
SerizII_A70CM Documentation

Release 0

Silica

Mar 16, 2017

Contents

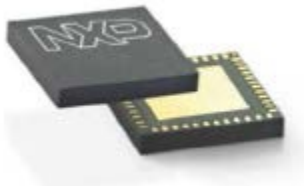
1	NXP Secure Authentication Microcontroller A70CM	3
2	INTRODUCTION	5
2.1	Developement tools	5
2.2	Document references	5

Version 1.00C

Copyright (C)2016 Avnet Silica company

Date 11 March 2015

NXP Secure Authentication Microcontroller A70CM



INTRODUCTION

The Silica SerizII with A70CM add-on board is designed to evaluate NXP A7001 ADPU secure module. The firmware will perform a sample demo application between LPC4350 cpu, A70CM Host API library and SCI2C library. It show the use of main basic A70CM API functions for AES encryption/decryption (using CBC mode) and a sample of authentication process between Server and Client. In addition, a Man-In-The-Middle simulator is implemented to change data and check security results. The firmware can be compiled in many different functionality by setting appropriate macros define:

Mode	Main features
Stand Alone	Use only 1 SerizII board with A70CM add-on. Messages are exchanged between the two A70CM onboard using RTOS queue. Man-In-The-Middle simulator
LAN Client	Use on SerizII Babylon client board. The messages are exchanged to Server using UDP queue
LAN Server	Use on SerizII Babylon server board. The messages are exchanged to Client using UDP queue. Man-In-The-Middle simulator

See at “Firmware Specification” paragraph for how to configure macros

Developement tools

Firmware was developed using: LPCXpresso 6-1-2 free version. For installation and configuration of the project, follow instruction inside [Quick start guide](#)

Document references

The A70CM reference documentation is available only under NDA agreement. Contact Silica for more informations about.

Contents:

Quick start guide

This guide will provide instructions to install the development environment and all the libraries needed to compile and debug the demo firmware SerizII A70CM. The main steps are:

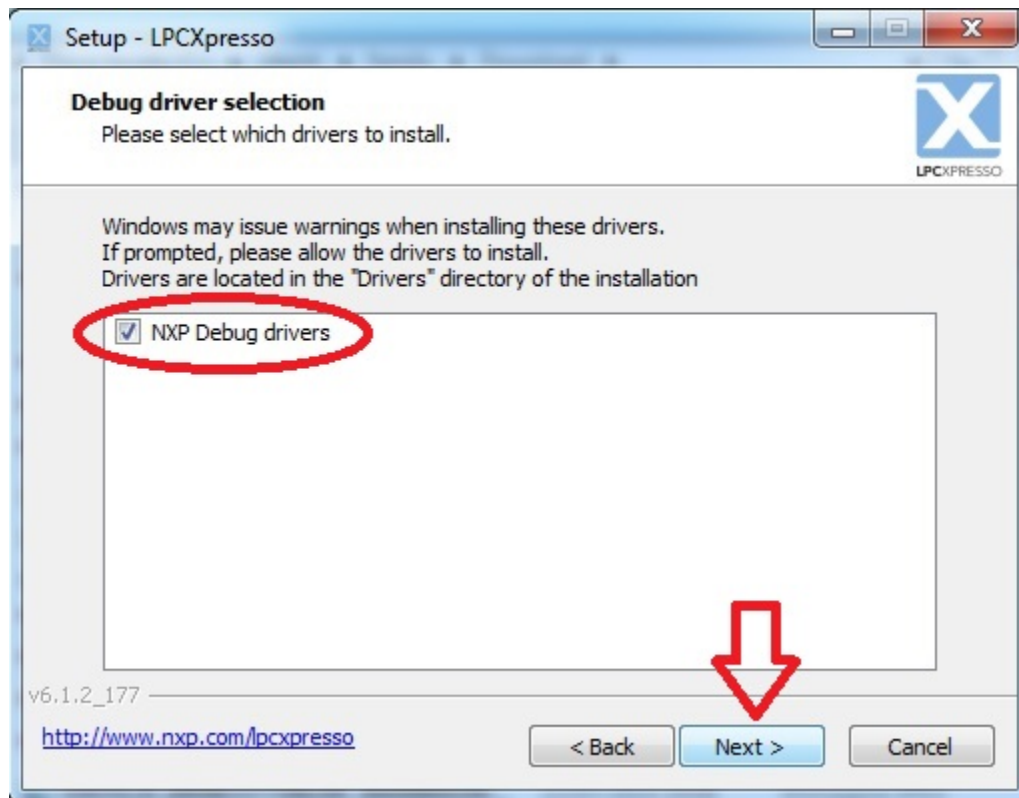
- Install and register LPCXpresso IDE
- Install LPCOpen
- Install EmWin
- Install firmware and configure LPCXpresso IDE

Load and install LPCXpresso

Download LPCXpresso 6.1.2 (or later) https://s3.amazonaws.com/LPCXpresso6/LPCXpresso_6.1.2_177.exe

- Install and register the software (this will enable to work up to 256KB code size);
- The tools will install under the C:\NXP subdirectory

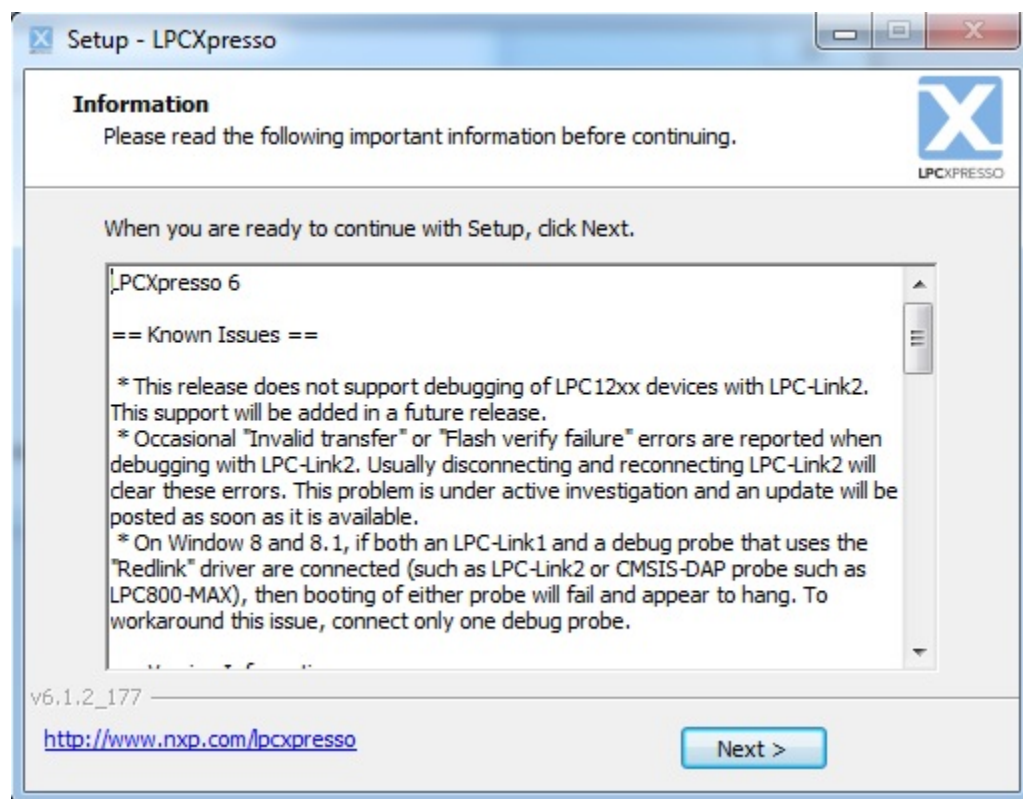
During installation tool will ask for debug driver install:



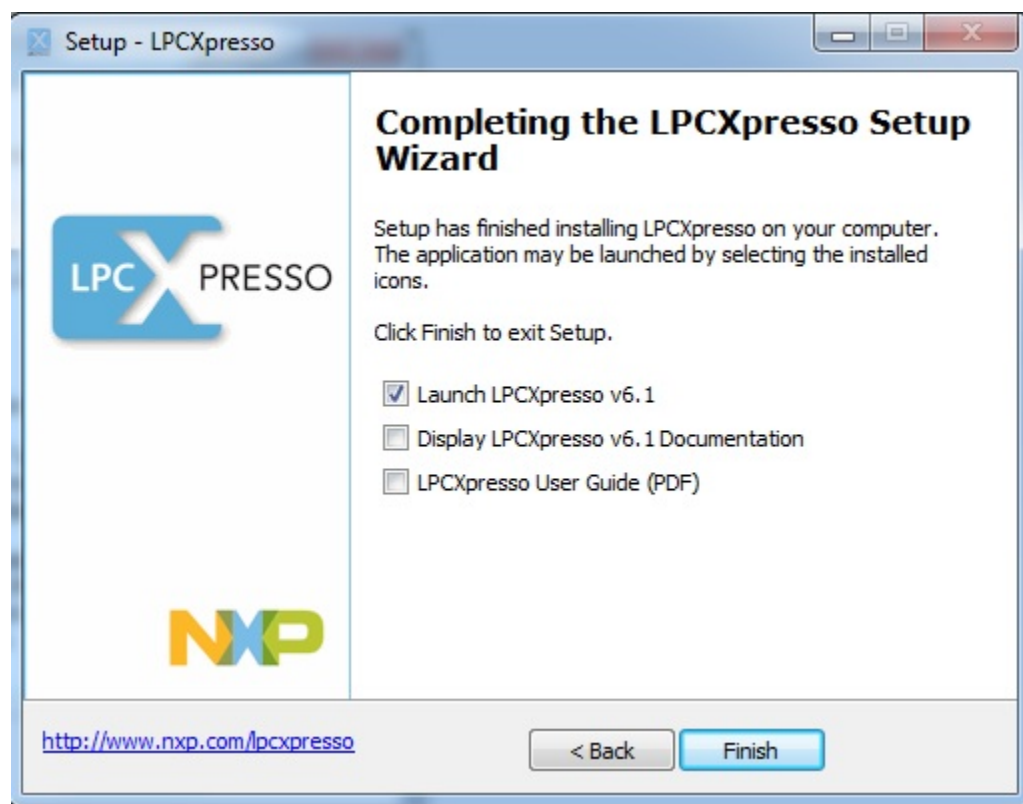
And then, Windows Security will ask about reliability of software driver:

- CDM driver package (tree times)
- Generic NXP driver (one times)

Click on “Install” to confirm

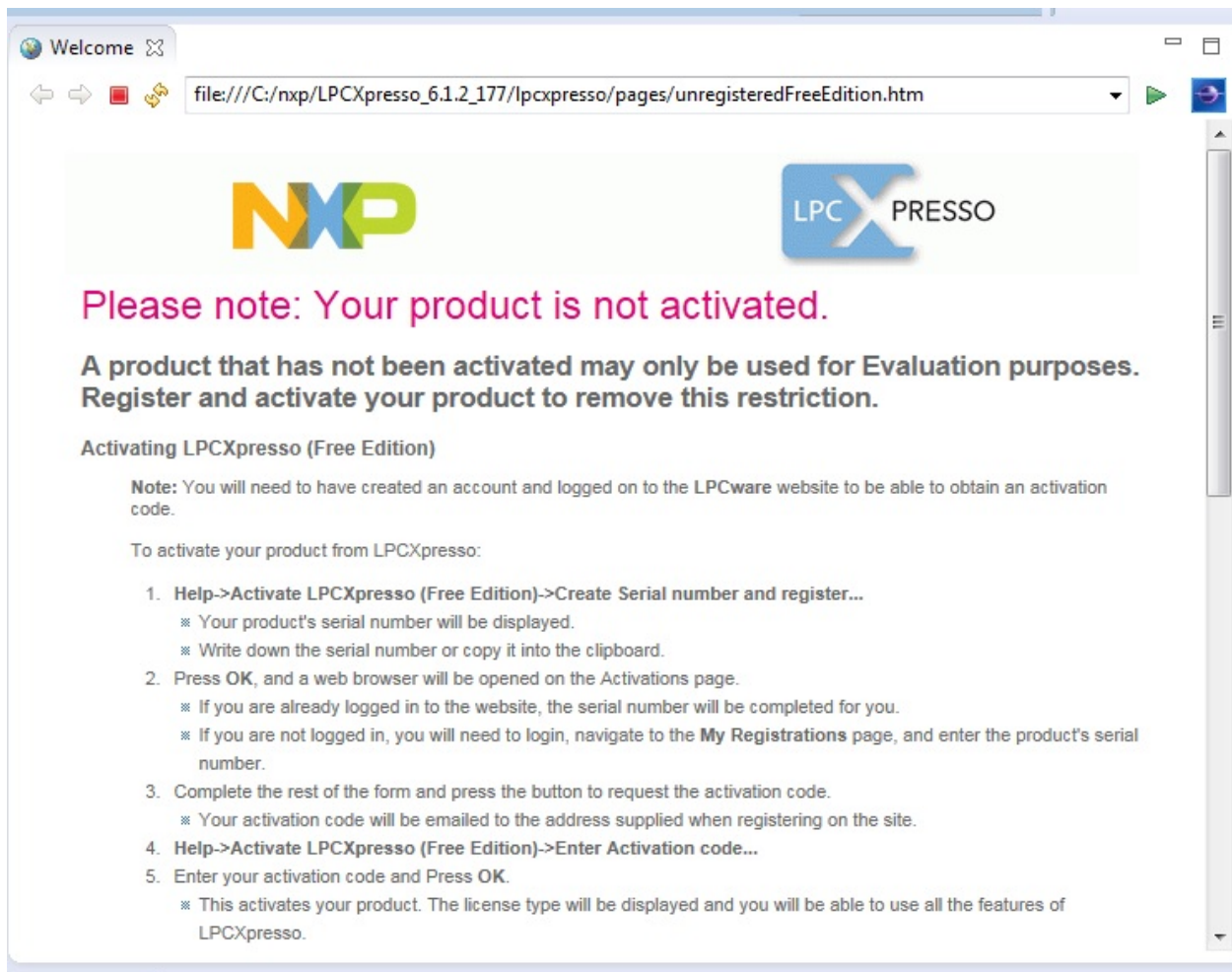


When all driver installation ends, click next

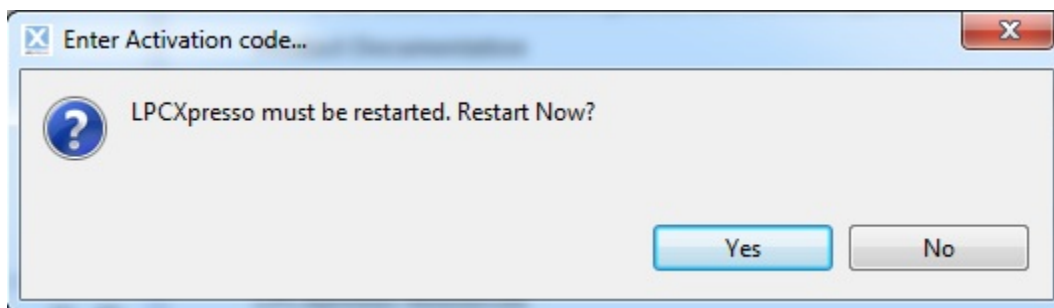


Select checkbox as figure above, then click finish.

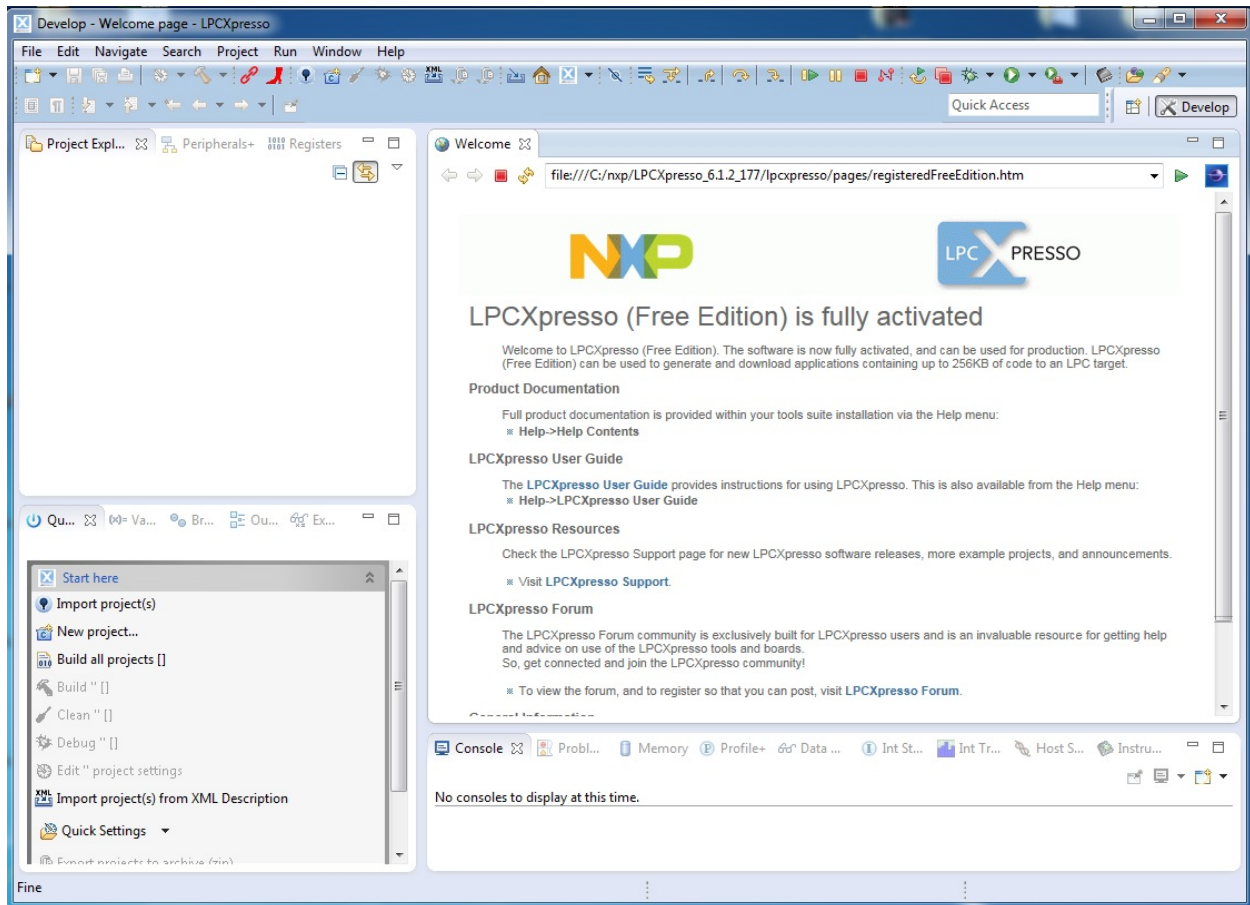
When LPCXpresso will open, follow free license registration instructions to work up to 256KB code size (see figure below)



After activation success, click OK to restart LPCXpresso.



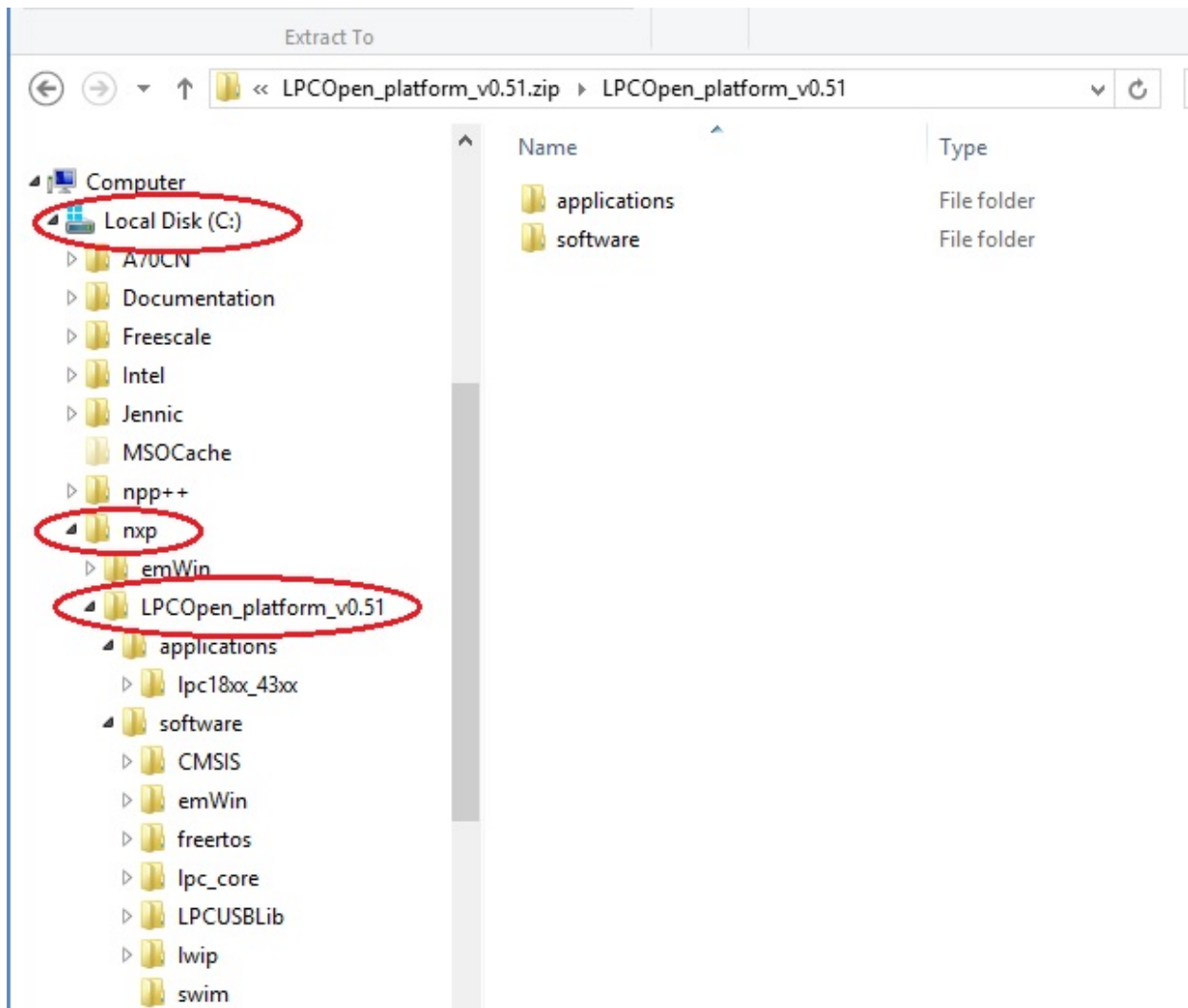
The "welcome" screen will appear, LPCXpresso successfully installed



Load and install LPCOpen

Download lpcOpen http://www.lpcware.com/system/files/LPCOpen_platform_v0.51.zip

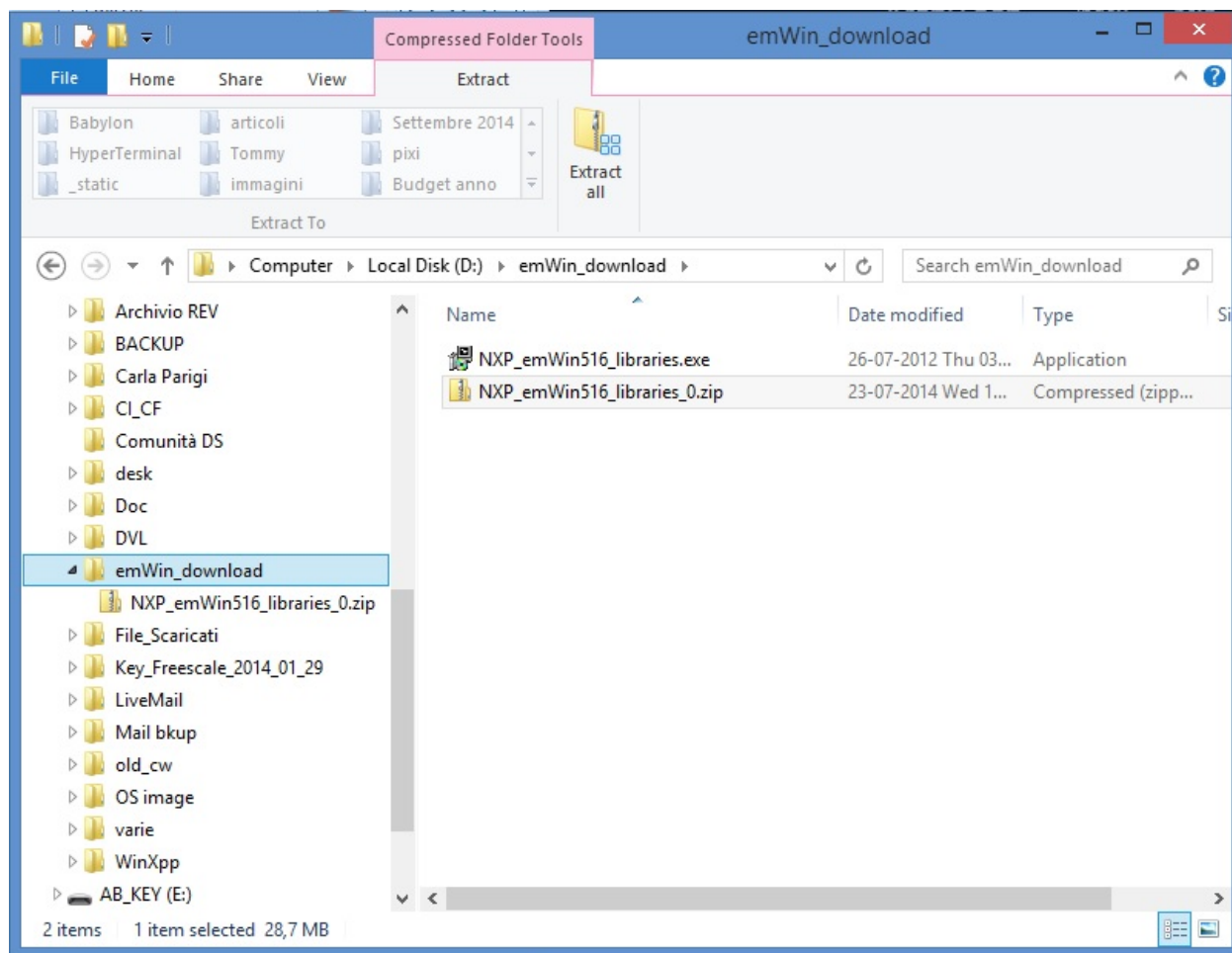
Extract the the zip content and put under the C:\NXP subdirectory. See figure below.



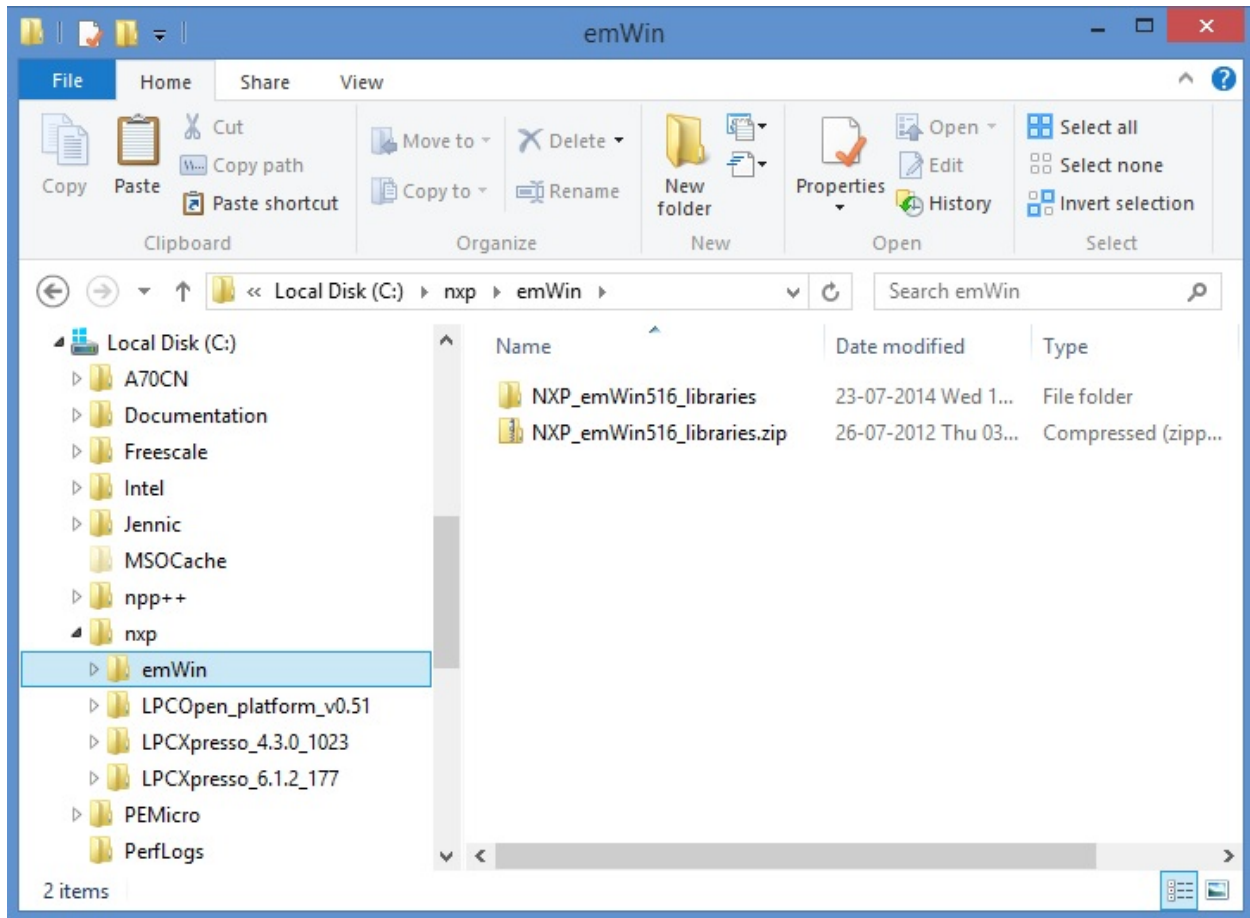
Load and install EmWin

Download emWin (not included for License issues) http://www.lpcware.com/system/files/NXP_emWin516_libraries_0.zip

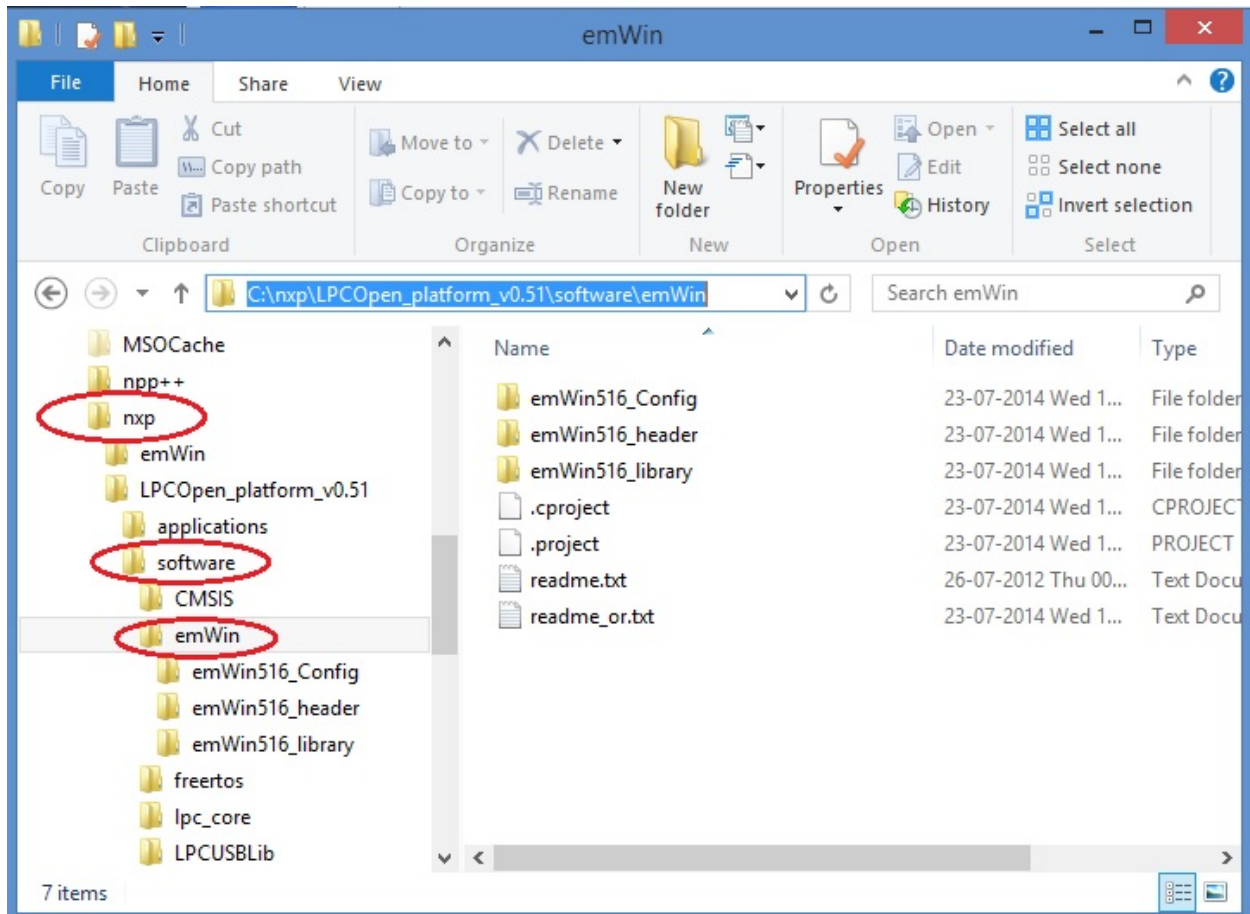
- Unzip the download compressed folder



- Run the resulting file `NXP_emWin516_libraries.exe`, which will self extract a file named '`NXP_emWin516_libraries.zip`' (located by default in `C:\NXP\emWin\`)
 - Extract the content of `NXP_emWin516_libraries.zip`, which is the folder `NXP_emWin516_libraries`.
- (sse figure above)



- Copy the contents of this folder to C:\NXP\LPCOpen_platform_v0.51\software\emWin

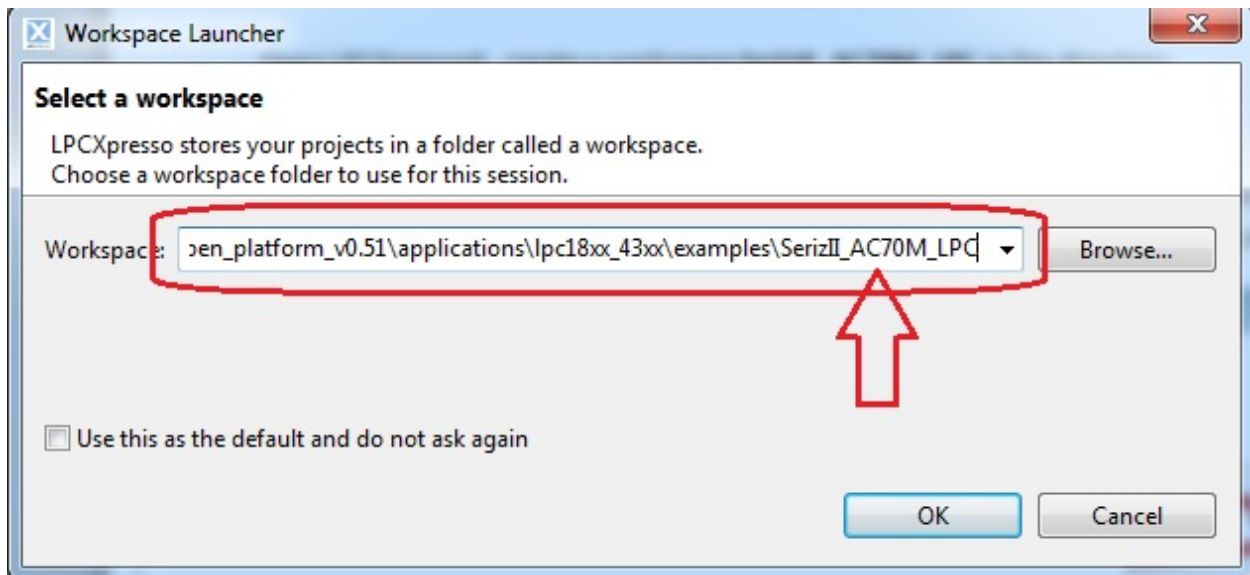


(it is important to follow the exact directory path, inside the project some files have relative reference to files included into the LPCOpen_platform_v0.51, any path change requires changes in this references)

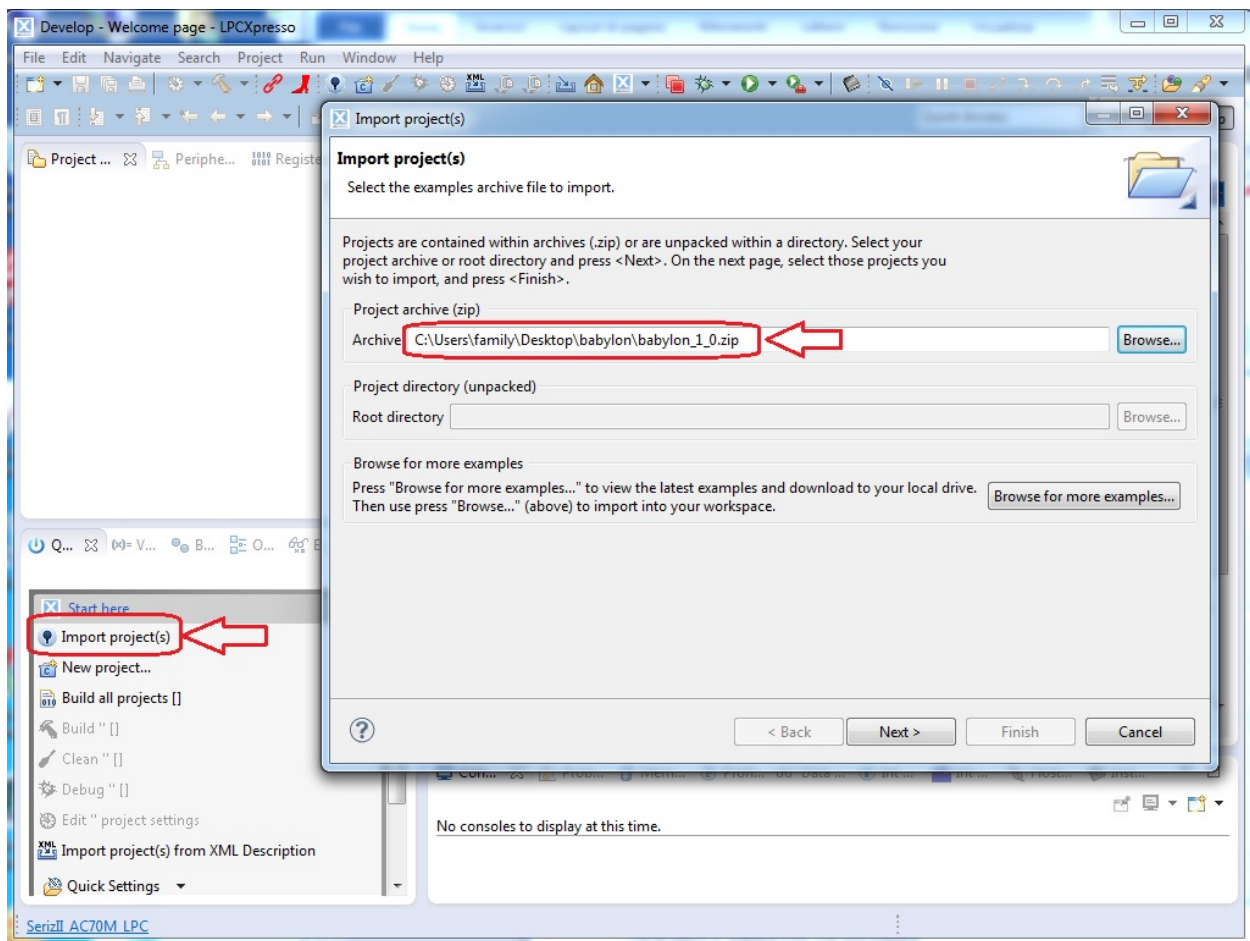
Run the Babylon A70CM demo

Open LPCXpresso6, create a workspace **SerizII_A70CM_LPC** in the directory `C:\nxp\LPCOpen_platform_v0.51\applications\lpc18xx_43xx\examples\`

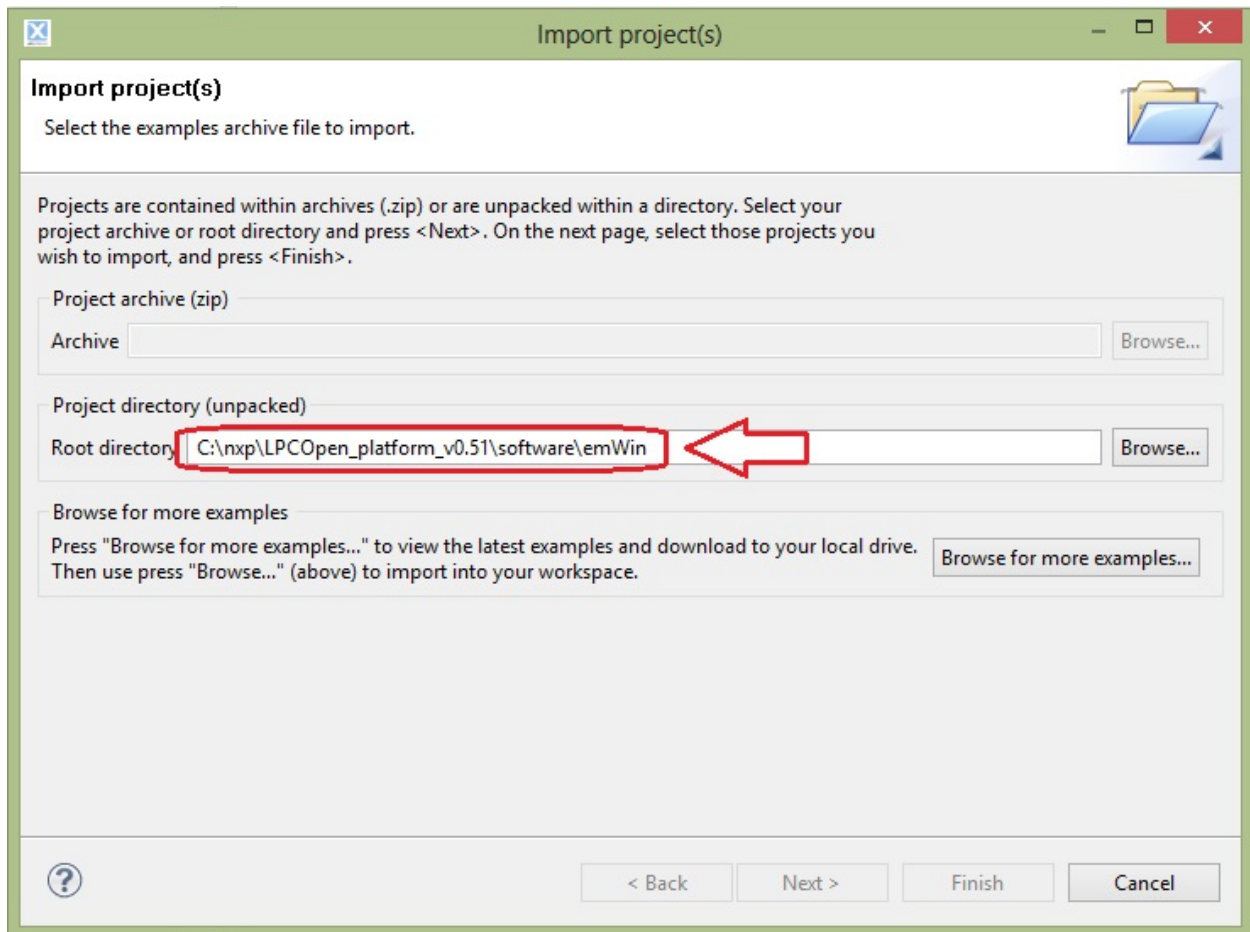
(it is important to follow the exact directory path, inside the project some files have relative reference to files included into the LPCOpen_platform_v0.51, any path change requires changes in this references)



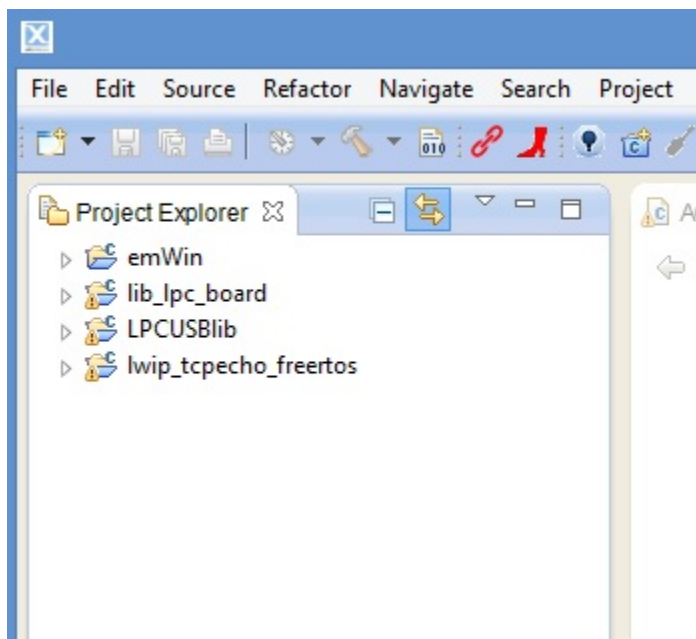
Import the **babylon_1_0.zip** into the workspace



Import project C:\NXP\LPCOpen_platform_v0.51\software\lemWin

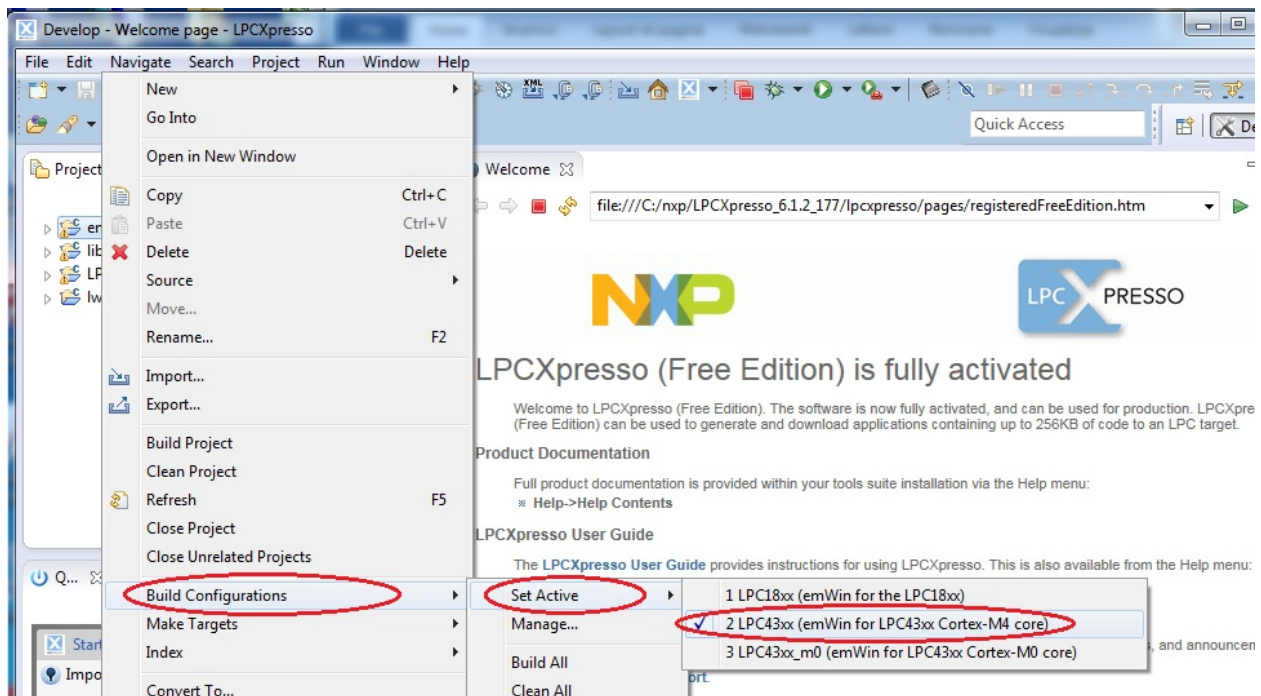


The project is loaded into the workspace



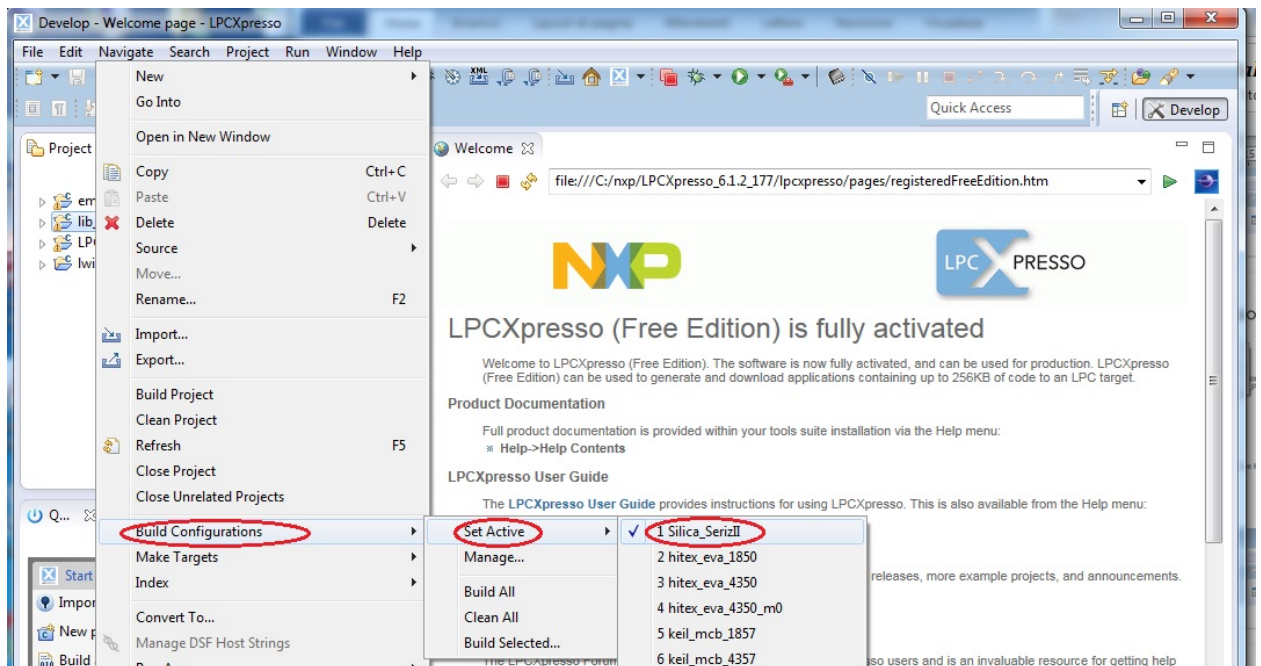
Configuring emWin

select the “LPC43xx” configuration as active



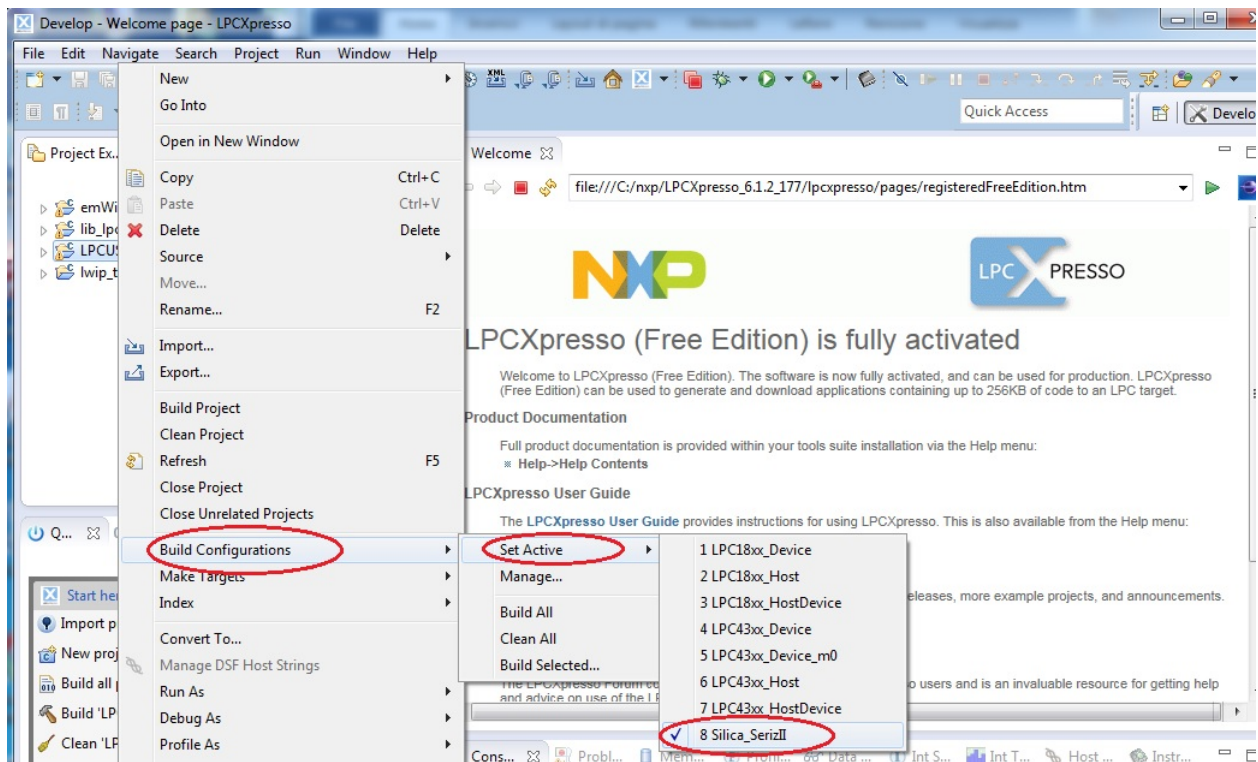
Configuring lib_lpc_board

select the “Silica_SerizII” configuration as active



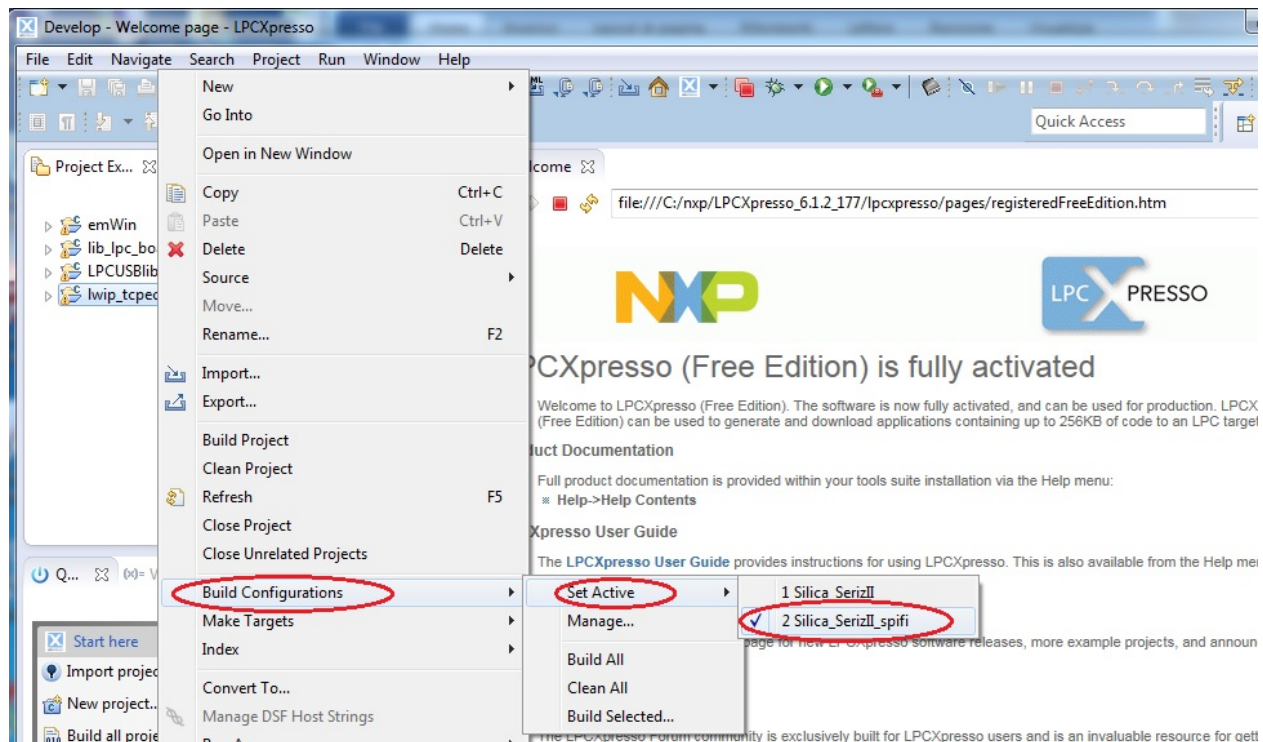
Configuring LPCUSBlib

select the “Silica_SerizII” configuration as active



Configuring lwip_tcpecho_freertos

select as active the “Silica_SerizII_spifi”

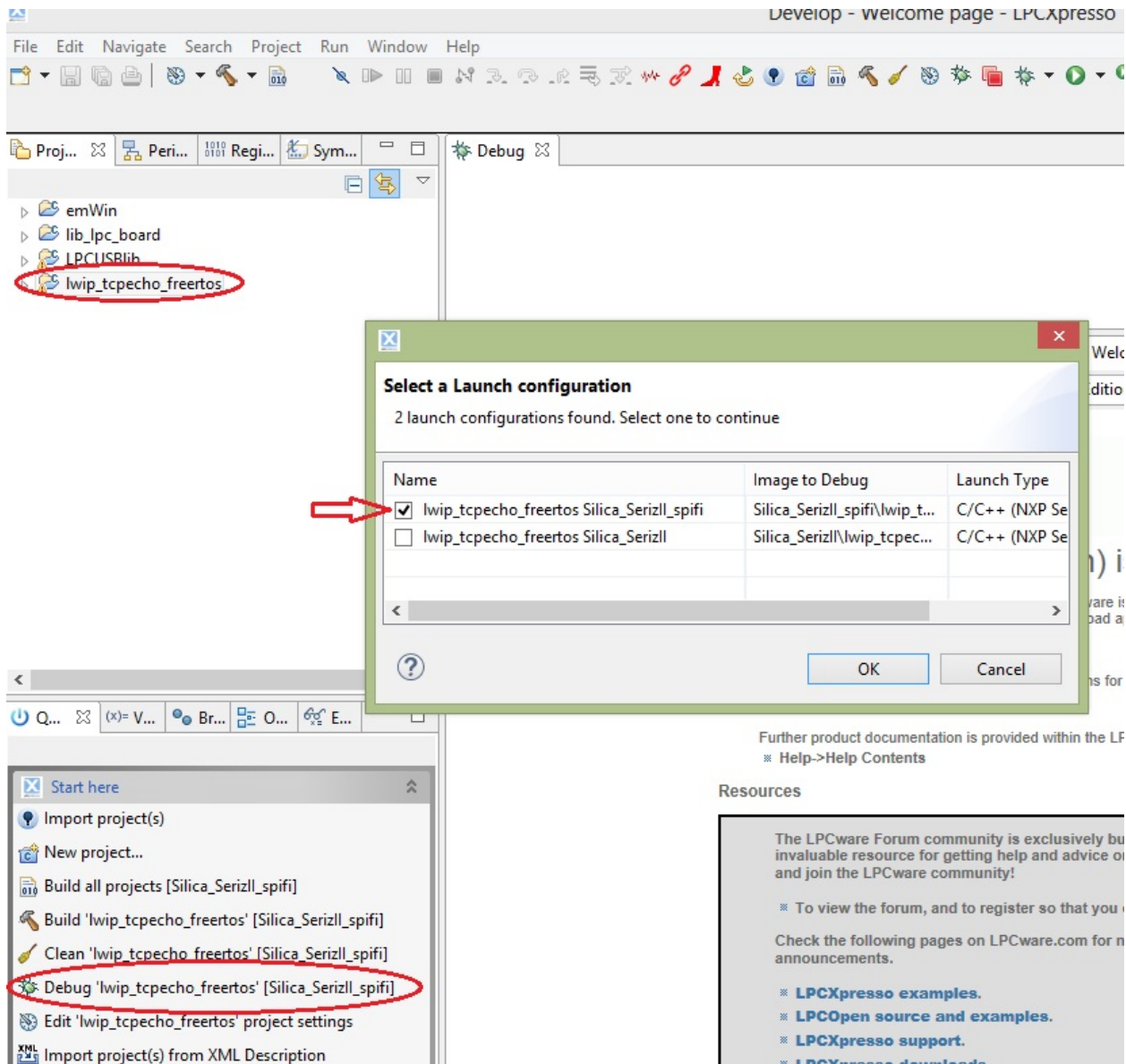
**Important:**

BEFORE COMPILING BABYLON PROJECT, YOU MUST DOWNLOAD A70CM NXP EXAMPLE LIBRARY AND APPLY PATCHES
CLOSE LPCXpresso IDE AND FOLLOW *Patching NXP A70CM libraries* BEFORE PROCEED

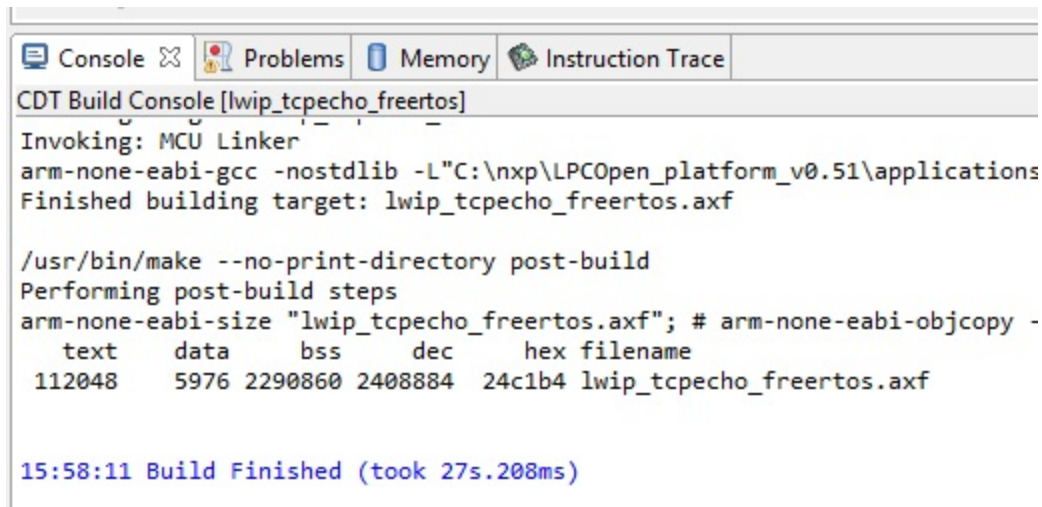
Compile and load firmware

Select the `lwip_tcpecho_freertos` project, select “debug”.

The compiler will warn there are no bin files, simple press OK, until the window “Select debug configuration for `lwip_tcpecho_freertos`” is shown.



Setting OK, the compilation will start (take few minutes to compile).



```

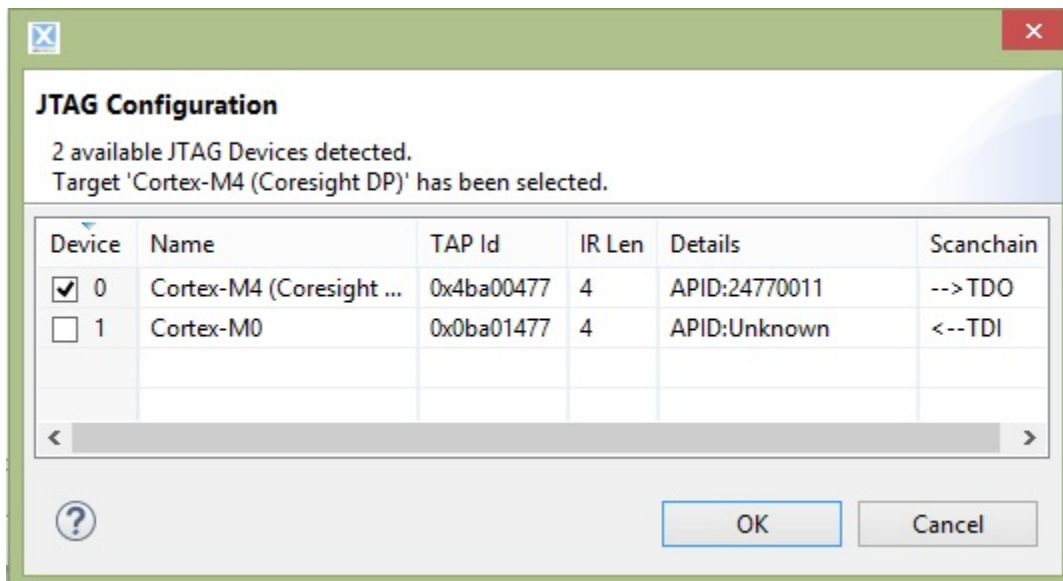
CDT Build Console [lwip_tcpecho_freertos]
Invoking: MCU Linker
arm-none-eabi-gcc -nostdlib -L"C:\nxp\LPCOpen_platform_v0.51\applications
Finished building target: lwip_tcpecho_freertos.axf

/usr/bin/make --no-print-directory post-build
Performing post-build steps
arm-none-eabi-size "lwip_tcpecho_freertos.axf"; # arm-none-eabi-objcopy -
    text    data    bss    dec    hex filename
  112048    5976  2290860  2408884  24c1b4 lwip_tcpecho_freertos.axf

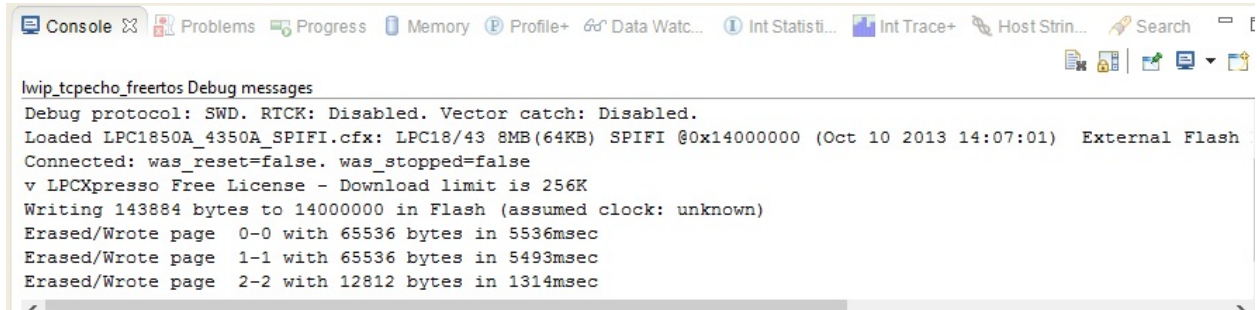
15:58:11 Build Finished (took 27s.208ms)

```

End of compilation, select the JTAG port to use



The executable is downloaded into flash



```

lwip_tcpecho_freertos Debug messages
Debug protocol: SWD. RTCK: Disabled. Vector catch: Disabled.
Loaded LPC1850A_4350A_SPIFI.cfx: LPC18/43 8MB(64KB) SPIFI @0x14000000 (Oct 10 2013 14:07:01) External Flash
Connected: was_reset=false. was_stopped=false
v LPCXpresso Free License - Download limit is 256K
Writing 143884 bytes to 14000000 in Flash (assumed clock: unknown)
Erased/Wrote page 0-0 with 65536 bytes in 5536msec
Erased/Wrote page 1-1 with 65536 bytes in 5493msec
Erased/Wrote page 2-2 with 12812 bytes in 1314msec

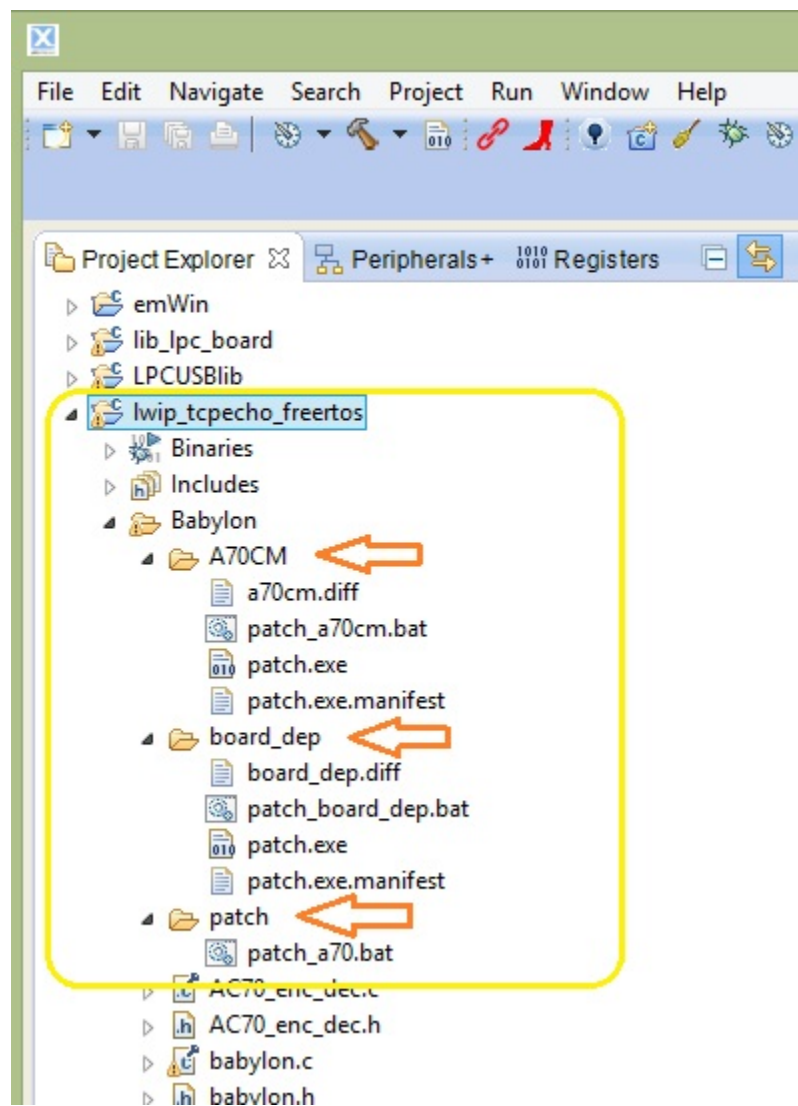
```

Switch off SerizII, unconnect the emulator; Mount on the SerizII A70CM (Babylon board) add-on board, then follow *Getting started Babylon A70CM* to start evaluation

Patching NXP A70CM libraries

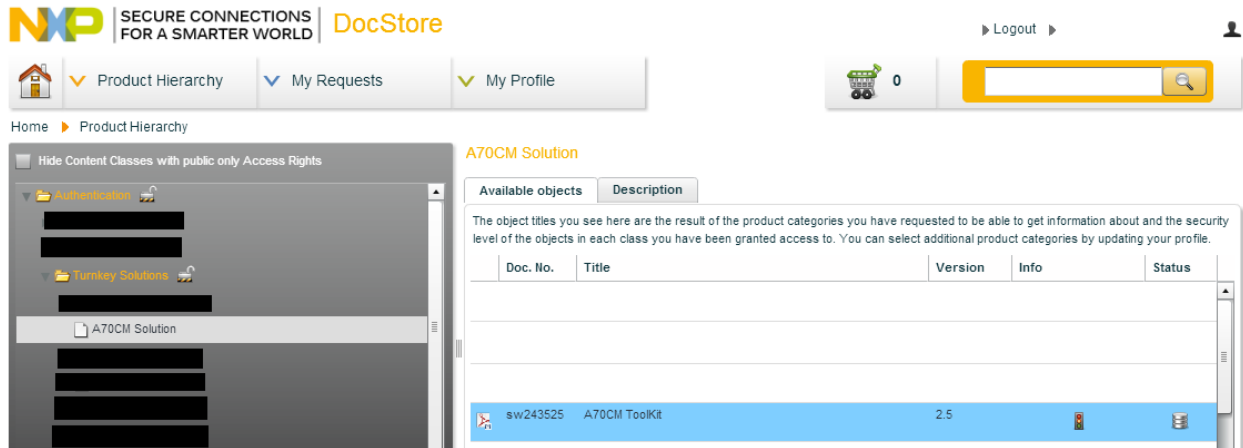
This chapter will show how to import and patch original NXP library for Babylon project

Babylon project comes with folders **a70cm** and **board_dep** containig only patch files required for patching original NXP libraries. (see image below)

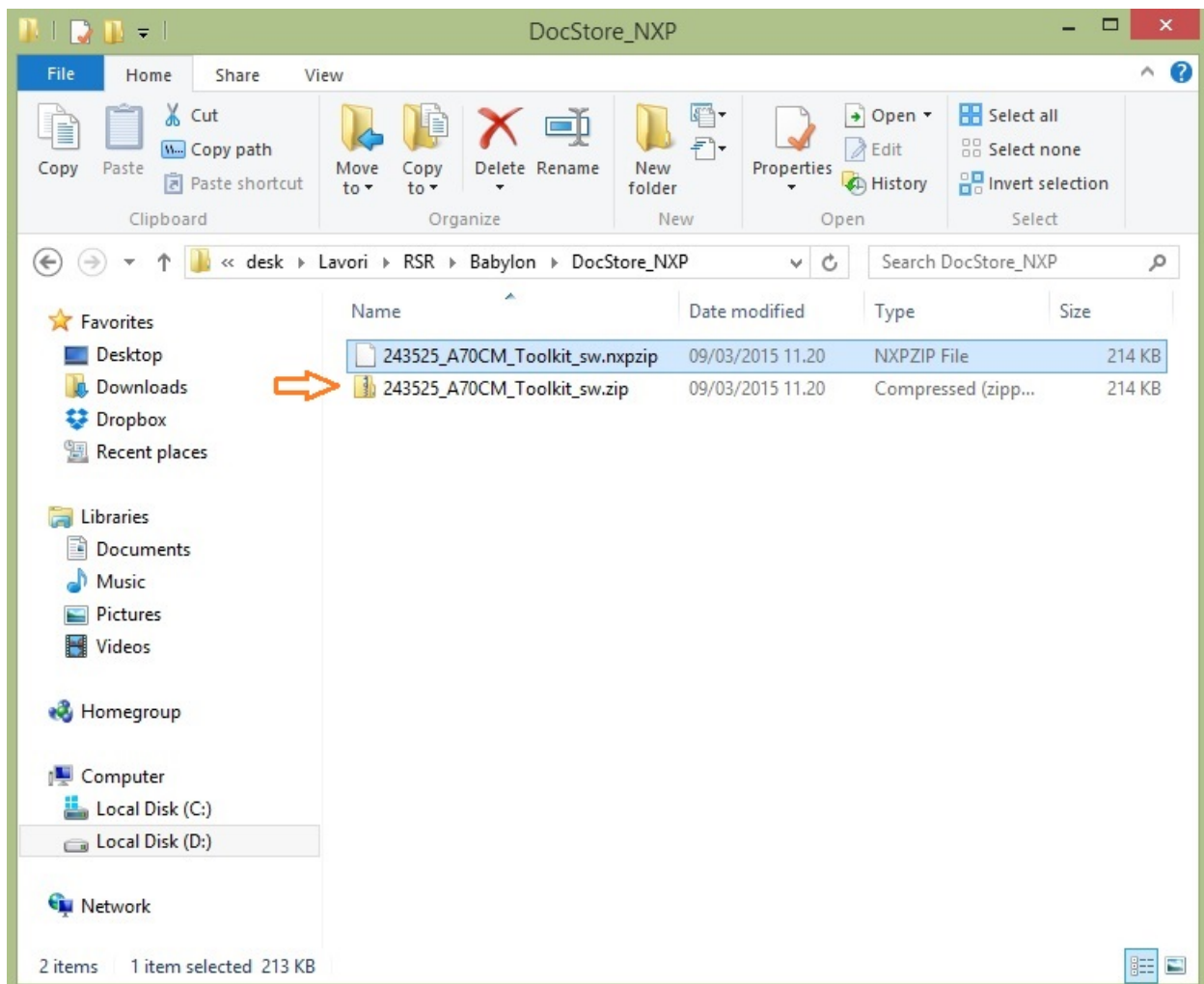


Import library

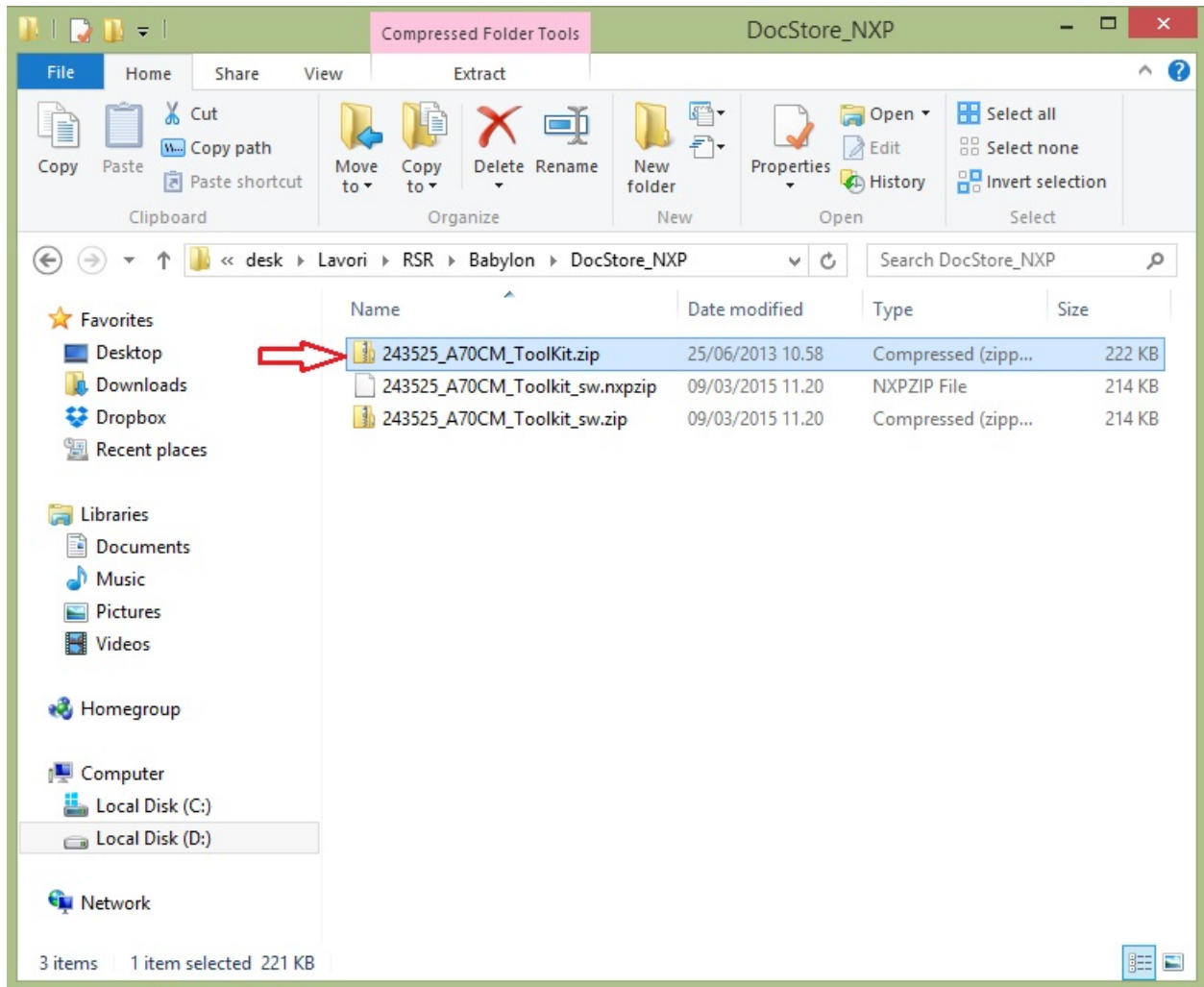
Before compile Babylon application firmware, you must download NXP A70cm example library. To do this, make a registration at link <https://www.docstore.nxp.com> When you receive authorization from NXP, you can download the A70cm example library as figure below



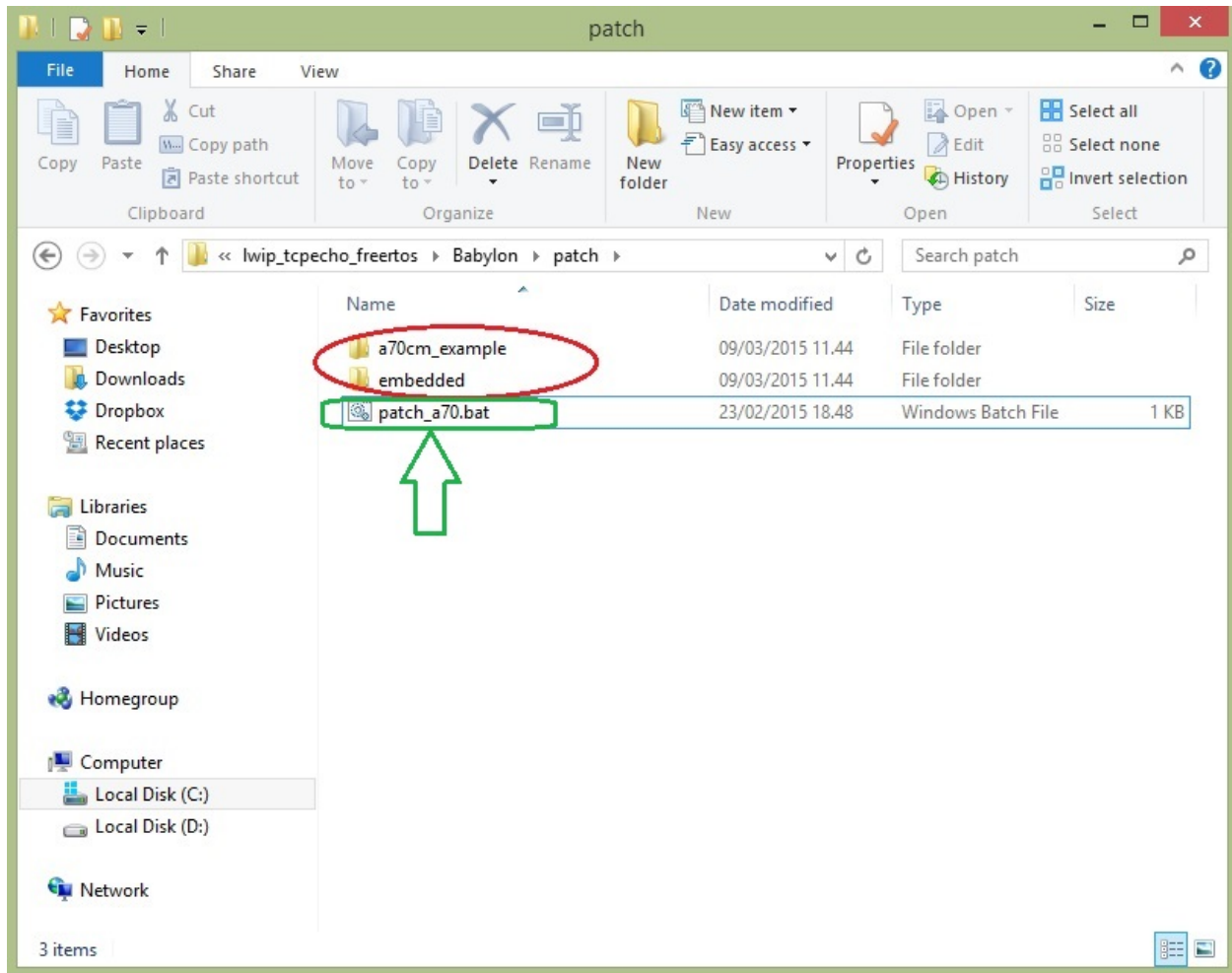
Rename the downloaded file “243525_A70CM_Toolkit_sw.nxpzip” into “**243525_A70CM_Toolkit_sw.zip**”



unzip the file “**243525_A70CM_Toolkit_sw.zip**” just renamed; your NXP library folder content should be as figure below:



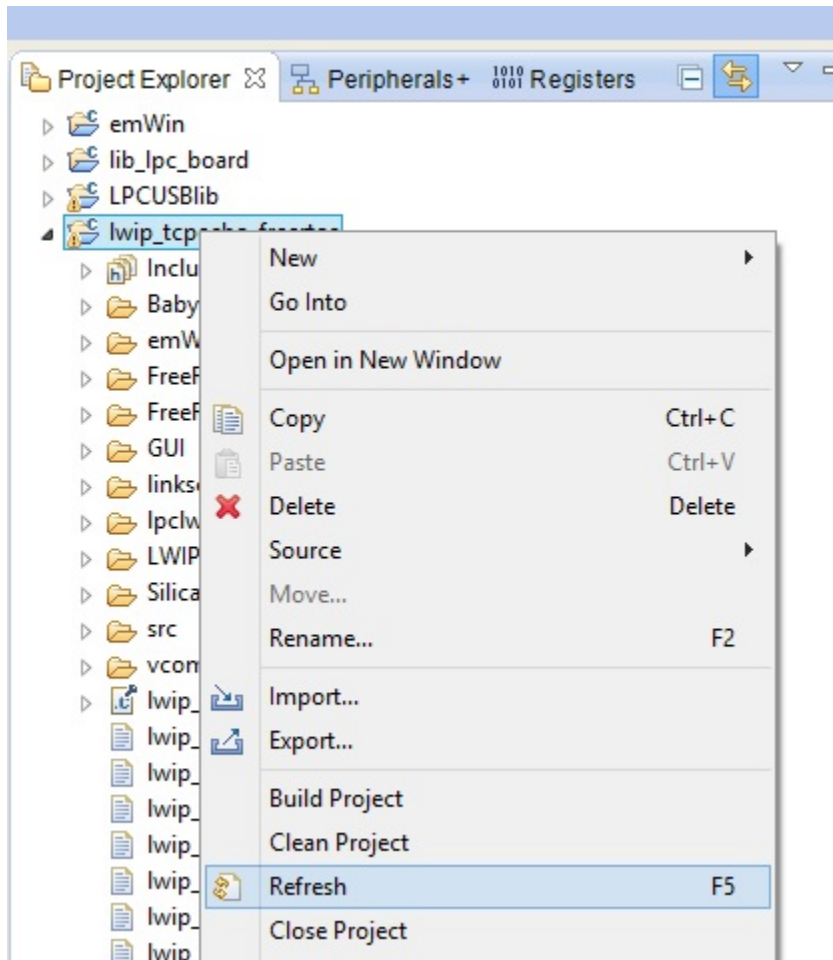
see at file “243525_A70CM_ToolKit.zip”; unzip this file into project folder “<workspace>\wip_tcpecho_freertos\Babylon\patch” located inside your workspace



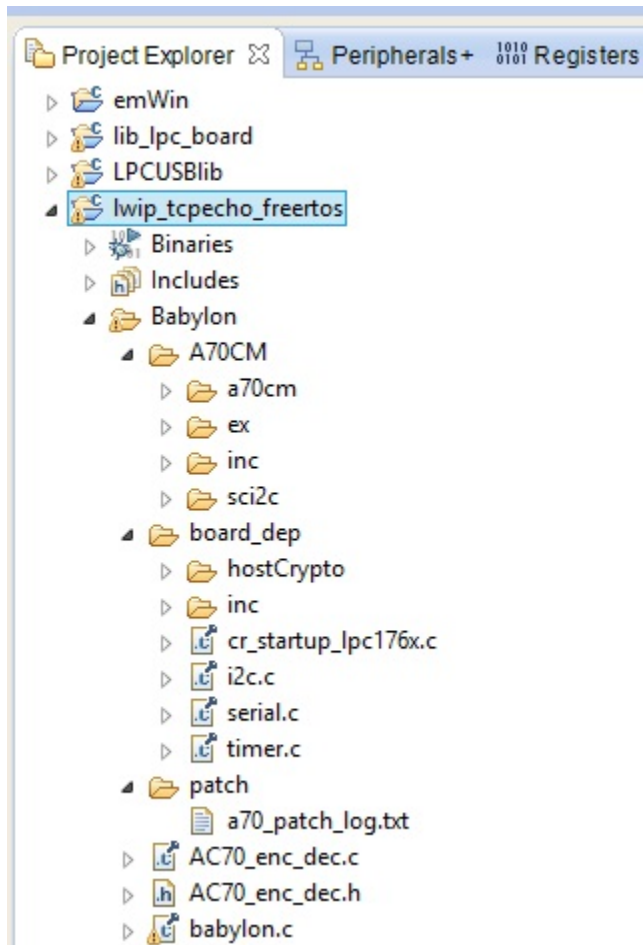
double click on file “**patch_a70.bat**” (green circled)

and follow screen instructions.

When batch process ends, open LPCXpresso, right click on “lwip_tcpecho_freertos” and select “Refresh” (sse image below)

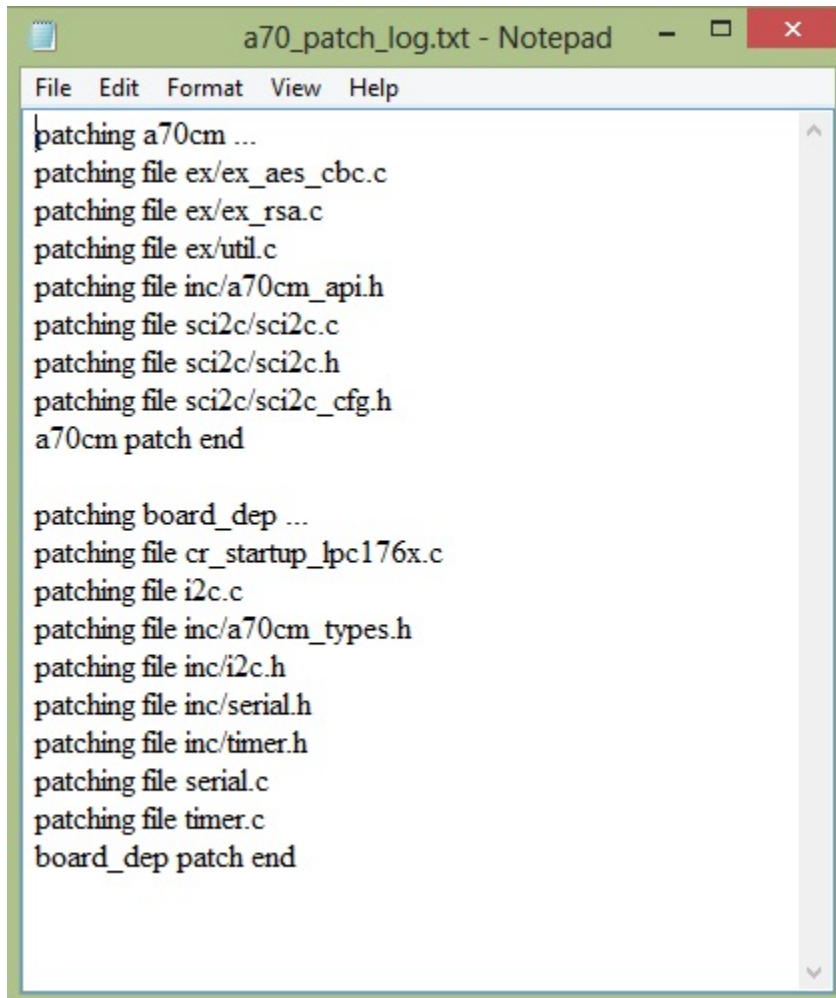


Now you can see the needed library files patched



Follow *Compile and load firmware* to compile and run babylon project

Note: you can find the result of batch process inside file “a70_patch_log.txt”



```
a70_patch_log.txt - Notepad
File Edit Format View Help
patching a70cm ...
patching file ex/ex_aes_cbc.c
patching file ex/ex_rsa.c
patching file ex/util.c
patching file inc/a70cm_api.h
patching file sci2c/sci2c.c
patching file sci2c/sci2c.h
patching file sci2c/sci2c_cfg.h
a70cm patch end

patching board_dep ...
patching file cr_startup_lpc176x.c
patching file i2c.c
patching file inc/a70cm_types.h
patching file inc/i2c.h
patching file inc/serial.h
patching file inc/timer.h
patching file serial.c
patching file timer.c
board_dep patch end
```

Firmware specification

Firmware overview

The Babylon A70CM demo firmware has been developed using **A70CM Host API library** and **SCI2C library** and performs RSA and AES-CBC encryption/decryption functions

It is designed only for evaluation purposes and to demonstrate how to use A70CM API functions.

End-user applications can use these examples to perform encryption/decryption functions, but it's mandatory that customers follow standard specifications to improve security data exchange.

Some of these specification can be found, for example, at **NIST** National Institute of Standards and Technology, section Computer Security Resource Center [CSRC](#)

Firmware restrictions

The Babylon A70CM demo firmware assumes the following restrictions:

- Authentication Process assumes both Client than Server has same RSA certificate inside. No Certificate exchange is performed

- The demo program use default NXP Certificate for Autentication Process
- Authentication process is started from Client and uses a simple AES KeyA-KeyB exchange check.
- The AES session Key is exchanged after authentication success.
- AES key is generated using secure wrapping algoritm RFC3394
- The user's text messages are sent from Server to Client (max lenght 64 char)
- All messages are 128 bytes fixed-length
- Signed user's messages uses SHA1 digital signature, stored inside of last 20 bytes of each message (from position 109 to position 128). (*more details at paragraph 19.2.3.1 - Secret Key Electronic Signatures on [this link](#))*
- To change AES session key, you must restart Authentication Process.
- AES key generation function is ONLY FOR DEBUG. Do not use for end-user application
- LAN mode has one board as Server and one board as Client, 3 terminal instance are needed

Main firmware features

By using two PC terminal software connected to Babylon board (see [Hardware setup](#)), you can perform:

- Main settings of application parameters
- Configure Man-In-The-Middle performance
- Launch Athentication Process
- Write plain custom messages (maximun 64 character) and send its to client
- Check results of sending

Operation mode macros

See at file “task_ac70.h” in project folder “\A70CM\inc” You can find these macros

```
#define AC70_TASK 0    //enable original main_task for A70CM Host Api demo application

#if AC70_TASK          //if Host api demo, no other task can be enabled
#define IP_TASK 0
#define VCOMM_TASK 0
#define BABYLON_TASK 0

#else                  //not api demo, select operation mode
#define VCOMM_TASK 1
#define BABYLON_TASK 1
#define IP_TASK 0      //IP task enable/disable

    #if IP_TASK
        #define UDP_SERVER 1
        #define UDP_CLIENT !UDP_SERVER
    #else
        #define UDP_SERVER 0
        #define UDP_CLIENT 0
    #endif
#endif

#endif
```

Macro **AC70_TASK** do not set to 1, only for A70CM API demo

Macro **IP_TASK** can be set 1 for LAN operation.

By default the firmware comes compiled in Stand-Alone operation mode (IP_TASK = 0).

To switch to LAN mode, you must set IP_TASK = 1, and then configure Server or Client compilation by setting:

UDP_SERVER 1 = Server application

or

UDP_SERVER 0 = Client application

You can also find these other macros:

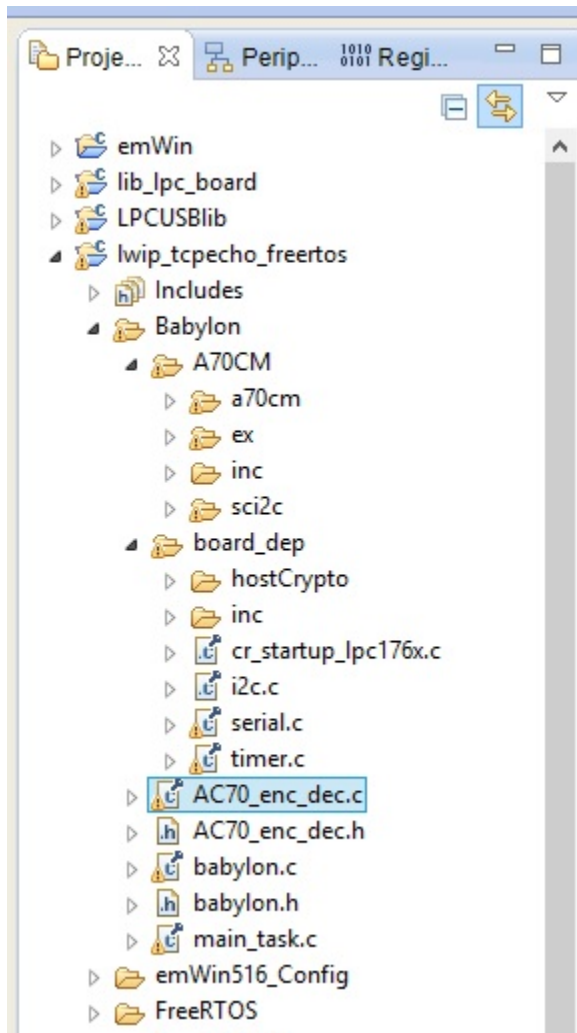
```
#define LCD_TASK 1
#define POLL_TASK 0
#define JEN_TASK 0
```

LCD_TASK enable/disable GUI interface task
POLL_TASK and JEN_TASK must be set to 0

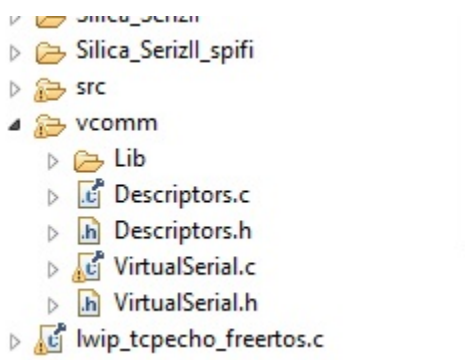
Project structure

The project “lwip_tcpecho_freertos” include the following specifying folders for A70CM application

- **Babylon – main file folder**
 - **A70CM** folder Host API and SCI2C library
 - **board_dep** folder SHA1 HostCrypto source files and other include file
 - **babylon.c** source file Main Application Tasks
 - **AC70_enc_dec.c** source file AC70 host API interface functions



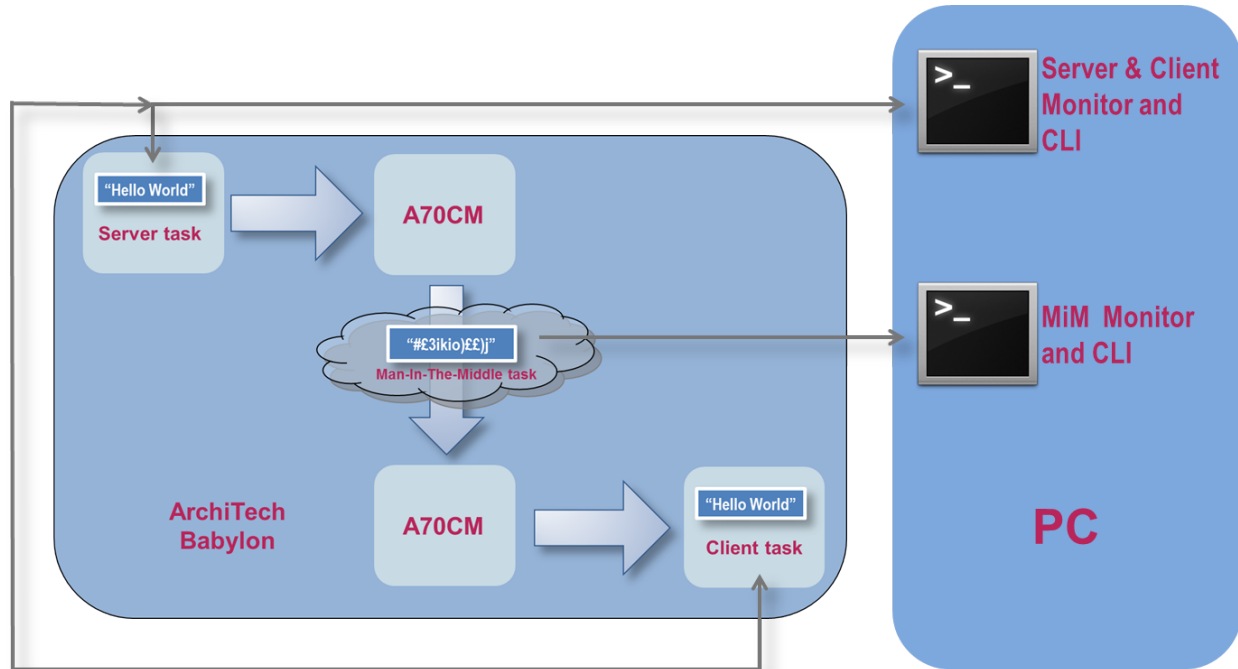
- **vcomm – Virtual Comm Port and Man-In-The-Middle file folder**
 - **VirtualSerial.c** source file Main Application Task



Brief of firmware performance

The firmware will include a minimal example how to initialize the A70CM and use Host API functions. It include also the source files of these Host API, the source files of SCI2C library and a layer to interface main application tasks with API functions.

In Stand Alone operation mode, there is only one Main Menu that will configure both Server and Client
 Also the results of all operation (Settings, Authentication, Message exchange) are shown an same terminal window
 As a demo of text messages, they are sent by Server Task to Client Task

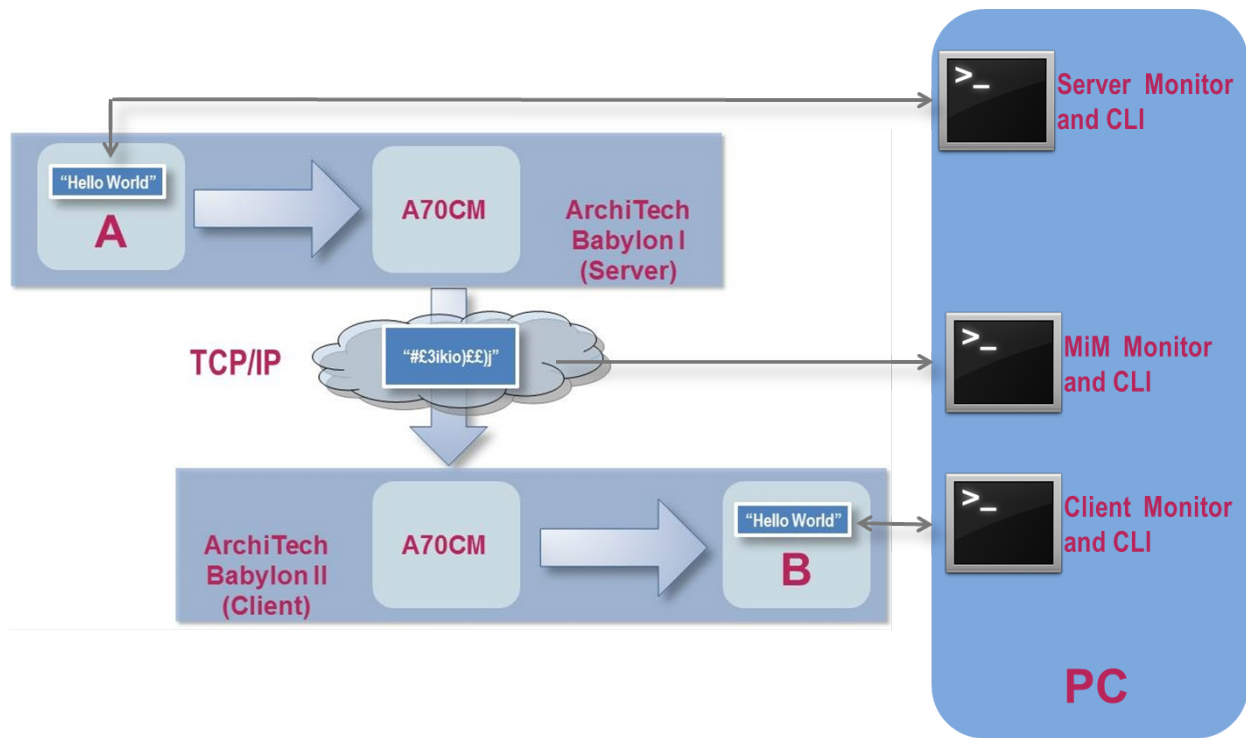


Using LAN configuration, Server and Client are on two separate boards. Each board has your own terminal connection, and show same configuration menu.

In this case, only the needed selections from Server or Client menu gives your own effects. Unneeded parameters will ignored.

The LAN operation mode is designed for getting closer to real world- Client/Server interaction using 2 boards

For mode detail about LAN mode see [Hardware setup LAN mode](#)



Project and tools download

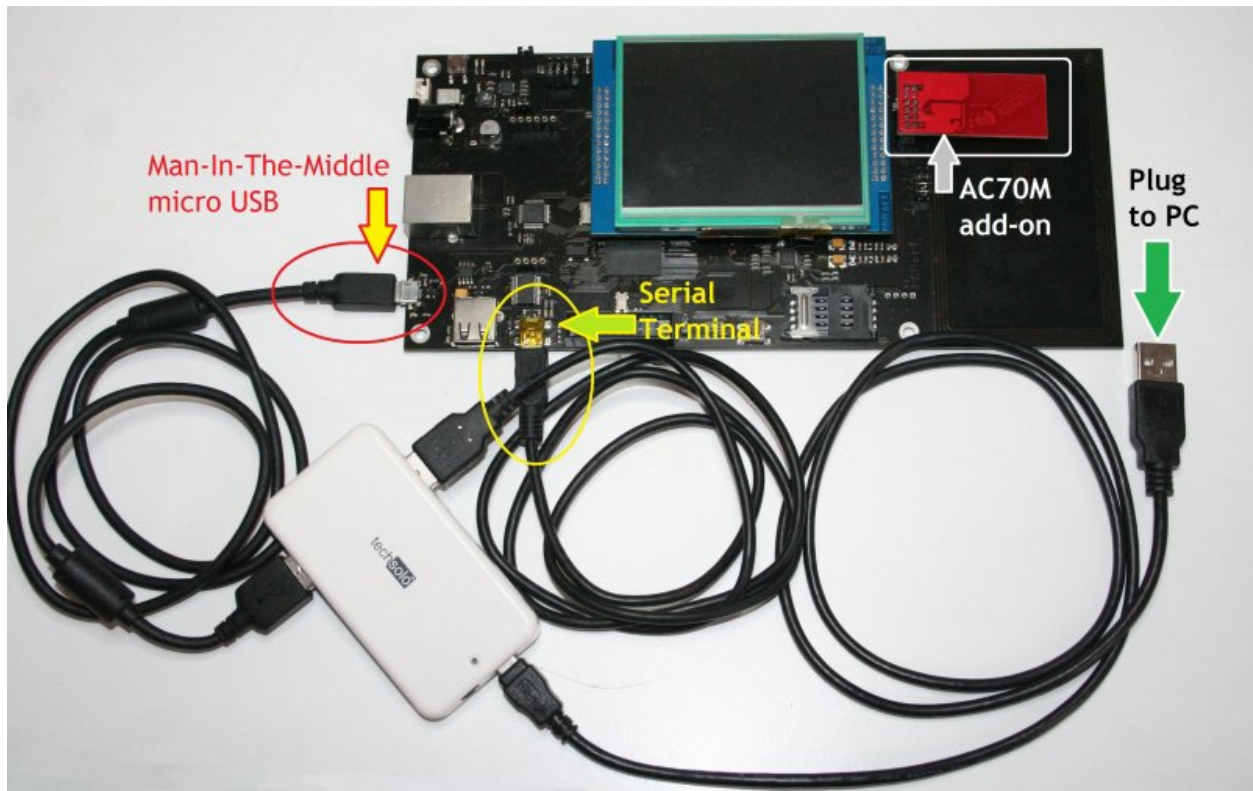
Instructions for tools and firmware installation can be found at Silica ArchiTech page under SerizII project section. Registration is needed to access at download section. Click [here](#) to go to ArchiTech main page.

Getting started Babylon A70CM

This chapter contain a brief of **Stand Alone operation mode** of Babylon application. More detail can be found in *More about firmware*

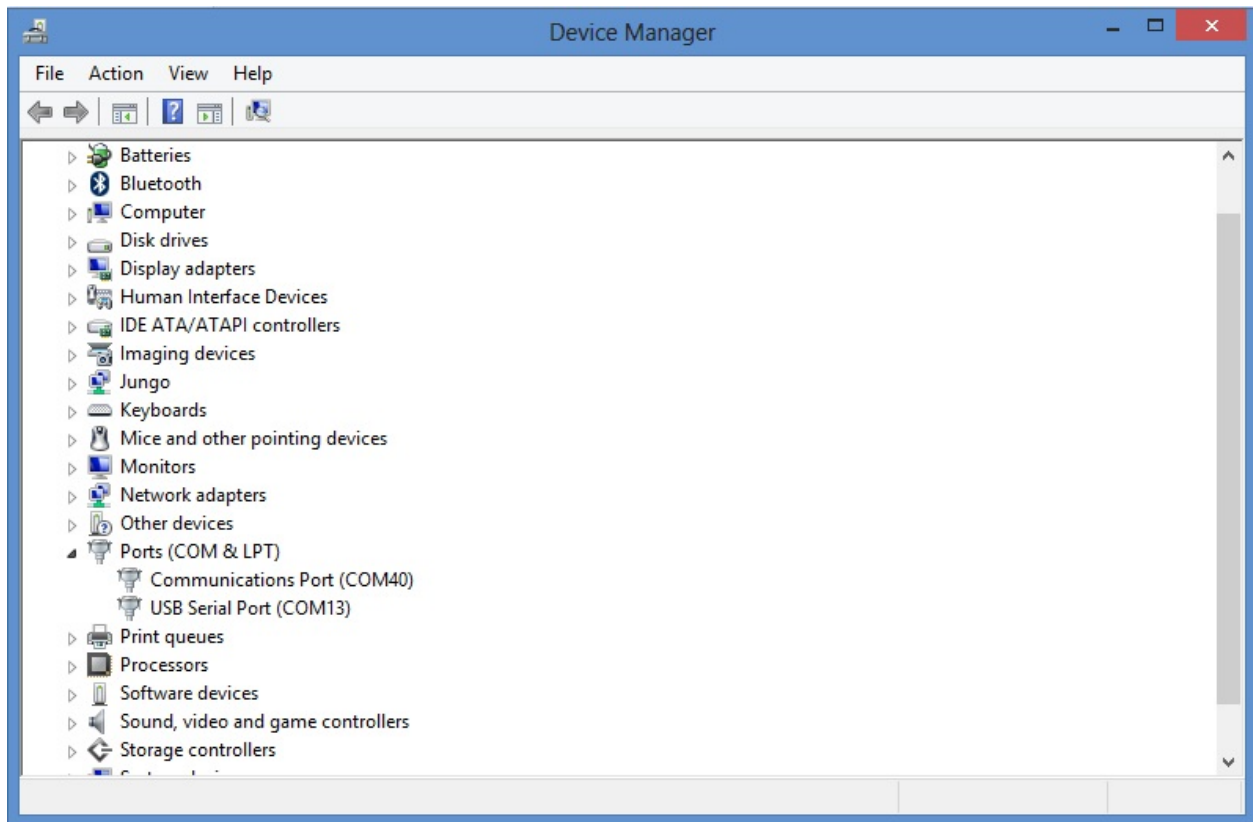
Hardware setup

Before use the application firmware, you must connect SerizII with its own Babylon board, as in figure below



When connected, you have 2 serial ports.

- one VCOMM port (need FTDI drivers) using FDTI.
- one VCOMM port (need a CDC driver - Babylon_1_0.zip file include driver folder named “USB CDC binaries”) being implemented on the USB of the LPC435X on SerizII Server



All ports work at 115200,n,8,1 no handshake.

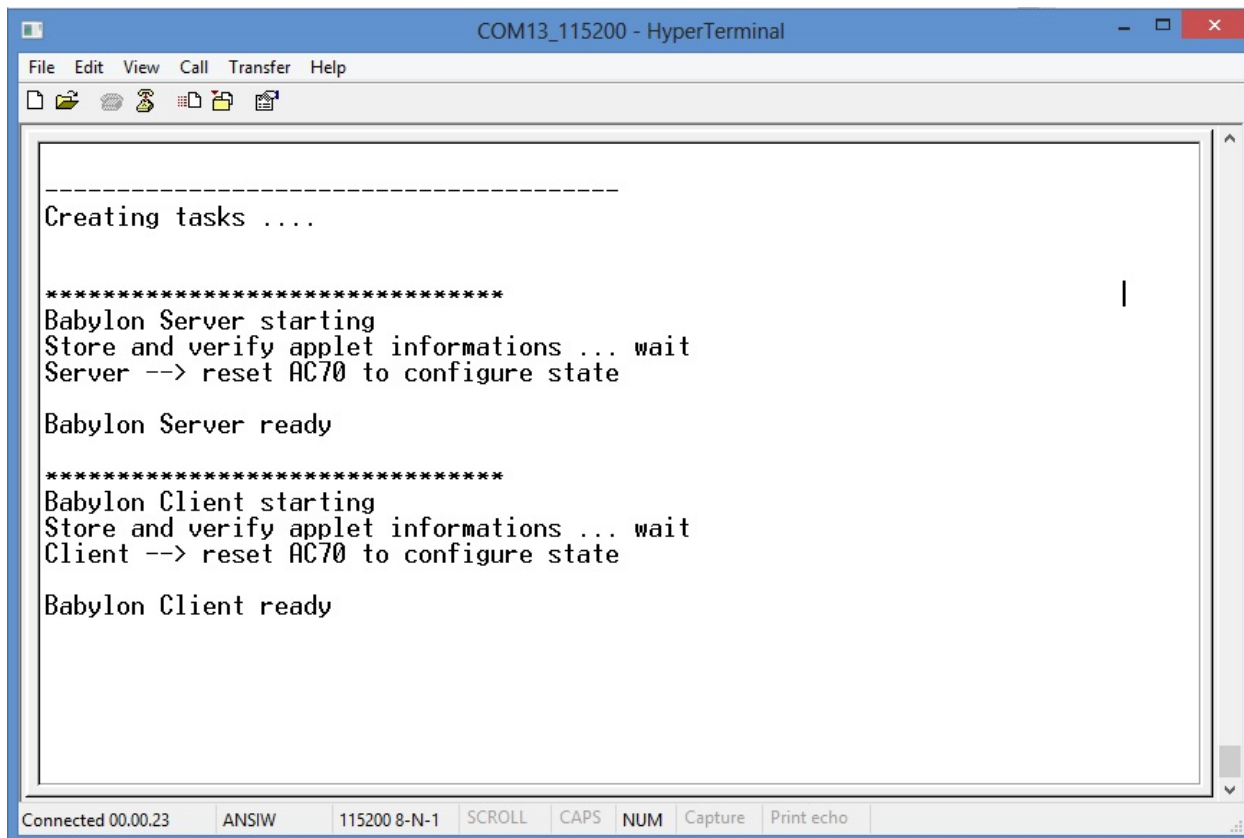
Open 1 terminal instances, connected to the MINI USB (FDTI). This is the Server and Client main menu. This VCOMM is immediate active as soon as the powered on.

When power up ends, SerizII display will show:



The VCOMM connected to the Micro USB is activated only after the Babylon has completed its initialization (this

will ends when FTDI terminal instance show the Babylon main menu, it takes about 10" after power on). Wait to open the Micro USB Vcomm port till the SerizII has completed initialization.



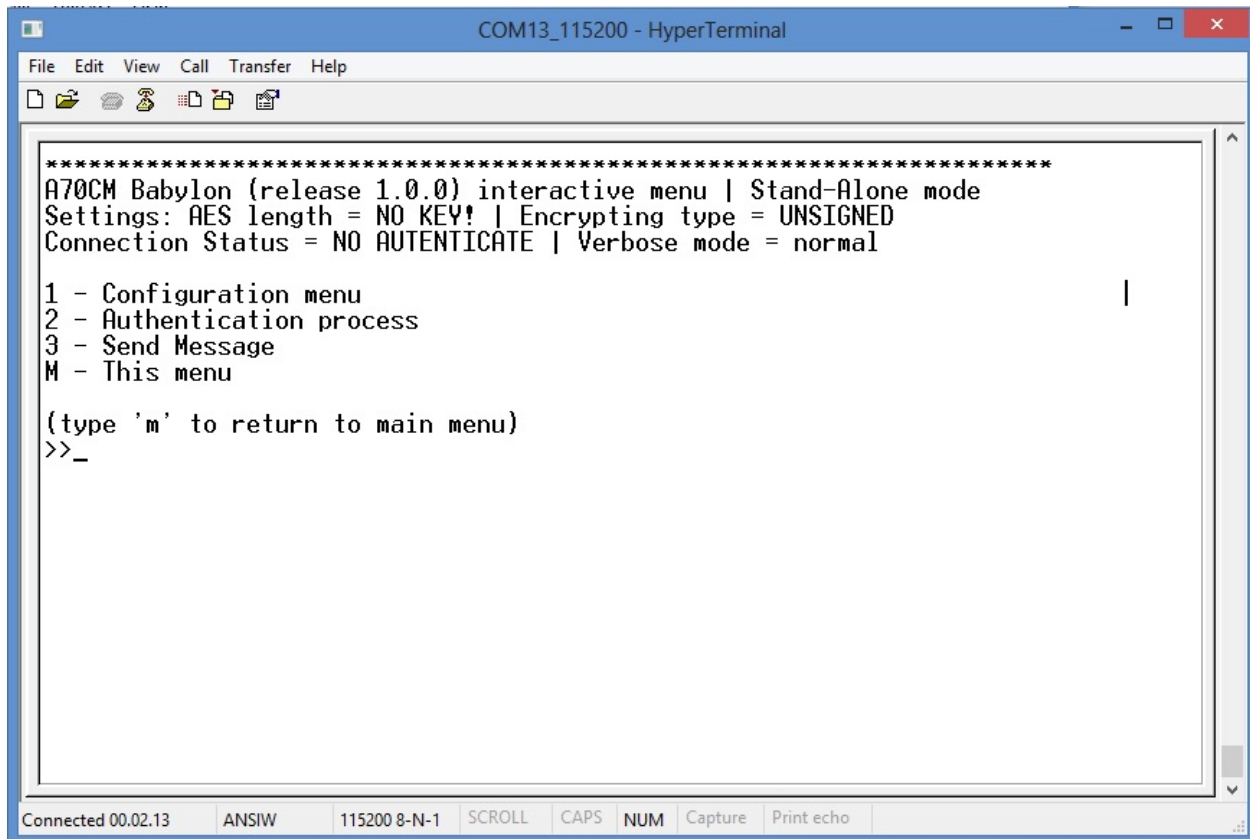
```
-----
Creating tasks ....

*****
Babylon Server starting
Store and verify applet informations ... wait
Server --> reset AC70 to configure state
Babylon Server ready

*****
Babylon Client starting
Store and verify applet informations ... wait
Client --> reset AC70 to configure state
Babylon Client ready
```

Connected 00.00.23 ANSIW 115200 8-N-1 SCROLL CAPS NUM Capture Print echo

On the instance connected to MINI USB (FTDI), after the initialization is complete, the babylon A70CM main menu will be displayed



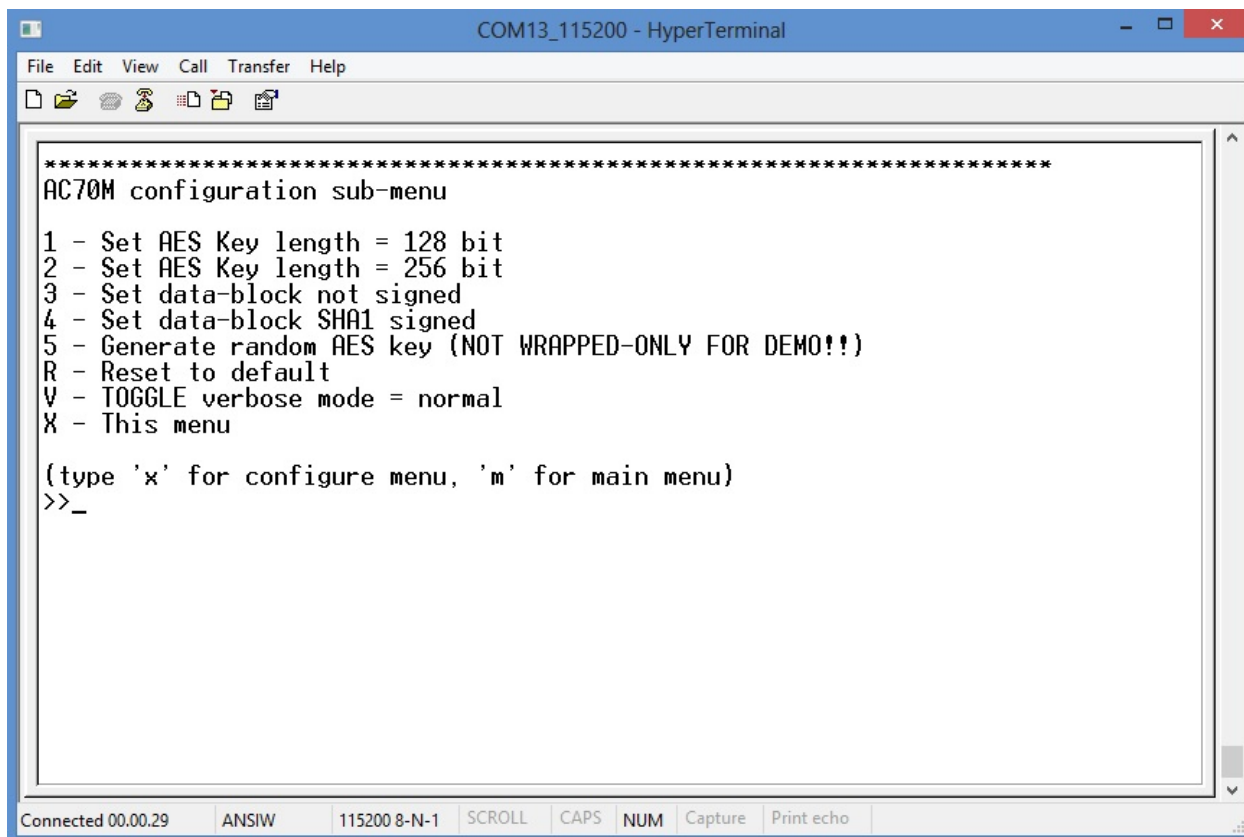
```
COM13_115200 - HyperTerminal
File Edit View Call Transfer Help
*****
A70CM Babylon (release 1.0.0) interactive menu | Stand-Alone mode
Settings: AES length = NO KEY! | Encrypting type = UNSIGNED
Connection Status = NO AUTHENTICATE | Verbose mode = normal

1 - Configuration menu
2 - Authentication process
3 - Send Message
M - This menu

(type 'm' to return to main menu)
>>_

Connected 00.02.13  ANSIW  115200 8-N-1  SCROLL  CAPS  NUM  Capture  Print echo
```

By pressing '1' you enter in the **configuration sub-menu**



```
COM13_115200 - HyperTerminal
File Edit View Call Transfer Help
*****
AC70M configuration sub-menu
1 - Set AES Key length = 128 bit
2 - Set AES Key length = 256 bit
3 - Set data-block not signed
4 - Set data-block SHA1 signed
5 - Generate random AES key (NOT WRAPPED-ONLY FOR DEMO!!)
R - Reset to default
V - TOGGLE verbose mode = normal
X - This menu

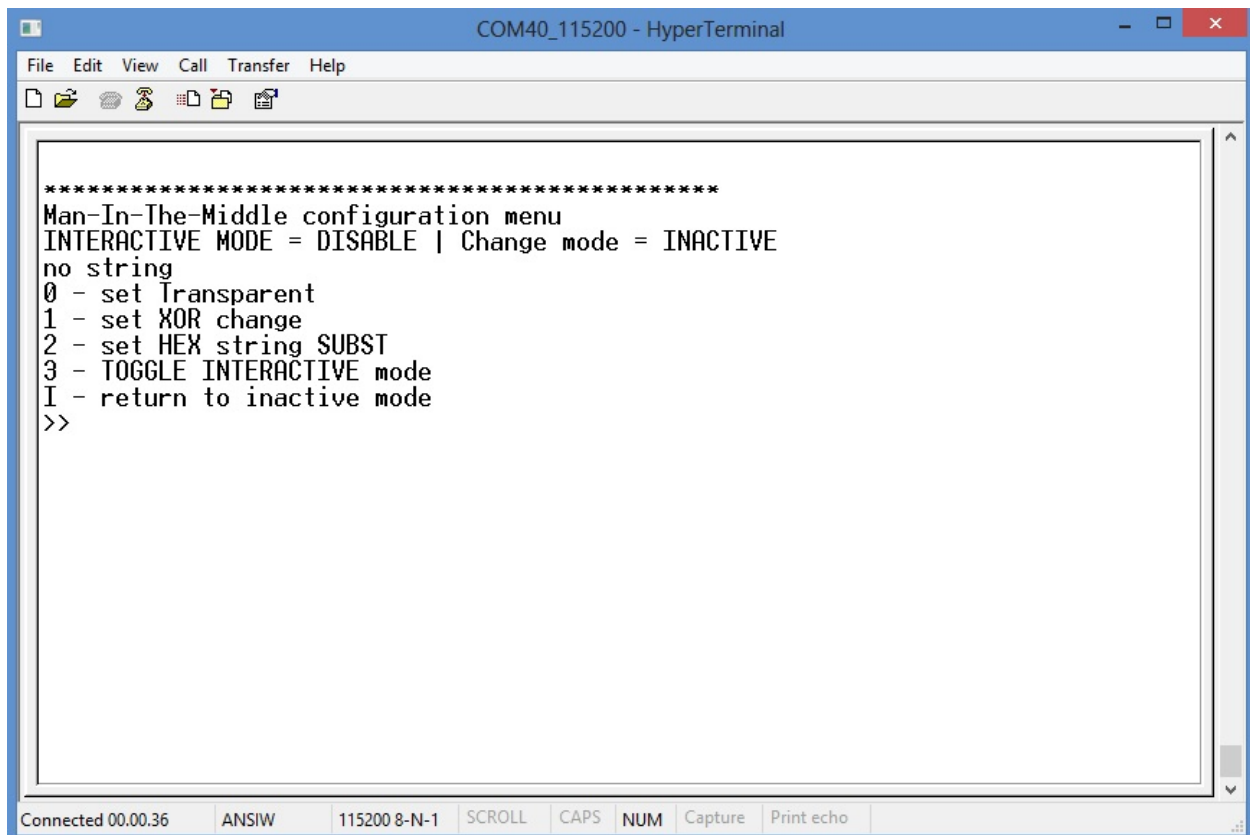
(type 'x' for configure menu, 'm' for main menu)
>>_
Connected 00.00.29  ANSIW  115200 8-N-1  SCROLL  CAPS  NUM  Capture  Print echo
```

This menu will enable:

- select the AES key length ('1' = 128, '2' = 256 bits)
- enable encryption only ('3'), or encrypted and signed ('4')
- toggle verbose mode for more details during data exchange
- reset main configuration parameter to default set
- return to main menu ('m' go to main menu)

Man-In-The-Middle menu

As soon as you open the instance connected to the “Man-In-The-Middle” (micro USB on Babylon SerizII), this will be displayed



```
*****
Man-In-The-Middle configuration menu
INTERACTIVE MODE = DISABLE | Change mode = INACTIVE
no string
0 - set Transparent
1 - set XOR change
2 - set HEX string SUBST
3 - TOGGLE INTERACTIVE mode
I - return to inactive mode
>>
```

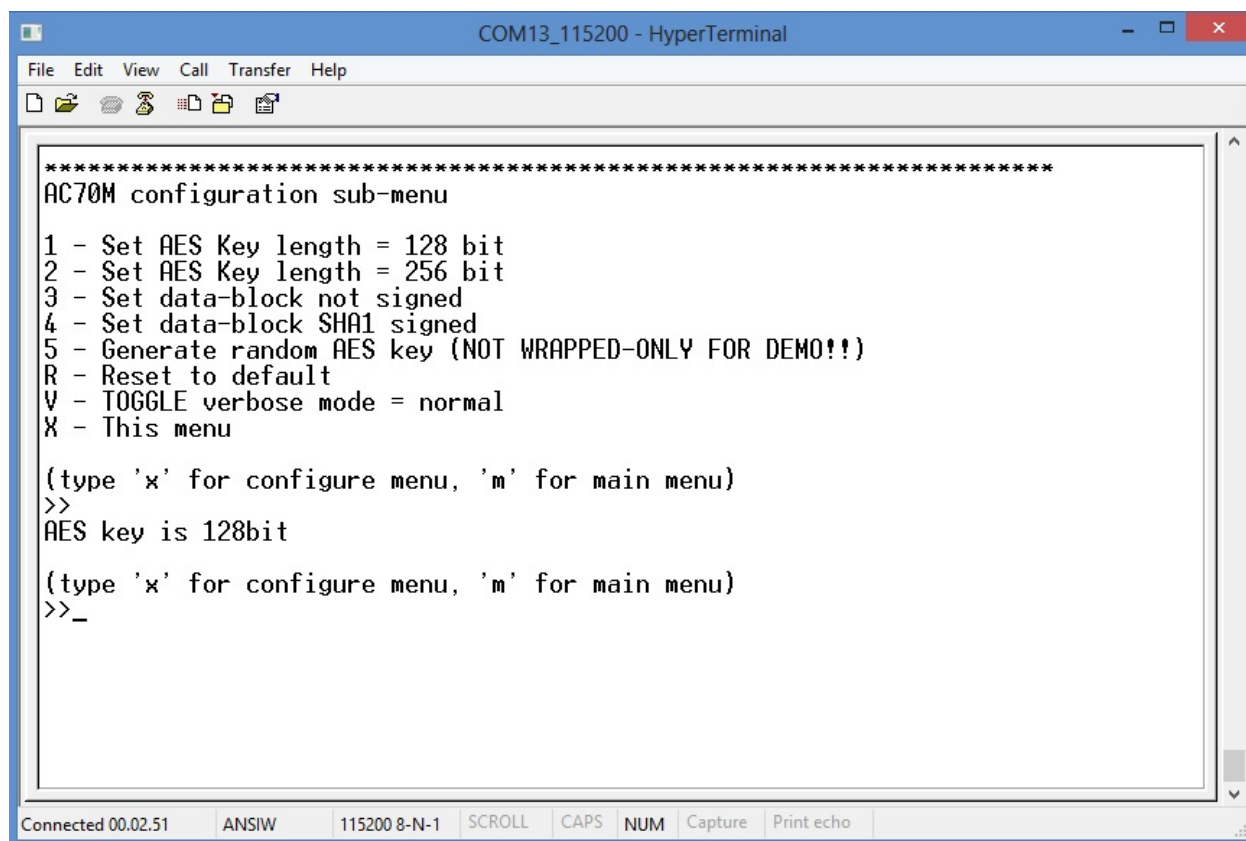
Connected 00.00.36 ANSIW 115200 8-N-1 SCROLL CAPS NUM Capture Print echo

When all the Terminal are activated, you are ready to start evaluation!!

First steps

First of all, you must set key length in configuration sub menu.

Type '1' from main menu, then select '1' for 128, '2' for 256 (see image above)



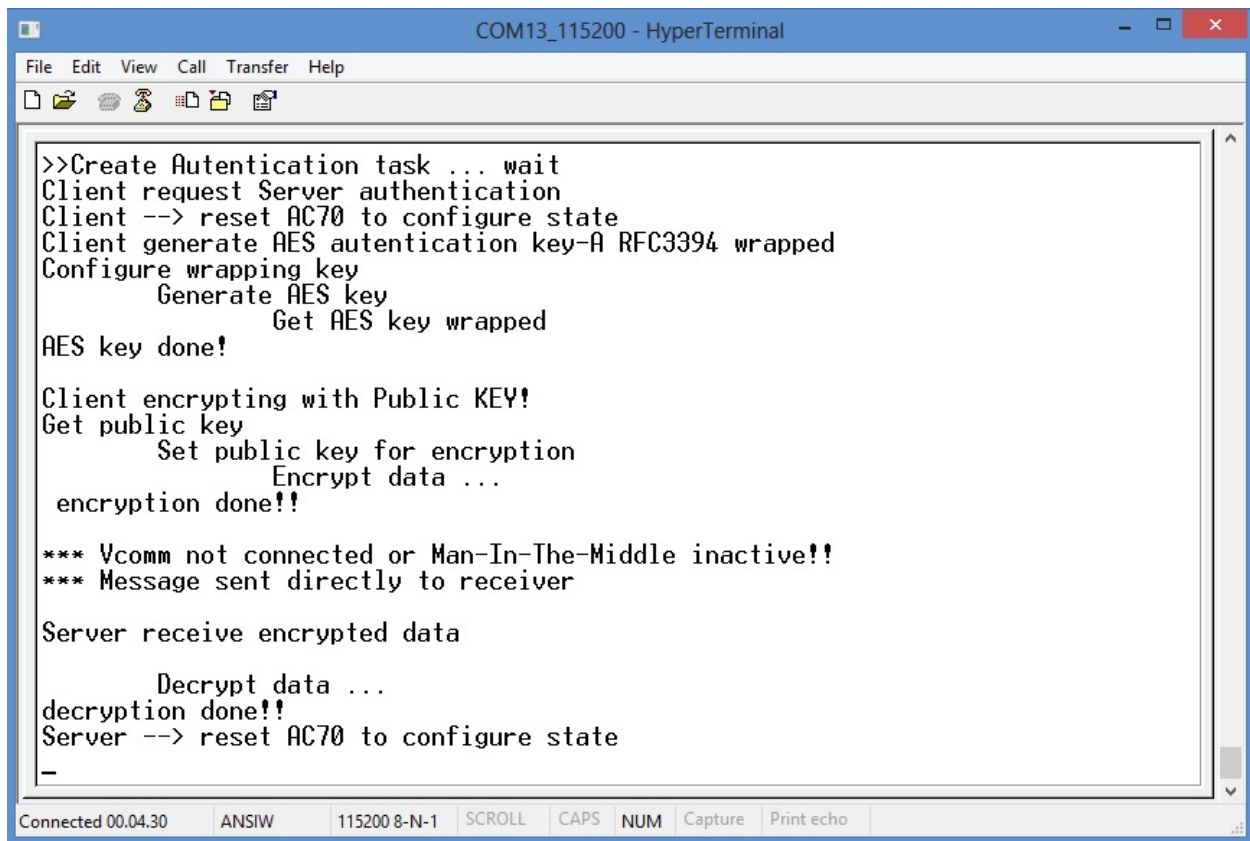
```
COM13_115200 - HyperTerminal
File Edit View Call Transfer Help
*****
AC70M configuration sub-menu
1 - Set AES Key length = 128 bit
2 - Set AES Key length = 256 bit
3 - Set data-block not signed
4 - Set data-block SHA1 signed
5 - Generate random AES key (NOT WRAPPED-ONLY FOR DEMO!!)
R - Reset to default
V - TOGGLE verbose mode = normal
X - This menu

(type 'x' for configure menu, 'm' for main menu)
>>
AES key is 128bit

(type 'x' for configure menu, 'm' for main menu)
>>_

Connected 00.02.51  ANSIW  115200 8-N-1  SCROLL  CAPS  NUM  Capture  Print echo
```

To start a work session and enable messages sending, it's mandatory to perform Authentication process. To do this, you must start the Authenticate operations. Press '2' from main menu:



```
>>Create Autentication task ... wait
Client request Server authentication
Client --> reset AC70 to configure state
Client generate AES authentication key-A RFC3394 wrapped
Configure wrapping key
    Generate AES key
    Get AES key wrapped
AES key done!

Client encrypting with Public KEY!
Get public key
    Set public key for encryption
    Encrypt data ...
encryption done!!

*** Vcomm not connected or Man-In-The-Middle inactive!!
*** Message sent directly to receiver

Server receive encrypted data
    Decrypt data ...
decryption done!!
Server --> reset AC70 to configure state
-
```

Connected 00.04.30 ANSIW 115200 8-N-1 SCROLL CAPS NUM Capture Print echo

when Authentication ends

```

COM13_115200 - HyperTerminal
File Edit View Call Transfer Help
*** Message sent directly to receiver
Client receive encrypted data
    Decrypt data ...
decryption done!!
Session AES key received from server, set as Client session key
Get public key
    Set public key for encryption
    Encrypt data ...
encryption done!!
Client encrypt AES session key and resent it to server for feedback
*** Vcomm not connected or Man-In-The-Middle inactive!!
*** Message sent directly to receiver
Server receive encrypted data
    Decrypt data ...
decryption done!!
Server receive from client AES session key feedbaack. Check it!
Client and Server authentication success! Go to connected state!
End authentication tasks, restart other tasks!!
Connected 00.06.28  ANSIW  115200 8-N-1  SCROLL  CAPS  NUM  Capture  Print echo

```

- a few messages are exchanged between the Server and Client tasks, and the messages are shown on the Babylon terminal as soon as the messages arrives and start from each task.
- by default the “Man-In-The-Middle” is INACTIVE; you can see the status of each actor (Babylon and “Man-In-The-Middle”) on the top of the menu display.
- here below an example of Babylon state when Authentication success

```

*****
A70CM Babylon (release 1.0.0) interactive menu | Stand-Alone mode
Settings: AES length = 128 | Encrypting type = UNSIGNED
Connection Status = AUTHENTICATED and CONNECTED | Verbose mode = normal

1 - Configuration menu
2 - Authentication process
3 - Send Message
M - This menu

(type 'm' to return to main menu)
>>_

```

When INACTIVE, no messages are sent to “Man-In-The-Middle” task. It will be ignored.

- The “Man-in-the-middle” go to the ACTIVE state when you select how it works, as:
 - transparent, doing nothing on the messages.
 - interactive change mode: when a message is received, the “Man-in-the-middle”, the message is stored in the “Man-In-The-Middle” buffer waiting action made using the menu.
 - automatic change mode: when a message is received by the “Man-in-the-middle” it is changes based on the change type selected.
- When the “Change” mode is active (either interactive or automatic), the message is modified based on the selection made
 - 0, set no change
 - 1, change by XORing some characters with a specified character string
 - 2, change by changing some characters with a specified character string
 - When in “automatic mode” selected, any incoming message is changed and sent to the destination
 - When in “interactive mode” (key ‘3’ toggle this mode), the message is stored, waiting for the “Man-In-The-Middle” operator to decide what to do with the message, The message is sent when the action is defined, and “CR” is pressed.

Sendig messages

Available only when Babylon is in “Authenticated and connected state”. See image below

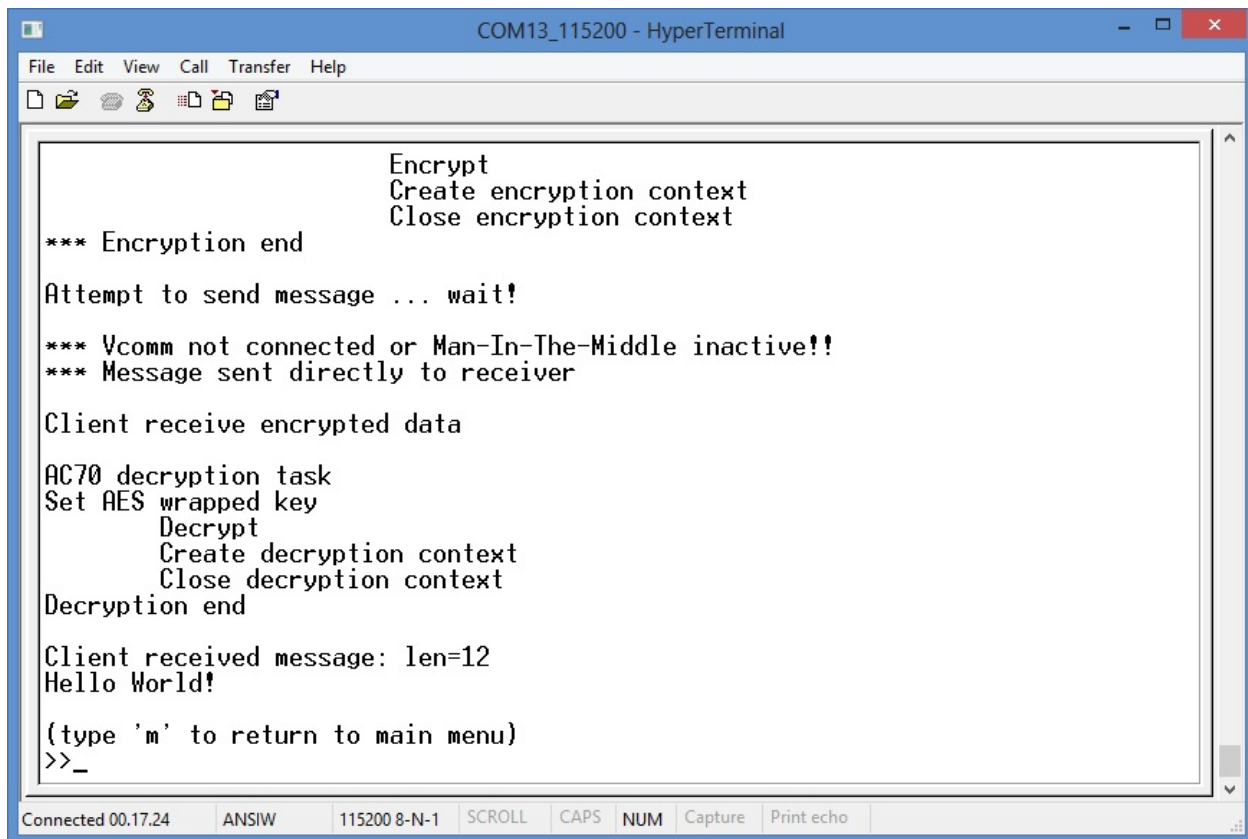
```
*****
A70CM Babylon (release 1.0.0) interactive menu
Settings: AES length = 128 | Encrypting type =
Connection Status = AUTHENTICATED and CONNECTED
```

The demo can start by leaving the “Man-In-The-Middle” inactive (default after reset). After authentication process ends successfully, by selecting to send a message:

- select ‘3’ on the “Babylon main menu”
- write an ascii string
- when the “return” key is pressed, the message is encrypted and sent by Babylon Server task; received, decrypted by Babylon Client task and shown on the Babylon terminal.

```
*****
AC70M send string Server-->Client
Type string (MAX 64 chars) and hit ENTER when finish
String = Hello World!
Server sending message <Hello World!>, length=12
Server encrypting message <Hello World!> length=12 mode=0
AC70 encryption task
Set AES wrapped key
Encrypt
Create encryption context
```

Connected 00.16.16 ANSIW 115200 8-N-1 SCROLL CAPS NUM Capture Print echo



```
COM13_115200 - HyperTerminal
File Edit View Call Transfer Help
[Icons]
Encrypt
Create encryption context
Close encryption context
*** Encryption end
Attempt to send message ... wait!
*** Vcomm not connected or Man-In-The-Middle inactive!!
*** Message sent directly to receiver
Client receive encrypted data
AC70 decryption task
Set AES wrapped key
    Decrypt
    Create decryption context
    Close decryption context
Decryption end
Client received message: len=12
Hello World!
(type 'm' to return to main menu)
>>_
Connected 00.17.24  ANSIW  115200 8-N-1  SCROLL  CAPS  NUM  Capture  Print echo
```

Then, the “Man-In-The-Middle” can be activated, pressing ‘3’ on its menu.

When a message is sent from Babylon Server task, the message is now received and shown on the “Man-In-The-Middle” terminal.

Pressing Return, the received message is sent with no modification to Babylon Client task and shown on the Babylon terminal.

The message can also be changed by the “Man-In-The-Middle” and the result is shown on the Babylon terminal.

```

COM13_115200 - HyperTerminal
File Edit View Call Transfer Help
*****
AC70M send string Server-->Client
Type string (MAX 64 chars) and hit ENTER when finish
String = Hello World!
Server sending message <Hello World!>, length=12
Server encrypting message <Hello World!> length=12 mode=0
AC70 encryption task
    Set AES wrapped key
    Encrypt
    Create encryption context
    Close encryption context
*** Encryption end
Attempt to send message ... wait!
MAN-IN-THE-MIDDLE intercepting!!
INTERACTIVE mode! Man-In-The-Middle waiting setting!

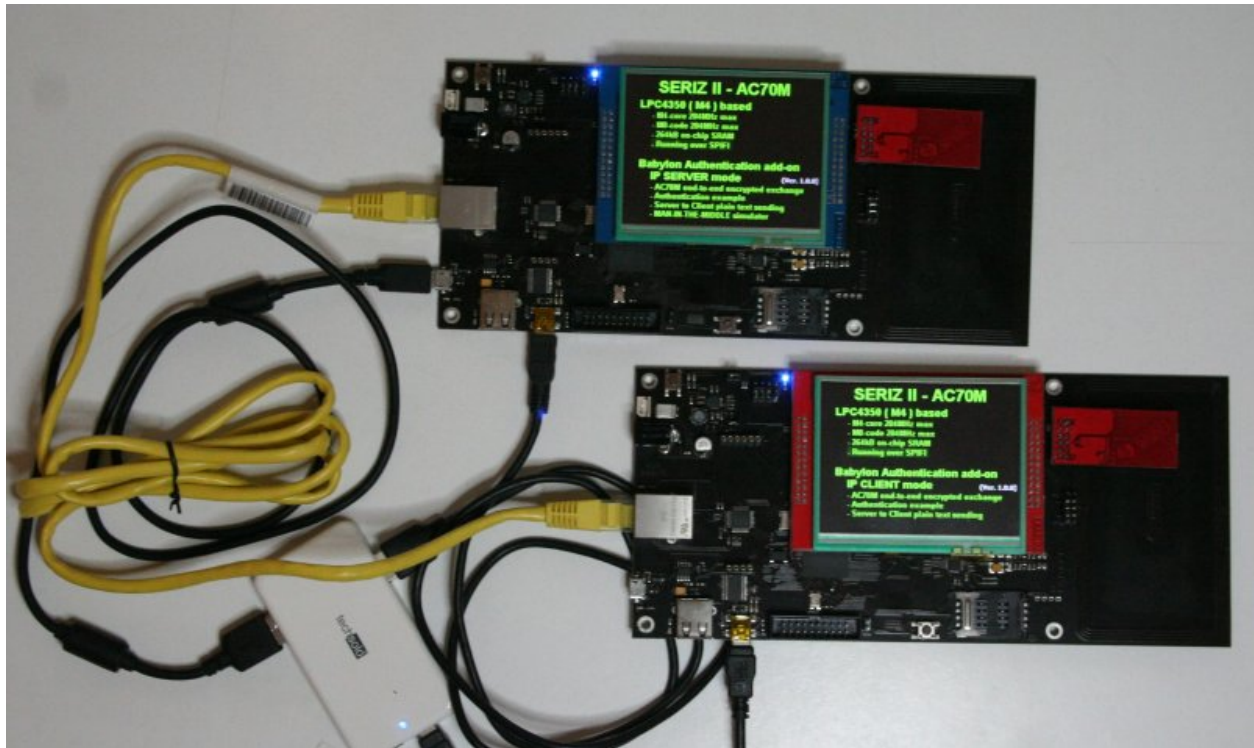
COM40_115200 - HyperTerminal
File Edit View Call Transfer Help
0 - set Transparent
1 - set XOR change
2 - set HEX string SUBST
3 - TOGGLE INTERACTIVE mode
I - return to inactive mode
>>Man-In-The-Middle intercept message:
0x71 0xb8 0x13 0xfb 0xa7 0xcf 0xa5 0x2f 0x54 0x76 0x6d 0x07 0x90 0xa7 0x3b 0xb2
0x28 0xcb 0xfb 0xe6 0xd7 0x26 0x87 0x3e 0xc7 0x8b 0x76 0xdd 0xf2 0xe2 0x8c 0x6b
0x45 0x16 0x64 0x48 0xd7 0x69 0x4b 0xe4 0x99 0xa6 0x45 0x2e 0x27 0xaa 0x6f 0x18
0xe5 0xc7 0x57 0xcb 0x7c 0x26 0xa9 0x81 0x14 0x96 0x48 0xa9 0xd0 0x04 0x0b 0x67
0x20 0x5c 0x3c 0xf1 0x97 0x48 0xef 0x4f 0x1e 0x11 0xb2 0xdf 0xf4 0xcb 0xc2 0x0e
0x2e 0x8e 0xe1 0xde 0x88 0xf8 0x89 0x71 0x9d 0x4c 0xd5 0x7a 0xf1 0xaa 0x80 0xc8
0x9d 0xab 0x0c 0x00 0x93 0x34 0xa2 0xae 0x76 0x23 0x74 0xb9 0x83 0x3d 0x48 0x4f
0x3f 0x5f 0xfd 0x2b 0xb8 0x87 0x3b 0x0c 0xb6 0x06 0xc4 0x6e 0xc9 0xc8 0x27 0x0c
*****
Man-In-The-Middle configuration menu
INTERACTIVE MODE = ENABLE | Change mode = TRANSPARENT
no string
0 - set Transparent
1 - set XOR change
2 - set HEX string SUBST
3 - TOGGLE INTERACTIVE mode
I - return to inactive mode
>> ← Waiting for input ...

```

Hardware setup LAN mode

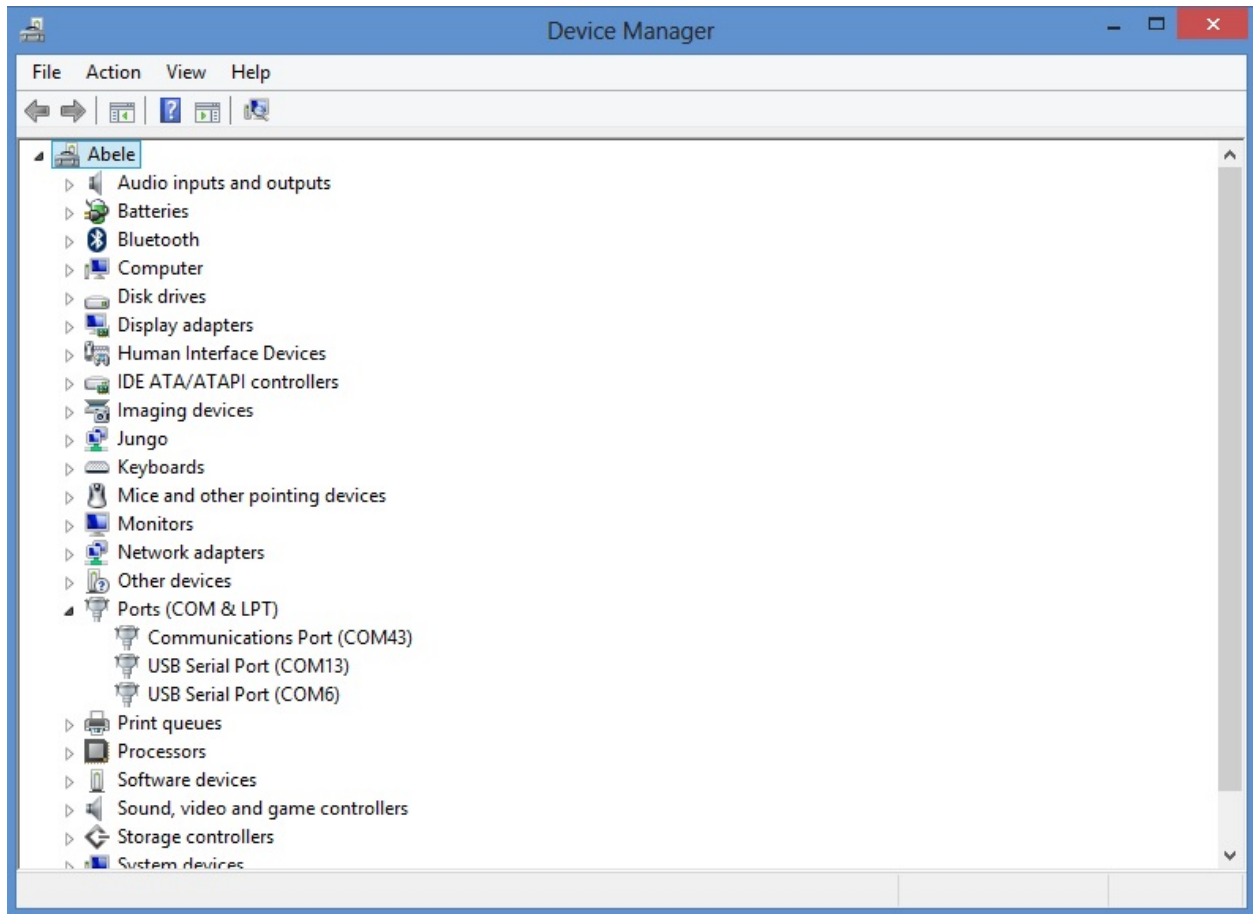
All functionality described in this paragraph can be performed using Babylon firmware compiled for LAN mode. Because Server and Client are on two separate board, each one has to be programmed compiling firmware with its own defines *Operation mode macros*

Before use the application firmware, you must connect 2 SerizII, each with its own Babylon board, as in figure below



When connected, you have 3 serial ports.

- two VCOMM port (need FTDI drivers) using FDTI, one from each SerizII.
- one VCOMM port (need a CDC driver) being implemented on the USB of the LPC435X on SerizII Server



The serizII which will show on the display **"Babylon IP client"** has only a single USB VCOMM connected to the mini USB (this port will use FTDI driver) and show a Client menu interface;



The serizII which will show on the display “Babylon IP server” has two USB connected, the Mini port is the Server menu interface (this port will use FTDI driver), and a micro USB which is connected to the “Man-in-the-middle” process (this port will need a CDC driver);

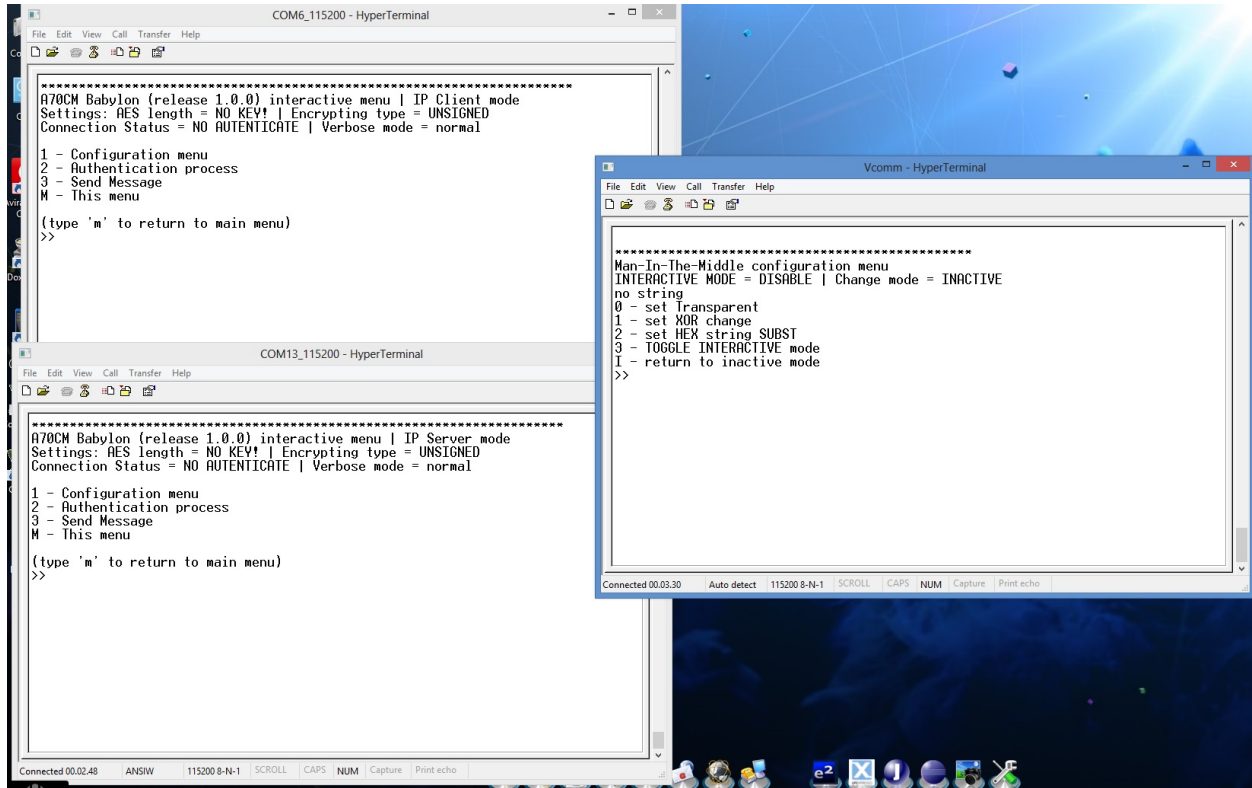


All ports work at 115200,n,8,1 no handshake.

Open 2 terminal instances, one for each VCOMM connected to the MINI USB (FTDI). There VCOMM are immediate active as soon as the powered on.

The VCOMM connected to the Micro USB is activated only after the Babylon Server SerizII has completed its initialization; wait to open this port till the SerizII has completed initialization (about 10" after power on).

On both instance connected to the IP Server and IP Client, after the initialization is complete, the A70CM main menu will be displayed



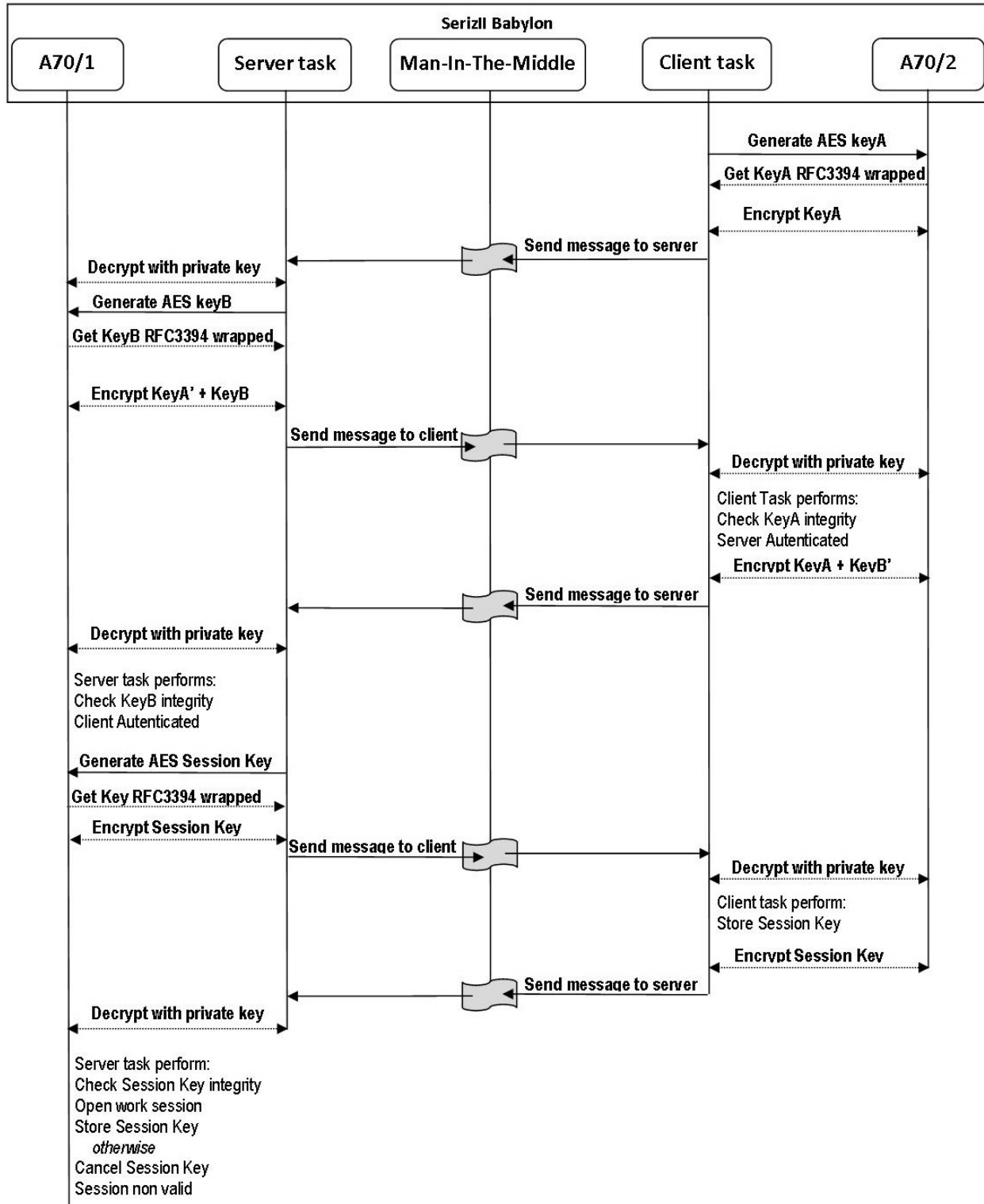
More about firmware

In this chapter you can find a flow of main functionality of Babylon A70CM firmware. The description is referred to Stand-Alone operation mode.

For LAN mode, remember that Server and Client are two separate boards

ATHENTICATION PROCESS

First of all, you must open a work session by performing Authentication Process (*First steps*)



ALL ENCRYPTION ARE MADE BY RSA PUBLIC KEY

Important:

The “gray flag” means Man-In-The-Middle activity:

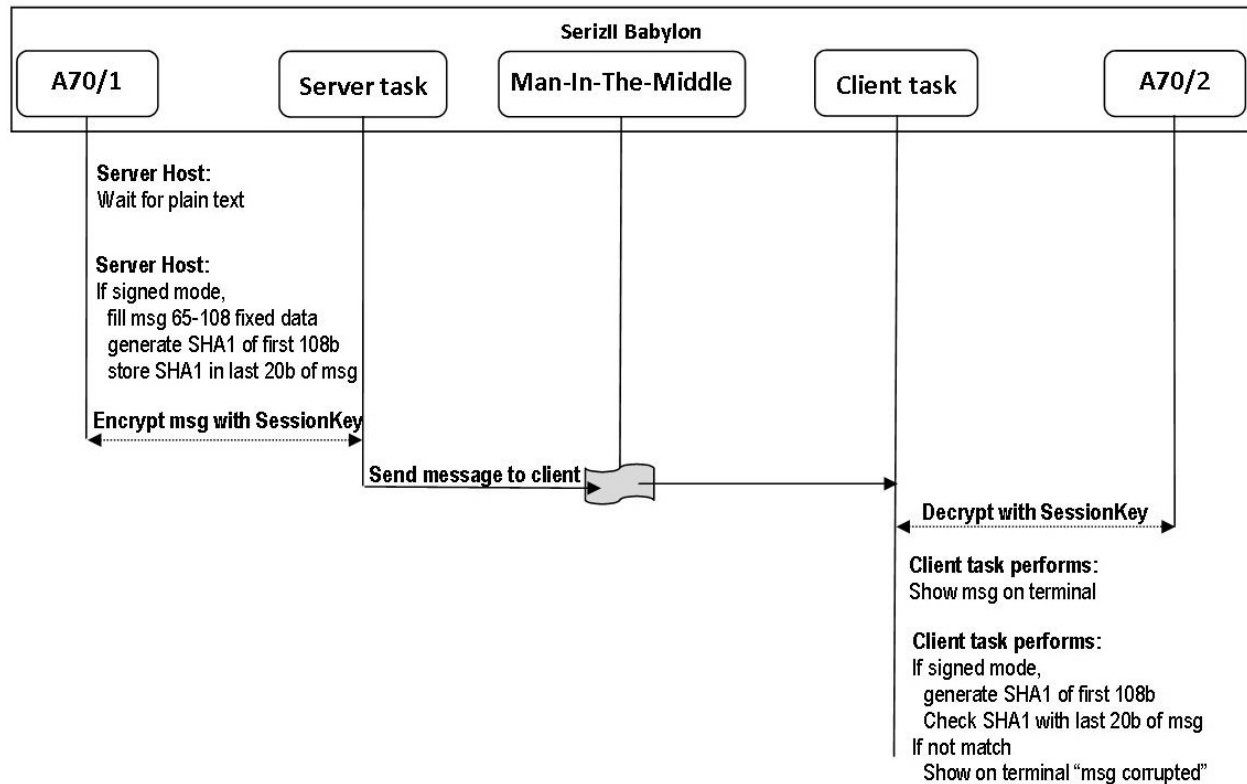
- if active, it intercept message and perform actions as your settings
- if inactive, it is skipped and messages comes out directly

Step by step description of authentication process

1. Client task generate one authentication KeyA wrapped with RFC3394 algo
2. Client task encrypt the KeyA with your RSA Public key, then send to Server
 - .. 2.1 *If Man-In-The-Middle is active, the intercept message and perform actions as your settings*
3. Server task receive the message, decrypt it, generate a second AES session KeyB also wrapped
4. Server task attach KeyB to the incoming message, encrypt with your RSA Public key, then send to Client task
 - .. 4.1 *If Man-In-The-Middle is active, the intercept message and perform actions as your settings*
5. Client task decrypt incoming message, check KeyA integrity; if ok, **Server is authenticated.**
6. Client task encrypt message as is with your RSA Public key, then feedback to Server task
 - .. 6.1 *If Man-In-The-Middle is active, the intercept message and perform actions as your settings*
7. Server task receive the message, decrypt it, check KeyB integrity; if OK, **Client is authenticated**
8. Server task generate a new AES key (session key), encrypt with your RSA Public key and send to Client task
 - .. 8.1 *If Man-In-The-Middle is active, the intercept message and perform actions as your settings*
9. Client task decrypt incoming message, store session Key as received from Server task
10. Client task encrypt session Key as is with your RSA Public key, then feedback to Server task
 - .. 10.1 *If Man-In-The-Middle is active, the intercept message and perform actions as your settings*
11. Server task decrypt incoming message, check session key integrity; **if OK, a valid work session is opened.**

SENDING/RECEIVING MESSAGES

After typing '3' on the main menu of Server, sending mode is entered

**Important:**

The “gray flag” means Man-In-The-Middle activity:

- if active, it intercept message and perform actions as your settings
- if inactive, it is skipped and messages comes out directly

Step by step description of sending messages

1. Server task wait for plain text input(max length 64)
2. if signed, Server task generate SHA1 signature and store in last 20 bytes of message
3. Server task encrypt message with AES session key.
- .. 3.1 If Man-In-The-Middle is active, the intercept message and perform actions as your settings
4. Client task decrypt incoming message with AES session key
5. if signed, Client task extract last 20 bytes of message, then generate SHA1 signature of remaining bytes
6. Client task show decrypted message on terminal.
7. if signed, Client task compare your own SHA1 signature with that just received from Server task
8. If signature don't match, Client task show on terminal “message corrupted”

Note:

- It assumes that both AC70 are same RSA certificate inside (the demo will use default NXP)
- All messages are 128 byte fixed length

- Plain text is stored on first 64 bytes of messages. All unused bytes are filled with fixed pattern (except for last 20 if signed message)
 - KeyA, KeyB, Session Key are stored at fixed position of the message. Both server and client know the relative position.
-

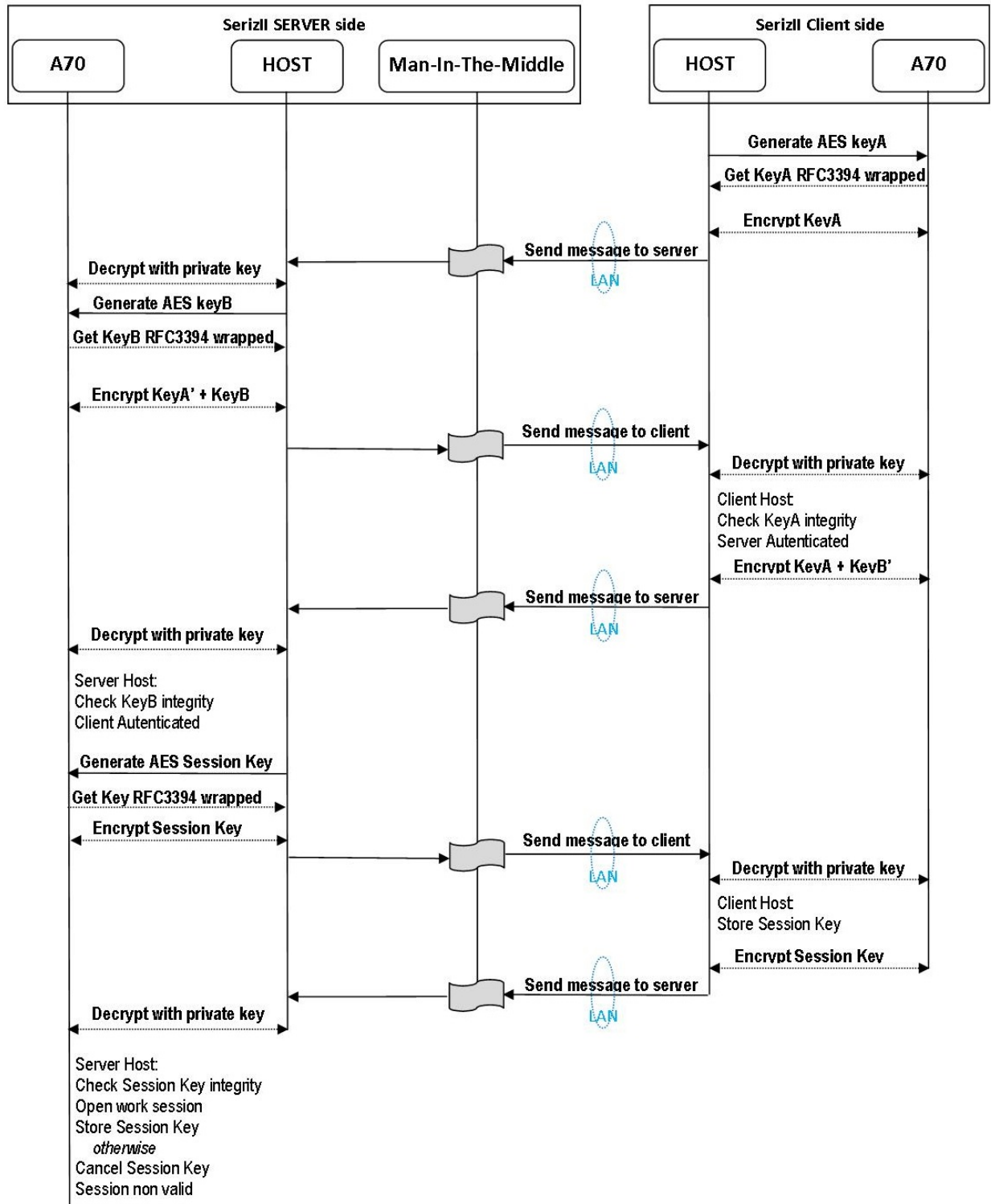
WORKING WITH LAN

LAN operation mode of firmware has Server and Client on two separate SerizII board. All messages (even authentication than user's text) are exchanged using UDP. For correct setup, see [Hardware setup LAN mode](#)

LAN AUTHENTICATION

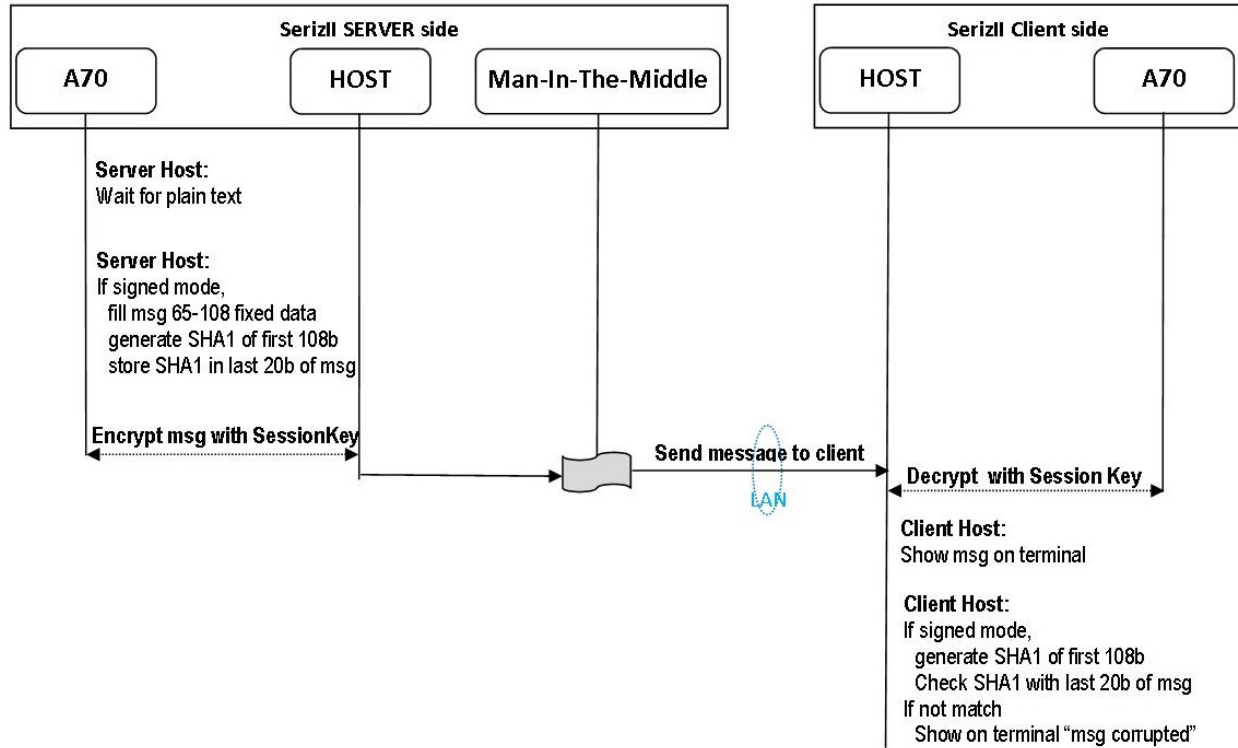
Lan Authentication will start automatically from Babylon IP CLIENT, by typing '2' from main menu. Before this, you must set key length in Client configuration sub menu. Type '1' from main menu, then select '1' for 128, '2' for 256

After this, to start a work session, both Server and Client must be in the Authenticate state.



LAN MESSAGE

User's plain text messages are exchanged using this flow To send plain text, type '3' from Babylon IP SERVER.



F

flow, [49](#)

I

index, [3](#)

P

patch, [20](#)

PrjFiles, [27](#)

Q

qs, [5](#)

U

using, [32](#)