_OTA/helloworld.bin 51.49% 59840
/samples/APP_OTA/helloworld.bin 56.17% 65280
/samples/APP_OTA/helloworld.bin 60.85% 70720
/samples/APP_OTA/helloworld.bin 65.53% 76160
/samples/APP_OTA/helloworld.bin 70.21% 81600
/samples/APP_OTA/helloworld.bin 74.89% 87040
/samples/APP_OTA/helloworld.bin 79.57% 92480
/samples/APP_OTA/helloworld.bin 84.25% 97920
/samples/APP_OTA/helloworld.bin 88.93% 103360
/samples/APP_OTA/helloworld.bin 93.61% 108800
/samples/APP_OTA/helloworld.bin 97.42% 113220
[DEBUG] 43.54  ftp_utils.c:608 - msgFTPTask{FTPOTA_TASK}$ download successful.
FTP quit...
[DEBUG] 43.77  ftp_utils.c:632 - msgFTPTask{FTPOTA_TASK}$ Deactivating PDP
[DEBUG] 43.77  ftp_utils.c:642 - msgFTPTask{FTPOTA_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 44.20  ftp_utils.c:407 - PdpCallback{pubTspt_0}$ Context deactive
[DEBUG] 45.44  app_utils.c:76 - update_app{FTPOTA_TASK}$ Application successfully configured.
[DEBUG] 45.45  app_utils.c:82 - update_app{FTPOTA_TASK}$ Deleting old application /mod/m2mapz.bin
€ÿStarting. This is v1.0.7 built on Apr  7 2020 17:02:52. LEVEL: 2

 Start Hello world Application  [ version: 2.000000 ]

 Hello world 2.0 [ 000001 ]
 Hello world 2.0 [ 000002 ]
 Hello world 2.0 [ 000003 ]
```

Figure 119

3.6.4 CJSON example:

Sample application showcasing how to manage JSON objects. Debug prints on **USB0**

Features

- How to read a JSON using cJSON library
- How to write a JSON
- How to manipulate JSON objects

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Parse an example string into a JSON object and print the result in a formatted string
- Print some test outcomes (e.g. non existing item correctly not found)
- Retrieve single elements from the parsed JSON object and use them to format a descriptive string
- Delete the JSON object
- Create a new JSON object appending elements to it
- Print the result JSON string from the object

```
Starting Logging demo app. This is v1.0.7 built on Apr  7 2020 08:33:03.
And here is what we got:
{
        "name":         "Atlantic Ocean",
        "format":       {
                "type":         "salt",
                "volume":       310410900,
                "depth":        -8486,
                "volume_percent":       23.300000,
                "tide":         -3.500000,
                "calm":         false,
                "life":         ["plankton", "corals", "fish", "mammals"]
        }
}
inexistent key not found
name found: Atlantic Ocean
format found (null)
Our JSON string contains info about an ocean named Atlantic Ocean,
has a volume of 310410900 km^3 of salt water with -8486 meters max depth,
represents 23.3% of total oceans volume,
has an average low tide of -3.5 meters,
hosts a huge number of living creatures such as plankton, corals, fish, mammals,
and is not always calm.

Let's build a TR50 command with a proprety.publish and an alarm.publish for MQTT (no auth).
And here is what we got:
{
        "1":    {
                "command":      "property.publish",
                "params":       {
                        "thingKey":     "mything",
                        "key": "mykey",
                        "value":        123.144000
                }
        },
        "2":    {
                "command":      "alarm.publish",
                "params":       {
                        "thingKey":     "mything",
                        "key": "mykey",
                        "state":        3,
                        "msg": "Message."
                }
        }
}
END.
```

Figure 120

3.6.5 Crypto Elliptic Curve Cryptography (ECC) example

Sample application showcasing how to manage Elliptic Curve Cryptography functionalities. Debug prints on **USB0**

Features

- How to initialize ECC contexts A (Alice) and B (Bob). Alice is emulating a remote host, from which a public key is known.
- How to generate keypairs for contexts and export public keys
- how to export keyblobs from a context (a keyblob is encrypted with hw specific keys, and can only be used on the module where it was created)
- How to save a keyblob in secured TrustZone.
- How to reload a keyblob from the TrustZone into an initialized context
- How to sign a message with ECDSA from context B (Bob) and verify it from another context A (Alice) with the signature and public key of Bob.
- How to make Bob and Alice derive a shared session keys using each other's public key.
- How to make Bob and Alice create an AES context with the newly created shared keys, encode data and decode it on the other side

Application workflow

`M2MB_main.c`

- Create Bob ECC context, create a keypair and export it in a keyblob
- Open a file in secured Trust Zone, then store the keyblob in it.
- Destroy Bob ECC context
- Recreate Bob ECC context, open the file from Trust Zone and read the keyblob.
- Import the keyblob in Bob context.
- Export Bob public key
- Create Alice ECC context, to simulate an external host. Generate a keypair and export the public key.
- Sign a message with Bob context, generating a signature.
- Use Alice to verify the signed message using Bob's signature and public key
- Derive a shared key for Bob, using Alice's public key
- Create an AES context for Bob
- Import the shared key into the AES context
- Encrypt a message using Bob's AES context.
- Derive a shared key for Alice, using Bob's public key

- Create an AES context for Alice
- Import the shared key into the AES context
- Decrypt the message using Alice’s AES context.
- Check the decrypted message and the original one match
- Clear all resources

```
Starting Crypto ECC demo app. This is v1.0.9-C1 built on May 11 2020 16:30:23.

Bob (local) and Alice (remote) scenario
Bob's keypair generated
Bob's keyblob length is 224
Bob exported the keyblob to be securely stored.

Bob already had an item in Secure Data Area, it was removed to create a new one
Bob securely saved the keyblob in Secure Data Area
Releasing resources

Close Bob's context...
Done. Now Bob context does not exist anymore.

Re-initialize Bob Context and load the keyblob from the secure zone
Bob securely loaded the keyblob from the SDA
Import keyblob in Bob's context..
Done. Now export Bob's public key...
Bob's public key successfully exported

Alice's keypair generated
Alice's public key successfully exported

Bob's message signed with ECDSA!
Alice verified bob's message with his pubkey and signature!

------------------------------------------------------------
Bob and Alice will now exchange a message with AES encrypt
------------------------------------------------------------

Bob retrieved the generated shared key size
Bob's shared keyblob length is: 32. Allocate the required memory to store it.
Bob created a shared key using Alice's public key!

Bob created an AEX context to exchange encrypted data with Alice
Bob's AES context imported the shared keyblob
Bob Encrypted the message using AES and the shared key!
Encrypted data:
 94EE531E3B84B2A4EF05502186BFF5DA

Alice retrieved the generated shared key size
Alice's shared keyblob length is: 32. Allocate the required memory to store it.
Alice created a shared key using Bob's public key!

Alice created an AEX context to exchange encrypted data with Bob
Alice's AES context imported the shared keyblob
Alice decrypted the message using AES and the shared key!
Decrypted:
 414094941E8942A4445548035BFAE943

Original, plain message:
 414094941E8942A4445548035BFAE943

Plain and decrypted messages match!
```

Figure 121

3.6.6 EEPROM 24AA256

Sample application showing how to communicate with a MicroChip 24AA256T I2C EEPROM chip using azx eeprom utility APIs. Debug prints on **USB0**

Features

- Initialize the logs on the output channel
- configure the EEPROM utility, setting the slave address and the memory parameters (page size, memory size)
- Write single bytes on a random address
- Read written bytes as a page
- Write data using pages
- Read the new data using pages
- Read again using sequential reading
- Read a single byte from a specific address
- Read next byte using read from current address
- Erase the EEPROM
- Deinit EEPROM utility

Application workflow

`M2MB_main.c`

- call azx_eeprom_init() to set the utility parameters (SDA and SCL pins, page and memory sizes)
- call azx_eeprom_writeByte() to store a single byte with value “5” at the address 0x0213
- call azx_eeprom_writeByte() to store a single byte with value “6” at the address 0x0214
- call azx_eeprom_readPages() from address 0x0213 to retrieve the 2 bytes from the EEPROM
- call azx_eeprom_writePages to write 1024 bytes from a buffer, starting from address 0x00
- call azx_eeprom_readPages() again, to read 256 bytes from address 0x00
- call azx_eeprom_readSequentially() to read 256 bytes from 0x00 by without pages (less overhead on I2C protocol)
- call azx_eeprom_readByte() to get a single byte from address 0x00
- call azx_eeprom_readByteFromCurrentAddress() to get a byte from next address (0x01)
- call azx_eeprom_eraseAll() to completely erase the EEPROM memory (this writes 0xFF in each byte)
- call azx_eeprom_readPages from address 0x0213 to get 2 bytes and verify the values have been written to 0xFF

- call azx_eeprom_deinit to close the eeprom handler and the I2C channel

```
Starting I2C EEPROM 24AA256T demo app. This is v1.0.13-C1 built on Nov  3 2020 16:28:23.

Configuring the I2C device...
Opening I2C channel /dev/I2C-160 ( device address is 0xA0 )
Writing 1 byte at address 0x0213...
Done.
Writing 1 byte at address 0x0214...
Done.
Reading the 2 bytes from address 0x0213...
Done. Data: [0xFF 0xFF]

Writing 1024 bytes at address 0x0000..
Done.

Reading 256 bytes from address 0x0000...
Done. Data:
<<ABCDEFGHIJKLMNOPQRSTUVWXYZ.....................................................abcdefghijklmnopqrstuvwxyz.................

Reading 256 bytes sequentially from address 0x0000...
Done. Data:
<<ABCDEFGHIJKLMNOPQRSTUVWXYZ.....................................................abcdefghijklmnopqrstuvwxyz.................


Reading 1 byte from address 0x0000...
Done. Data: 'A'

Reading 1 byte from current address (should be 0x0001)...
Done. Data: 'B'


[DEBUG] 17.47  M2MB_main:177 - run_I2C_EEPROM_Demo{M2M_DamsStart}$ Erasing all the eeprom...
[DEBUG] 28.05  M2MB_main:185 - run_I2C_EEPROM_Demo{M2M_DamsStart}$ Done

Reading the 2 bytes from address 0x0213...
Done. Data: [0xFF 0xFF]

Deinit EEPROM...
Done
```

Figure 122

3.6.7 Easy AT example

Sample application showcasing Easy AT functionalities. Debug prints on **USB0**

Features

- Shows how to register custom commands

3.6.8 Events

Sample application showcasing events setup and usage. Debug prints on **USB0**

Features

- How to setup OS events with a custom bitmask
- How to wait for events and generate them in callback functions to synchronize blocks of code

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create an event handler
- Create a timer to generate an event, with a 2 seconds expiration time
- Wait for a specific event bit on the event handler
- At timer expiration, set the same event bit and verify that the code flow went through after the event.

```
Starting Events demo app. This is v1.0.7 built on Apr  7 2020 08:44:29.
[DEBUG] 20.55  M2MB_main.c:171 - M2MB_main{M2M_DamsStart}$ m2mb_os_ev_init success
Set the timer attributes structure success.
Timer successfully created
[DEBUG] 20.57  M2MB_main.c:125 - setup_timer{M2M_DamsStart}$ Start the timer, success.
[DEBUG] 22.60  M2MB_main.c:60 - hwTimerCb{pubTspt_0}$ Timer Callback, generate event!
[DEBUG] 22.61  M2MB_main.c:183 - M2MB_main{M2M_DamsStart}$ event occurred!
```

Figure 123

3.6.9 Events - Barrier (multi events)

Sample application showcasing how to setup and use multiple events to create a barrier. Debug prints on **USB0**

Features

- How to setup OS events to be used as a barrier
- How to wait for multiple events in the same point, and generate them in call-back functions to synchronize blocks of code

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create an event handler
- Create a timer to generate an event, with a 3 seconds expiration time
- Create another timer to generate an event, with a 6 seconds expiration time
- Start both timers
- Wait for both event bits on the event handler (each one will be set by one of the timers)
- At first timer expiration, set the first event bit and verify that the code flow does not procede.
- At second timer expiration, set the second event bit and verify that the code flow went through after the event (implementing a barrier).

```
Starting Barrier demo app. This is v1.0.7 built on Apr  7 2020 08:48:30.
[DEBUG] 20.01  M2MB_main.c:179 - M2MB_main{M2M_DamsStart}$ m2mb_os_ev_init success
Set the timer attributes structure success.
Timer successfully created with 3000 timeout (ms)
Set the timer attributes structure success.
Timer successfully created with 6000 timeout (ms)
[DEBUG] 23.08  M2MB_main.c:66 - hwTimerCb1{pubTspt_0}$ Timer Callback, generate event 1!
[DEBUG] 26.12  M2MB_main.c:75 - hwTimerCb2{pubTspt_0}$ Timer Callback, generate event 2!
[DEBUG] 26.13  M2MB_main.c:214 - M2MB_main{M2M_DamsStart}$ BOTH events occurred!
```

Figure 124

3.6.10 FOTA example

Sample application showcasing FOTA usage with M2MB API. Debug prints on **USB0**

Features

- How download a delta file from a remote server
- How to apply the delta and update the module firmware

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a main task to manage connectivity.
- create a fota task to manage FOTA and start it with INIT option

`fota.c`

fotaTask()

- Initialize FOTA system then reset parameters.
- Check current FOTA state, if not in IDLE, return error.
- Send a message to mainTask so networking is initialized.
- after PdPCallback() notifies the correct context activation, configure the fota client parameters such as FTP server URL, username and password
- get delta file from server. when it is completed, FOTADownloadCallback is called.
- If delta download went fine, check it.
- If delta file is correct, apply it. Once complete, restart the module.

mainTask()

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context. Event will be received on **PdP-Callback** function
- Disable PDP context when required to stop the app

PdpCallback()

- When PDP context is enabled, send a message to fotaTask to start the download

```
Starting FOTA demo app. This is v1.0.7 built on Apr  7 2020 16:24:29.
[DEBUG] 27.68  fota.c:185 - fotaTask{FOTA_TASK}$ Init FOTA...
Session file not present, procede with FOTA...
[DEBUG] 27.69  fota.c:229 - fotaTask{FOTA_TASK}$ m2mb_fota_reset PASS
[DEBUG] 27.70  fota.c:253 - fotaTask{FOTA_TASK}$ m2mb_fota_state_get  M2MB_FOTA_STATE_IDLE
[DEBUG] 27.71  fota.c:369 - mainTask{MAIN_TASK}$ Case INIT
[DEBUG] 27.71  fota.c:374 - mainTask{MAIN_TASK}$ Case WAIT_FOR_REGISTRATION
[DEBUG] 27.72  fota.c:378 - mainTask{MAIN_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 27.72  fota.c:385 - mainTask{MAIN_TASK}$ Waiting for registration...
[DEBUG] 27.73  fota.c:130 - NetCallback{pubTspt_0}$ Module is registered to cell 0xC4CF!
[DEBUG] 27.74  fota.c:395 - mainTask{MAIN_TASK}$ REGISTERED
[DEBUG] 27.74  fota.c:400 - mainTask{MAIN_TASK}$ Pdp context activation
[DEBUG] 27.75  fota.c:404 - mainTask{MAIN_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 29.71  fota.c:419 - mainTask{MAIN_TASK}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 30.44  fota.c:151 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 30.45  fota.c:154 - PdpCallback{pubTspt_0}$ IP address: 37.118.165.97

[DEBUG] 30.45  fota.c:278 - fotaTask{FOTA_TASK}$
Trying to download "samples/FOTA/30.00.006.2_to_30.00.006.2_ME910C1_NANVWW.bin" delta file...
[DEBUG] 30.47  fota.c:288 - fotaTask{FOTA_TASK}$ m2mb_fota_get_delta OK - Waiting for the completion callback
[DEBUG] 39.03  fota.c:95 - FOTADownloadCallBack{pubTspt_0}$ FOTA download Success - performing packet validation...
[DEBUG] 39.05  fota.c:294 - fotaTask{FOTA_TASK}$ Validating delta file...
[DEBUG] 122.34  fota.c:310 - fotaTask{FOTA_TASK}$ Packet is valid, start update...
[DEBUG] 122.38  fota.c:322 - fotaTask{FOTA_TASK}$ m2mb_fota_start PASS
[DEBUG] 124.39  fota.c:335 - fotaTask{FOTA_TASK}$
Rebooting...After reboot there will be the new FW running on module!

#OTAEV: Module Upgraded To New Fw

Starting FOTA demo app. This is v1.0.7 built on Apr  7 2020 16:24:29.
[DEBUG] 33.31  fota.c:185 - fotaTask{FOTA_TASK}$ Init FOTA...

Session file is already present, stop.
```

Figure 125

3.6.11 FTP

Sample application showcasing FTP client demo with AZX FTP. Debug prints on **USB0**

Features

- How to check module registration and activate PDP context
- How to connect to a FTP server
- How to exchange data with the server

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage FTP client and start it

`ftp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Init FTP client and set the debug function for it
- Connect to the server
- Perform log in
- Check remote file size and last modification time
- Download file from server to local filesystem. A data callback is set to report periodic info about the download status
- Upload the same file to the server with a different name. A data callback is set to report periodic info about the upload status
- Download another file content in a buffer instead of a file. A data callback is set to report periodic info about the download status
- Close the connection with FTP server
- Disable PDP context

```
Starting FTP demo app. This is v1.0.7 built on Apr  7 2020 11:17:36.
[DEBUG] 21.23  ftp_test.c:290 - msgFTPTask{FTP_TASK}$ INIT
[DEBUG] 21.23  ftp_test.c:304 - msgFTPTask{FTP_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.23  ftp_test.c:310 - msgFTPTask{FTP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.23  ftp_test.c:318 - msgFTPTask{FTP_TASK}$ Waiting for registration...
[DEBUG] 21.25  ftp_test.c:214 - NetCallback{pubTspt_0}$ Module is registered to network
[DEBUG] 21.26  ftp_test.c:331 - msgFTPTask{FTP_TASK}$ Pdp context activation
[DEBUG] 21.27  ftp_test.c:335 - msgFTPTask{FTP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.31  ftp_test.c:344 - msgFTPTask{FTP_TASK}$ Activate PDP with APN web.omnitel.it on cid 3....
[DEBUG] 24.09  ftp_test.c:241 - PdpCallback{pubTspt_0}$ Context active
[DEBUG] 24.10  ftp_test.c:244 - PdpCallback{pubTspt_0}$ IP address: 176.244.166.181
Start ftp client...
[DEBUG] 24.82  ftp_test.c:373 - msgFTPTask{FTP_TASK}$ Connected.
[DEBUG] 26.32  ftp_test.c:386 - msgFTPTask{FTP_TASK}$ FTP login successful.
Get remote file /samples/pattern_big.txt size
[DEBUG] 26.69  ftp_test.c:428 - msgFTPTask{FTP_TASK}$ Done. File size: 20026.
Get remote file /samples/pattern_big.txt last modification date
[DEBUG] 26.89  ftp_test.c:450 - msgFTPTask{FTP_TASK}$ Done. File last mod date: 20200407090654
.

Starting download of remote file /samples/pattern_big.txt into local /mod/_pattern_big.txt
/samples/pattern_big.txt 47.54% 9520
/samples/pattern_big.txt 100.00% 20026
[DEBUG] 29.75  ftp_test.c:488 - msgFTPTask{FTP_TASK}$ download successful.
[DEBUG] 29.76  ftp_test.c:522 - msgFTPTask{FTP_TASK}$
Local file /mod/_pattern_big.txt size: 20026

Starting upload of local file /mod/_pattern_big.txt
/mod/_pattern_big.txt 81.81% 16384
Upload successful.

Starting download of remote file /samples/pattern.txt into local buffer
Getting remote file /samples/pattern.txt size..
[DEBUG] 32.97  ftp_test.c:583 - msgFTPTask{FTP_TASK}$ Done. File size: 988.
Starting download of remote file /samples/pattern.txt to buffer
[DEBUG] 34.08  ftp_test.c:145 - buf_data_cb{FTP_TASK}$ Received START event
[DEBUG] 34.09  ftp_test.c:149 - buf_data_cb{FTP_TASK}$ Received DATA: 988 bytes on buffer 0x400399e0
[DEBUG] 34.26  ftp_test.c:153 - buf_data_cb{FTP_TASK}$ Received END event
[DEBUG] 34.26  ftp_test.c:623 - msgFTPTask{FTP_TASK}$ Download successful. Received 988 bytes<<<
0    |---------|    |---------|    |---------|    |---------|    |---------|   *
1    |    A    |    |    A    |    |    A    |    |    A    |    |    A    |    *
2    |   AAA   |    |   AAA   |    |   AAA   |    |   AAA   |    |   AAA   |     *
3    |  AAAAA  |    |  AAAAA  |    |  AAAAA  |    |  AAAAA  |    |  AAAAA  |      *
4    | AAAAAAA |    | AAAAAAA |    | AAAAAAA |    | AAAAAAA |    | AAAAAAA |       *
5    |AAAAAAAAA|    |AAAAAAAAA|    |AAAAAAAAA|    |AAAAAAAAA|    |AAAAAAAAA|        *
6    | AAAAAAA |    | AAAAAAA |    | AAAAAAA |    | AAAAAAA |    | AAAAAAA |       *
7    |  AAAAA  |    |  AAAAA  |    |  AAAAA  |    |  AAAAA  |    |  AAAAA  |      *
8    |   AAA   |    |   AAA   |    |   AAA   |    |   AAA   |    |   AAA   |     *
9    |    A    |    |    A    |    |    A    |    |    A    |    |    A    |    *
10   |---------|    |---------|    |---------|    |---------|    |---------|   *
11                                                                             *
12   |---------|    |---------|    |---------|    |---------|    |--->>>
```

Figure 126

3.6.12 File System example

Sample application showcasing M2MB File system API usage. Debug prints on **USB0**

Features

- How to open a file in write mode and write data in it
- How to reopen the file in read mode and read data from it

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Open file in write mode
- Write data in file
- Close file
- Reopen file in read mode
- Read data from file and print it
- Close file and delete it

```
Starting FileSystem demo app. This is v1.0.7 build on Mar 26 2020 09:50:19. LEVEL: 2
Opening /my_text_file.txt in write mode..
Buffer written successfully into file. 15 bytes were written.
Closing file.
Opening /my_text_file.txt in read only mode..
Received 15 bytes from file:
<Hello from file>
Closing file.
Deleting File
File deleted
App Completed
```

Figure 127

3.6.13 GNSS example

Sample application showing how to use GNSS functionality. Debug prints on **USB0**

Features

- How to enable GNSS receiver on module
- How to collect location information from receiver

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Init gnss, enable position report and start it.
- When a fix is available, a message will be printed by the GNSS callback function

```
START GNSS TEST APP
m2mb_gnss_enable OK
m2mb_gnss_start OK
latitude_valid: 1 - latitude: 39.228245
longitude_valid: 1 - longitude: 9.069106
altitude_valid: 1 - altitude: 12.000000
uncertainty_valid: 1 - uncertainty: 30.000000
velocity_valid: 1 - codingType: 0
speed_horizontal: 0.000000
bearing: 0.000000
timestamp_valid: 1 -timestamp: 1563376148000
speed_valid: 1 - speed: 0.060000
```

Figure 128

3.6.14 GPIO interrupt example

Sample application showing how to use GPIOs and interrupts. Debug prints on **USB0**

Features

- How to open a GPIO in input mode with interrupt
- How to open a second GPIO in output mode to trigger the first one

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Open GPIO 4 as output
- Open GPIO 3 as input and set interrupt for any edge (rising and falling). **A jumper must be used to short GPIO 3 and 4 pins.**
- Toggle GPIO 4 status high and low every second
- An interrupt is generated on GPIO 3

```
Starting GPIO interrupt demo app. This is v1.0.7 built on Mar 26 2020 16:33:01.
Setting gpio 3 interrupt...
Setting GPIO 4 HIGH
CALLBACK->Interrupt on GPIO 3! Value: 1
Setting GPIO 4 LOW
CALLBACK->Interrupt on GPIO 3! Value: 0
Setting GPIO 4 HIGH
CALLBACK->Interrupt on GPIO 3! Value: 1
Setting GPIO 4 LOW
CALLBACK->Interrupt on GPIO 3! Value: 0
Setting GPIO 4 HIGH
CALLBACK->Interrupt on GPIO 3! Value: 1
Setting GPIO 4 LOW
CALLBACK->Interrupt on GPIO 3! Value: 0
```

Figure 129

3.6.15 HTTP Client

Sample application showing how to use HTTPs client functionalities. Debug prints on **USB0**

Features

- How to check module registration and activate PDP context
- How to initialize the http client, set the debug hook function and the data call-back to manage incoming data
- How to perform GET, HEAD or POST operations

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage HTTP client and start it

`httpTaskCB`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create HTTP client options and initialize its functionality
- Create HTTP SSL config and initialize the SSL options
- Configure data management options for HTTP client
- Appy all configurations to HTTP client
- Perform a GET request to a server
- Disable PDP context

`DATA_CB`

- Print incoming data
- Set the abort flag to 0 to keep going.

```
Starting HTTP(s) client demo app. This is v1.0.13-C1 built on Aug 11 2020 16:56:28.
[DEBUG] 15.19  M2MB_main:259 - activatePdP{HttpClient}$ m2mb_os_ev_init success
[DEBUG] 15.20  M2MB_main:265 - activatePdP{HttpClient}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 15.20  M2MB_main:273 - activatePdP{HttpClient}$ Waiting for registration...
[DEBUG] 15.21  M2MB_main:119 - NetCallback{pubTspt_0}$ Module is registered to cell 0xC4CF!
[DEBUG] 15.22  M2MB_main:287 - activatePdP{HttpClient}$ Pdp context initialization
[DEBUG] 17.26  M2MB_main:297 - activatePdP{HttpClient}$ Activate PDP with APN NXT17.NET....
[DEBUG] 17.97  M2MB_main:146 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 17.97  M2MB_main:149 - PdpCallback{pubTspt_0}$ IP address: 100.77.54.97
Performing a GET request...

Host Address: linux-ip.net Port 80

Socket connected! |
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="author" content="Martin A. Brown" />
    <meta name="robots" content="index, follow"/>

    <meta property="og:title" content="http://linux-ip.net/"/>
    <meta property="og:url" content="http://linux-ip.net"/>
    <meta property="og:site_name" content="http://linux-ip.net/"/>
    <meta property="og:type" content="website"/>

    <link rel="canonical" href="http://linux-ip.net" />

    <title>http://linux-ip.net/</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" type="text/css" href="//netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-awesome.css" />
    <link rel="stylesheet" type="text/css" href="//netdna.bootstrapcdn.com/twitter-bootstrap/2.3.2/css/bootstrap-combined.min.css" />
    <link rel="stylesheet" type="text/css" href="http://linux-ip.net/theme/css/main.css" />

</head>

   ...
    <footer id="site-footer">
        <div class="row-fluid">
            <div class="span10 offset1">
                <address>
                    <p>
                        Powered by <a href="http://getpelican.com/">Pelican</a>
                        and <a href="http://python.org">Python</a>.
                        Theme based on <a href="http://github.com/jsliang/pelican-fresh/">Fresh</a>
                        by <a href="http://jsliang.com/">jsliang</a>
                    </p>
                </address>
            </div>
        </div>
    </footer>
</body>
</html>
Done.
[DEBUG] 22.44  M2MB_main:155 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 130

3.6.16 HW Timer (Hardware Timer)

The sample application shows how to use HW Timers M2MB API. Debug prints on **USB0**

Features

- How to open configure a HW timer
- How to use the timer to manage recurring events

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create hw timer structure
- Configure it with 100 ms timeout, periodic timer (auto fires when expires) and autostart
- Init the timer with the parameters
- Wait 10 seconds
- Stop the timer

`TimerCb`

- Print a message with an increasing counter

```
Starting HW Timers demo app. This is v1.0.7 built on Mar 26 2020 13:04:14.
[DEBUG] 14.06  M2MB_main.c:114 - M2MB_main{M2M_DamsStart}$ Set the timer attributes structure: success.
Timer successfully created
Start the timer, success.
[DEBUG] 14.18  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [0]
[DEBUG] 14.28  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [1]
[DEBUG] 14.38  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [2]
[DEBUG] 14.48  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [3]
[DEBUG] 14.58  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [4]
[DEBUG] 14.69  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [5]
[DEBUG] 14.79  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [6]
[DEBUG] 14.88  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [7]
[DEBUG] 14.98  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [8]
[DEBUG] 15.08  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [9]


[DEBUG] 23.90  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [96]
[DEBUG] 24.01  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [97]
[DEBUG] 24.11  M2MB_main.c:55 - TimerCb{pubTspt_0}$ Callback Count: [98]
Stop a running timer: success
Application end
```

Figure 131

3.6.17 Hello World

The application prints “Hello World!” over selected output every two seconds. Debug prints on **USB0**, using AZX log example functions

Features

- How to open an output channel using AZX LOG sample functions
- How to print logging information on the channel using AZX LOG sample functions

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print “Hello World!” every 2 seconds in a while loop

```
Starting. This is v1.0.7 built on Mar 26 2020 09:34:16. LEVEL: 2

 Start Hello world Application  [ version: 2.000000 ]

 Hello world 2.0 [ 000001 ]
 Hello world 2.0 [ 000002 ]
 Hello world 2.0 [ 000003 ]
 Hello world 2.0 [ 000004 ]
 Hello world 2.0 [ 000005 ]
 Hello world 2.0 [ 000006 ]
 Hello world 2.0 [ 000007 ]
 Hello world 2.0 [ 000008 ]
 Hello world 2.0 [ 000009 ]
```

Figure 132

3.6.18 I2C example

Sample application showing how to communicate with an I2C slave device. Debug prints on **USB0**

Features

- How to open a communication channel with an I2C slave device
- How to send and receive data to/from the slave device

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Open I2C bus, setting SDA an SCL pins as 2 and 3 respectively
- Set registers to configure accelerometer -Read in a loop the 6 registers carrying the 3 axes values and show the g value for each of them

```
Starting I2C demo app. This is v1.0.7 built on Mar 26 2020 16:50:40.

Configuring the Kionix device...
opening channel /dev/I2C-30
[DEBUG] 20.18  M2MB_main.c:218 - test_I2C{M2M_DamsStart}$ -
WHOAMI content: 0x01
Configuring I2C Registers - Writing 0x4D into 0x1D register (CTRL_REG3)...
Write: success

I2C reading data from 0x1D register (CTRL_REG3)...
Read: success.
Accelerometer Enabled. ODR tilt: 12.5Hz, ODR directional tap: 400Hz, ORD Motion Wakeup: 50Hz
Configuring I2C Registers - Writing 0xC0 into 0x1B register (CTRL_REG1)...
Write: success

I2C reading data from 0x1B register (CTRL_REG1)...
Read: success.
Accelerometer Enabled. Operative mode, 12bit resolution
I2C read axes registers
------------
Reading Success.

X: -0.050 g
Y: -0.046 g
Z: 1.006 g
Reading Success.

X: -0.049 g
Y: -0.044 g
Z: 1.004 g
Reading Success.

X: -0.052 g
Y: -0.044 g
Z: 1.007 g
Reading Success.

X: -0.048 g
Y: -0.045 g
Z: 1.005 g
```

Figure 133

3.6.19 I2C Combined

Sample application showing how to communicate with an I2C slave device with I2C raw mode. Debug prints on **USB0**

Features

- How to open a communication channel with an I2C slave device
- How to send and receive data to/from the slave device using raw mode API

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Open I2C bus, setting SDA an SCL pins as 2 and 3 respectively
- Set registers to configure accelerometer -Read in a loop the 6 registers carrying the 3 axes values and show the g value for each of them

```
Starting I2C raw demo app. This is v1.0.13-C1 built on Jul 30 2020 11:28:18.

Configuring the I2C device...
Opening I2C channel /dev/I2C-30 ( device address is 0x0F << 1 )
Accelerometer Enabled. ODR tilt: 12.5Hz, ODR directional tap: 400Hz, ORD Motion Wakeup: 50Hz
Accelerometer Enabled. Operative mode, 12bit resolution
I2C read axes registers
------------

X: 0.000 g
Y: 0.000 g
Z: 0.000 g

X: -0.270 g
Y: 0.016 g
Z: 0.917 g

X: -0.268 g
Y: 0.013 g
Z: 0.925 g

X: -0.271 g
Y: 0.015 g
Z: 0.922 g

X: -0.267 g
Y: 0.016 g
Z: 0.918 g

X: -0.274 g
Y: 0.019 g
Z: 0.915 g
```

Figure 134

3.6.20 LWM2M

Sample application showcasing LWM2M client usage with M2MB API. Debug prints on **USB0**

Features

- Configure LWM2M client and enable it
- Create an instance of a custom object
- Set an integer value on a read only resource
- Set two integer values on a multi-instance read only resource
- write a string on a read/write resource
- Manage exec requests from the portal
- Manage write, read and monitoring resources changed from the portal

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create a task to manage the LWM2M client and start it

`lwm2m_demo.c`

`msgLWM2MTask` - Check registration status

- Configure APN to the correct one for CID 1
- Initialize LWM2M client,
- Check for XML file fo custom object
- Enable unsolicited messages from client
- Create a task (lwm2m_taskCB is its callback function)to manage events from Portal
- Enable LwM2M client
- Create a new instance for the custom object
- Wait for client to register to Portal
- Send integer and string values
- Wait for events from server

`lwm2mIndicationCB`

- Manage events arriving from client (operations completion status and unsolicited events)
- Run lwm2m_taskCB when a monitored resource changes, to manage the action to be done

3.6.20.1 Custom Object configuration

The XML file content must be loaded on the Telit IoT Portal for the demo application to be fully executed.

First, enter Developer section from the top menu

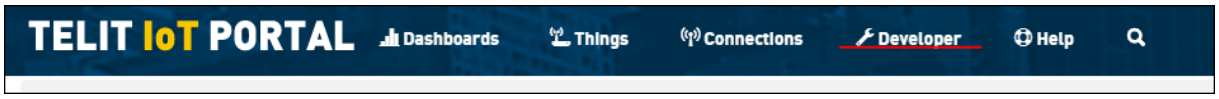

Figure 135

Choose Object Registry

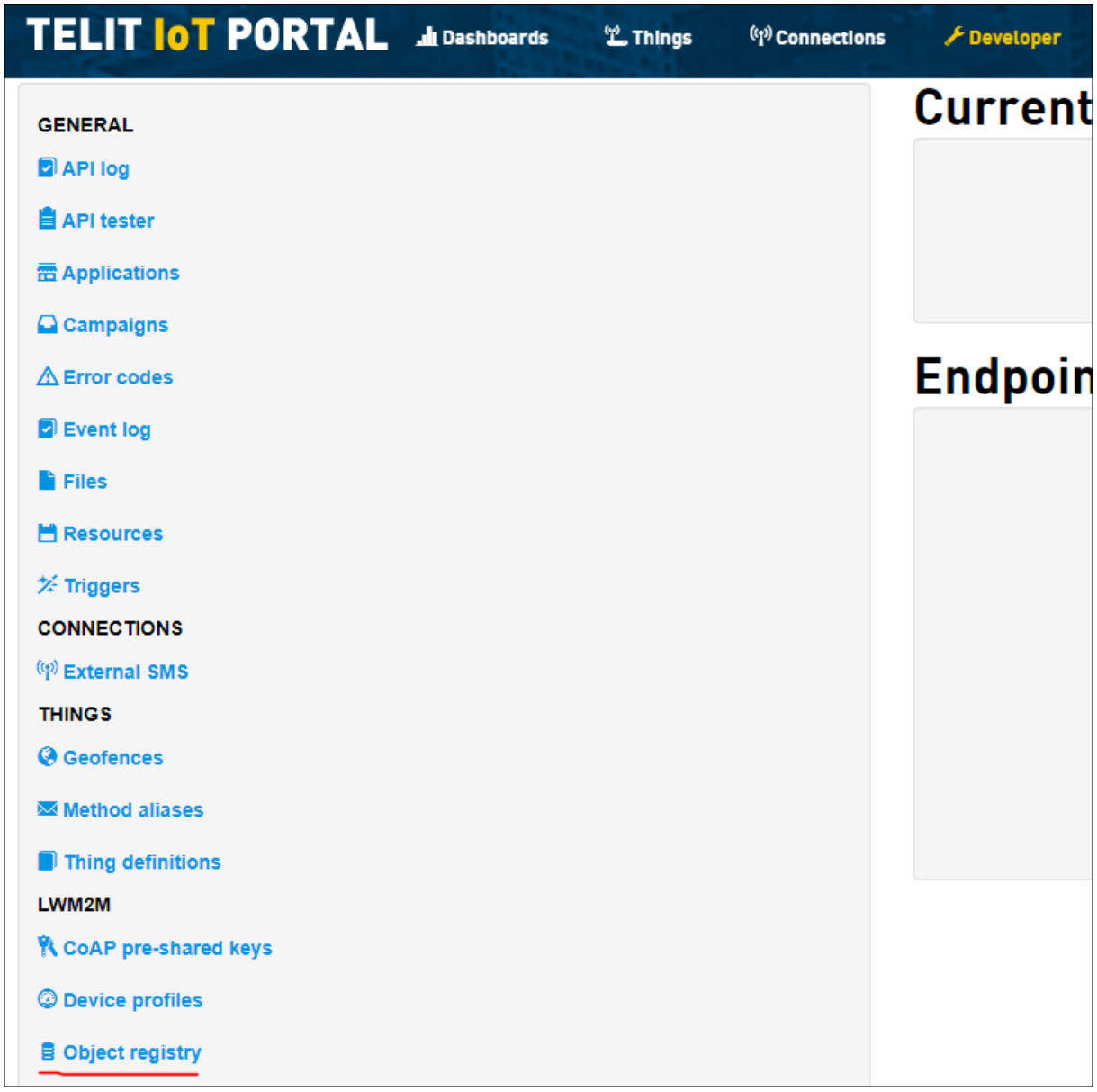

Figure 136

Create a New Object

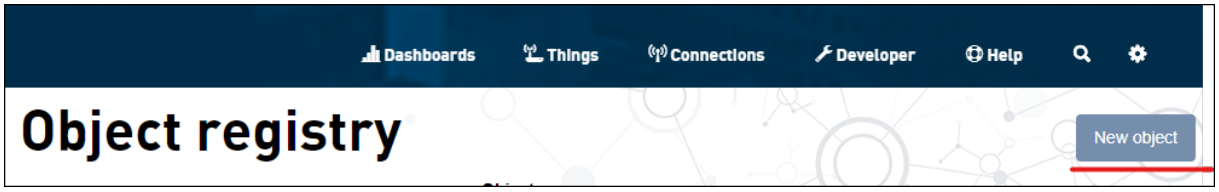

Figure 137

Copy the xml file content and paste it in the new Object form

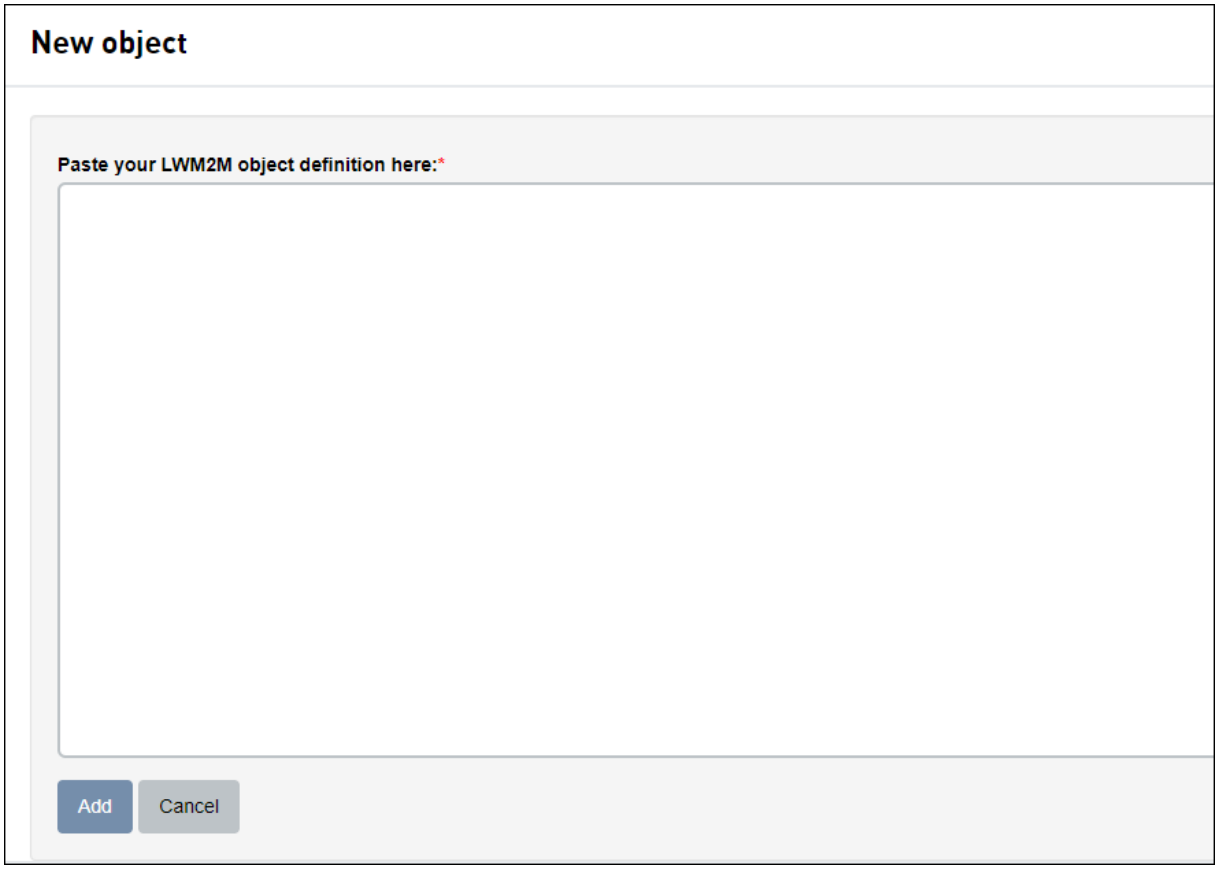

Figure 138

Also, the application requires the XML file `/xml/object_35000.xml` (provided with the sample files) to be stored in module's `/XML/` folder. It can be done with

`AT#M2MWRITE=/XML/object_35000.xml,<size_in_bytes>`

To load the XML file in the module, Telit AT Controller (TATC) can be used. Once the command above is issued, press the load content button:

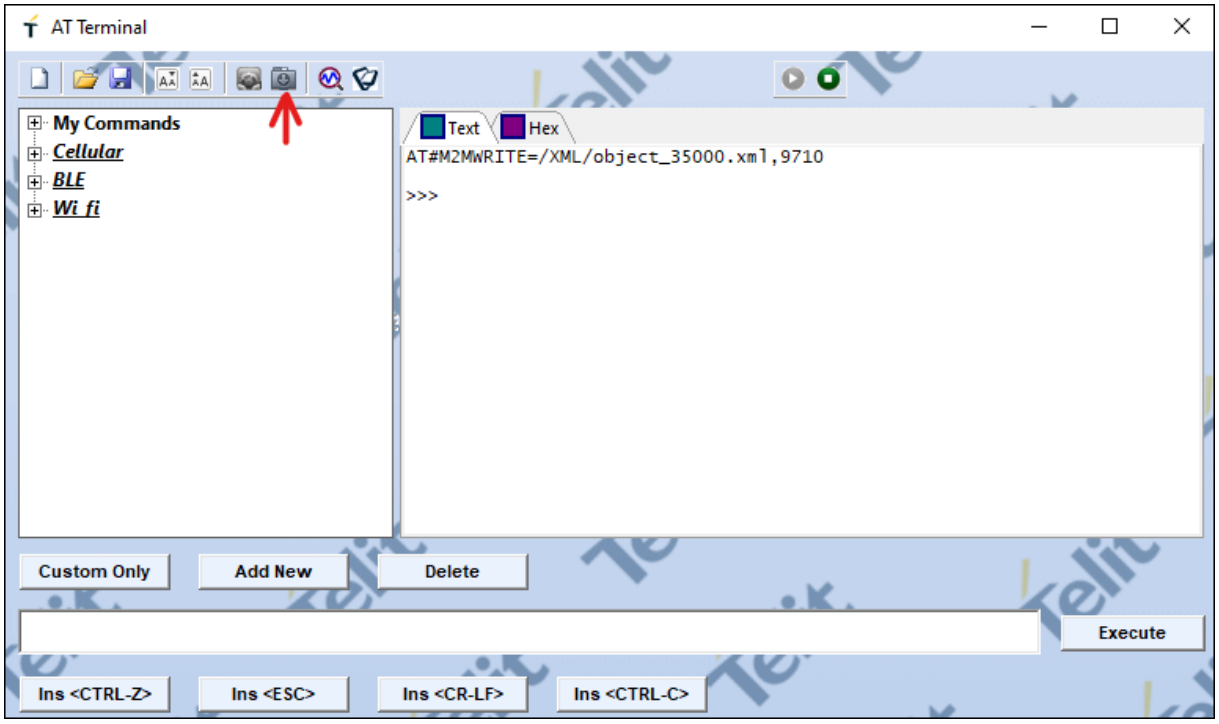

Figure 139

Select the file from your computer

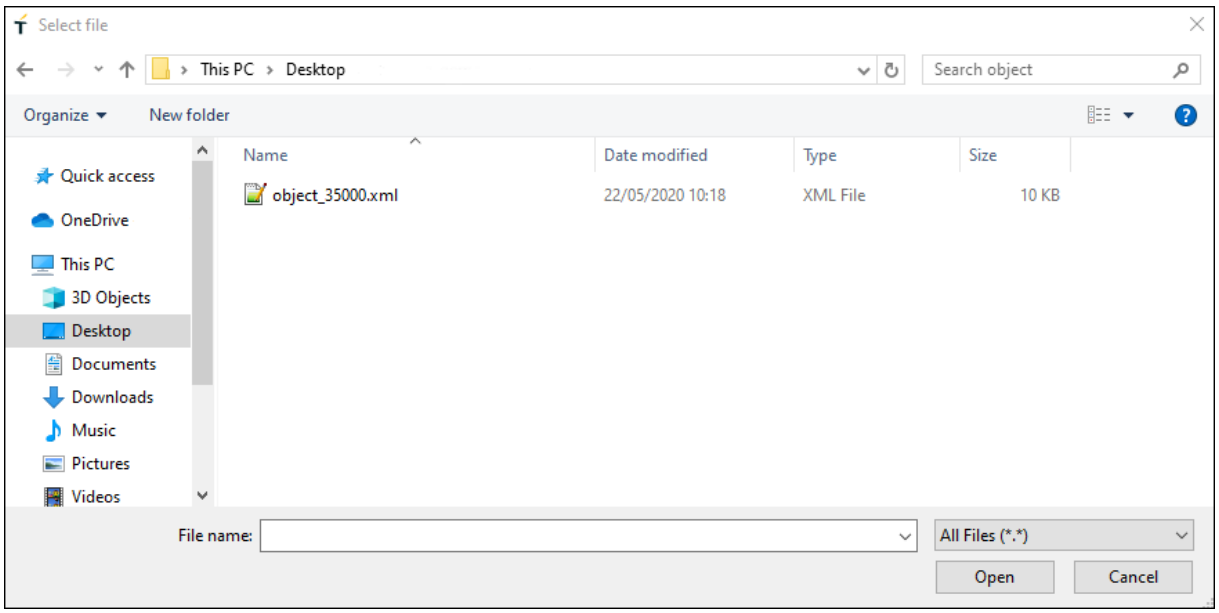

Figure 140

The file is successfully loaded on the module

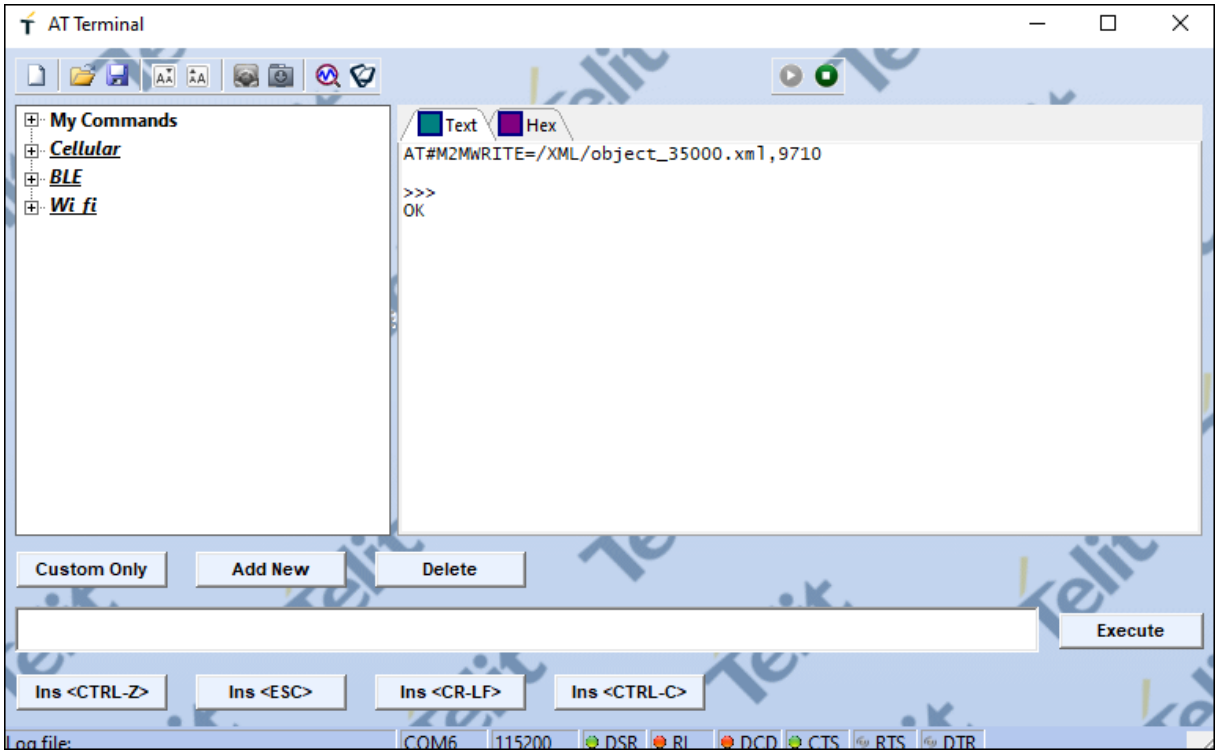

Figure 141

3.6.20.2 Application execution example

```
Starting lwm2m demo. This is v1.0.13-C1 built on Jul  6 2020 06:54:58.
On OneEdge portal, be sure that observations are enabled for the following object resources:
{35000/0/01} {35000/0/02} {35000/0/03} {35000/0/04} {35000/0/05} {35000/0/06} {35000/0/07}
{35000/0/11} {35000/0/12} {35000/0/13} {35000/0/14} {35000/0/15} {35000/0/16} {35000/0/17}
{35000/0/21} {35000/0/22} {35000/0/23} {35000/0/24} {35000/0/25} {35000/0/26} {35000/0/27}
{35000/0/31} {35000/0/32} {35000/0/33} {35000/0/34} {35000/0/35} {35000/0/36} {35000/0/37}

Initializing resources...
LWM2M enable result OK
[DEBUG] 32.39  lwm2m_demo:637 - lwm2mIndicationCB{pubTspt_0}$ Monitoring enabled.

Waiting LWM2M Registering (120 seconds timeout)...
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPING
resp->info == M2MB_LWM2M_CL_STATE_BOOTSTRAPPED
resp->info == M2MB_LWM2M_CL_STATE_REGISTERING
resp->info == M2MB_LWM2M_CL_STATE_REGISTERED

Waiting for events from portal. Write on monitored resource or call an exec

GET STATUS.
IF Status: M2MB_LWM2M_IF_STATE_ACTIVE
Client Status: M2MB_LWM2M_CL_STATE_REGISTERING
```

Figure 142

```
Setting integer resource {35000/0/2} value to 50 on LWM2M client.

Setting integer resource {35000/0/22/0} value to 10 on LWM2M client.

Resource /35000/0/2/0 changed!

Setting integer resource {35000/0/22/1} value to 11 on LWM2M client.
[DEBUG] 52.05  lwm2m_demo:470 - lwm2mIndicationCB{pubTspt_0}$ Read type: 3; size: 4

Resource /35000/0/22/0 changed!

Integer data in {35000/0/2/0} resource was updated to new value: 50

Writing string resource {35000/0/11} value to demo_string on LWM2M client.
[DEBUG] 52.57  lwm2m_demo:470 - lwm2mIndicationCB{pubTspt_0}$ Read type: 3; size: 4

Resource /35000/0/22/1 changed!

Integer data in {35000/0/22/0} resource was updated to new value: 10
[DEBUG] 54.19  lwm2m_demo:470 - lwm2mIndicationCB{pubTspt_0}$ Read type: 3; size: 4

Resource /35000/0/11/0 changed!

Integer data in {35000/0/22/1} resource was updated to new value: 11
[DEBUG] 54.71  lwm2m_demo:470 - lwm2mIndicationCB{pubTspt_0}$ Read type: 2; size: 11

String data in {35000/0/11/0} resource was updated to new content: <demo_string>
```

Figure 143

After the Demo completes the initialization, it is possible to access the object resources from the Portal Object Browser

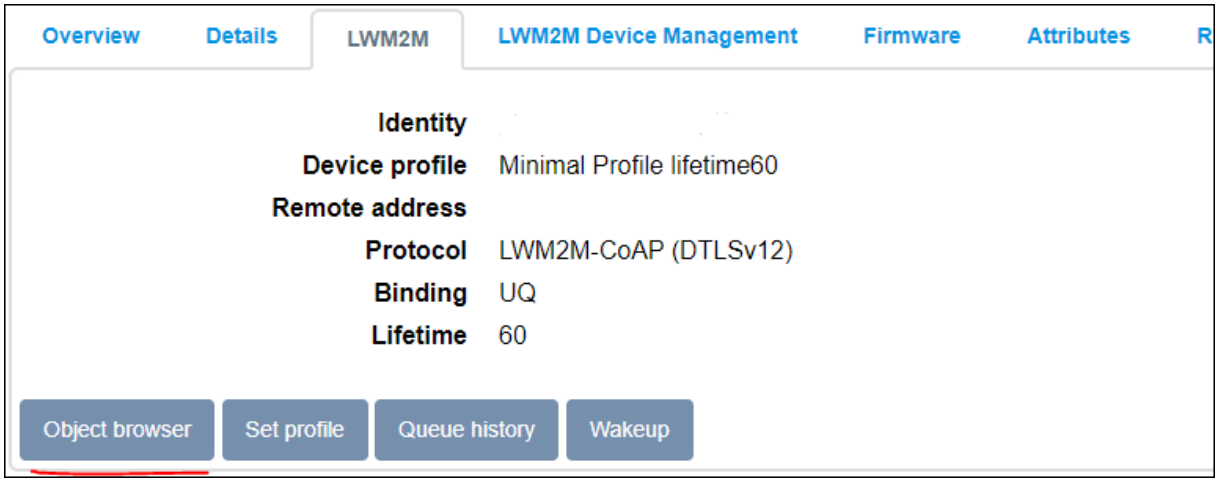

Figure 144

An instance of the object will be present and the resources can be modified.

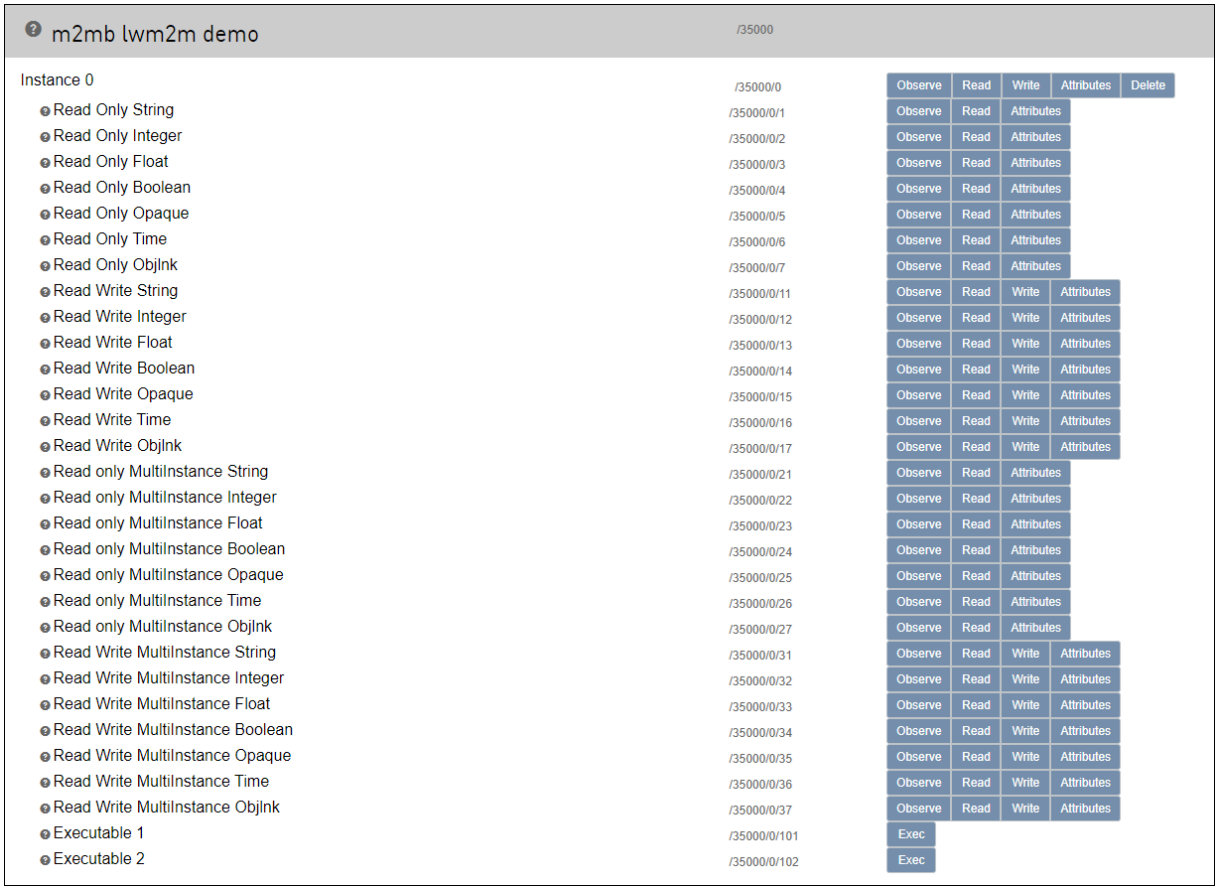

Figure 145

For example, executing the two Exec Resources at the bottom of the list, the application will react accordingly:

```
Info Exec Ind: 35000/0/101/0
Asked to execute resource 101

Info Exec Ind: 35000/0/102/0
Asked to execute resource 102
```

Figure 146

Writing a string resource (id /35000/0/11), the application will notify the change

```
Resource /35000/0/11/0 changed!
[DEBUG] 1249.86  lwm2m_demo:470 - lwm2mIndicationCB{pubTspt_0}$ Read type: 2; size: 5

String data in {35000/0/11/0} resource was updated to new content: <hello>
```

Figure 147

3.6.21 Logging Demo

Sample application showing how to print on one of the available output interfaces. Debug prints on **USB0**

Features

- How to open a logging channel
- How to set a logging level
- How to use different logging macros

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Print a message with every log level

```
Starting Logging demo app. This is v1.0.7 built on Mar 26 2020 13:57:06.
[WARN ] 20.17  M2MB_main.c:74 - M2MB_main{M2M_DamsStart}$ This is a WARNING MESSAGE
[ERROR] 20.18  M2MB_main.c:76 - M2MB_main{M2M_DamsStart}$ THIS IS AN ERROR MESSAGE
[CRITICAL] 20.19  M2MB_main.c:78 - M2MB_main{M2M_DamsStart}$ THIS IS AN CRITICAL MESSAGE
[DEBUG] 20.19  M2MB_main.c:80 - M2MB_main{M2M_DamsStart}$ This is a DEBUG message
[TRACE] 20.20  M2MB_main.c:82 - M2MB_main{M2M_DamsStart}$ This is a TRACE message
END.
```

Figure 148

3.6.22 MD5 example

Sample application showing how to compute MD5 hashes using m2mb crypto. Debug prints on **USB0**

Features

- Compute MD5 hash of a file
- Compute MD5 hash of a string

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create a temporary file with the expected content
- Compute MD5 hash of the provided text file
- Compare the hash with the expected one
- Compute MD5 hash of a string
- Compare the hash with the expected one
- Delete test file

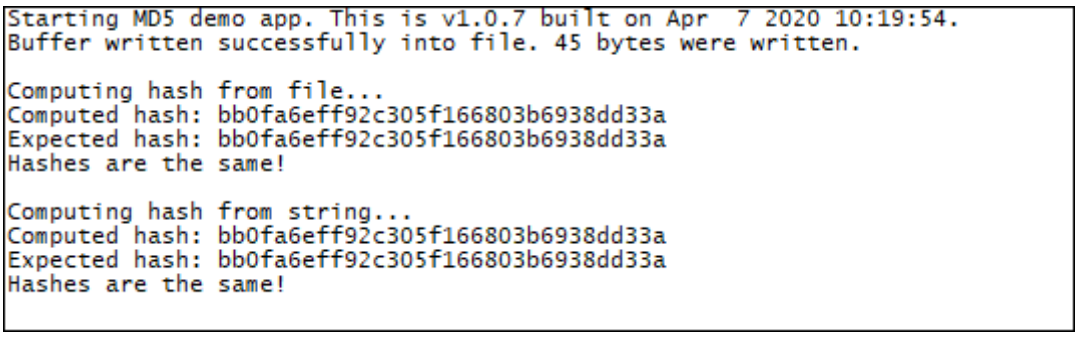

Figure 149

3.6.23 MQTT Client

Sample application showcasing MQTT client functionalities (with SSL). Debug prints on **USB0**

Features

- How to check module registration and enable PDP context
- How to configure MQTT client parameters
- How to connect to a broker with SSL and exchange data over a subscribed topic

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage MQTT client and start it

`mqtt_demo.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Init MQTT client
- Configure it with all parameters (Client ID, username, password, PDP context ID, keepalive timeout...)
- Connect MQTT client to broker
- Subscribe to two topics
- Publish 10 messages with increasing counter. Even messages are sent to topic 1, odd messages on topic 2.
- Print received message in mqtt_topc_cb function
- Disconnect MQTT client and deinit it
- Disable PDP context

```
Starting MQTT demo app. This is v1.0.7 built on Apr  7 2020 10:34:08.
[DEBUG] 16.18  mqtt_demo.c:192 - MQTT_Task{MQTT_TASK}$ INIT
[DEBUG] 16.18  mqtt_demo.c:206 - MQTT_Task{MQTT_TASK}$ m2mb_os_ev_init success
[DEBUG] 16.19  mqtt_demo.c:214 - MQTT_Task{MQTT_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 16.19  mqtt_demo.c:221 - MQTT_Task{MQTT_TASK}$ Waiting for registration...
[DEBUG] 16.20  mqtt_demo.c:131 - NetCallback{pubTspt_0}$ Module is registered
[DEBUG] 16.21  mqtt_demo.c:232 - MQTT_Task{MQTT_TASK}$ Pdp context activation
[DEBUG] 18.26  mqtt_demo.c:246 - MQTT_Task{MQTT_TASK}$ Activate PDP with APN web.omnitel.it on CID 3....
[DEBUG] 18.95  mqtt_demo.c:155 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 18.96  mqtt_demo.c:159 - PdpCallback{pubTspt_0}$ IP address: 37.118.201.56
[DEBUG] 18.96  mqtt_demo.c:268 - MQTT_Task{MQTT_TASK}$ Init MQTT
[DEBUG] 18.97  mqtt_demo.c:278 - MQTT_Task{MQTT_TASK}$ m2mb_mqtt_init succeeded

Connecting to broker <api-dev.devicewise.com>:1883...
Done.
Subscribing to test_topic and test_topic2..
[DEBUG] 20.35  mqtt_demo.c:367 - MQTT_Task{MQTT_TASK}$ Done.

[DEBUG] 20.36  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 2> to topic test_topic
[DEBUG] 20.37  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 20.71  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 20.72  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 2>
[DEBUG] 23.37  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 3> to topic test_topic2
[DEBUG] 23.38  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 23.92  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 23.93  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 3>
[DEBUG] 26.40  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 4> to topic test_topic
[DEBUG] 26.41  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 26.93  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 26.93  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 4>
[DEBUG] 29.42  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 5> to topic test_topic2
[DEBUG] 29.43  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 29.99  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 30.00  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 5>
[DEBUG] 32.46  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 6> to topic test_topic
[DEBUG] 32.48  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 33.00  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 33.01  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 6>
[DEBUG] 35.47  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 7> to topic test_topic2
[DEBUG] 35.48  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 36.01  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 36.02  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 7>
[DEBUG] 38.50  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 8> to topic test_topic
[DEBUG] 38.51  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 39.15  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 27
[DEBUG] 39.16  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 8>
[DEBUG] 41.52  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 9> to topic test_topic2
[DEBUG] 41.53  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 42.10  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 27
[DEBUG] 42.12  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 9>
[DEBUG] 44.56  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 10> to topic test_topic
[DEBUG] 44.57  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 45.09  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic; data len: 28
[DEBUG] 45.11  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 10>
[DEBUG] 47.58  mqtt_demo.c:392 - MQTT_Task{MQTT_TASK}$ PUBLISHING <Hello from M2MB MQTT! ID: 11> to topic test_topic2
[DEBUG] 47.59  mqtt_demo.c:397 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 48.12  mqtt_demo.c:103 - mqtt_topic_cb{MQTT_Async}$ MQTT Message on Topic test_topic2; data len: 28
[DEBUG] 48.13  mqtt_demo.c:107 - mqtt_topic_cb{MQTT_Async}$ Message: <Hello from M2MB MQTT! ID: 11>

Disconnecting from MQTT broker..
[DEBUG] 50.60  mqtt_demo.c:414 - MQTT_Task{MQTT_TASK}$ Done.
[DEBUG] 50.61  mqtt_demo.c:443 - MQTT_Task{MQTT_TASK}$ application exit
[DEBUG] 50.62  mqtt_demo.c:453 - MQTT_Task{MQTT_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 50.63  mqtt_demo.c:457 - MQTT_Task{MQTT_TASK}$ Application complete.
[DEBUG] 51.23  mqtt_demo.c:164 - PdpCallback{pubTspt_0}$ Context deactivated!
```

Figure 150

3.6.24 MultiTask

Sample application showcasing multi tasking functionalities with M2MB API. Debug prints on **USB0**

Features

- How to create tasks using azx utilities
- How to use send messages to tasks
- How to use a semaphore to synchronize two tasks

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create three tasks with the provided utility (this calls public m2mb APIs)
- Send a message to the task1, its callback function azx_msgTask1 will be called

`azx_msgTask1`

- Print received parameters from main
- Send modified parameters to task2 (its callback function azx_msgTask2 will be called)
- wait for an InterProcess Communication semaphore to be available (released by task3)
- Once the semaphore is available, print a message and return

`azx_msgTask2`

- Print received parameters from caller
- If first parameter is bigger than a certain value, Send modified parameters to task3
- Else, use the second parameter as a task handle and print the corresponding name plus the value of the first parameter

`azx_msgTask3`

- Print received parameters from task 2
- release IPC semaphore
- send message to task 2 with first parameter below the threshold and second parameter with task3 handle

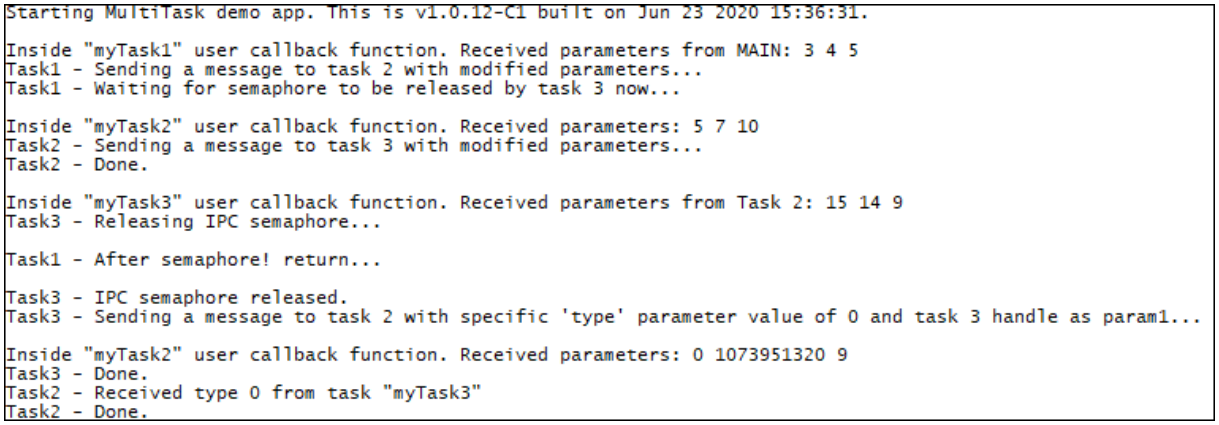
Starting MultiTask demo app. This is v1.0.12-C1 built on Jun 23 2020 15:36:31.
Inside "myTask1" user callback function. Received parameters from MAIN: 3 4 5
Task1 - Sending a message to task 2 with modified parameters...
Task1 - Waiting for semaphore to be released by task 3 now...
Inside "myTask2" user callback function. Received parameters: 5 7 10
Task2 - Sending a message to task 3 with modified parameters...
Task2 - Done.
Inside "myTask3" user callback function. Received parameters from Task 2: 15 14 9
Task3 - Releasing IPC semaphore...
Task1 - After semaphore! return...
Task3 - IPC semaphore released.
Task3 - Sending a message to task 2 with specific 'type' parameter value of 0 and task 3 handle as param1...
Inside "myTask2" user callback function. Received parameters: 0 1073951320 9
Task3 - Done.
Task2 - Received type 0 from task "myTask3"
Task2 - Done.

Figure 151

3.6.25 MutEx

Sample application showing mutex usage, with ownership and prioritization usage. Debug prints on **USB0**

Features

- How to create a mutex
- How to use the mutex with tasks having different priorities
- how to reorder the pending tasks queue for the mutex

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create four tasks with the provided utility (this calls public m2mb APIs). The first task is a “producer”, putting data on a shared buffer. The second is a “consumer” of said data, the other two are used for prioritization demo
- run producer and consumer tasks at the same pace. the shared buffer will stay empty, because the resource is consumed right after creation
- run producer twice as fast as consumer. The buffer is slowly filled
- run consumer twice as fast as publisher. The buffer is always empty.
- reserve the mutex in the main task and run producer, support and support2 tasks (in this order). Then release the mutex and check the execution order. It should be by arrival.
- reserve the mutex in the main task and run the same three task, but before releasing the mutex, call the prioritization API. the task with highest priority (producer) is put as first in the queue.

```
Starting MutEx app. This is v1.0.12-C1 built on Jul  1 2020 08:37:15.
[DEBUG] 14.50  M2MB_main:90 - mutex_init{M2M_DamsStart}$ [MUTEX] Mutex initialized

[CASE 1 ] Producer and consumer have same idle time

[DEBUG] 14.51  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 14.52  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 14.53  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 14.53  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 14.54  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 14.54  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 14.55  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 14.56  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 15.56  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 15.56  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 15.57  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 15.58  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 15.58  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 15.59  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 15.60  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 15.60  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 16.61  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 16.61  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 16.62  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 16.63  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 16.63  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 16.64  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 16.64  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 16.65  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
```

Figure 152

```
[CASE 2 ] Producer has double idle time

[DEBUG] 17.56  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 17.56  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 17.57  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 17.58  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 17.58  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 17.59  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 17.59  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 17.60  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 18.63  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 18.64  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 0 items
[DEBUG] 18.64  M2MB_main:268 - msgConsumer{CONSUMER}$ Can't consume anything, buffer size is 0
[DEBUG] 18.65  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 19.62  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 19.62  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 19.63  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 19.64  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 19.68  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 19.69  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 19.69  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 19.70  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 20.73  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 20.74  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 0 items
[DEBUG] 20.75  M2MB_main:268 - msgConsumer{CONSUMER}$ Can't consume anything, buffer size is 0
[DEBUG] 20.75  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 21.67  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 21.68  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 21.68  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 21.69  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 21.77  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 21.79  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 21.80  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 21.80  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
```

Figure 153

```
[CASE 3 ] Producer has half idle time

[DEBUG] 22.62  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 22.63  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 22.64  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 22.64  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 22.65  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 22.65  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 22.66  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 22.67  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 23.67  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 23.68  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 23.68  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 23.69  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 24.71  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 24.72  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 1 items
[DEBUG] 24.72  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 0
[DEBUG] 24.73  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 24.74  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 24.74  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 0 items
[DEBUG] 24.75  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 0
[DEBUG] 24.76  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 25.79  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 25.79  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 1 items
[DEBUG] 25.80  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 1
[DEBUG] 25.81  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
[DEBUG] 26.78  M2MB_main:250 - msgConsumer{CONSUMER}$ Mutex acquired
[DEBUG] 26.78  M2MB_main:251 - msgConsumer{CONSUMER}$ Now there are 2 items
[DEBUG] 26.79  M2MB_main:261 - msgConsumer{CONSUMER}$ [CONSUMER]I consumed 99 from index 1
[DEBUG] 26.79  M2MB_main:308 - msgConsumer{CONSUMER}$ Mutex released
[DEBUG] 26.84  M2MB_main:119 - msgProducer{PRODUCER}$ Mutex acquired
[DEBUG] 26.84  M2MB_main:120 - msgProducer{PRODUCER}$ Now there are 1 items
[DEBUG] 26.85  M2MB_main:125 - msgProducer{PRODUCER}$ Produced item 99 at index 1
[DEBUG] 26.86  M2MB_main:176 - msgProducer{PRODUCER}$ Mutex released
```

Figure 154

```
[CASE 4 ] NO HTPF

Reserve MUTEX so all tasks are enqueued
[DEBUG] 30.77  M2MB_main:387 - msgSupport{HPTF_SUPPORT}$ freepos = 0 | evaluate[freepos]= 3
[DEBUG] 30.78  M2MB_main:416 - msgSupport2{HPTF_SUPPORT2}$ freepos = 1 | evaluate[freepos]= 4
[DEBUG] 30.79  M2MB_main:223 - msgProducer{PRODUCER}$ producer: freepos = 2 | evaluate[freepos]= 1
[DEBUG] 35.85  M2MB_main:586 - M2MB_main{M2M_DamsStart}$ EVALUATE SEQUENCE IS 3 4 1. Expected: 3 4 1
NO HTPF OK

[CASE 4.1 ] HTPF USED

Reserve MUTEX so all tasks are enqueued
m2mb_os_mtx_hptf OK
[DEBUG] 41.98  M2MB_main:223 - msgProducer{PRODUCER}$ producer: freepos = 0 | evaluate[freepos]= 1
[DEBUG] 41.99  M2MB_main:387 - msgSupport{HPTF_SUPPORT}$ freepos = 1 | evaluate[freepos]= 3
[DEBUG] 42.00  M2MB_main:416 - msgSupport2{HPTF_SUPPORT2}$ freepos = 2 | evaluate[freepos]= 4
[DEBUG] 44.03  M2MB_main:650 - M2MB_main{M2M_DamsStart}$ EVALUATE SEQUENCE IS 1 3 4, expected 1 3 4
HTPF DEMO OK
The application has ended...
```

Figure 155

3.6.26 SMS PDU

Sample application showcasing how to create and decode PDUs to be used with m2mb_sms_* API set. A SIM card and antenna must be present. Debug prints on **USB0**

Features

- How to enable SMS functionality
- How to use encode an SMS PDU to be sent with m2mb_api
- How to decode a received SMS response from PDU to ASCII mode.

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Init sms functionality
- Create PDU from text message
- Send message to destination number
- Wait for response
- When SMS PDU response is received, decode it and print information about it, plus the message content

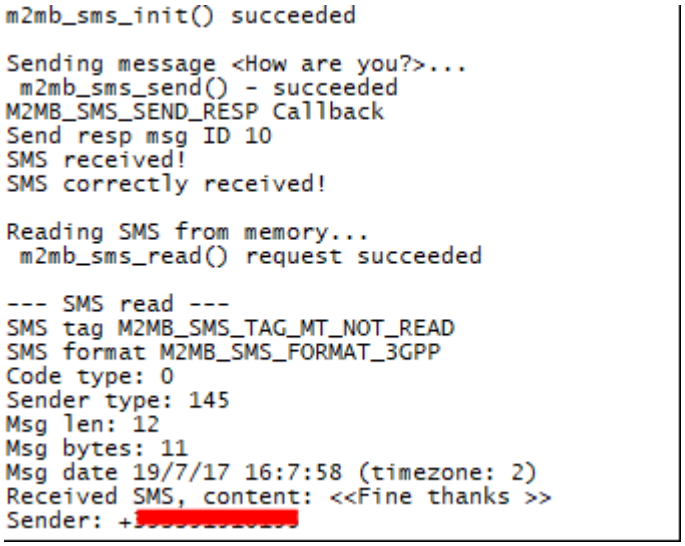

Figure 156

3.6.27 SPI Echo

Sample application showing how to communicate over SPI with m2mb API. Debug prints on **USB0**

Features

- How to open an SPI bus. MOSI and MISO will be shorted, to have an echo.
- How to communicate over SPI bus

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Open SPI bus, set parameters
- Send data on MOSI and read the same in MISO

```
Starting SPI demo app. This is v1.0.7 built on Apr  1 2020 13:48:05.
Transfer successful. Received: hello from spi echo
```

Figure 157

3.6.28 SPI sensors

Sample application showing SPI usage, configuring two ST devices: a magnetometer (ST LIS3MDL) and a gyroscope (ST L3G4200D). The application will read values from both devices using GPIO4 and 3 (respectively) as magnetometer CS and gyro CS. Debug prints on **USB0**

Features

- How to open an SPI bus with a slave device
- How to communicate with the device over the SPI bus

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Open SPI bus, set parameters
- Configure `GPIO 3` and `GPIO 4` as output, set them high (idle)
- Set registers to configure magnetometer
- Read in a loop (10 iterations) the registers carrying the 3 axes values and show the gauss value for each of them. A metal object is put close to the sensor to change the read values.
- Set registers to configure gyroscope
- Read in a loop (10 iterations) the registers carrying the 3 axes values and show the degrees per second value for each of them. The board is rotated to change the read values.

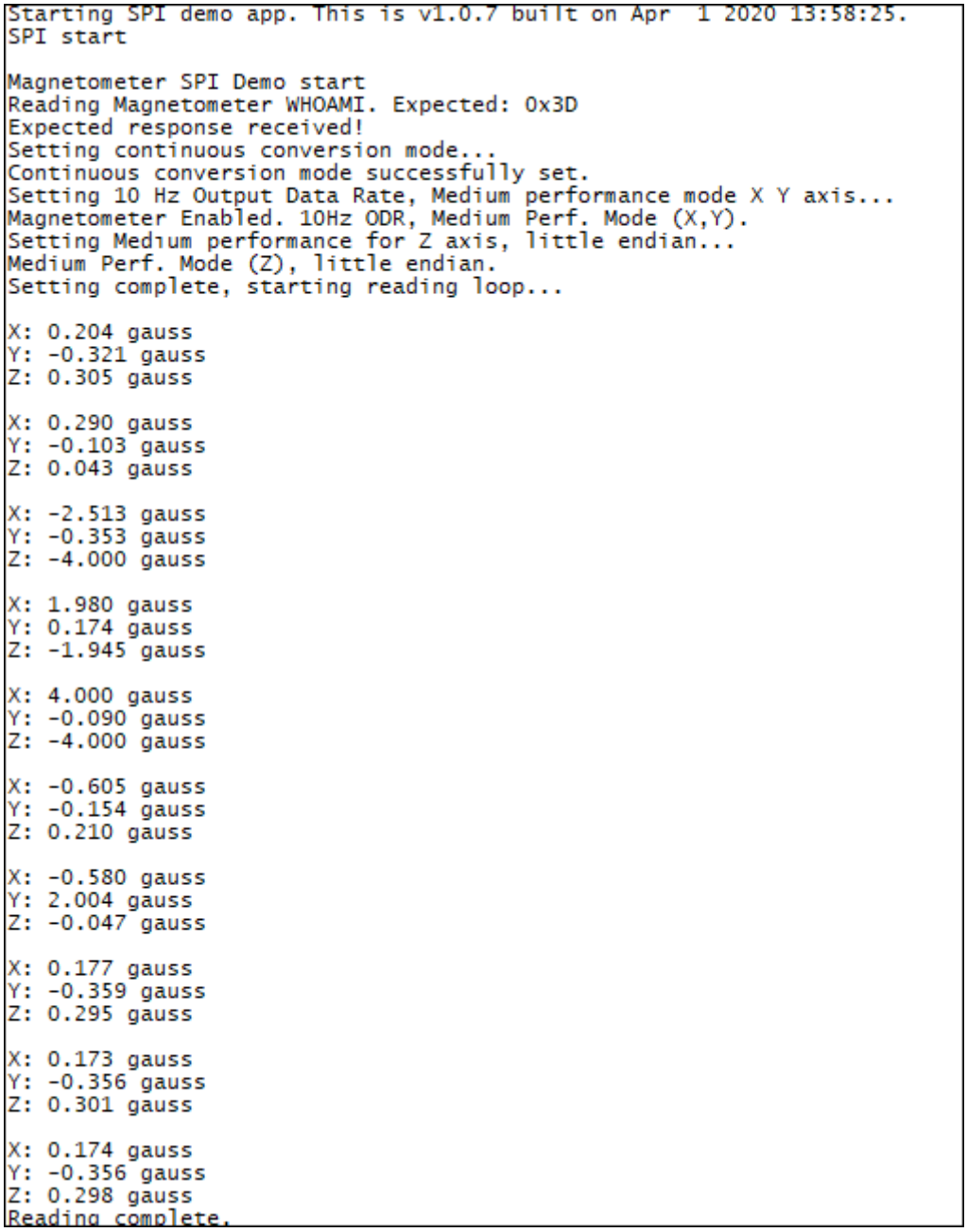
Starting SPI demo app. This is v1.0.7 built on Apr 1 2020 13:58:25.
SPI start
Magnetometer SPI Demo start
Reading Magnetometer WHOAMI. Expected: 0x3D
Expected response received!
Setting continuous conversion mode...
Continuous conversion mode successfully set.
Setting 10 Hz Output Data Rate, Medium performance mode X Y axis...
Magnetometer Enabled. 10Hz ODR, Medium Perf. Mode (X,Y).
Setting Medium performance for Z axis, little endian...
Medium Perf. Mode (Z), little endian.
Setting complete, starting reading loop...
X: 0.204 gauss
Y: -0.321 gauss
Z: 0.305 gauss
X: 0.290 gauss
Y: -0.103 gauss
Z: 0.043 gauss
X: -2.513 gauss
Y: -0.353 gauss
Z: -4.000 gauss
X: 1.980 gauss
Y: 0.174 gauss
Z: -1.945 gauss
X: 4.000 gauss
Y: -0.090 gauss
Z: -4.000 gauss
X: -0.605 gauss
Y: -0.154 gauss
Z: 0.210 gauss
X: -0.580 gauss
Y: 2.004 gauss
Z: -0.047 gauss
X: 0.177 gauss
Y: -0.359 gauss
Z: 0.295 gauss
X: 0.173 gauss
Y: -0.356 gauss
Z: 0.301 gauss
X: 0.174 gauss
Y: -0.356 gauss
Z: 0.298 gauss
Reading complete.

Figure 158

3.6.29 SW Timer (Software Timer)

The sample application shows how to use SW Timers M2MB API. Debug prints on **USB0**

Features

- How to open configure a SW timer
- How to use the timer to manage recurring events

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create sw timer structure
- Configure it with 4 seconds timeout, periodic timer (auto fires when expires)
- Init the timer with the parameters
- Start the timer
- Wait 10 seconds
- Stop the timer

`timerCb`

- Print a message with inside the callback

```
Starting SW Timers demo app. This is v1.0.7 built on Apr  7 2020 09:51:25.
timer expired!
[DEBUG] 21.41  M2MB_main.c:59 - timerCb{pubTspt_0}$ timer handle: 0x4002b004
timer expired!
[DEBUG] 25.47  M2MB_main.c:59 - timerCb{pubTspt_0}$ timer handle: 0x4002b004
stopping the timer
Stop a running timer: success
Application end
```

Figure 159

3.6.30 Secure MicroService

Sample application showcasing how to manage secure microservice functionalities. Debug prints on **USB0**

Features

- Write data in Secure Data Area (SDA), non protected
- Read the written data and compare with the original buffer
- Write a cripty key in Secure Data Area (SDA), non protected
- Perform a rotate of the written key data
- Perform MD5 sum of written data from TZ file
- Compare computed digest with expected one
- Write data in trust zone as a trusted object (it will not be possible to read it again but only use its content for crypto operations)
- Try to read the trusted object and verify it fails
- Rotate trusted item and verify retrieving the content fails
- compute MD5 sum of trusted item and compare with the expected one
- Try to pass data from a trusted item to a non trusted item using untrusted TZ buffers, and verify it fails

Application workflow

`M2MB_main.c`

- Write a buffer in a SDA item using `m2mb_secure_ms_write`
- Read the same item using `m2mb_secure_ms_read`
- Write a buffer containing some cripty key in a SDA item using `m2mb_secure_ms_write`
- Rotate the content of the key item
- Read it with `m2mb_secure_ms_read`
- Load the key content using `m2mb_secure_ms_crypto_alloc` and `m2mb_secure_crypto_add_it` in a SECURE_MS buffer
- Compute MD digest with `m2mb_secure_ms_crypto_md`
- Write a buffer containing some cripty key in a SDA item using `m2mb_secure_ms_write` but with **TRUSTED** option in `m2mb_secure_ms_open`
- Verify that `m2mb_secure_ms_read` on the trusted item fails
- Verify that `m2mb_secure_ms_crypto_rotate` fails for the trusted item
- Verify the MD5 digest
- Try to copy the trusted item data in a SECURE_MS buffer with `m2mb_secure_ms_crypto_alloc` and `m2mb_secure_crypto_add_item`, then load it in an untrusted object with `m2mb_secure_ms_crypto_write`, and verify it fails.

```
Starting secure ms demo app. This is v1.0.13-C1 built on Jul 30 2020 12:19:02.
Writing data in normal item
Stored input data in Secure Data Area
Reading data from normal item
Data length in SDA: 11 bytes
Securely loaded the data from the SDA
Read 11 bytes: <hello world>
original and retrieved strings are the same

Writing key in normal item
Stored input data in Secure Data Area

Rotate data in normal item
Original key: AA_THIS_IS_MY_SECRET_KEY_BB
Rotated key:

Compute MD5 of data in normal item
Data length in SDA: 27 bytes
MD5: 8EDAD26E26E1C74C7C02386C1C7F541D
hash is the expected one!

Writing data in trusted item
Stored input data in Secure Data Area
Reading data from trusted item (should fail!)
Data length in SDA: 27 bytes
m2mb_secure_ms_read() failed for trusted item, as expected!

Rotate data in trusted item
[ERROR] 17.01  M2MB_main:329 - read_rotate{M2M_DamsStart}$ Cannot read data from SECURE_MS_BUFFER to user buffer
Original key: AA_THIS_IS_MY_SECRET_KEY_BB
Rotated key:

Compute MD5 of data in trusted item
Data length in SDA: 27 bytes

MD5: 8EDAD26E26E1C74C7C02386C1C7F541D
Hash is the expected one!

Try to pass data from trusted to untrusted through TZ buffers
Cannot store data from SECURE_MS_BUFFER to SDA 'non-trusted', as expected
```

Figure 160

3.6.31 TCP IP

Sample application showcasing TCP echo demo with M2MB API. Debug prints on **USB0**

Features

- How to check module registration and activate PDP context
- How to open a TCP client socket
- How to communicate over the socket

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Connect to the server
- Send data and receive response
- Close socket
- Disable PDP context

```
Starting TCP-IP demo app. This is v1.0.7 built on Mar 26 2020 16:20:30.
[DEBUG] 21.23  m2m_tcp_test.c:201 - M2M_msgTCPTask{TCP_TASK}$ INIT
[DEBUG] 21.25  m2m_tcp_test.c:217 - M2M_msgTCPTask{TCP_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.26  m2m_tcp_test.c:223 - M2M_msgTCPTask{TCP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.26  m2m_tcp_test.c:231 - M2M_msgTCPTask{TCP_TASK}$ Waiting for registration...
[DEBUG] 21.28  m2m_tcp_test.c:128 - NetCallback{pubTspt_0}$ Module is registered to cell 0x816B!
[DEBUG] 21.29  m2m_tcp_test.c:244 - M2M_msgTCPTask{TCP_TASK}$ Pdp context activation
[DEBUG] 21.30  m2m_tcp_test.c:248 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.34  m2m_tcp_test.c:263 - M2M_msgTCPTask{TCP_TASK}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 24.52  m2m_tcp_test.c:155 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.52  m2m_tcp_test.c:158 - PdpCallback{pubTspt_0}$ IP address: 83.225.44.56
[DEBUG] 24.54  m2m_tcp_test.c:273 - M2M_msgTCPTask{TCP_TASK}$ Creating Socket...
[DEBUG] 24.54  m2m_tcp_test.c:284 - M2M_msgTCPTask{TCP_TASK}$ Socket created
[DEBUG] 24.55  m2m_tcp_test.c:294 - M2M_msgTCPTask{TCP_TASK}$ Socket ctx set to 3
[DEBUG] 24.95  m2m_tcp_test.c:307 - M2M_msgTCPTask{TCP_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 25.17  m2m_tcp_test.c:322 - M2M_msgTCPTask{TCP_TASK}$ Socket Connected!
[DEBUG] 25.18  m2m_tcp_test.c:329 - M2M_msgTCPTask{TCP_TASK}$ Sending data over socket..
[DEBUG] 25.19  m2m_tcp_test.c:342 - M2M_msgTCPTask{TCP_TASK}$ Data send successfully (16 bytes)
[DEBUG] 27.20  m2m_tcp_test.c:356 - M2M_msgTCPTask{TCP_TASK}$ trying to receive 16 bytes..
[DEBUG] 27.21  m2m_tcp_test.c:364 - M2M_msgTCPTask{TCP_TASK}$ Data received (16): <hello from m2mb!>
[DEBUG] 27.21  m2m_tcp_test.c:373 - M2M_msgTCPTask{TCP_TASK}$ application exit
[DEBUG] 27.22  m2m_tcp_test.c:385 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 27.24  m2m_tcp_test.c:388 - M2M_msgTCPTask{TCP_TASK}$ Application complete.
[DEBUG] 29.43  m2m_tcp_test.c:164 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 161

3.6.32 TCP Socket status

Sample application showcasing how to check a TPC connected socket current status. Debug prints on **USB0**

Features

- How to check module registration and activate PDP context
- How to open a TCP client socket
- How to check if the TCP socket is still valid

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Connect to the server
- Check in a loop the current socket status using the adv_select function with a 2 seconds timeout
- Close socket when the remote host closes it
- Disable PDP context

```
Starting TCP socket status check demo app. This is v1.0.14-C1 built on Sep  8 2020 14:59:25.
[DEBUG] 21.33  m2m_tcp_tes:324 - M2M_msgTCPTask{TCP_TASK}$ INIT
[DEBUG] 21.34  m2m_tcp_tes:338 - M2M_msgTCPTask{TCP_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.34  m2m_tcp_tes:344 - M2M_msgTCPTask{TCP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.35  m2m_tcp_tes:352 - M2M_msgTCPTask{TCP_TASK}$ Waiting for registration...
[DEBUG] 21.36  m2m_tcp_tes:365 - M2M_msgTCPTask{TCP_TASK}$ Pdp context activation
[DEBUG] 21.37  m2m_tcp_tes:369 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.41  m2m_tcp_tes:384 - M2M_msgTCPTask{TCP_TASK}$ Activate PDP with APN NXT17.NET....
[DEBUG] 24.09  m2m_tcp_tes:281 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.10  m2m_tcp_tes:284 - PdpCallback{pubTspt_0}$ IP address: 100.77.5.223
[DEBUG] 24.10  m2m_tcp_tes:394 - M2M_msgTCPTask{TCP_TASK}$ Creating Socket...
[DEBUG] 24.11  m2m_tcp_tes:405 - M2M_msgTCPTask{TCP_TASK}$ Socket created
[DEBUG] 24.11  m2m_tcp_tes:415 - M2M_msgTCPTask{TCP_TASK}$ Socket ctx set to 3
[DEBUG] 24.60  m2m_tcp_tes:428 - M2M_msgTCPTask{TCP_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 24.93  m2m_tcp_tes:443 - M2M_msgTCPTask{TCP_TASK}$ Socket Connected!
[DEBUG] 26.98  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 29.03  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...

...

[DEBUG] 82.18  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 84.23  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 86.28  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 88.31  m2m_tcp_tes:461 - M2M_msgTCPTask{TCP_TASK}$ Socket does not have any event, try again...
[DEBUG] 88.90  m2m_tcp_tes:154 - adv_select{TCP_TASK}$ Data is available on socket <0x40032b3c>
[DEBUG] 88.92  m2m_tcp_tes:160 - adv_select{TCP_TASK}$ There are <0> pending bytes on socket
Socket was closed by remote!
[DEBUG] 88.92  m2m_tcp_tes:494 - M2M_msgTCPTask{TCP_TASK}$ application exit
[DEBUG] 88.94  m2m_tcp_tes:506 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 88.94  m2m_tcp_tes:509 - M2M_msgTCPTask{TCP_TASK}$ Application complete.
[DEBUG] 89.31  m2m_tcp_tes:290 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 162

3.6.33 TCP Server

Sample application showcasing TCP listening socket demo with M2MB API. Debug prints on **USB0**

Features

- How to check module registration and activate PDP context
- How to open a TCP listening socket
- How to manage external hosts connection and exchange data

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task to manage socket and start it

`m2m_tcp_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and set it in non-blocking mode
- Bind the socket to the listening port
- Start listening for incoming connection
- Check if a connection is incoming using m2mb_socket_bsd_select function
- If a client connects, perform accept on the child socket
- Send a “START” message to the client
- Send some data
- Wait for data from client and print it
- Close the child socket
- Start listening again, up to 3 times
- Close listening socket
- Disable PDP context

Debug Log

```
Starting TCP Server demo app. This is v1.0.7 built on Apr  7 2020 13:28:24.
[DEBUG] 14.55  m2m_tcp_test.c:220 - M2M_msgTCPTask{TCP_TASK}$ INIT
[DEBUG] 14.55  m2m_tcp_test.c:236 - M2M_msgTCPTask{TCP_TASK}$ m2mb_os_ev_init success
[DEBUG] 14.57  m2m_tcp_test.c:242 - M2M_msgTCPTask{TCP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 14.57  m2m_tcp_test.c:250 - M2M_msgTCPTask{TCP_TASK}$ Waiting for registration...
[DEBUG] 14.58  m2m_tcp_test.c:138 - NetCallback{pubTspt_0}$ Module is registered to cell 0x5222!
[DEBUG] 14.59  m2m_tcp_test.c:263 - M2M_msgTCPTask{TCP_TASK}$ Pdp context activation
[DEBUG] 14.60  m2m_tcp_test.c:267 - M2M_msgTCPTask{TCP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 16.57  m2m_tcp_test.c:282 - M2M_msgTCPTask{TCP_TASK}$ Activate PDP with APN ibox.tim.it....
[DEBUG] 17.16  m2m_tcp_test.c:165 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 17.17  m2m_tcp_test.c:168 - PdpCallback{pubTspt_0}$ IP address: 2.195.165.137


-------------------------------------------
|                                         |
|  Start TCP server                       |
|                                         |
-------------------------------------------


[DEBUG] 19.15  m2m_tcp_test.c:301 - M2M_msgTCPTask{TCP_TASK}$ Creating Socket...
[DEBUG] 19.15  m2m_tcp_test.c:312 - M2M_msgTCPTask{TCP_TASK}$ Socket created
[DEBUG] 19.16  m2m_tcp_test.c:313 - M2M_msgTCPTask{TCP_TASK}$ m2mb_socket_bsd_socket(): valid socket ID [0x4002E79C] - PASS
[DEBUG] 20.16  m2m_tcp_test.c:319 - M2M_msgTCPTask{TCP_TASK}$ issuing m2m_socket_bsd_ioctl() to set non-blocking mode ...
[DEBUG] 20.17  m2m_tcp_test.c:331 - M2M_msgTCPTask{TCP_TASK}$ Binding Socket...
[DEBUG] 22.12  m2m_tcp_test.c:343 - M2M_msgTCPTask{TCP_TASK}$ Socket Bind Pass


Start TCP listening on port 6500...

[DEBUG] 24.13  m2m_tcp_test.c:368 - M2M_msgTCPTask{TCP_TASK}$ select...
Select result: 0
[DEBUG] 28.13  m2m_tcp_test.c:368 - M2M_msgTCPTask{TCP_TASK}$ select...
Select result: 1

TCP Server Coming Connection

--> Accept
[DEBUG] 30.52  m2m_tcp_test.c:397 - M2M_msgTCPTask{TCP_TASK}$ Socket Accept Pass


Connected! (socket dial n.1)

[DEBUG] 30.53  m2m_tcp_test.c:403 - M2M_msgTCPTask{TCP_TASK}$ Client Source Address: 185.86.42.254
[DEBUG] 30.54  m2m_tcp_test.c:404 - M2M_msgTCPTask{TCP_TASK}$ Client Port: 58658
[DEBUG] 30.54  m2m_tcp_test.c:405 - M2M_msgTCPTask{TCP_TASK}$ Client Family: 2
[DEBUG] 31.56  m2m_tcp_test.c:410 - M2M_msgTCPTask{TCP_TASK}$
-------------------------------
[DEBUG] 31.57  m2m_tcp_test.c:411 - M2M_msgTCPTask{TCP_TASK}$ |    Send/receive data test    |
[DEBUG] 31.57  m2m_tcp_test.c:412 - M2M_msgTCPTask{TCP_TASK}$ -------------------------------

[DEBUG] 32.58  m2m_tcp_test.c:416 - M2M_msgTCPTask{TCP_TASK}$
--> issuing m2mb_socket_bsd_send(): transmit "__START____" packet...
[DEBUG] 32.59  m2m_tcp_test.c:423 - M2M_msgTCPTask{TCP_TASK}$  --> done (11 have been transmitted)
[DEBUG] 32.60  m2m_tcp_test.c:425 - M2M_msgTCPTask{TCP_TASK}$      ALL data transmitted - PASS
[DEBUG] 32.61  m2m_tcp_test.c:430 - M2M_msgTCPTask{TCP_TASK}$
--> issuing m2mb_socket_bsd_send(): transmit 58 bytes...
[DEBUG] 32.62  m2m_tcp_test.c:437 - M2M_msgTCPTask{TCP_TASK}$  --> done (58 have been transmitted)
[DEBUG] 32.63  m2m_tcp_test.c:440 - M2M_msgTCPTask{TCP_TASK}$      ALL data transmitted - PASS
[DEBUG] 32.64  m2m_tcp_test.c:448 - M2M_msgTCPTask{TCP_TASK}$
Waiting for data...

[DEBUG] 39.64  m2m_tcp_test.c:457 - M2M_msgTCPTask{TCP_TASK}$ test
[DEBUG] 99.61  m2m_tcp_test.c:465 - M2M_msgTCPTask{TCP_TASK}$
m2mb_socket_bsd_recv() has received 6 bytes

[DEBUG] 102.60  m2m_tcp_test.c:469 - M2M_msgTCPTask{TCP_TASK}$
Server TCP is closing the current connection ...
```

Figure 163

Data on a PuTTY terminal

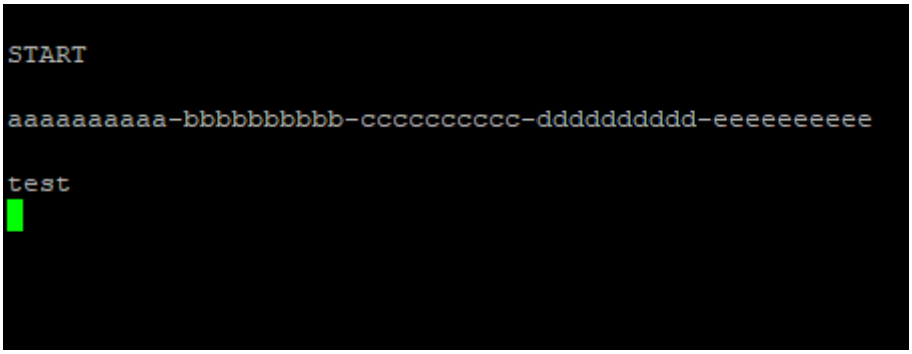
START
aaaaaaaaaa-bbbbbbbbbb-cccccccccc-dddddddddd-eeeeeeeeee
test

Figure 164

3.6.34 TLS SSL Client

Sample application showcasing TLS/SSL with client certificates usage with M2MB API. Debug prints on **USB0**

Features

- How to check module registration and enable PDP context
- How to open a SSL client socket
- How to communicate over SSL socket

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Create a task to manage the connection and start it

`ssl_test.c`

- Initialize Network structure and check registration
- Initialize PDP structure and start PDP context
- Create socket and link it to the PDP context id
- Connect to the server over TCP socket
- Initialize the TLS parameters (TLS1.2) andh auth mode (server+client auth in the example)
- Create SSL context
- Read certificates files and store them
- Create secure socket and connect to the server using SSL
- Send data and receive response
- Close secure socket
- Close socket
- Delete SSL context
- Disable PDP context

The application requires the certificates to be stored in `/test_ssl_certs/` folder. It can be created with `AT#M2MMKDIR=/test_ssl_certs`

```
Starting TLS-SSL demo app. This is v1.0.7 built on Mar 26 2020 16:27:00.
[DEBUG] 21.27  ssl_test.c:253 - msgHTTPSTask{TLS_TASK}$ INIT
[DEBUG] 21.27  ssl_test.c:267 - msgHTTPSTask{TLS_TASK}$ m2mb_os_ev_init success
[DEBUG] 21.28  ssl_test.c:271 - msgHTTPSTask{TLS_TASK}$ Init SSL session test app
[DEBUG] 21.28  ssl_test.c:286 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_config sslConfigHndl = 0x400330a8, sslRes= 0
[DEBUG] 21.29  ssl_test.c:295 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_config PASSED
[DEBUG] 21.30  ssl_test.c:307 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_ctxt PASSED
[DEBUG] 21.31  ssl_test.c:312 - msgHTTPSTask{TLS_TASK}$ loading CA CERT from file /test_ssl_certs/modulesCA.crt
[DEBUG] 21.32  ssl_test.c:316 - msgHTTPSTask{TLS_TASK}$ file size: 1740
[DEBUG] 21.32  ssl_test.c:329 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1740
Buffer successfully received from file. 1740 bytes were loaded.
Closing file.
[DEBUG] 21.34  ssl_test.c:458 - msgHTTPSTask{TLS_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 21.35  ssl_test.c:466 - msgHTTPSTask{TLS_TASK}$ Waiting for registration...
[DEBUG] 21.36  ssl_test.c:172 - NetCallback{pubTspt_0}$ Module is registered to cell 0xC4CF!
[DEBUG] 21.36  ssl_test.c:478 - msgHTTPSTask{TLS_TASK}$ Pdp context activation
[DEBUG] 21.37  ssl_test.c:482 - msgHTTPSTask{TLS_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 23.41  ssl_test.c:497 - msgHTTPSTask{TLS_TASK}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 24.24  ssl_test.c:198 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.24  ssl_test.c:201 - PdpCallback{pubTspt_0}$ IP address: 37.118.158.27
[DEBUG] 24.25  ssl_test.c:515 - msgHTTPSTask{TLS_TASK}$ Creating Socket...
[DEBUG] 24.26  ssl_test.c:526 - msgHTTPSTask{TLS_TASK}$ Socket created
[DEBUG] 24.26  ssl_test.c:536 - msgHTTPSTask{TLS_TASK}$ Socket ctx set to 3
[DEBUG] 24.61  ssl_test.c:549 - msgHTTPSTask{TLS_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 24.87  ssl_test.c:563 - msgHTTPSTask{TLS_TASK}$ Socket Connected!
[DEBUG] 26.14  ssl_test.c:588 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_connect ret 0
[DEBUG] 28.17  ssl_test.c:594 - msgHTTPSTask{TLS_TASK}$ Sending bytes..
[DEBUG] 28.17  ssl_test.c:597 - msgHTTPSTask{TLS_TASK}$ SSL write result = 44
[DEBUG] 32.18  ssl_test.c:609 - msgHTTPSTask{TLS_TASK}$ pending bytes: 1087
[DEBUG] 32.19  ssl_test.c:612 - msgHTTPSTask{TLS_TASK}$ trying to receive bytes..
[DEBUG] 32.19  ssl_test.c:618 - msgHTTPSTask{TLS_TASK}$ Server response: (269)<HTTP/1.1 200 OK
Date: Thu, 26 Mar 2020 15:29:43 GMT
Server: Apache/2.2.15 (CentOS)
Last-Modified: Mon, 22 Jan 2018 10:57:39 GMT
ETag: "1fffc-27f-5635b4c6f12b3"
Accept-Ranges: bytes
Content-Length: 639
Connection: close
Content-Type: text/html; charset=UTF-8

>
[DEBUG] 32.23  ssl_test.c:635 - msgHTTPSTask{TLS_TASK}$ application exit
[DEBUG] 32.23  ssl_test.c:653 - msgHTTPSTask{TLS_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 32.26  ssl_test.c:656 - msgHTTPSTask{TLS_TASK}$ Application complete.
[DEBUG] 32.89  ssl_test.c:207 - PdpCallback{pubTspt_0}$ Context deactivated!
```

Figure 165

3.6.35 UDP client

Sample application showcasing UDP echo demo with M2MB API. Debug prints on **USB0**

Features

- How to check module registration and activate PDP context
- How to open a UDP client socket
- How to communicate over the socket

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Print welcome message
- Create a task and start it

`m2m_udp_test.c` - Initialize Network structure and check registration - Initialize PDP structure and start PDP context - Create socket and link it to the PDP context id - Send data and receive response - Close socket - Disable PDP context

```
Starting UDP client demo app. This is v1.0.7 built on Apr  1 2020 14:57:13.
INIT
[DEBUG] 21.23  m2m_udp_test.c:223 - M2M_msgUDPTask{UDP_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
Waiting for registration...
[DEBUG] 21.25  m2m_udp_test.c:131 - NetCallback{pubTspt_0}$ Module is registered to cell 0xC4CF!
[DEBUG] 21.26  m2m_udp_test.c:241 - M2M_msgUDPTask{UDP_TASK}$ Pdp context initialization
[DEBUG] 21.26  m2m_udp_test.c:245 - M2M_msgUDPTask{UDP_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
Activate PDP with APN web.omnitel.it....
[DEBUG] 24.11  m2m_udp_test.c:157 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 24.11  m2m_udp_test.c:160 - PdpCallback{pubTspt_0}$ IP address: 109.113.222.12
[DEBUG] 24.12  m2m_udp_test.c:268 - M2M_msgUDPTask{UDP_TASK}$ Creating Socket...
[DEBUG] 24.13  m2m_udp_test.c:280 - M2M_msgUDPTask{UDP_TASK}$ Socket created
Socket ctx set to 3
[DEBUG] 24.41  m2m_udp_test.c:306 - M2M_msgUDPTask{UDP_TASK}$ Retrieved IP: 185.86.42.218
Socket ready.
Data successfully sent (16 bytes)
Socket recv...
[DEBUG] 26.47  m2m_udp_test.c:352 - M2M_msgUDPTask{UDP_TASK}$ m2mb_socket_bsd_set_sock_opt() M2MB_SOCKET_BSD_SO_RCVTIMEO - success
trying to receive 16 bytes..
Data received (16): <hello from m2mb!>
[DEBUG] 26.48  m2m_udp_test.c:377 - M2M_msgUDPTask{UDP_TASK}$ application exit
Socket Closed
[DEBUG] 26.49  m2m_udp_test.c:399 - M2M_msgUDPTask{UDP_TASK}$ m2mb_pdp_deactivate returned success
Application complete.
[DEBUG] 27.04  m2m_udp_test.c:166 - PdpCallback{pubTspt_0}$ Context successfully deactivated!
```

Figure 166

3.6.36 ZLIB example

Sample application showing how to compress/uncompress with ZLIB. Debug prints on **USB0**

Features

- How to compress a file
- How to uncompress a file

In order to execute the entire test, copy `test.gz` file into your module running the following AT command:

```
AT#M2MWRITE="/mod/test.gz",138
>>> here receive the prompt; then type or send the file, sized 138 bytes
```

Application workflow

`M2MB_main.c`

- Open USB/UART/UART_AUX
- Test the compression and decompression of a data string
- Test the decompression of a .gz file (test.gz), expected to be in `/mod` folder, into its content `test.txt`. The file must be uploaded by the user (see steps above).

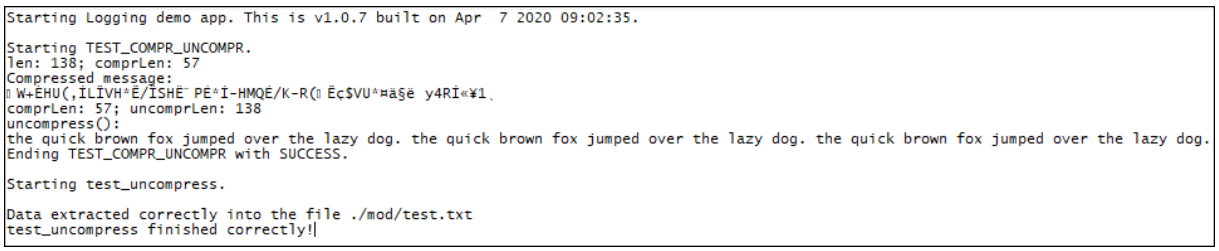

Figure 167

3.6.37 Little fs2

Sample application showing how use lfs2 porting with RAM disk and SPI data flash. Debug prints on **USB0**

Features

- How to create and manage Ram Disk
- How to manage file-system in Ram disk partition
- How to create and manage SPI Flash memory partition
- How to manage file-system in SPI Flash memory partition

Application workflow

`M2MB_main.c`

- Init logging system
- Call Ram Disk tests
- Call Flash memory tests

`ram_utils_usage.c`

- Initialize Ram Disk
- Format and Mount partition
- List files
- Files creation and write content
- List files
- Read files
- Unmount and Release resources

`spi_utils_usage.c` - Initialize SPI Flash chip - Initialize SPI Flash Disk - Format and Mount partition - List files - Files creation and write content - List files - Read files - Delete files - Directories creation and deletion - Unmount and Release resources

Notes: For SPI Flash a JSC memory is used with chip select pin connected to module GPIO2 pin. For better performances, a 33kOhm pull-down resistor on SPI clock is suggested. Please refer to SPI_echo sample app for SPI connection details.

```
Starting lfs2 demo app. This is v1.0.14-C1 built on Oct 22 2020 09:43:08.

>>>>>>> Starting RAMDiskDemo ...
[DEBUG] 18.28  azx_lfs_uti:125 - azx_ram_initialize{M2M_DamsStart}$ Ram Memory allocated correctly from 0x40042228 to 0x40046228!!

Mounting partition...

Formatting...

Mounting...


Mounted partition...

<><><><>fileListUtils

List:
., 0, 2
.., 0, 2

file_name: file000.txt
size: 10
buffer: content000
mode: 0
RAM TYPE size: 10000


File created and closed: file000.txt


<><><><>fileListUtils
____INSIDE --->file000.txt, 10, 1

List:
., 0, 2
.., 0, 2
file000.txt, 10, 1

------>File reading

File: file000.txt, Size: 10, Buffer: content000

Nand released
Partition unmounted
[DEBUG] 20.31  azx_lfs_uti:165 - azx_ram_releaseResources{M2M_DamsStart}$ Ram Memory released correctly!!
```

```
>>>>>>> Starting FlashDiskDemo ...
Starting initialization...

table id[0] = 191
table id[1] = 1
table id[2] = 0

nandLFSCallback Callback event <1>
NAND Callback event: NAND_JSC_INITIALIZED <1>
nandLFSCallback Callback event <1>
NAND Callback event: NAND_JSC_INITIALIZED <1>

Mounting partition...

Formatting...
spiErase: address = 0, len = 131072

spiErase: address = 131072, len = 131072


Mounting...


Mounted partition...


<><><><>fileListUtils

List:
., 0, 2
.., 0, 2


Formatting...
spiErase: address = 0, len = 131072

spiErase: address = 131072, len = 131072


Mounting...


Mounted partition...


<><><><>fileListUtils
List:
., 0, 2
.., 0, 2

file_name: file000.txt
size: 10
buffer: content000
mode: 0


File created and closed: file000.txt
```

```
<><><><fileListUtils

List:
., 0, 2
.., 0, 2
file000.txt, 10, 1
file001.txt, 10, 1
file002.txt, 10, 1
file003.txt, 10, 1
file004.txt, 10, 1

------>File reading

File: file000.txt, Size: 10, Buffer: content000


File: file004.txt, Size: 10, Buffer: content004


File: file002.txt, Size: 10, Buffer: content002
------>File removing
file001.txt<<<<<<<


File removed: file001.txt
file000.txt<<<<<<<


File removed: file000.txt
file004.txt<<<<<<<


File removed: file004.txt


<><><><fileListUtils

List:
., 0, 2
.., 0, 2
file002.txt, 10, 1
file003.txt, 10, 1

spiErase: address = 59637760, len = 131072

[DEBUG] 58.61  azx_lfs_uti:648 - azx_lfsDirCreationByContext{M2M_DamsStart}$ Directory created: dir000!!
[DEBUG] 59.78  azx_lfs_uti:631 - azx_lfsDirCreationByContext{M2M_DamsStart}$ Directory already exists: dir000!!
spiErase: address = 59899904, len = 131072

[DEBUG] 61.70  azx_lfs_uti:648 - azx_lfsDirCreationByContext{M2M_DamsStart}$ Directory created: dir001!!
spiErase: address = 60162048, len = 131072

[DEBUG] 63.67  azx_lfs_uti:648 - azx_lfsDirCreationByContext{M2M_DamsStart}$ Directory created: dir002!!
```

```
<><><><fileListUtils

List:
., 0, 2
.., 0, 2
dir000, 0, 2
dir001, 0, 2
dir002, 0, 2
file002.txt, 10, 1
file003.txt, 10, 1



<><><><fileListUtils

List:
., 0, 2
.., 0, 2
dir001, 0, 2
dir002, 0, 2
file002.txt, 10, 1
file003.txt, 10, 1

Nand released
Partition unmounted

Unmounted process ended...

testAllInOneFunction ended...
```

4 Installing beta version libraries Plug-in

4.1 New beta plug-in installation

To install a new plug-in for a beta firmware into the IDE, first receive plug-in ".zip" packet, then unzip the file in a local folder and open the SDK IDE.

PLEASE DO NOT USE BETA PLUGINS FOR PRODUCTION DEPLOYMENTS, SOFTWARE IS PROVIDED AS IS AND CUSTOMER ACKNOWLEDGES THAT IT IS POSSIBLE THE DEVICE MAY MISFUNCTION. PLEASE REFER TO Contact Information, Support section

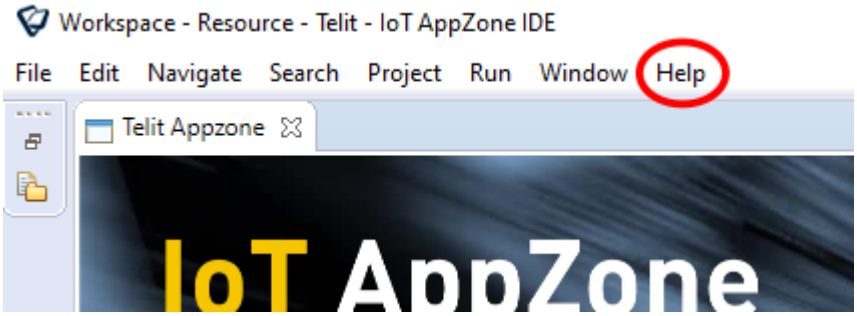

Figure 168

Click on "Help" tag and choose "Install New Software...". This window will appear:

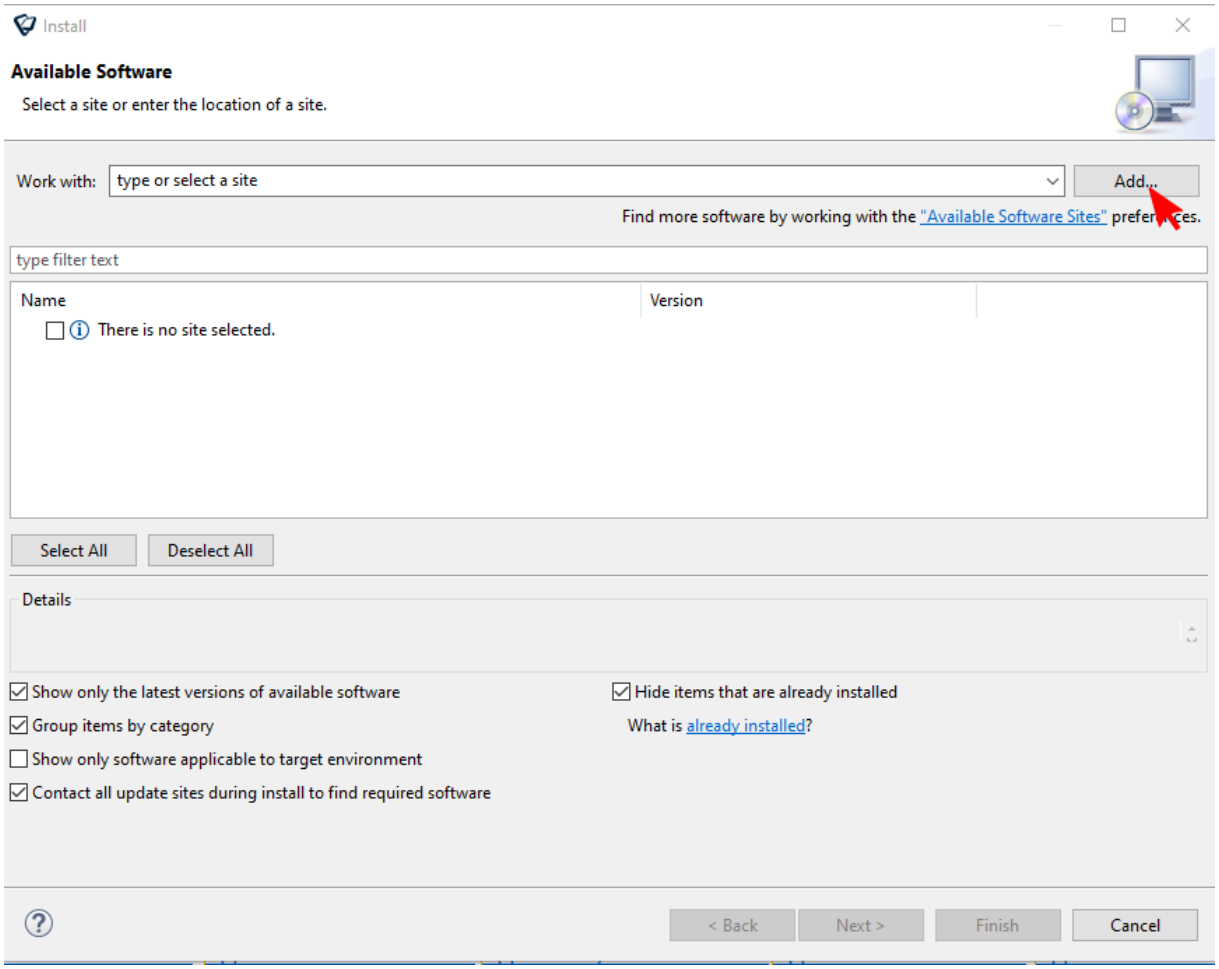

Figure 169

Click on “Add…” button and then in the following window click on “Local…” to select the unzipped folder with the plug-in content.

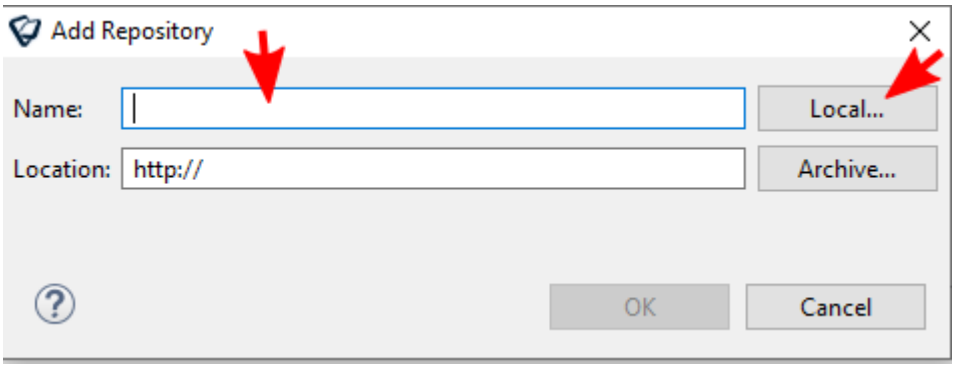

Figure 170

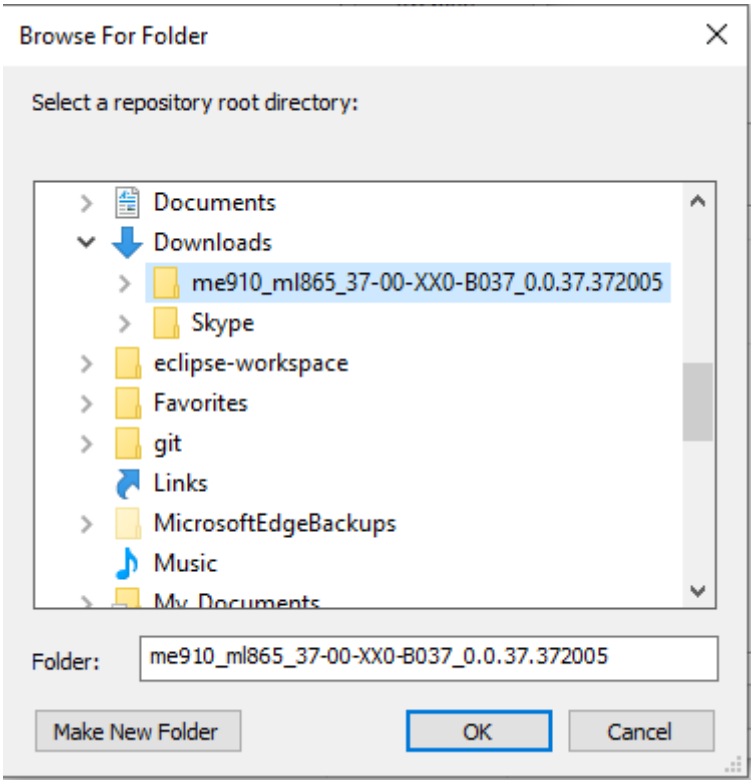

Figure 171

Once selected the plug-in folder, the “Location:” form will present the selected path. Now in “Name:” write a name for the new libraries (for example 37.00.xx0_B037) and click on “OK” button.

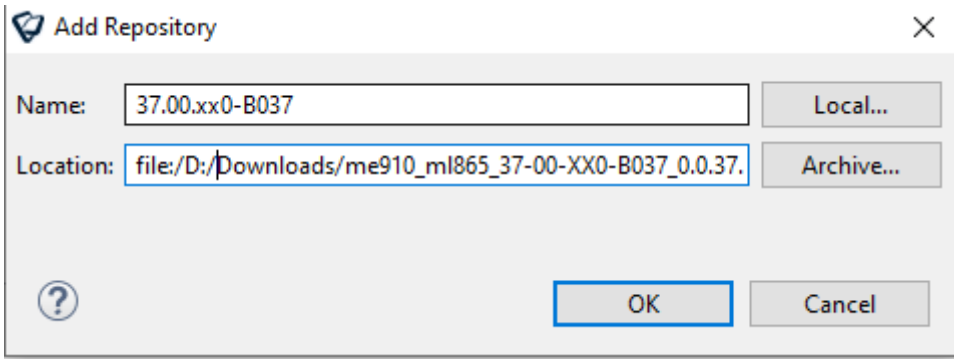

Figure 172

The new packet is now ready to be installed: select it and click on “Next >” button until “Review Licenses” window will appear.

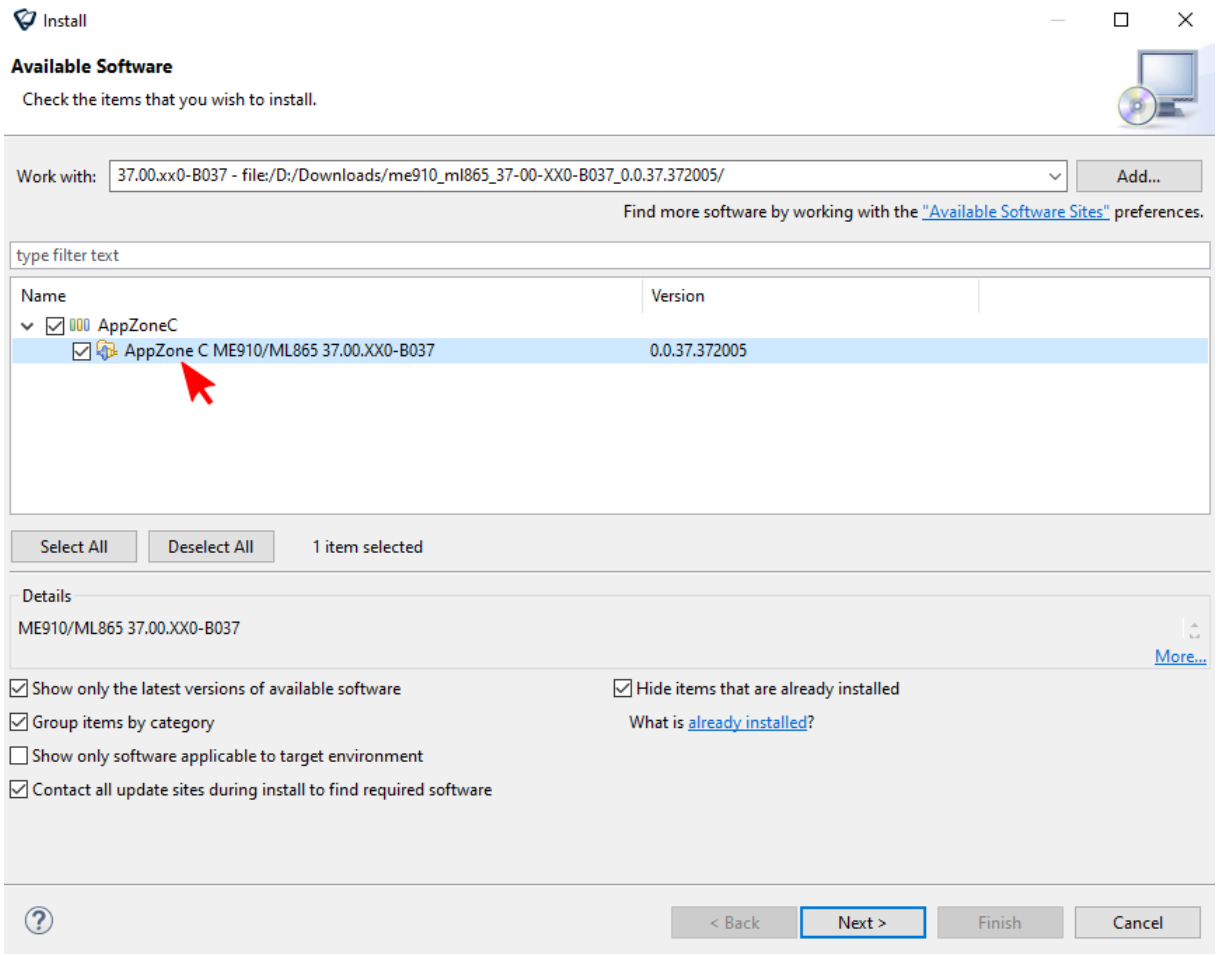

Figure 173

Accept the licenses when required and click on “Finish” button to complete the installation.

4.2 Change existing project libraries

To align an old project to the new libraries, right click on the project and choose “Properties”.

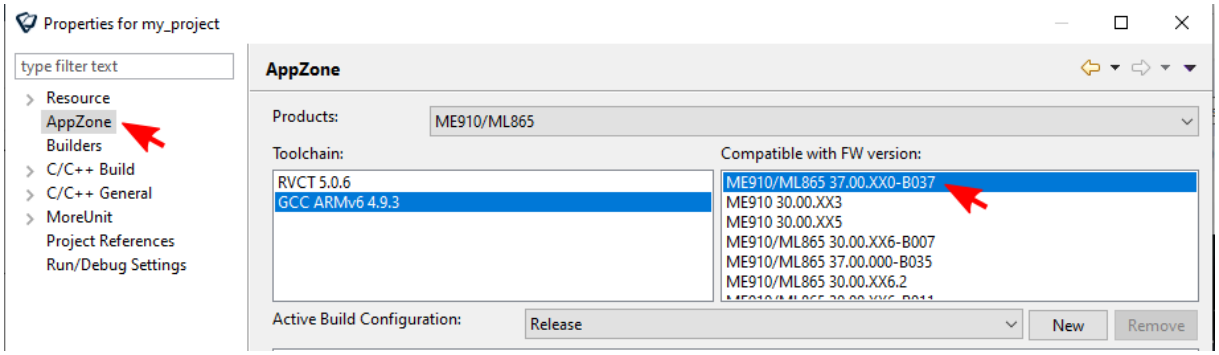

Figure 174

Now select “AppZone” on the left side of the window, and on the right choose the packet with the same name as the firmware version to be used. Then click on “OK” (or “Apply”) button.

4.3 Create a project with the new plug-in

To use the new libraries, create a new project: “File”-> “New” -> “Telit Project”

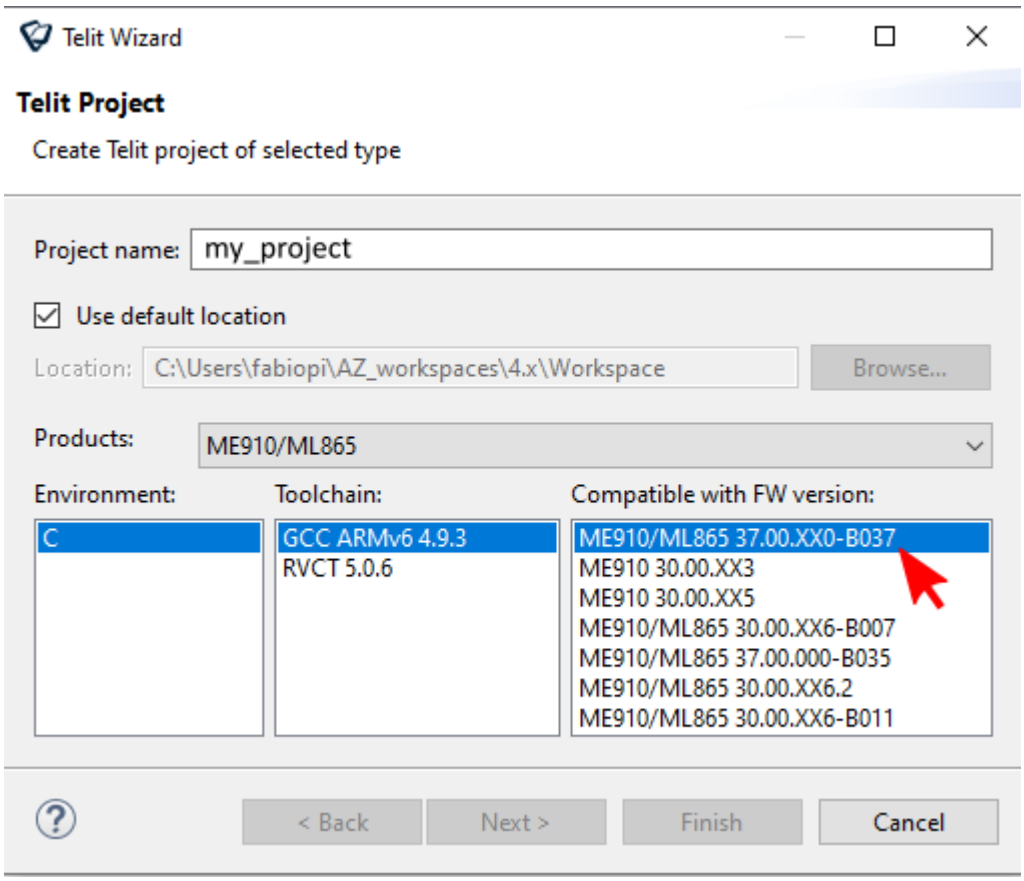

Figure 175

Select the new firmware version (37.00.xx0-B037) and create an empty project.

SUPPORT INQUIRIES

Link to **www.telit.com** and contact our technical support team for any questions related to technical issues.

www.telit.com

Telit Communications S.p.A.
Via Stazione di Prosecco, 5/B
I-34010 Sgonico (Trieste), Italy

Telit IoT Platforms LLC
5300 Broken Sound Blvd, Suite 150
Boca Raton, FL 33487, USA

Telit Wireless Solutions Inc.
3131 RDU Center Drive, Suite 135
Morrisville, NC 27560, USA

Telit Wireless Solutions Co., Ltd.
8th Fl., Shinyoung Securities Bld.
6, Gukjegeumyung-ro8-gil, Yeongdeungpo-gu
Seoul, 150-884, Korea

Telit Wireless Solutions Ltd.
10 Habarzel St.
Tel Aviv 69710, Israel

Telit Wireless Solutions
Technologia e Servicos Ltda
Avenida Paulista, 1776, Room 10.C
01310-921 São Paulo, Brazil