iC880A QuickStart Guide

How to get started with the iC880A

Document ID: 4100/40140/0078

Category: confidential

IMST GmbH

Carl-Friedrich-Gauss-Str. 2-4

D-47475 Kamp-Lintfort

Document Information

File name	iC880A_QuickStartGuide.docx
Created	2015-03-15
Total pages	15

Revision History

Version	Description
0.1	Preliminary version.
0.2	Figure 2.1 updated
0.3	Added information about the polarity of the DC-Jack / Power supply connector
0.4	Picture of iC880A updated
0.5	Added version information about the referenced github repository; added hint about the WiMOD LR Studio
0.6	Changed link URL to libmpsse

Aim of this Document

Aim of this document is to give some quick start instructions how to start working with the WiMOD iC880A.

Confidentiality Note

This document has to be treated confidentially. Its content must not be published, duplicated or passed to third parties without our express permission.

Table of Contents

1 Precaution: Important note

Caution: Before connecting the iC880A to an external power supply, please read this note carefully!

The WiMOD iC880A concentrator is a versatile device that needs extra power to operate properly. The USB specification allows a device to draw up to 500 mA in normal operation conditions. Within the defined performance range the concentrator needs more than 500 mA. Depending on your Linux host that is to be connected with the iC880A, the USB interface cannot supply enough power. Therefore an external power supply is highly recommended.

When using an external power supply a certain order of connecting the cables must be followed:

1) First: power up the iC880A with an external power supply
2) Second: connect the USB cable between the iC880A and the host.

Make sure the external power supply is on and supplies 5 V before connecting the power cable. If there is no power available and the connector is in the DC jack, the internal voltage regulators of the iC880A may get damaged!

The polarity of the external DC-Jack is shown in Figure 1-1:

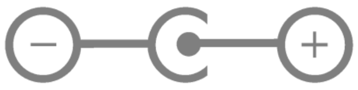


Figure 1-1 Polarity of the 5V DC-Jack

2 Overview

Purpose of this documentation is to give some useful information how to start developing your own software based on the open source LoRa HAL published on the github platform.

The concentrator module iC880A is meant to be connected to a host system forming an integrated gateway.

2.1 Hardware PCB

Figure 2-1 depicts the printed circuit board of the iC880A. As main interfaces there are the DC –Power Jack and the USB-Connector to a host system that runs the driver software. A SMA connector is supplied in order to connect any suitable antenna to this board. The board offers some LEDs to give a visual feedback of the current operation status of the board. The concrete meaning of the LEDs is software dependant. As default setting the LEDs is:

1) Backhaul packet
2) TX packet
3) RX Sensor packet
4) RX FSK packet
5) RX buffer not empty
6) Power

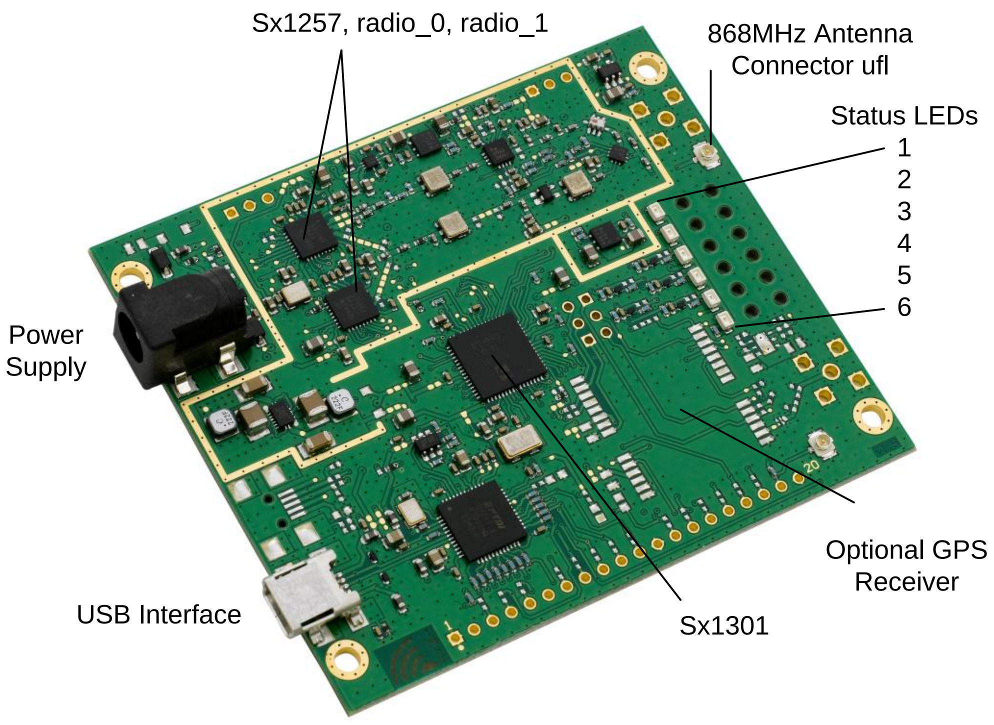


Figure 2-1 Picture of iC880A-USB

2.2 Basic System Concept

Figure 2-2 shows the basic system concept for the LoRA WAN system. The iC880A is the central hardware solution for all LoRa based radio communication. It receives and transmits radio messages. Processing of the radio messages as well as the protocol related tasks is done by (external) host system(s). The concrete segmentation of the protocol related tasks is outside the scope of this document.

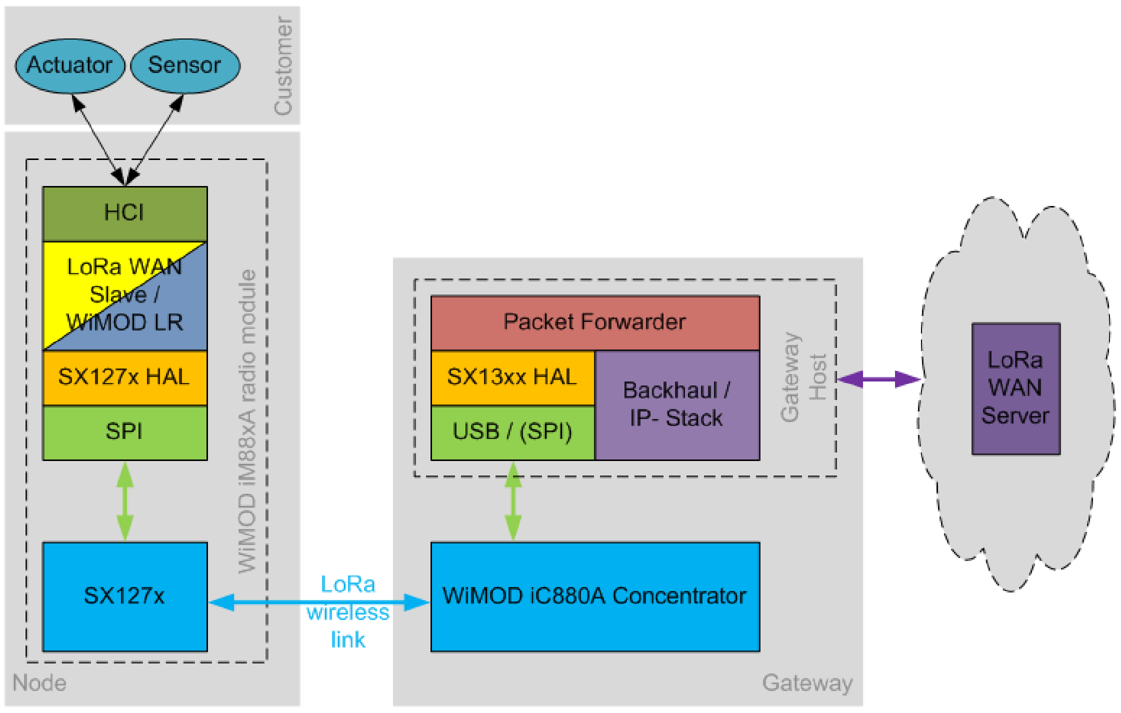


Figure 2-2: Basic System Concept

2.3 Linux Host System

As a host system an (embedded) Linux host system is planned to be used. This quick start guide assumes that the user has got an (embedded) Linux host system (e.g. Raspberry PI, or similar), is familiar with the basic concept of a Linux console and knows how to compile source code. All commands shown in this document are tested with a Debian 8 "Jessie" Linux distribution on a Raspberry PI B+. It is possible to use any other host system and Linux distribution, but there may be some variations in the commands, that are not within the scope of this document.

The basic setup of the (embedded) Linux host system (keyboard layout, time zone, network connectivity, accessibility, …) is outside of the scope of this document, too.

2.4 Hardware Connection

The iC880A provides a mini USB connector as main interface to the Linux host system. Before connecting the iC880A and the host, please read this document carefully. In order to establish a logical connection the host needs to have some drivers installed.

3 Open Source Driver on Github

Semtech provides an open source driver for SX1301 based LoRa solutions. The complete repository is hosted on the github platform (https://github.com/).

The main vanilla source repository for the LoRa related projects maintained by Semtech can be found at:

https://github.com/Lora-net

There you will find an implementation for the HAL (Hardware Abstraction Layer) of SX1301based designs. This implementation is called "lora_gateway" (https://github.com/Lora-net/lora_gateway). **This document references to version 1.7 of the lora_gateway repository. Newer versions of this software may behave differently or may require a different configuration. Changes / updates within all relevant software repositories are subject to change without notice.**

3.1 Download of the Open Source Driver

Install a git client on your (embedded) host system: On Debian based systems the git client can be installed by calling the following command in a terminal emulator / console:

```
sudo apt-get install git
```

Create local folder that should contain the local copy of the repository:

```
mkdir -p ~/LoRa/lora_gateway
```

Get a copy of the repository:

```
cd ~/LoRa/lora_gateway

git clone https://github.com/Lora-net/lora_gateway.git
```

3.2 Downloading Additional Drivers

The iC880A device is connected to an (embedded) Linux host system via an USB link. In order to establish a local USB link, a driver called "`libmpsse`" must be installed on the host system.

Create a folder for the libmpsse files:

```
mkdir -p ~/LoRa/libmpsse
```

Download the newest version of the libmpsse files:

```
cd ~/LoRa/libmpsse

wget      https://storage.googleapis.com/google-code-archive-
downloads/v2/code.google.com/libmpsse/libmpsse-1.3.tar.gz

tar -xzvf libmpsse-1.3.tar.gz
```

Follow the installation instructions from the hosted website: https://code.google.com/p/libmpsse/wiki/Installation:

```
sudo apt-get install libftdi-dev

cd ~/LoRa/libmpsse/libmpsse-1.3/src

./configure --disable-python && make && sudo make install

sudo ldconfig
```

If the `configure` step fails, follow the instructions and try to install the missing dependencies.

3.3 Configuration of the Driver

The driver must be configured in order to work with the iC880A. The following chapter describes the necessary configuration options to make.

In the folder "`LoRa/github/lora_gateway/libloragw`" there is a file called "`libraray.cfg`". This file contains the most important options for the HAL-library and must be modified to your needs before compiling the software.

3.3.1 File: lora_gateway/libloragw/library.cfg

This file is used as general configuration file for the whole `libloragw` system library. In order to fit the needs of the iC880A, some parameters must be set to the following values prior (re-) compiling the library:

```
CFG_SPI= ftdi
CFG_CHIP= sx1301
CFG_RADIO= sx1257
CFG_BAND= eu868
```

The parameter called "`CFG_BRD`" must be left empty or set to:

```
CFG_BRD= ref_1301_868
```

Additionally there are some optional debug settings that can be turned on (`1`) or off (`0`):

```
DEBUG_AUX= 0
DEBUG_SPI= 0
DEBUG_REG= 0
DEBUG_HAL= 1
DEBUG_GPS= 0
```

These flags are used by the internal modules of the library to enable / disable additional output for debugging purposes.

3.3.2 File: lora_gateway/libloragw/src/loragw_spi.ftdi.c

As second step a small modification must be done in one of the low level connection files. The WiMOD iC880A LoRa Concentrator board uses a FTDI USB to SPI converter chip that is not supported in Semtech's reference code. The only difference is another USB PID. Therefore the USB PID settings in the `libloragw` library have to be adjusted prior compiling the library.

Open the file lora_gateway/libloragw/src/loragw_spi.ftdi.c with your favorite editor (e.g. nano)

```
nano lora_gateway/libloragw/src/loragw_spi.ftdi.c
```

Find the block with the "`PRIVATE CONSTANTS`" (around line 50 in the file) and change the line

```
#define PID  0x6010
```

to

```
#define PID  0x6014
```

Afterwards save the file and close the editor.

3.4 Compilation of the Library

In order to compile the `libloragw` library it is assumed that a gcc/g++ compiler and a make utility is already installed and runable.

Enter the lora_gateway folder and execute the make command:

```
cd lora_gateway

make
```

This will compile the library and some of the basic test utilities located in the subfolders of `lora_gateway`.

After a successful compilation a library file called "`libloragw.a`" has been created and is ready for usage now.

3.5 Setting up "udev" Rules

The FTDI USB chip needs to be run in the SPI mode. Therefore an "udev" rule must be installed on the Linux host system. This rule will configure the right mode automatically when the iC880A device is connected to the host system. In order to install the udev rule the following steps must be done:

In the folder `lora_gateway/libloragw` there is a template file present. Open the file in your favorite editor (e.g. nano).

```
nano lora_gateway/libloragw/99-libftdi.rules
```

locate the line

```
SUBSYSTEM=="usb",ENV{DEVTYPE}=="usb_device",
ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6010",
MODE="0664", GROUP="plugdev"
```

and change the entry

```
ATTRS{idProduct}=="6010”
```

to:

```
ATTRS{idProduct}=="6014”
```

Save the file and leave the editor. Next copy the file to the folder `/etc/udev/rules.d/`

```
sudo cp 99-libftdi.rules /etc/udev/rules.d/
```

3.6 Connection of iC880A and the Linux Host

3.7 Example Utilities

In the folder `lora_gateway` there are several example utilities showing the usage of the `libloragw` library. This quick start guide will show the “`util_pkt_logger`” and the “`util_tx_test`”.

3.7.1 util_tx_test

This is a small utility demonstrating the ability to send out some data via the iC880A concentrator device. To start this utility go into the folder and start it:

```
cd lora_gateway/util_tx_test

./util_tx_test
```

The program should start and print out some output like that:

```
Sending -1 packets on 866500000 Hz (BW 125 kHz, SF 10, CR 1,
16 bytes payload, 8 symbols preamble) at 14 dBm, with 1000 ms
between each

INFO: concentrator started, packet can be sent

Sending packet number 1 ...OK

Sending packet number 2 ...OK

Sending packet number 3 ...OK
```

Each time a new packet is being transmitted the TX-LED of the iC880A should flash. The tool can be stopped by pressing the key combination `Ctrl+C`. For more information about this tool, please consult the online help by calling:

```
./util_tx_test --help
```

3.7.2 util_pkt_logger

This utility tool is able to demonstrate the receiving ability of the iC880A device. It configures the iC880A device for RX operation and writes each received packet into a CSV file. That file can be opened and inspected afterwards.

This tool needs a configuration file that holds parameters relevant to the RX operation of the concentrator. Please take a look into this file called `global_conf.json` .

An example global_conf.json looks like that:

```
{
    "SX1301_conf": {
        "radio_0": {
            "enable": true,
            "freq": 868200000
        },
        "chan_multiSF_0": {
            /* LoRa MAC channel, 125kHz, all SF, 868.1 MHz */
            "enable": true,
            "radio": 0,
            "if": -100000
        },
        "chan_multiSF_1": {
            /* LoRa MAC channel, 125kHz, all SF, 868.3 MHz */
            "enable": true,
            "radio": 0,
            "if": 100000
        },
        "chan_multiSF_2": {
            /* LoRa MAC channel, 125kHz, all SF, 868.5 MHz */
            "enable": true,
            "radio": 0,
            "if": 300000
        },
        "chan_Lora_std": {
            /* LoRa MAC channel, 250kHz, SF7, 868.3 MHz */
            "enable": true,
            "radio": 0,
            "if": 100000,
            "bandwidth": 250000,
            "spread_factor": 7
        },
        "chan_FSK": {
            /* FSK 100kbps channel, 868.3 MHz */
            "enable": true,
            "radio": 0,
            "if": 100000,
            "bandwidth": 250000,
            "datarate": 100000
        }
    },
    "gateway_conf": {
        "gateway_ID": "AA555A0000000000"
    }
}
```

In the head of the file there is one config block for each radio of the concentrator. Each radio can be turned on / off and operates on a certain base frequency. (See block "`radio_0`" and "`radio_1`"). The following blocks define the parameters used by the virtual multi SF channels. Each logical channel configuration consists of the radio to be used and the RF frequency. The RF frequency is defined as difference to the base frequency defined in the corresponding radio config block.

After checking the configuration the tool can be started via:

```
cd lora_gateway/util_pkt_logger

./util_ptk_logger
```

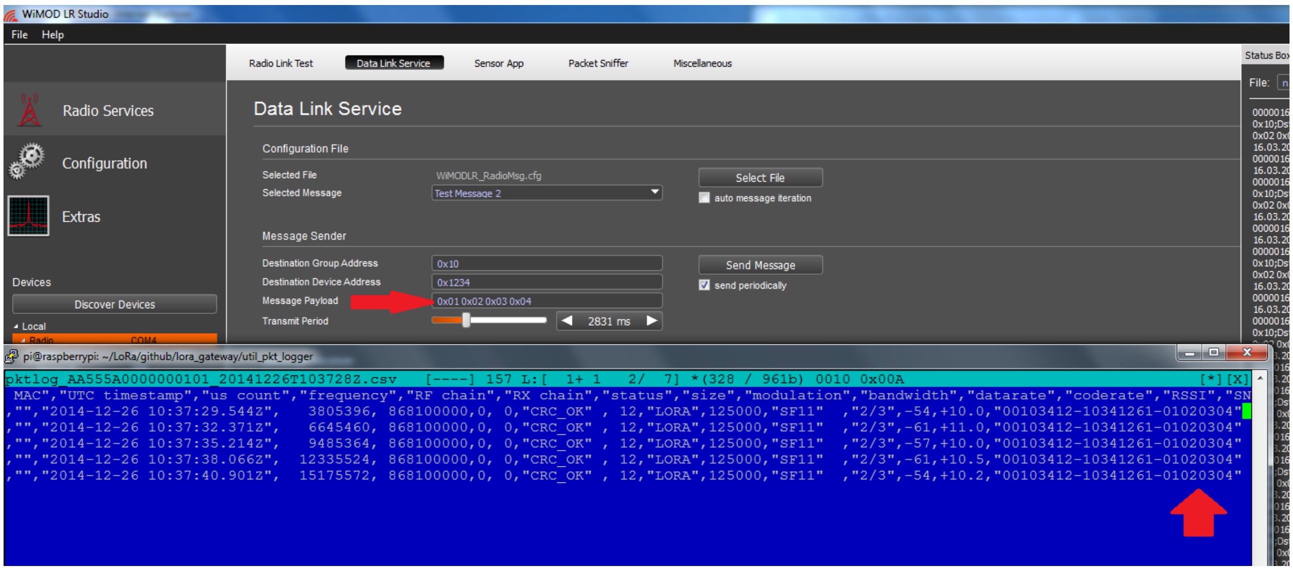
After starting the packet logger, please send out some data with another radio module. In this document it is assumed that the user is familiar with the WiMOD iM88xA radio module or the iU88xA USB-Stick and the Windows Software called "WiMOD LR Studio"[1] or "WiMOD ExpLoRa Studio" or "WiMOD LoRaWAN EndNode Studio".

In order to send some data start the Windows software and navigate to "Configuration" and make sure the TX setup matches one of the configurations in the `global_conf.json` file within the `util_pkt_logger` folder on the linux host. After setting up the radio TX configuration, navigate to "Radio Services" / "Data Link Service" and start sending some test messages.

In order to inspect the RX data from the iC880A device stop the `util_pkt_logger` by pressing `Ctrl+C` and open the CSV file in the current folder.

The following screen shot shows the TX and RX data:

[1] The WiMOD LR Studio is intended to be used with a WiMOD Module running the WiMOD LR-Base firmware. The other PC software is intended to be used in conjunction with a WiMOD Module running the WiMOD LoRaWAN firmware.



```
pktlog_AA555A0000000101_20141226T103728Z.csv   [----] 157 L:[  1+ 1   2/  7] *(328 / 961b) 0010 0x00A                [*][X]
 MAC","UTC timestamp","us count","frequency","RF chain","RX chain","status","size","modulation","bandwidth","datarate","coderate","RSSI","SN
,"","2014-12-26 10:37:29.544Z",   3805396, 868100000,0, 0,"CRC_OK" , 12,"LORA",125000,"SF11"  ,"2/3",-54,+10.0,"00103412-10341261-01020304"
,"","2014-12-26 10:37:32.371Z",   6645460, 868100000,0, 0,"CRC_OK" , 12,"LORA",125000,"SF11"  ,"2/3",-61,+11.0,"00103412-10341261-01020304"
,"","2014-12-26 10:37:35.214Z",   9485364, 868100000,0, 0,"CRC_OK" , 12,"LORA",125000,"SF11"  ,"2/3",-57,+10.0,"00103412-10341261-01020304"
,"","2014-12-26 10:37:38.066Z",  12335524, 868100000,0, 0,"CRC_OK" , 12,"LORA",125000,"SF11"  ,"2/3",-61,+10.5,"00103412-10341261-01020304"
,"","2014-12-26 10:37:40.901Z",  15175572, 868100000,0, 0,"CRC_OK" , 12,"LORA",125000,"SF11"  ,"2/3",-54,+10.2,"00103412-10341261-01020304"
```

Figure 3-1: Screen shot of the recorded CSV file and the WiMOD LR studio

Note: The CSV file records all data as hexadecimal bytes with no respect to any fields defined in the LoRa WAN specification. Therefore you have to look at the end of the payload within the CSV file to see the payload data message itself.

4 Next Steps

After stepping through this guide the user is able to start developing his/her own iC880A based solution.

As a final hint there are some more interesting open source projects within the gitbub repository

https://github.com/Lora-net/

For example there is a project called “packet_forwarder” that demonstrates how to receive LoRa packets and forward the data to a remote server which may run a LoRa WAN Server implementation or any other proprietary solution.

For more information please refer to the User Guides and specific data sheets of the corresponding products.

5 Regulatory Compliance Information

The use of radio frequencies is limited by national regulations. The radio module has been designed to comply with the European Union's R&TTE (Radio & Telecommunications Terminal Equipment) directive 1999/5/EC and can be used free of charge within the European Union. Nevertheless, restrictions in terms of maximum allowed RF power or duty cycle may apply.

The radio module has been designed to be embedded into other products (referred as "final products"). According to the R&TTE directive, the declaration of compliance with essential requirements of the R&TTE directive is within the responsibility of the manufacturer of the final product. A declaration of conformity for the radio module is available from IMST GmbH on request.

The applicable regulation requirements are subject to change. IMST GmbH does not take any responsibility for the correctness and accuracy of the aforementioned information. National laws and regulations, as well as their interpretation can vary with the country. In case of uncertainty, it is recommended to contact either IMST's accredited Test Center or to consult the local authorities of the relevant countries.

This Development Kit/Starter Kit does not fall within the scope of the European Union directives regarding electromagnetic compatibility, restricted substances (RoHS), recycling (WEEE), FCC, CE or UL, and therefore may not meet the technical requirements of these directives or other related directives.

6 Important Notice

6.1 Disclaimer

IMST GmbH points out that all information in this document is given on an “as is” basis. No guarantee, neither explicit nor implicit is given for the correctness at the time of publication. IMST GmbH reserves all rights to make corrections, modifications, enhancements, and other changes to its products and services at any time and to discontinue any product or service without prior notice. It is recommended for customers to refer to the latest relevant information before placing orders and to verify that such information is current and complete. All products are sold and delivered subject to “General Terms and Conditions” of IMST GmbH, supplied at the time of order acknowledgment.

IMST GmbH assumes no liability for the use of its products and does not grant any licenses for its patent rights or for any other of its intellectual property rights or third-party rights. It is the customer’s duty to bear responsibility for compliance of systems or units in which products from IMST GmbH are integrated with applicable legal regulations. Customers should provide adequate design and operating safeguards to minimize the risks associated with customer products and applications. The products are not approved for use in life supporting systems or other systems whose malfunction could result in personal injury to the user. Customers using the products within such applications do so at their own risk.

Any reproduction of information in datasheets of IMST GmbH is permissible only if reproduction is without alteration and is accompanied by all given associated warranties, conditions, limitations, and notices. Any resale of IMST GmbH products or services with statements different from or beyond the parameters stated by IMST GmbH for that product/solution or service is not allowed and voids all express and any implied warranties. The limitations on liability in favor of IMST GmbH shall also affect its employees, executive personnel and bodies in the same way. IMST GmbH is not responsible or liable for any such wrong statements.

Copyright © 2015, IMST GmbH

6.2 Contact Information

IMST GmbH

Carl-Friedrich-Gauss-Str. 2-4
47475 Kamp-Lintfort
Germany

T +49 2842 981 0
F +49 2842 981 299
E wimod@imst.de
I www.wireless-solutions.de