



AppZone m2mb Sample Apps

80000NT11840A Rev. 1 - 2021-01-29

TELIT
TECHNICAL
DOCUMENTATION

1 AppZone m2mb Sample Apps

Package Version: **1.1.3-CxL**

Minimum Firmware Version: **25.20.XX8**

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 **25.20.XX8** FW version flashed (AT#SWPKGV will give you the FW version)
2. Copy m2mapz.bin to /data/azc/mod/

```
AT#M2MWRITE="/data/azc/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="/data/azc/mod/m2mapz.bin"`
 - Delete everything, reflash and retry
`AT#M2MDEL="/data/azc/mod/m2mapz.bin"`
`AT#M2MDEL="/data/azc/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 to build and try the sample app on your device.

Contents

1 AppZone m2mb Sample Apps	2
1.1 Features	2
2 Quick start	2
2.1 Deployment Instructions	2
2.2 References	2
2.3 Known Issues	2
2.4 Contact Information, Support	3
2.5 Troubleshooting	3
2.6 Making source code changes	4
2.6.1 Folder structure	4
2.7 Import a Sample App into an IDE project	4
3 Applications	9
3.1 AUX UART	9
3.1.1 ATI (AT Instance)	9
3.1.2 App update OTA via FTP	11
3.1.3 CJSON example:	13
3.1.4 Easy AT example	15
3.1.5 Events	16
3.1.6 Events - Barrier (multi events)	17
3.1.7 FTP	18
3.1.8 File System example	20
3.1.9 GNSS example	21
3.1.10GPIO interrupt example	22
3.1.11HTTP Client	23
3.1.12HW Timer (Hardware Timer)	25
3.1.13Hello World	26
3.1.14Logging Demo	27
3.1.15MD5 example	28
3.1.16MultiTask	29
3.1.17SMS PDU	31
3.1.18SW Timer (Software Timer)	32
3.1.19TCP IP	33
3.1.20TCP Socket status	35
3.1.21TCP Server	37
3.1.22TLS SSL Client	40
3.1.23UDP client	43
3.1.24ZLIB example	44
3.2 MISC	45

3.2.1	GPIO toggle example	45
3.3	MAIN UART	46
3.3.1	ATI (AT Instance)	46
3.3.2	AT Tunnel	48
3.3.3	App update OTA via FTP	50
3.3.4	CJSON example:	52
3.3.5	Easy AT example	54
3.3.6	Events	55
3.3.7	Events - Barrier (multi events)	56
3.3.8	FTP	57
3.3.9	File System example	59
3.3.10	GNSS example	60
3.3.11	GPIO interrupt example	61
3.3.12	HTTP Client	62
3.3.13	HW Timer (Hardware Timer)	64
3.3.14	Hello World	65
3.3.15	Logging Demo	66
3.3.16	MD5 example	67
3.3.17	MultiTask	68
3.3.18	SMS PDU	70
3.3.19	SPI Echo	71
3.3.20	SW Timer (Software Timer)	72
3.3.21	TCP IP	73
3.3.22	TCP Socket status	75
3.3.23	TCP Server	77
3.3.24	TLS SSL Client	80
3.3.25	Uart To Server	83
3.3.26	UDP client	84
3.3.27	ZLIB example	85
3.4	BASIC	86
3.4.1	Basic Hello World (Aux UART)	86
3.4.2	Basic Hello World (Main UART)	87
3.4.3	Basic Hello World (USB0)	88
3.5	USB0	89
3.5.1	ATI (AT Instance)	89
3.5.2	App update OTA via FTP	91
3.5.3	CJSON example:	93
3.5.4	Easy AT example	95
3.5.5	Events	96
3.5.6	Events - Barrier (multi events)	97
3.5.7	FTP	98

3.5.8 File System example	100
3.5.9 GNSS example	101
3.5.10GPIO interrupt example	102
3.5.11HTTP Client	103
3.5.12HW Timer (Hardware Timer)	105
3.5.13Hello World	106
3.5.14Logging Demo	107
3.5.15MD5 example	108
3.5.16MultiTask	109
3.5.17SMS PDU	111
3.5.18SPI Echo	112
3.5.19SW Timer (Software Timer)	113
3.5.20TCP IP	114
3.5.21TCP Socket status	116
3.5.22TCP Server	118
3.5.23TLS SSL Client	121
3.5.24UDP client	124
3.5.25ZLIB example	125

4 Installing beta version libraries Plug-in	126
4.1 New beta plug-in installation	126
4.2 Change existing project libraries	129
4.3 Create a project with the new plug-in	130

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
MOB.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
MOB.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 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 `/data/azc/mod` folder, for example with

```
AT#M2MWRITE="/data/azc/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 `/data/azc/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: <<_ - - - ->>
Set FTP PASS to: <<_ - - - ->>
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
€yStarting. 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 4

3.1.3 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 property.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 5

3.1.4 Easy AT example

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

Features

- Shows how to register custom commands

3.1.5 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 6

3.1.6 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 7

3.1.7 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 | AAAAAAAA | | AAAAAAAA | | AAAAAAAA | | AAAAAAAA | | AAAAAAAA | | *
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 8

3.1.8 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 9

3.1.9 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 10

3.1.10 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 11

3.1.11 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
- Apply 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:287 - 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 12

3.1.12 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 13

3.1.13 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 14

3.1.14 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 15

3.1.15 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 16

3.1.16 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 17

3.1.17 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 18

3.1.18 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 19

3.1.19 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 20

3.1.20 TCP Socket status

Sample application showcasing how to check a TCP 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 21

3.1.21 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

```

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 22

Data on a PuTTY terminal

```
START  
aaaaaaaaaa-bbbbbbbbbb-ccccccccc-ddddddddd-eeeeeeeeee  
test  
█
```

Figure 23

3.1.22 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 /data/azc/mod/ssl_certs/ folder. It can be created with

```
AT#M2MMKDIR=/data/azc/mod/ssl_certs
```

Certificates can then be loaded with

```
AT#M2MWRITE="/data/azc/mod/ssl_certs/data/azc/modulesCA.crt",1740
```


and providing the file content in RAW mode (for example using the “Transfer Data” button in Telit AT Controller)

For client certificates (if required), the commands will be

AT#M2MWRITE="/data/azc/mod/ssl_certs/data/azc/modulesClient.crt",1651

AT#M2MWRITE="/data/azc/mod/ssl_certs/data/azc/modulesClient_pkcs1.key",1679

PLEASE NOTE: always verify the file sizes to be used in the commands above as they might change

```
Starting TLS-SSL demo app. This is v1.1.2 built on Mar  3 2021 10:15:00.
[DEBUG] 10.85 ssl_test:252 - msgHTTPSTask{TLS_TASK}$ INIT
[DEBUG] 10.85 ssl_test:266 - msgHTTPSTask{TLS_TASK}$ m2mb_os_ev_init success
[DEBUG] 10.85 ssl_test:270 - msgHTTPSTask{TLS_TASK}$ Init SSL session test app
[DEBUG] 10.85 ssl_test:285 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_config sslConfigHnd1 = 0x40037958, sslRes= 0
[DEBUG] 10.85 ssl_test:294 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_config PASSED
[DEBUG] 10.85 ssl_test:306 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_ctxt PASSED
[DEBUG] 10.85 ssl_test:311 - msgHTTPSTask{TLS_TASK}$ loading CA CERT from file /mod/ssl_certs/modulesCA.crt
[DEBUG] 10.85 ssl_test:315 - msgHTTPSTask{TLS_TASK}$ file size: 1740
[DEBUG] 10.85 ssl_test:328 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1740
Buffer successfully received from file. 1740 bytes were loaded.
Closing file.
[DEBUG] 10.85 ssl_test:361 - msgHTTPSTask{TLS_TASK}$ loading client CERT from file /mod/ssl_certs/modulesClient.crt
[DEBUG] 10.85 ssl_test:365 - msgHTTPSTask{TLS_TASK}$ file size: 1651
[DEBUG] 10.85 ssl_test:378 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1651
Buffer successfully received from file. 1651 bytes were loaded.
Closing file.
[DEBUG] 10.85 ssl_test:401 - msgHTTPSTask{TLS_TASK}$ loading client KEY from file /mod/ssl_certs/modulesClient_pkcs1.key
[DEBUG] 10.85 ssl_test:405 - msgHTTPSTask{TLS_TASK}$ file size: 1679
[DEBUG] 10.85 ssl_test:418 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1679
Buffer successfully received from file. 1679 bytes were loaded.
Closing file.
[DEBUG] 10.85 ssl_test:448 - msgHTTPSTask{TLS_TASK}$ certificates successfully stored!
[DEBUG] 10.85 ssl_test:457 - msgHTTPSTask{TLS_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 10.85 ssl_test:465 - msgHTTPSTask{TLS_TASK}$ Waiting for registration...
[DEBUG] 10.86 ssl_test:171 - NetCallback{pubTspt_0}$ Module is registered to cell 0x468E!
[DEBUG] 10.86 ssl_test:477 - msgHTTPSTask{TLS_TASK}$ Pdp context activation
[DEBUG] 10.86 ssl_test:481 - msgHTTPSTask{TLS_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 12.87 ssl_test:496 - msgHTTPSTask{TLS_TASK}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 13.71 ssl_test:197 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 13.71 ssl_test:200 - PdpCallback{pubTspt_0}$ IP address: 2.41.76.63
[DEBUG] 13.71 ssl_test:514 - msgHTTPSTask{TLS_TASK}$ Creating Socket...
[DEBUG] 13.71 ssl_test:525 - msgHTTPSTask{TLS_TASK}$ Socket created
[DEBUG] 13.71 ssl_test:535 - msgHTTPSTask{TLS_TASK}$ Socket ctx set to 3
[DEBUG] 13.92 ssl_test:548 - msgHTTPSTask{TLS_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 14.05 ssl_test:562 - msgHTTPSTask{TLS_TASK}$ Socket Connected!
[DEBUG] 15.97 ssl_test:587 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_connect ret 0
[DEBUG] 17.99 ssl_test:593 - msgHTTPSTask{TLS_TASK}$ Sending bytes..
```

```
[DEBUG] 17.99 ssl_test:593 - msgHTTPSTask{TLS_TASK}$ Sending bytes..
[DEBUG] 17.99 ssl_test:596 - msgHTTPSTask{TLS_TASK}$ SSL write result = 44
[DEBUG] 22.03 ssl_test:608 - msgHTTPSTask{TLS_TASK}$ pending bytes: 1087
[DEBUG] 22.03 ssl_test:612 - msgHTTPSTask{TLS_TASK}$ trying to receive 1087 bytes..
[DEBUG] 22.03 ssl_test:618 - msgHTTPSTask{TLS_TASK}$ Server response: (269)<HTTP/1.1 200 OK
Date: Wed, 03 Mar 2021 09:18:22 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] 22.03 ssl_test:634 - msgHTTPSTask{TLS_TASK}$ pending bytes: 762
[DEBUG] 22.03 ssl_test:638 - msgHTTPSTask{TLS_TASK}$ trying to receive remaining 762 bytes..
[DEBUG] 22.03 ssl_test:644 - msgHTTPSTask{TLS_TASK}$ Server response: (639)<<html>
<head>
  <title>module.telit.com</title>
  <meta content="text/html; charset=utf-8" />
</head>
<body>
  <table border=0 align=center>
    <tr>
      <td height="100" align=center><h2>modules.telit.com - Test HTML page</h2></td>
    </tr>
    <tr>
      <td align=center><img src=Telit.jpg alt="Telit logo" height="126" width="410"></img></td>
    </tr>
    <tr>
      <td height="200" align=center> This is a simple HTML page, </br>
      made with simple HTML code,</br>
      just for test!
    </td>
    </tr>
    <tr>
      <td height="100" align=center><font size="3">Telit &copy; 2015 - 2017 All rights reserved</font></td>
    </tr>
  </table>
</body>
</html>
>
[DEBUG] 22.03 ssl_test:662 - msgHTTPSTask{TLS_TASK}$ application exit
[DEBUG] 22.03 ssl_test:680 - msgHTTPSTask{TLS_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 22.03 ssl_test:683 - msgHTTPSTask{TLS_TASK}$ Application complete.
[DEBUG] 22.77 ssl_test:206 - PdpCallback{pubTspt_0}$ Context deactivated!
```

3.1.23 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 0xC4CF1
[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 rcv...
[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 24

3.1.24 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="/data/azc/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 /data/azc/mod folder, into its content test.txt. The file must be uploaded by the user (see steps above).

```
Starting Logging demo app. This is v1.0.7 built on Apr 7 2020 09:02:35.
Starting TEST_COMPR_UNCOMPR.
len: 138; comprLen: 57
Compressed message:
W+EHU(,ILIVH^E/ISHE`PE*I-HMQE/K-R(ÛÊç$VU^#ašê y4RI«¥1.
comprLen: 57; uncomprLen: 138
uncompress():
the quick brown fox jumped over the lazy dog. the quick brown fox jumped over the lazy dog. the quick brown fox jumped over the lazy dog.
Ending TEST_COMPR_UNCOMPR with SUCCESS.
Starting test_uncompress.
Data extracted correctly into the file ./mod/test.txt
test_uncompress finished correctly!
```

Figure 25

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
MOB.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 26

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
MOB.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 27

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:

```
Starting AT tunnel demo app. Waiting for AT commands...
AT+CGMM
ME910C1-P2

OK
AT+CGREG?
+CGREG: 0,1

OK
```

Figure 28

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 29

3.3.3 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 `/data/azc/mod` folder, for example with

```
AT#M2MWRITE="/data/azc/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 `/data/azc/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: <<ftp_telit>>
Set FTP PASS to: <<ftp_telit>>
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
€yStarting. 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 30

3.3.4 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 property.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 31

3.3.5 Easy AT example

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

Features

- Shows how to register custom commands

3.3.6 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 32

3.3.7 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 33

3.3.8 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 | AAAAAAAA | | AAAAAAAA | | AAAAAAAA | | AAAAAAAA | | AAAAAAAA | | *
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 34

3.3.9 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 35

3.3.10 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 36

3.3.11 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 37

3.3.12 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
- Apply 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:287 - 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 38

3.3.13 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 39

3.3.14 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 40

3.3.15 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 41

3.3.16 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 42

3.3.17 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 43

3.3.18 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 44

3.3.19 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 45

3.3.20 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 46

3.3.21 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 47

3.3.22 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 48

3.3.23 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

```

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 49

Data on a PuTTY terminal

```
START  
aaaaaaaaa-bbbbbbbbb-bbbbbbbbbb-cccccccccc-ddddddddd-eeeeeeeeee  
test  
█
```

Figure 50

3.3.24 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 /data/azc/mod/ssl_certs/ folder. It can be created with

```
AT#M2MMKDIR=/data/azc/mod/ssl_certs
```

Certificates can then be loaded with

```
AT#M2MWRITE="/data/azc/mod/ssl_certs/data/azc/modulesCA.crt",1740
```


and providing the file content in RAW mode (for example using the “Transfer Data” button in Telit AT Controller)

For client certificates (if required), the commands will be

AT#M2MWRITE="/data/azc/mod/ssl_certs/data/azc/modulesClient.crt",1651

AT#M2MWRITE="/data/azc/mod/ssl_certs/data/azc/modulesClient_pkcs1.key",1679

PLEASE NOTE: always verify the file sizes to be used in the commands above as they might change

```
Starting TLS-SSL demo app. This is v1.1.2 built on Mar  3 2021 10:15:00.
[DEBUG] 10.85 ssl_test:252 - msgHTTPSTask{TLS_TASK}$ INIT
[DEBUG] 10.85 ssl_test:266 - msgHTTPSTask{TLS_TASK}$ m2mb_os_ev_init success
[DEBUG] 10.85 ssl_test:270 - msgHTTPSTask{TLS_TASK}$ Init SSL session test app
[DEBUG] 10.85 ssl_test:285 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_config sslConfigHnd1 = 0x40037958, sslRes= 0
[DEBUG] 10.85 ssl_test:294 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_config PASSED
[DEBUG] 10.85 ssl_test:306 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_ctxt PASSED
[DEBUG] 10.85 ssl_test:311 - msgHTTPSTask{TLS_TASK}$ loading CA CERT from file /mod/ssl_certs/modulesCA.crt
[DEBUG] 10.85 ssl_test:315 - msgHTTPSTask{TLS_TASK}$ file size: 1740
[DEBUG] 10.85 ssl_test:328 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1740
Buffer successfully received from file. 1740 bytes were loaded.
Closing file.
[DEBUG] 10.85 ssl_test:361 - msgHTTPSTask{TLS_TASK}$ loading client CERT from file /mod/ssl_certs/modulesClient.crt
[DEBUG] 10.85 ssl_test:365 - msgHTTPSTask{TLS_TASK}$ file size: 1651
[DEBUG] 10.85 ssl_test:378 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1651
Buffer successfully received from file. 1651 bytes were loaded.
Closing file.
[DEBUG] 10.85 ssl_test:401 - msgHTTPSTask{TLS_TASK}$ loading client KEY from file /mod/ssl_certs/modulesClient_pkcs1.key
[DEBUG] 10.85 ssl_test:405 - msgHTTPSTask{TLS_TASK}$ file size: 1679
[DEBUG] 10.85 ssl_test:418 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1679
Buffer successfully received from file. 1679 bytes were loaded.
Closing file.
[DEBUG] 10.85 ssl_test:448 - msgHTTPSTask{TLS_TASK}$ certificates successfully stored!
[DEBUG] 10.85 ssl_test:457 - msgHTTPSTask{TLS_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 10.85 ssl_test:465 - msgHTTPSTask{TLS_TASK}$ Waiting for registration...
[DEBUG] 10.86 ssl_test:171 - NetCallback{pubTspt_0}$ Module is registered to cell 0x468E!
[DEBUG] 10.86 ssl_test:477 - msgHTTPSTask{TLS_TASK}$ Pdp context activation
[DEBUG] 10.86 ssl_test:481 - msgHTTPSTask{TLS_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 12.87 ssl_test:496 - msgHTTPSTask{TLS_TASK}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 13.71 ssl_test:197 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 13.71 ssl_test:200 - PdpCallback{pubTspt_0}$ IP address: 2.41.76.63
[DEBUG] 13.71 ssl_test:514 - msgHTTPSTask{TLS_TASK}$ Creating Socket...
[DEBUG] 13.71 ssl_test:525 - msgHTTPSTask{TLS_TASK}$ Socket created
[DEBUG] 13.71 ssl_test:535 - msgHTTPSTask{TLS_TASK}$ Socket ctx set to 3
[DEBUG] 13.92 ssl_test:548 - msgHTTPSTask{TLS_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 14.05 ssl_test:562 - msgHTTPSTask{TLS_TASK}$ Socket Connected!
[DEBUG] 15.97 ssl_test:587 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_connect ret 0
[DEBUG] 17.99 ssl_test:593 - msgHTTPSTask{TLS_TASK}$ Sending bytes..
```

```
[DEBUG] 17.99 ssl_test:593 - msgHTTPSTask{TLS_TASK}$ Sending bytes..
[DEBUG] 17.99 ssl_test:596 - msgHTTPSTask{TLS_TASK}$ SSL write result = 44
[DEBUG] 22.03 ssl_test:608 - msgHTTPSTask{TLS_TASK}$ pending bytes: 1087
[DEBUG] 22.03 ssl_test:612 - msgHTTPSTask{TLS_TASK}$ trying to receive 1087 bytes..
[DEBUG] 22.03 ssl_test:618 - msgHTTPSTask{TLS_TASK}$ Server response: (269)<HTTP/1.1 200 OK
Date: Wed, 03 Mar 2021 09:18:22 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] 22.03 ssl_test:634 - msgHTTPSTask{TLS_TASK}$ pending bytes: 762
[DEBUG] 22.03 ssl_test:638 - msgHTTPSTask{TLS_TASK}$ trying to receive remaining 762 bytes..
[DEBUG] 22.03 ssl_test:644 - msgHTTPSTask{TLS_TASK}$ Server response: (639)<<html>
<head>
  <title>module.telit.com</title>
  <meta content="text/html; charset=utf-8" />
</head>
<body>
  <table border=0 align=center>
    <tr>
      <td height="100" align=center><h2>modules.telit.com - Test HTML page</h2></td>
    </tr>
    <tr>
      <td align=center><img src=Telit.jpg alt="Telit logo" height="126" width="410"></img></td>
    </tr>
    <tr>
      <td height="200" align=center> This is a simple HTML page, </br>
      made with simple HTML code,</br>
      just for test!
    </td>
    </tr>
    <tr>
      <td height="100" align=center><font size="3">Telit &copy; 2015 - 2017 All rights reserved</font></td>
    </tr>
  </table>
</body>
</html>
>
[DEBUG] 22.03 ssl_test:662 - msgHTTPSTask{TLS_TASK}$ application exit
[DEBUG] 22.03 ssl_test:680 - msgHTTPSTask{TLS_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 22.03 ssl_test:683 - msgHTTPSTask{TLS_TASK}$ Application complete.
[DEBUG] 22.77 ssl_test:206 - PdpCallback{pubTspt_0}$ Context deactivated!
```

3.3.25 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:

```
Ready to receive data and send to socket.  
<<<test message  
<<<test 2
```

Figure 51

Debug log on AUX UART:

```
Starting. This is build: Jul 17 2019 16:39:24. MASK: 000F  
Waiting for registration...  
Activate PDP with APN internet.wind.biz....  
Context activated!  
Socket created  
Server IP address: 185.86.42.218  
Socket Connected and ready to receive data!  
Uart opened, setting callback for data..  
Waiting for data from uart.  
UART IN: <test message>. Sending to socket...  
Data sent to socket!  
Response from server (12 bytes): <test message>  
UART IN: <test 2>. Sending to socket...  
Data sent to socket!  
Response from server (6 bytes): <test 2>
```

Figure 52

3.3.26 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 0xC4CF1
[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 rcv...
[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 53

3.3.27 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="/data/azc/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 /data/azc/mod folder, into its content test.txt. The file must be uploaded by the user (see steps above).

```
Starting Logging demo app. This is v1.0.7 built on Apr 7 2020 09:02:35.
Starting TEST_COMPR_UNCOMPR.
len: 138; comprLen: 57
Compressed message:
W+EHU(,ILIVH^E/ISHE`PE*I-HMQE/K-R(Ë Êç$VU*#ašê y4RI«¥1.
comprLen: 57; uncomprLen: 138
uncompress():
the quick brown fox jumped over the lazy dog. the quick brown fox jumped over the lazy dog. the quick brown fox jumped over the lazy dog.
Ending TEST_COMPR_UNCOMPR with SUCCESS.
Starting test_uncompress.
Data extracted correctly into the file ./mod/test.txt
test_uncompress finished correctly!
```

Figure 54

3.4 BASIC

Basic applications showing simple operations with minimum code overhead

3.4.1 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**

```
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 55

3.4.2 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 56

3.4.3 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 57

3.5 USB0

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

3.5.1 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
MOB.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 58

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
MOB.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 59

3.5.2 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 `/data/azc/mod` folder, for example with

```
AT#M2MWRITE="/data/azc/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 `/data/azc/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: <<ftp_telit>>
Set FTP PASS to: <<ftp_telit>>
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
€yStarting. 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 60

3.5.3 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 property.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 61

3.5.4 Easy AT example

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

Features

- Shows how to register custom commands

3.5.5 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 62

3.5.6 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 63

3.5.7 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 | AAAAAAAA | | AAAAAAAA | | AAAAAAAA | | AAAAAAAA | | AAAAAAAA | *
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 64

3.5.8 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 65

3.5.9 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 66

3.5.10 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 67

3.5.11 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 callback 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
- Apply 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:287 - 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 68

3.5.12 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 69

3.5.13 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 70

3.5.14 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 71

3.5.15 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

```
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 72

3.5.16 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 73

3.5.17 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

```
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 74

3.5.18 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 75

3.5.19 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 76

3.5.20 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 77

3.5.21 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 78

3.5.22 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 79

Data on a PuTTY terminal

```
START  
aaaaaaaaaa-bbbbbbbbbb-ccccccccc-ddddddddd-eeeeeeeeee  
test  
█
```

Figure 80

3.5.23 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 /data/azc/mod/ssl_certs/ folder. It can be created with

```
AT#M2MMKDIR=/data/azc/mod/ssl_certs
```

Certificates can then be loaded with

```
AT#M2MWRITE="/data/azc/mod/ssl_certs/data/azc/modulesCA.crt",1740
```

and providing the file content in RAW mode (for example using the “Transfer Data” button in Telit AT Controller)

For client certificates (if required), the commands will be

AT#M2MWRITE="/data/azc/mod/ssl_certs/data/azc/modulesClient.crt",1651

AT#M2MWRITE="/data/azc/mod/ssl_certs/data/azc/modulesClient_pkcs1.key",1679

PLEASE NOTE: always verify the file sizes to be used in the commands above as they might change

```
Starting TLS-SSL demo app. This is v1.1.2 built on Mar  3 2021 10:15:00.
[DEBUG] 10.85 ssl_test:252 - msgHTTPSTask{TLS_TASK}$ INIT
[DEBUG] 10.85 ssl_test:266 - msgHTTPSTask{TLS_TASK}$ m2mb_os_ev_init success
[DEBUG] 10.85 ssl_test:270 - msgHTTPSTask{TLS_TASK}$ Init SSL session test app
[DEBUG] 10.85 ssl_test:285 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_config sslConfigHnd1 = 0x40037958, sslRes= 0
[DEBUG] 10.85 ssl_test:294 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_config PASSED
[DEBUG] 10.85 ssl_test:306 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_create_ctxt PASSED
[DEBUG] 10.85 ssl_test:311 - msgHTTPSTask{TLS_TASK}$ loading CA CERT from file /mod/ssl_certs/modulesCA.crt
[DEBUG] 10.85 ssl_test:315 - msgHTTPSTask{TLS_TASK}$ file size: 1740
[DEBUG] 10.85 ssl_test:328 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1740
Buffer successfully received from file. 1740 bytes were loaded.
Closing file.
[DEBUG] 10.85 ssl_test:361 - msgHTTPSTask{TLS_TASK}$ loading client CERT from file /mod/ssl_certs/modulesClient.crt
[DEBUG] 10.85 ssl_test:365 - msgHTTPSTask{TLS_TASK}$ file size: 1651
[DEBUG] 10.85 ssl_test:378 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1651
Buffer successfully received from file. 1651 bytes were loaded.
Closing file.
[DEBUG] 10.85 ssl_test:401 - msgHTTPSTask{TLS_TASK}$ loading client KEY from file /mod/ssl_certs/modulesClient_pkcs1.key
[DEBUG] 10.85 ssl_test:405 - msgHTTPSTask{TLS_TASK}$ file size: 1679
[DEBUG] 10.85 ssl_test:418 - msgHTTPSTask{TLS_TASK}$ Reading content from file. Size: 1679
Buffer successfully received from file. 1679 bytes were loaded.
Closing file.
[DEBUG] 10.85 ssl_test:448 - msgHTTPSTask{TLS_TASK}$ certificates successfully stored!
[DEBUG] 10.85 ssl_test:457 - msgHTTPSTask{TLS_TASK}$ m2mb_net_init returned M2MB_RESULT_SUCCESS
[DEBUG] 10.85 ssl_test:465 - msgHTTPSTask{TLS_TASK}$ Waiting for registration...
[DEBUG] 10.86 ssl_test:171 - NetCallback{pubTspt_0}$ Module is registered to cell 0x468E!
[DEBUG] 10.86 ssl_test:477 - msgHTTPSTask{TLS_TASK}$ Pdp context activation
[DEBUG] 10.86 ssl_test:481 - msgHTTPSTask{TLS_TASK}$ m2mb_pdp_init returned M2MB_RESULT_SUCCESS
[DEBUG] 12.87 ssl_test:496 - msgHTTPSTask{TLS_TASK}$ Activate PDP with APN web.omnitel.it....
[DEBUG] 13.71 ssl_test:197 - PdpCallback{pubTspt_0}$ Context activated!
[DEBUG] 13.71 ssl_test:200 - PdpCallback{pubTspt_0}$ IP address: 2.41.76.63
[DEBUG] 13.71 ssl_test:514 - msgHTTPSTask{TLS_TASK}$ Creating Socket...
[DEBUG] 13.71 ssl_test:525 - msgHTTPSTask{TLS_TASK}$ Socket created
[DEBUG] 13.71 ssl_test:535 - msgHTTPSTask{TLS_TASK}$ Socket ctx set to 3
[DEBUG] 13.92 ssl_test:548 - msgHTTPSTask{TLS_TASK}$ Retrieved IP: 185.86.42.218
[DEBUG] 14.05 ssl_test:562 - msgHTTPSTask{TLS_TASK}$ Socket Connected!
[DEBUG] 15.97 ssl_test:587 - msgHTTPSTask{TLS_TASK}$ m2mb_ssl_connect ret 0
[DEBUG] 17.99 ssl_test:593 - msgHTTPSTask{TLS_TASK}$ Sending bytes..
```

```
[DEBUG] 17.99 ssl_test:593 - msgHTTPSTask{TLS_TASK}$ Sending bytes..
[DEBUG] 17.99 ssl_test:596 - msgHTTPSTask{TLS_TASK}$ SSL write result = 44
[DEBUG] 22.03 ssl_test:608 - msgHTTPSTask{TLS_TASK}$ pending bytes: 1087
[DEBUG] 22.03 ssl_test:612 - msgHTTPSTask{TLS_TASK}$ trying to receive 1087 bytes..
[DEBUG] 22.03 ssl_test:618 - msgHTTPSTask{TLS_TASK}$ Server response: (269)<HTTP/1.1 200 OK
Date: Wed, 03 Mar 2021 09:18:22 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] 22.03 ssl_test:634 - msgHTTPSTask{TLS_TASK}$ pending bytes: 762
[DEBUG] 22.03 ssl_test:638 - msgHTTPSTask{TLS_TASK}$ trying to receive remaining 762 bytes..
[DEBUG] 22.03 ssl_test:644 - msgHTTPSTask{TLS_TASK}$ Server response: (639)<<html>
<head>
  <title>module.telit.com</title>
  <meta content="text/html; charset=utf-8" />
</head>
<body>
  <table border=0 align=center>
    <tr>
      <td height="100" align=center><h2>modules.telit.com - Test HTML page</h2></td>
    </tr>
    <tr>
      <td align=center><img src=Telit.jpg alt="Telit logo" height="126" width="410"></img></td>
    </tr>
    <tr>
      <td height="200" align=center> This is a simple HTML page, </br>
      made with simple HTML code,</br>
      just for test!
    </td>
    </tr>
    <tr>
      <td height="100" align=center><font size="3">Telit &copy; 2015 - 2017 All rights reserved</font></td>
    </tr>
  </table>
</body>
</html>
>
[DEBUG] 22.03 ssl_test:662 - msgHTTPSTask{TLS_TASK}$ application exit
[DEBUG] 22.03 ssl_test:680 - msgHTTPSTask{TLS_TASK}$ m2mb_pdp_deactivate returned success
[DEBUG] 22.03 ssl_test:683 - msgHTTPSTask{TLS_TASK}$ Application complete.
[DEBUG] 22.77 ssl_test:206 - PdpCallback{pubTspt_0}$ Context deactivated!
```

3.5.24 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 0xC4CF1
[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 rcv...
[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 81

3.5.25 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="/data/azc/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 /data/azc/mod folder, into its content test.txt. The file must be uploaded by the user (see steps above).

```
Starting Logging demo app. This is v1.0.7 built on Apr 7 2020 09:02:35.
Starting TEST_COMPR_UNCOMPR.
len: 138; comprLen: 57
Compressed message:
W+EHU(,ILIVH^E/ISHE`PE*I-HMQE/K-R(ÛÊç$VU*#ašê y4RI«¥1.
comprLen: 57; uncomprLen: 138
uncompress():
the quick brown fox jumped over the lazy dog. the quick brown fox jumped over the lazy dog. the quick brown fox jumped over the lazy dog.
Ending TEST_COMPR_UNCOMPR with SUCCESS.
Starting test_uncompress.
Data extracted correctly into the file ./mod/test.txt
test_uncompress finished correctly!
```

Figure 82

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

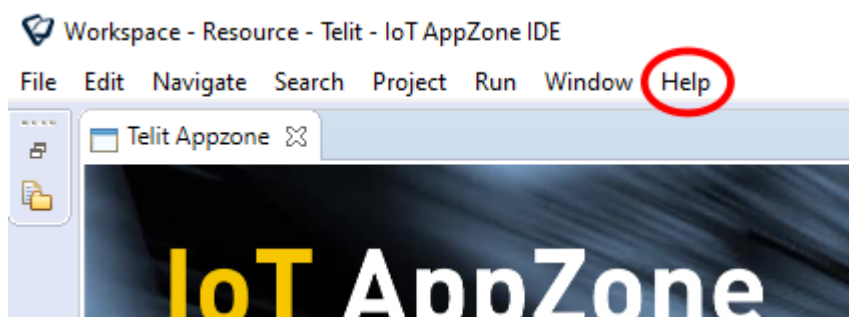
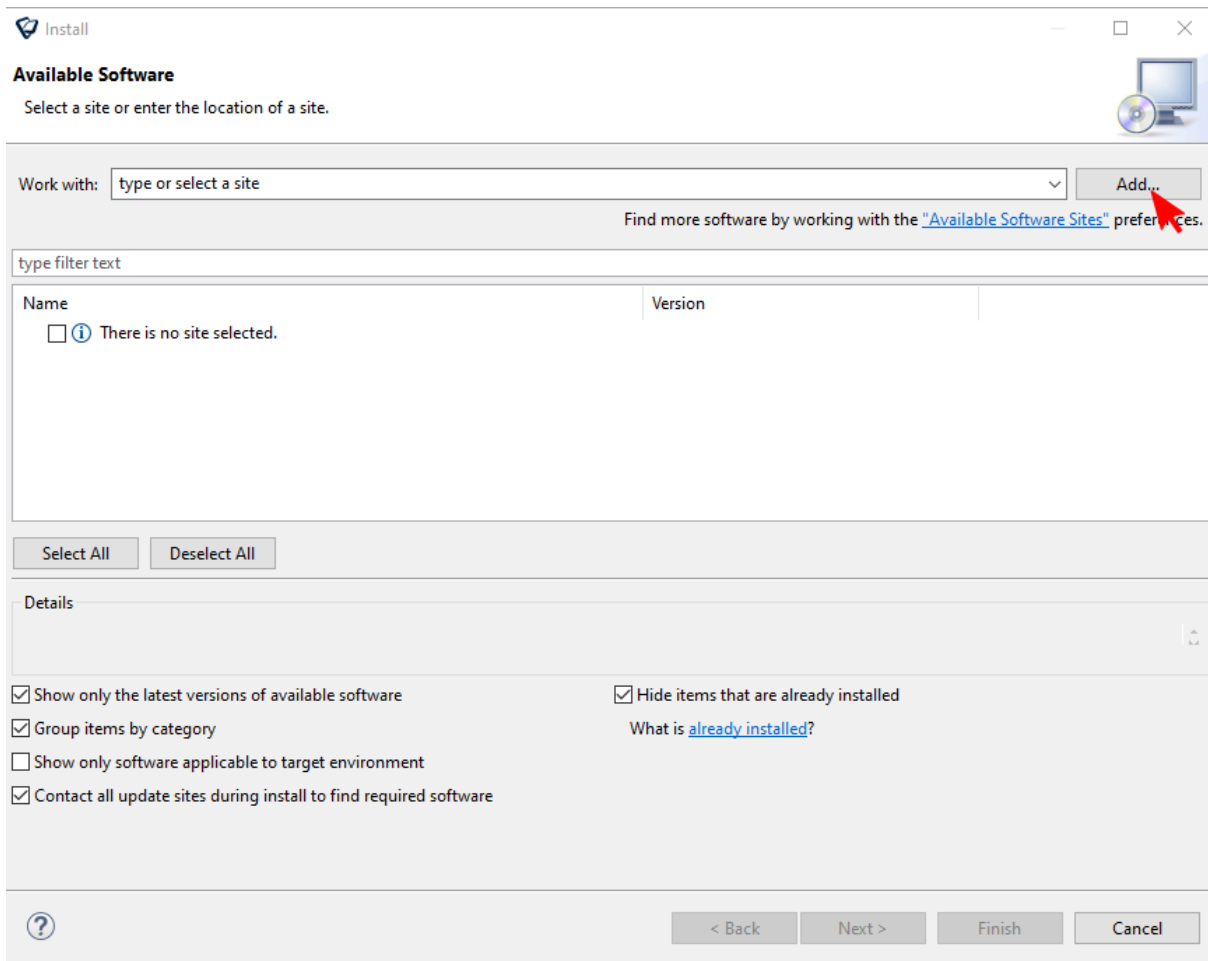
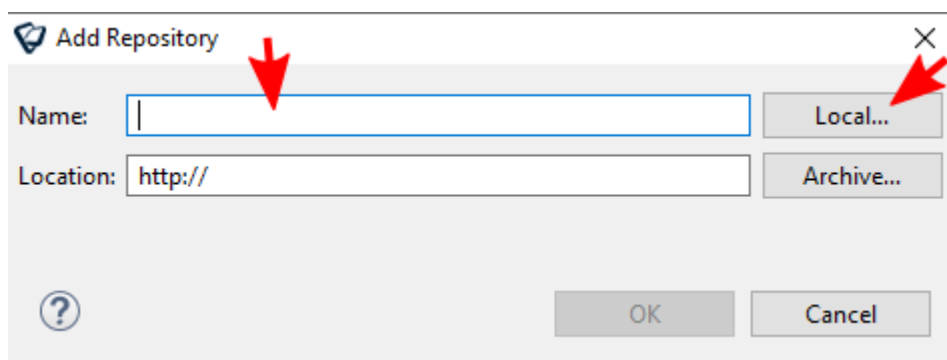


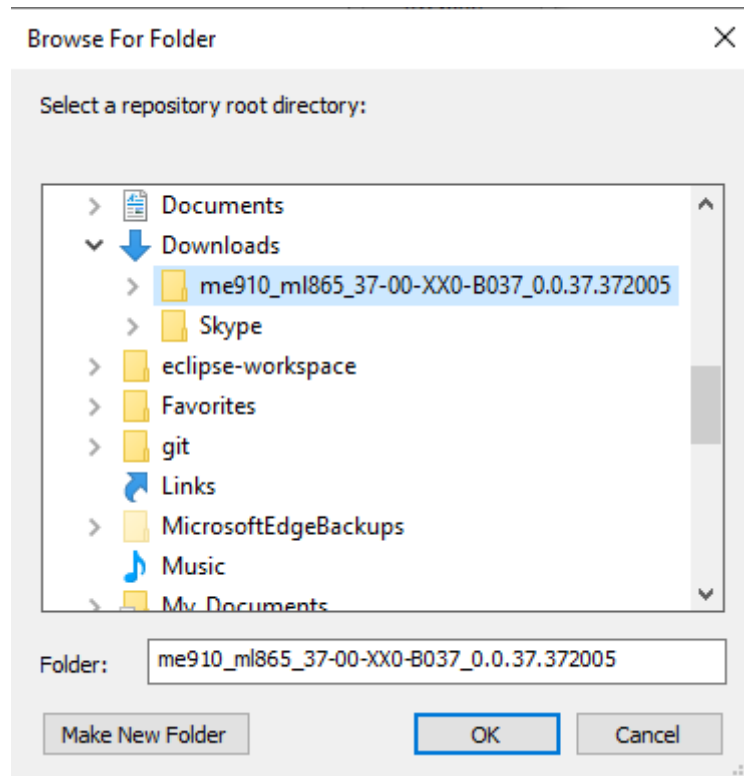
Figure 83

Click on “Help” tag and choose “Install New Software...”. This window will appear:

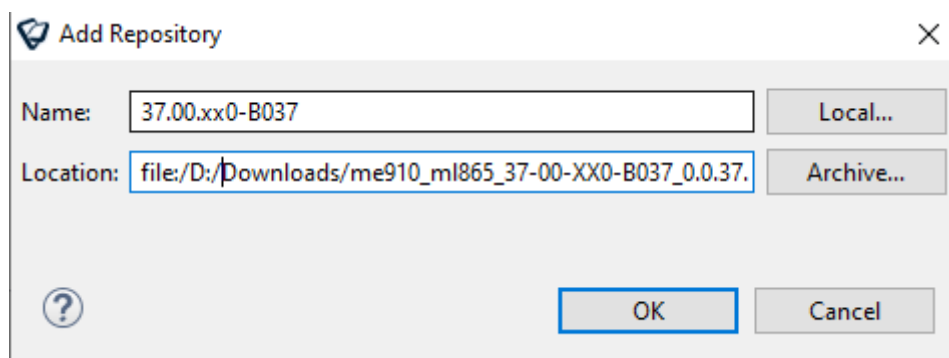
**Figure 84**

Click on "Add..." button and then in the following window click on "Local..." to select the unzipped folder with the plug-in content.

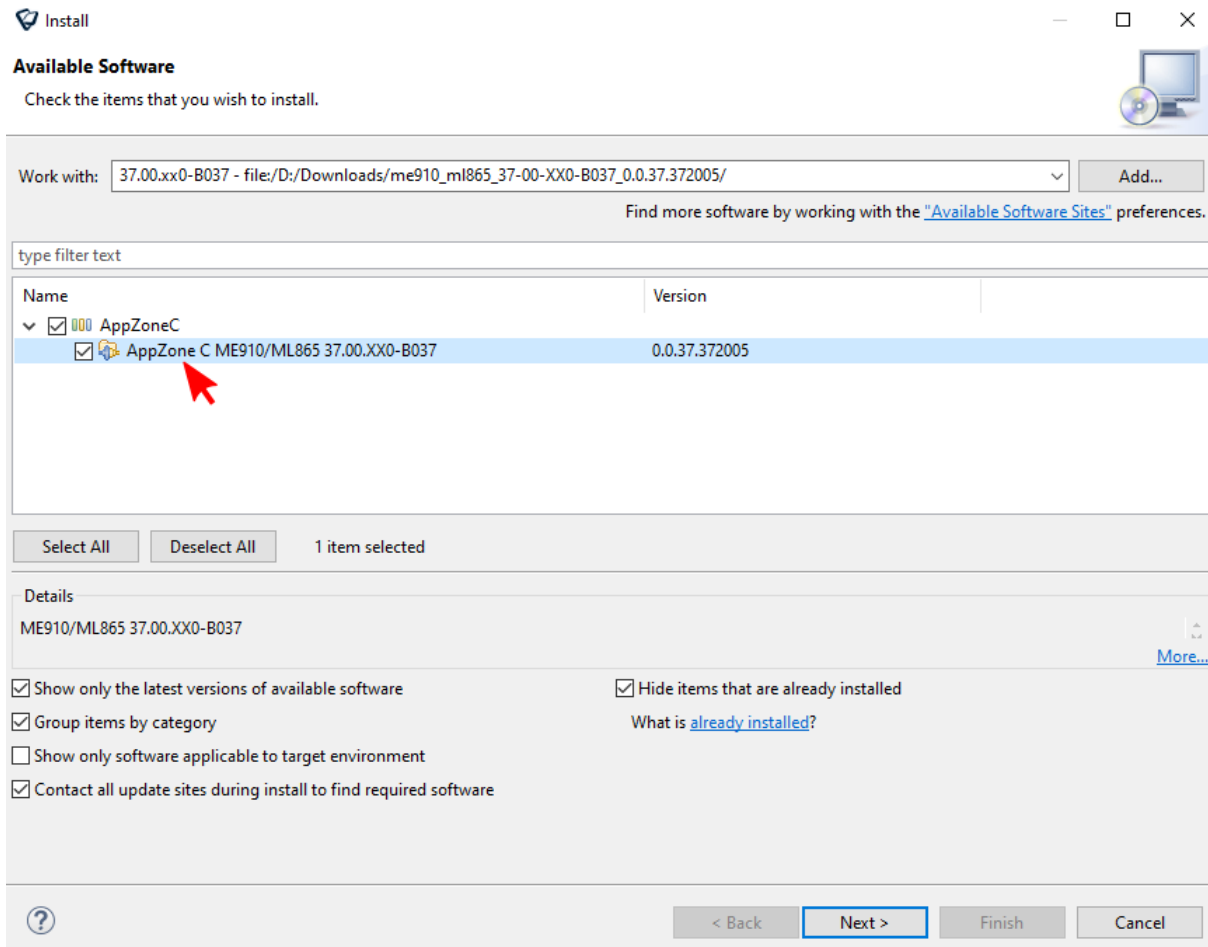
**Figure 85**

**Figure 86**

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 87**

The new packet is now ready to be installed: select it and click on “Next >” button until “Review Licenses” window will appear.

**Figure 88**

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”.

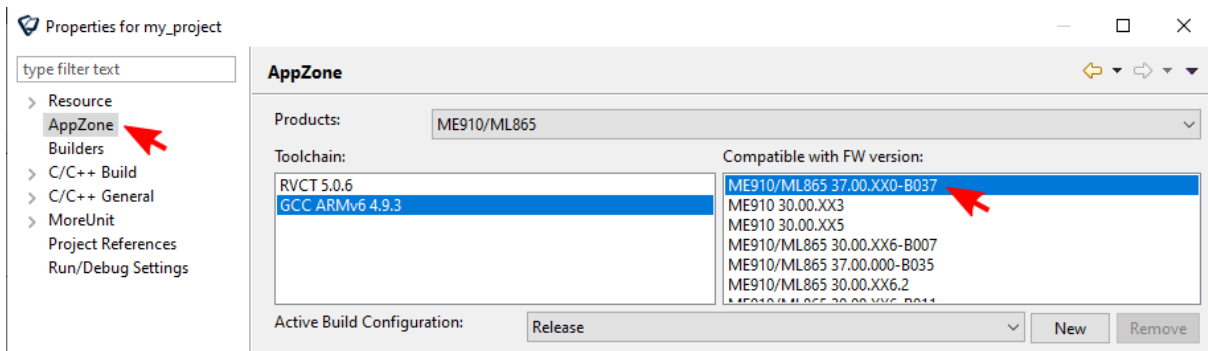


Figure 89

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”

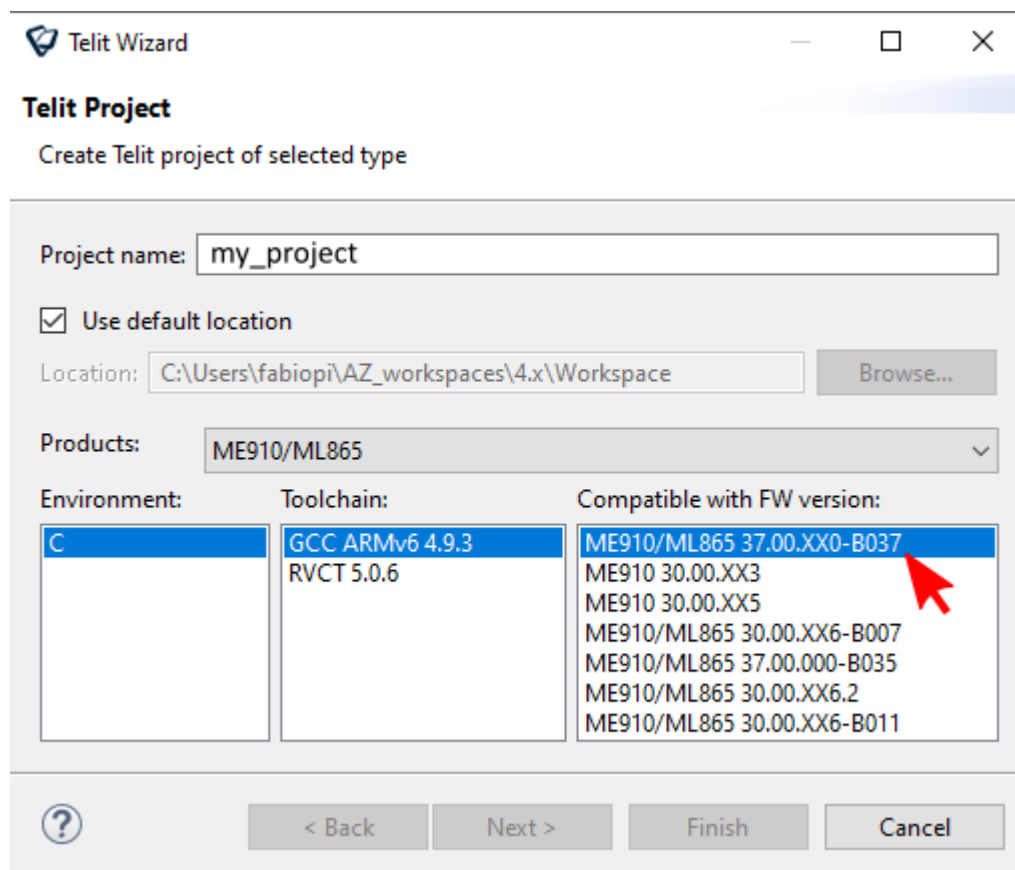


Figure 90

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
Tecnologia e Servicos Ltda
Avenida Paulista, 1776, Room 10.C
01310-921 São Paulo, Brazil

Telit reserves all rights to this document and the information contained herein. Products, names, logos and designs described herein may in whole or in part be subject to intellectual property rights. The information contained herein is provided "as is". No warranty of any kind, either express or implied, is made in relation to the accuracy, reliability, fitness for a particular purpose or content of this document. This document may be revised by Telit at any time. For most recent documents, please visit www.telit.com

Copyright © 2016, Telit