UART to UART Wireless Communication

ABSTRACT

This document explains the required steps to establish a simple wireless communication between two Microcontrollers (MCUs) using two Y-Lynx YLX-TRM8053-xxx-05 wireless modules via their UART interface. This application note covers the requested hardware configuration and the main three communications Host-Modem, Modem-Modem and Modem-Host via UART.

Key Words

- UART to UART
- MCU to Modem communication
- Modem to MCU communication
- Communication troubleshooting

Key Commands

- BEACON MODE
- CMD_GET_LAST_BEACON_INFO
- RF ADDRESS / RF DEST ADDRESS
- SEND DATA / SEND ECHO DATA
- RECEIVED DATA

Introduction

The explanations consider an embedded application composed of two identical RF enabled systems System1 and System2.

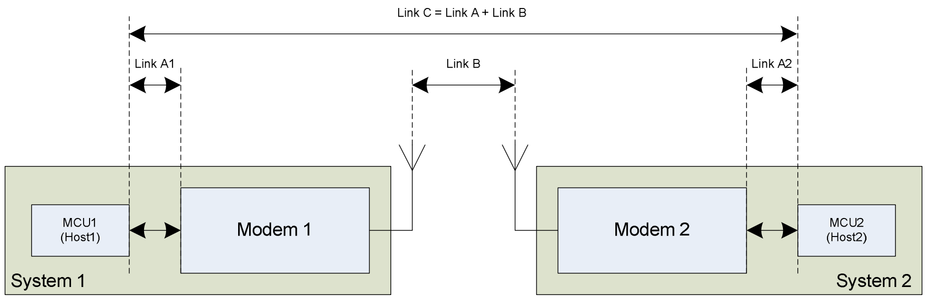
To enable the end-to-end communication between both MCUs (or hosts), each near-to-near intermediate link has to be previously established.

Step by step, this document describes how to build up a wireless enabled system. One chapter will be devoted to each of these steps:

1) Hardware configuration
2) Host-Modem serial communication (Link A1/2)
3) Modem-Modem RF communication (Link B)
4) Host-Host communication (Link C)

At each step, the modem behavior and requirements are evocated and moreover, simple sequences and troubleshooter are proposed.

As a result, the obtained communicating system may be used as a reference. It should be considered as a developer starting point, ready to receive the application desired RF parameters for tests purposes and deployment.



Hardware Configuration

Host Interface

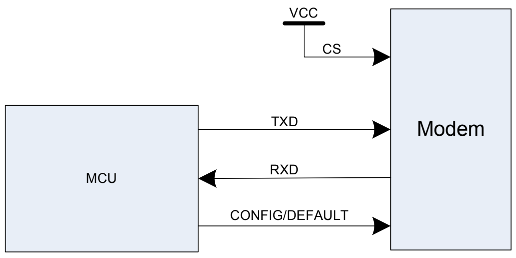
The host can communicate with the modem in several modes and ways (UART, SPI in either data or command mode) which may imply the use of different signals.

With the proposed connections, the host communicates with the modem through its UART interface and switches between data and command modes by driving the modem CONFIG_DEFAULT pin. (If the CONFIG_DEFAULT pin cannot be used, it is possible to switch between both modes using a BREAK condition. For more information, please refer to YLX-TRM datasheets).

- CONFIG_DEFAULT selects either the modem's data ('1') or command ('0') mode.

- RXD and TXD are the asynchronous data lines between the host and the modem, which constitutes the UART.

- CS signal is kept at '1' to select UART as host communication interface. If the host needs to drive the CS pin, the UART cannot be selected again after having been unselected (CS = '0').



Unused Pin Termination

To ensure a proper behavior and avoid unnecessary consumption, a good practice is to follow the recommended termination of the unused pins.
Most of the time, digital inputs can be connected to GND to avoid a floating potential while digital outputs can be left open.

To avoid initialization electrical conflicts, the default direction of configurable pins is often input. When those configurable pins are unused, they can be considered as simple inputs and then be connected to GND.

In this example, the following connections should be used for the modem unused pins:

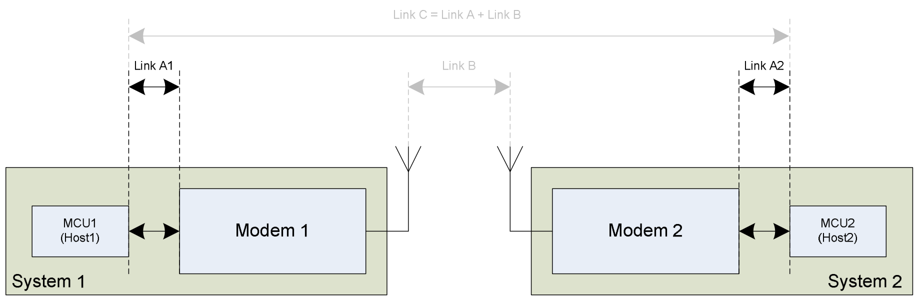
PIN NUMBER	PIN NAME	CONNECTION	DESCRIPTION
7	MOSI	GND	Input
8	MISO	Open	Output
9	SCLK	Open	Internal pull-up
11	HOP	Open	Output
12	SYNC	Open	Output
13	GP1	GND	Input (default)
14	GP2	GND	Input (default)
36	BAT_LEVEL	VCC	Input (reserved)
37	AINT	GND	Input
38	TINT	Open	Output

Host-Modem serial link

Serial Link background

The first link required to enable an RF communication is the one between the host and the modem. In a further step, it will be used as a basis to configure the modem and finally to communicate wirelessly with the other system.

As the modem offers multiple host interfaces and modes, several ways may be used to validate that link.
The one being presented in here is much simpler than the others, which explains its selection.



The modem embeds a flash memory enabling the storage of a user defined configuration set. This feature offers the possibility to separate the modem configuration phase, which can be done before assembly, from the modem use phase, which occurs when the application has been deployed.

However, when an initial communication has to be established for test purposes, it is interesting to start from a well known starting point. This is why the modem has to be powered up with its CONFIG_DEFAULT pin set to ‘0’. This sequence, also called as “DEFAULT RESET”, requests the modem to keep its DEFAULT configuration instead of loading the flash located one. At a power up, the modem takes some time to settle, so that a delay of 100ms should be inserted before considering the modem as ready.

Once the host knows the modem uses its default UART configuration, it can configure its own parameters to the same value.
The modem default UART parameters are: 9600 baud, 8 data bits, no parity bit, 1 stop bit and no flow control.

When the host communicates with a modem, it can access either its “data” or “configuration” entity which is selected by the CONFIG_DEFAULT pin.
As the modem has been powered up with its CONFIG_DEFAULT pin set to ‘0’ to invoke a “DEFAULT RESET”, its configuration entity is selected until CONFIG_DEFAULT is raised to ‘1’. In this configuration mode, the modem interprets each host frame as a command to execute and acknowledge.

This default configuration insures the modem uses well known parameters. Thus, this is an ideal context allowing the validation of the communication between the host and the modem.

With its UART interface configured to be compatible with the modem’s one (9600, 8, N, 1), the host can send a simple command frame to the modem.
The only caution still to be taken is not to allow the occurrence of a BYTE_TIMEOUT. This error indicated by the modem as an IND_ERROR happens in configuration mode when the amount of time between the beginnings of two consecutive bytes in a given frame reaches BYTE_TIMEOUT value (which default value is 2ms).

In configuration mode, a command/response frame is composed of three fields:
- [SIZE]
- [COMMAND]
- [ARGUMENTS]

The two first are mandatory and one byte sized while the last field has a variable size which can also be null in some cases.
The simple request frame, named CMD_GET_VERSION, can be used to validate the communication with the modem.

Serial link validation

The serial link validation sequence which has to be executed on both systems is described below:

1. Power up the modem with CS=’1’ and CONFIG_DEFAULT=’0’.
2. Wait 100ms to give the modem enough time to settle.
3. Retrieve the version number by sending CMD_GET_VERSION (0x01 0x10), at 9600 baud 8N1 on TXD and check the received response on RXD. The answer’s argument has to be interpreted in ASCII.

Action	Line	Frame		
		Size	Command	Arguments
MCU sends CMD_GET_VERSION	TXD	0x01	0x10	-
Modem answers the version “YLX” in ASCII	RXD	0x04	0x10	0x59 0x4C 0x58

When the previous sequence is successful on both boards the links between the MCUs and their own modem can be considered as correct. In that case, the next step is to establish the first radio communication. Otherwise the troubleshoot chapter hereafter will help finding out the issue.

Serial Link Troubleshooting

This chapter enumerates the symptoms that may result from an erroneous execution of the previously given sequence.

To find out which step is faulty, please double check the points proposed under the described modem behavior.

- *No answer from the modem.*
  - The modem is correctly power supplied.
  - The voltage applied to the modem inputs are within the expected limits for logical ‘0’ or ‘1’.
  - Every unused pin is correctly terminated.
  - Every connection between host and modem is correct according to the desired schematic.
  - Every modem connection is electrically correct.
  - The host sends data on TXD and expects data on RXD.
  - RESET = ’1’ after POR capacity loaded.
  - CS = ‘1’ since modem power up.
  - CONFIG_DEFAULT = ‘0’ since power up.
  - The host UART configuration matches (9600, 8, N, 1) and follows UART specifications.
  - The time between two consecutive bytes’ beginnings sent by the host is less than BYTE_TIMEOUT (2ms by default).

- *The modem answers but its frame content is not coherent.*
  - CONFIG_DEFAULT = ‘0’ since power up.
  - The host UART configuration matches (9600, 8, N, 1) and follows UART specifications.

- *The modem sends unexpected frames with COMMAND byte = 0xD3.*
  - The modem, in command mode, forwards RF received data frames encapsulated within the IND_RECEIVED_DATA indicator. You should either discard them or avoid their reception by putting the modem in power sleep mode during its initialization phase. Don’t forget to put the modem back in its normal mode when trying to receive RF communication.

- *The modem answers with the following error code:*
  - IND_ERROR ERR_TIMEOUT
    = 0x02 0x03 0x05

    This error indicates that a byte timeout occurred. Please make sure that the time between two consecutive bytes’ beginnings sent by the host is less than BYTE_TIMEOUT (2ms by default).

  - Any other error code like IND_ERROR ANY = 0x02 0x03 0xXX.

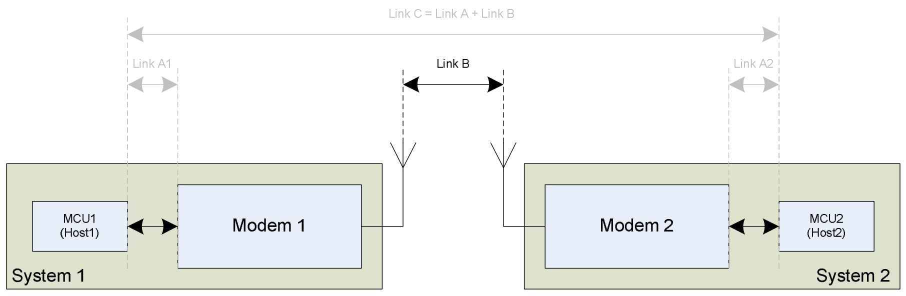
    The frame has been interpreted as erroneous. Please make sure that:
    - CONFIG_DEFAULT = ‘0’ since power up.
    - The host UART configuration matches (9600, 8, N, 1) and follows UART specifications.
    - The host sent the documented frame.

Modem-Modem RF communication

RF Link Background

Based on the already validated local links, each host can configure its own modem to enable the wireless communication with the other one.

Once this point will have been reached, the last step allowing a communication between each system's MCU can be achieved.



In order to reach the lowest communication power consumption, the modem communication protocol is based on a time synchronized mechanism. As a consequence, every communicating device needs to be synchronized before being allowed to receive/send data from/to the network.

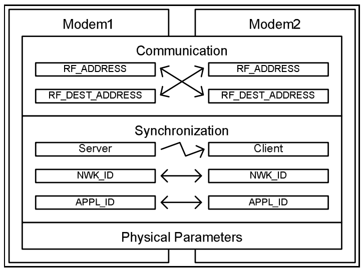
To achieve this principle, one modem among the network's devices has to be designated and configured as a beacon server. The device playing this role sends regular beacon frames to synchronize the clients.

A client having compatible parameters with a server and being in its RF range will become synchronized with the first received beacon and then will maintain its synchronization with the following beacons.

In the current case, both modems have been powered up with CONFIG_DEFAULT = '0'. This means that their parameters are fully compatible with each other and that the only requirement to enable the RF link is to configure one of them as beacon server. A third modem could also be used as a dedicated server but this is beyond the scope of this document.

The fact that both modems have the same own address, which is also by default their RF destination address, could simplify simple communication tests. Nevertheless, this should be avoided when more than two devices are used with an enabled acknowledgment mode, which is the case by default. To fulfill any further application needs, one should attribute each modem a unique RF_ADDRESS within its network and modify accordingly the RF_DEST_ADDRESS of the other device.

The following figure reminds that synchronization is a prerequisite to any RF communication. Apart from the physical layer parameters, the synchronization compatibility only relies on the network addressing implemented by APPL_ID and NWK_ID. Therefore those parameters need to be identical on both server and client sides. Afterwards the RF communication only requires that each RF_DEST_ADDRESS equals the other's RF_ADDRESS.



The synchronization being a communication enabler, the host has the possibility to check whether or not its modem is synchronized. This is necessary only for the clients because by definition the server is always synchronized to itself. This status is retrievable either by the SYNC pin or by the command GET_LAST_BEACON_INFO. This last possibility is used because it doesn't require any additional pin connection. The request command is simply sent with the argument byte asking for status.

To facilitate the test of the RF parameters, there is also a convenient command named CMD_SEND_ECHO_DATA. It is used to encapsulate data to be sent by RF. The pro of this command among others is that the modem, which receives the RF data, echoes it back without involving its own host. So, when the originator host receives back the sent data, the RF compatibility is proved. Note that because the originator host has kept the modem in command mode (CONFIG_DEFAULT = '0'), the received echoed data will be encapsulated with the IND_RECEIVED_DATA indicator.

RF Link Validation

Here is the RF link validation sequence and its usage recommendations.

Please note that this sequence has to be achieved only when all MCU-Modem links have already been validated with the previous explanation.
If during the sequence any modem is reset, the whole sequence has to be executed again.

As previously, each command frame has to be sent with CONFIG_DEFAULT = '0' and at (9600, 8, N, 1) on TXD.
The easiest way insuring the modem is ready to receive the next command is to wait and interpret the previous command's acknowledgement before sending the new one.

1. MCU2 attributes a new own address to Modem2

Action	Line	Frame		
		Size	Command	Arguments
MCU2 sends CMD_SET_RF_ADDRESS	TXD	0x02	0x44	0x33
Modem2 acknowledges the command	RXD	0x01	0x44	-

2. MCU1 configures Modem1 as beacon server

Action	Line	Frame		
		Size	Command	Arguments
MCU1 sends CMD_SET_BEACON_MODE	TXD	0x02	0x54	0x01
Modem1 acknowledges the command	RXD	0x01	0x54	-

3. MCU1 adapts Modem1 destination address to match Modem2 new address

Action	Line	Frame		
		Size	Command	Arguments
MCU1 sends CMD_SET_RF_DEST_ADDRESS	TXD	0x02	0x4A	0x33
Modem1 acknowledges the command	RXD	0x01	0x4A	-

4. MCU2 requests Modem2 synchronization status

Action	Line	Frame		
		Size	Command	Arguments
MCU2 sends CMD_GET_LAST_BEACON_INFO	TXD	0x02	0x31	0x01
Modem2 answers the status (0x01 when synchronized)	RXD	0x02	0x31	0x01

5. MCU1 tests RF communication link by sending encapsulated ASCII “Test1” which has to be echoed by Modem2

Action	Line	Frame		
		Size	Command	Arguments
MCU1 sends CMD_SEND_ECHO_DATA	TXD	0x06	0xD4	0x54 0x65 0x73 0x74 0x31
Modem1 acknowledges the command	RXD	0x01	0xD4	-
Modem1 forwards the received echoed data encapsulated with IND_RECEIVED_DATA	RXD	0x06	0xD3	0x54 0x65 0x73 0x74 0x31

6. MCU2 tests RF communication link by sending encapsulated ASCII “Test2” which has to be echoed by Modem1

Action	Line	Frame		
		Size	Command	Arguments
MCU2 sends CMD_SEND_ECHO_DATA	TXD	0x06	0xD4	0x54 0x65 0x73 0x74 0x32
Modem2 acknowledges the command	RXD	0x01	0xD4	-
Modem2 forwards the received echoed data encapsulated with IND_RECEIVED_DATA	RXD	0x06	0xD3	0x54 0x65 0x73 0x74 0x32

The sequence can be considered as successful as soon as both MCU have correctly received the echoed data encapsulated with IND_RECEIVED_DATA. When every intermediate links has been verified as functional, the next and final step is to exploit the MCU1-MCU2 two ways wireless link. On the contrary, if the sequence failed at any step, the following troubleshoots may help finding out the issue.

RF Link Troubleshooting

This chapter enumerates the symptoms that may result from an erroneous execution of the previously given sequence.
To find out which step is faulty, please double check the points proposed under the described modem behavior.

- *No answer from the modem in command mode (CONFIG_DEFAULT=’0’).*
  - The serial link between the host and the modem should be validated according to the dedicated chapter.

- *The client modem is not synchronized to the server modem beacon.*
  (This issue is independent of the modems own address and destination address)
  - Modems are correctly power supplied (Voltage and current requirements are fulfilled).
  - Modems have well plugged antennas and are far from 3 to 10 meters from each others.
  - Modems have been powered up or reset with CONFIG_DEFAULT=’0’ to insure a default reset.
  - The beacon server modem has acknowledged the SET_BEACON_MODE command.
  - Each modem’s channel0 frequency is identical (it can be retrieved using GET_CHANNEL_FREQ CH0 = 0x03 0x2D 0x00).

- *The modem is synchronized to the server, or is the server, but the RF data is never received.*
  - If this is the server, the other device is not a server but a client synchronized to it (and not to another server).
  - The initiator’s destination address equals the destination’s own address (The link can be tested with addresses default values).
  - All parameters are identical in both modems apart from the beacon mode of the server, the own address of a modem and the destination address of the other.

- *The modem is synchronized to the server, or is the server, but some RF data is missed.*
  - Modems have been powered up or reset with CONFIG_DEFAULT='0' to insure a default reset.
  - All parameters are identical in both modems exception made for the beacon mode of the server and the own address of a modem and the destination address of the other.
  - The involved client doesn't lose its synchronization. Indeed, synchronization is a prerequisite to participate in a network. To reduce the impact of a beacon lost, increasing the clients' MAX_BEACON_LOST parameter could be considered.
  - The used RF frequencies are not disturbed by any external factor. The presence of a perturbator can be verified using a unique frequency on every channel. The communication can therefore be tested with several frequencies chosen sufficiently far from each other. The command CMD_SET_CHANNEL_FREQ can be used for this purpose.

- *The modem reloaded its default parameters values unexpectedly*
  - The modem has not been reset (Voltage and current requirements are fulfilled).
  - The modem didn't receive 20 consecutive break conditions, which is an alternate way to load its default parameters.
    Note that break conditions can be interpreted when the host sends UART data at bitrates lower than the modem one.

- *The modem answers with the following error code.*

  - IND_ERROR ERR_TIMEOUT
    = 0x02 0x03 0x05

    This error indicates that a byte timeout occurred. Please make sure that:
    - The time between two consecutive bytes' beginnings sent by the host is less than BYTE_TIMEOUT (2ms by default).

  - IND_ERROR ERR_CONTEXT
    = 0x02 0x03 0x06

    This error happens when a command execution cannot take place within the current context.
    This is the case when the host sends a RF data frame in command mode with CMD_SEND_(ECHO_)DATA and the modem is a client which is not synchronized or when there is no data channel. This last condition is met when BEACON_PERIOD = 1 and CHANNEL_COUNT = 1 so that every channel receives a beacon.

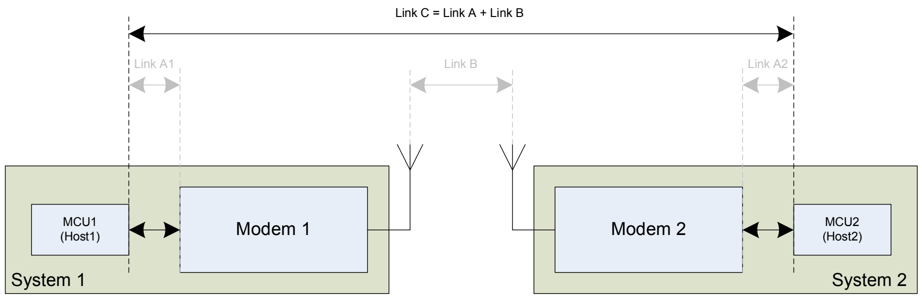
  - Any other error code like: IND_ERROR ANY = 0x02 0x03 0xXX

    The frame has been interpreted as erroneous. Please make sure that:
    - CONFIG_DEFAULT = '0' since power up.
    - The host UART configuration matches (9600, 8, N, 1) and follows UART specifications.
    - The host sent the exact documented frame.

Host-Host global communication

Global Link Background

The local and wireless links having been checked as operational, the system hosts can now use the modems to communicate seamlessly with each other. Afterwards the developer can use the communicating solution as a base point for his application.



The application can benefit from the presence of the intermediate links to emulate a Host-Host link.

As mentioned, the modem offers a command and a data mode which can both be used to send RF data frames. The difference between the command and the data mode to send and receive RF frames only takes place in the protocol between the host and the modem. In the first case the data is encapsulated while it is raw in the second one. This means that a working RF link in command mode also works in data mode.

In the command mode the RF data frames to be sent are encapsulated in CMD_SEND_DATA commands while the modem uses the IND_RECEIVED_DATA indicator to forward the RF received data.

In the data mode the RF data frames can simply be transmitted to the modem for transmission like the modem itself forwards seamlessly the received data to the host.

The new functionality involved in this step is the mode switching from command to data. In the same way as the CMD_SEND_ECHO_DATA has been used previously, the first end to end communication can be established thanks to the CMD_SEND_DATA command.

Once the MCUs communicate together according to the described sequence, the modem parameters may be adapted to the application needs.

Global Link Validation

Here is the summarized sequence proposed to use the wireless link offered by the modems.
The following sequence is based on the already validated intermediate links and allows both MCU to communicate with each other in command mode then in data mode.

1. MCU1 sends MCU2 some RF data frames “Test1” in command mode (CONFIG_DEFAULT='0' on both modems)

Action	Line	Frame		
		Size	Command	Arguments
MCU1 sends CMD_SEND_DATA	TXD	0x06	0xD2	0x54 0x65 0x73 0x74 0x31
Modem1 acknowledges the command	RXD	0x01	0xD2	-
Modem2 forwards the received data encapsulated with IND_RECEIVED_DATA	RXD	0x06	0xD3	0x54 0x65 0x73 0x74 0x31

2. MCU2 sends MCU1 some RF data frames “Test2” in command mode (CONFIG_DEFAULT=’0’ on both modems)

Action	Line	Frame		
		Size	Command	Arguments
MCU2 sends CMD_SEND_DATA	TXD	0x06	0xD2	0x54 0x65 0x73 0x74 0x32
Modem2 acknowledges the command	RXD	0x01	0xD2	-
Modem1 forwards the received data encapsulated with IND_RECEIVED_DATA	RXD	0x06	0xD3	0x54 0x65 0x73 0x74 0x32

3. MCU1 and MCU2 put their modem in data mode by setting CONFIG_DEFAULT=’1’

4. MCU1 sends MCU2 some RF data frames “Test1” in data mode (CONFIG_DEFAULT=’1’ on both modems)

Action	Line	Frame
MCU1 sends “Test1”	TXD	0x54 0x65 0x73 0x74 0x31
MCU2 receives “Test1”	RXD	0x54 0x65 0x73 0x74 0x31

5. MCU2 sends MCU1 some RF data frames “Test2” in data mode (CONFIG_DEFAULT=’1’ on both modems)

Action	Line	Frame
MCU1 sends “Test2”	TXD	0x54 0x65 0x73 0x74 0x32
MCU2 receives “Test2”	RXD	0x54 0x65 0x73 0x74 0x32

The sequence and the whole technical note can be considered as fully functional when the data sent by MCU1 is received by MCU2 and reciprocally.
Now the initial default communication link has been validated. The application which requires it could progressively modify the modem parameters to match its requirements. In this last case, the new parameters have to be set in both modems to insure the compatibility and the link should be retested each time.

Global Link Troubleshooting

This chapter enumerates the symptoms that may result from an erroneous execution of the previously given sequence.
To find out which step is faulty, please double check the points proposed under the described modem behavior.

- *No answer from the modem in command mode (CONFIG_DEFAULT=’0’).*
  - The serial link between the host and the modem should be validated with the dedicated chapter.

- *The RF synchronization or the RF data communication doesn’t work.*
  - The RF link should be validated with the dedicated chapter.

- *Data are sent to the modem, but instead of sending them over RF it answers with error indications or command acknowledgements.*
  - The modem CONFIG_DEFAULT=’0’ and no break condition has been sent on the UART previously to the data.
    Note that break conditions can be interpreted when the host sends UART data at bitrates lower than the modem one.

Conclusion

Throughout this document, the reader should have learned the required steps enabling a first wireless communication in an embedded system using the UART interface of two Y-Lynx YLX-TRM8053-xxx-05 modems. At this stage, the gap separating the final application should also have been considerably reduced since only specific modems parameterization remains.

More than the exactly described sequences, the steps of this document can also be used or adapted to break down the development phase leading to a first end to end wireless communication. Thus, a similar approach should be used when, for example, the SPI host communication interface has been selected for the application.

Documentation History

Revision	Modifications	Date
Rev 1.0	Original version - AN0829-003.1.0	July, 2008

Contact Information

Address:

Y-Lynx Sàrl

Rue de Galilée 15
CH - 1400 YVERDON
SWITZERLAND

Phone: +41 24 423 92 05
Email: info@y-lynx.com
Web site: http://www.y-lynx.com