TEKZITEL

PROTOFOX USER GUIDE

Abstract

This document will assist any person who wants to get started quickly on using the development kit.

Development Kit Terms of Usage

The development kit is designed to be used for research and development in a laboratory environment. This development kit is not intended to be a finished appliance, nor is it intended for incorporation into finished appliances that are made commercially available as single functional units to end users.

This development kit uses the WISOL Sigfox module which is SIGFOX verified.

Stresses exceeding those listed in the Maximum Ratings table may damage the device. If any of these limits are exceeded, device functionality should not be assumed, damage may occur and reliability may be affected.

Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Contents

1. Introduction

This prototyping kit enables solution development for IoT devices communicating on the Sigfox network. The kit has 2 core parts, a Wisol based Sigfox module and a microcontroller to drive and control the messaging functions. The kit is supported by peripheral pins that can be used to sample and acquire data. The kit is powered and driven by a USB cable connected to a PC. The microcontroller is an Atmel ATMEGA32U4-AU. Given this, the firmware programming can be done using the popular Arduino IDE based on the Leonardo design. The kit includes an SMA antenna that is matched to the Wisol module. The Wisol module used is dependent on the zone selected.

ATMEGA32U4-AU is a USB slave device that enables firmware programming without the need of an additional component for interfacing to the PC. Thus reducing component count and costs.

The kit supports all RC zones. However, hardware purchased for a particular zone is limited to that specific zone only.

The kit also has ESD protection as recommended by Wisol.

To summarize, this kit is a useful piece of hardware for anyone who wishes to prototype and experiment solutions, eventually enabling the user to develop a final solution for a device to meet industry or community needs.

2. Features

Worldwide zone coverage

- RCZ1 - Europe, Oman, Iran, South Africa, Tunisia, UAE
- RCZ2 - USA/Mexico/Brazil
- RCZ3 - Japan
- RCZ4 - Australia/New Zealand/South East Asia/Rest of South America

Usable with Arduino IDE for Leonardo boards

Technical support available

Micro USB port compatible with commonly used USB cables

Jumper provided on board to measure current consumption, enabling design of low power devices.

Kit layout compatible with Arduino shields, giving freedom to the developer to use sensor shields as per their requirements.

SPI interface for both the microcontroller and Wisol available.

12 IO ports available from the microcontroller including 4 ADC inputs

4 IO ports available from the Wisol module including 2 ADC inputs

5V and 3V power available

On-board LEDs to indicate device activities

3. Prerequisites

1. Make sure you are in a Sigfox covered area by checking the coverage map. If not, either use the Sigfox Network Emulator Dongle or check your coverage options at the I need Sigfox page.
2. Basic knowledge of embedded systems, C/C++ programming and IoT concepts.
3. If you're new to Sigfox and want to jump start your Sigfox development, you'll find everything you need here.

4. Development Kit Package Contents

1. Micro USB cable.
2. SMA Antenna male.
3. Development board.

5. Tools Needed

1. Arduino IDE
2. PC

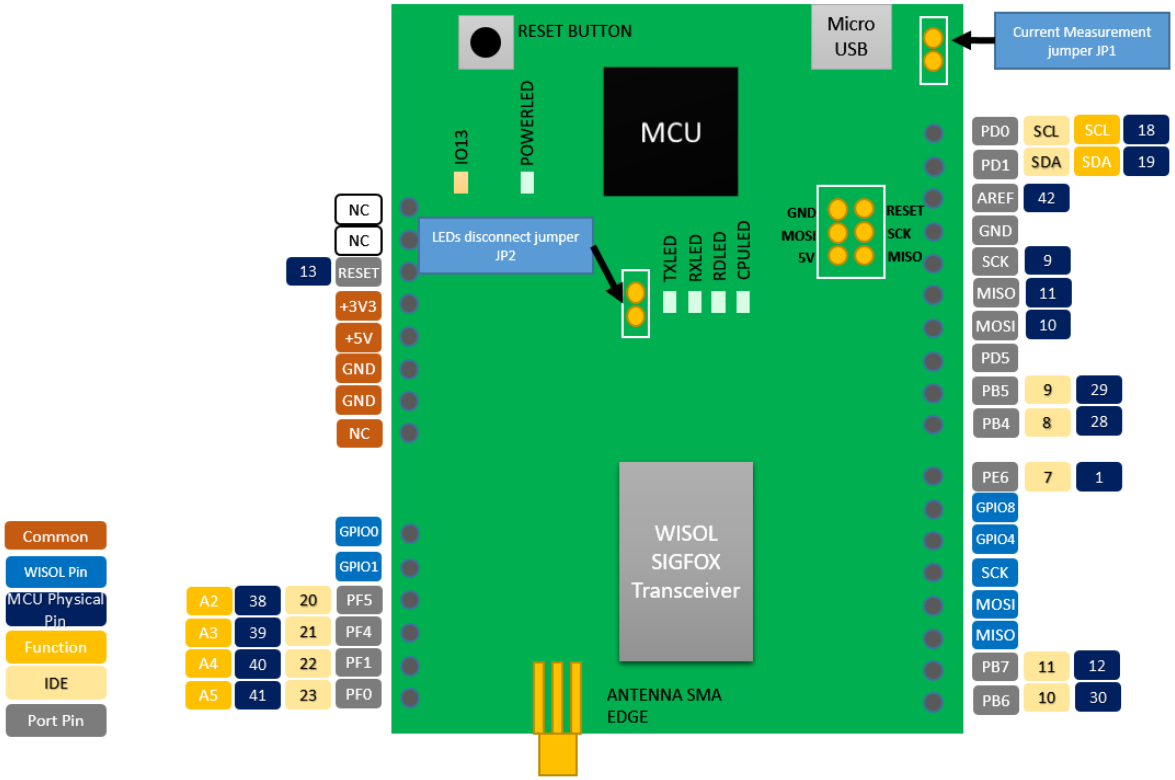
6. Software Installation

The firmware required to run the microcontroller, so that it is able to talk to the Wisol module, has to be developed in an IDE. The development kit is based on the ATmega32u4 microcontroller, which is the same that is used on Leonardo boards. So, by using the popular Arduino IDE, one should be able to develop firmware and flash it to the board through the USB.

The IDE if not yet installed on your PC can be downloaded from https://www.arduino.cc/en/Main/Software . The link has step-by-step instructions for your operating system.

Drivers should be automatically installed when plugging the board with a USB cable to your PC, but with some versions of the Windows operating system (like Windows 7, Vista and 10) it can happen that your board won't be recognized and you will get the message “Unknown USB device”. If this happens then it is necessary to manually install them following the guide manually install Drivers on Windows.

7. Pinout and Board Functions



The picture above shows the pin functions and the core components of the development kit. The core components are,

1. Wisol Module – This is the module that performs the Sigfox protocol stack execution and the transmission and reception functions. It receives inputs from the microcontroller through the serial RX/TX lines.
2. Microcontroller – This runs the user developed firmware that controls data acquisition and communication with the Wisol module. The microcontroller is required as the Wisol module executes AT commands that are sent to it, to provide the functionality.
3. Reset Button – Reset the microcontroller.
4. SMA Antenna – This is a crucial component that is matched to the Wisol module and the appropriate center frequency for the RC zone with an acceptable VSWR. A mismatched antenna can produce significant reflections that degrades transmission and reception of radio waves.

The pins are a mix of the ones available from the microcontroller and from the Wisol module. The intention of the development kit is to enable the user to try out the various features of the Wisol Sigfox module, to be able to prototype Sigfox based applications. The availability of the Wisol pins on the board enables this objective.

The IDE pin numbers refer to the digital pin number used by the Arduino IDE.

The table below refers to the pins and their associated descriptions.

Pin	Function	Type
PB6	General purpose IO (Microcontroller)	I/O
PB7	General purpose IO (Microcontroller)	I/O
MISO, MOSI, SCK (Blue)	General purpose IO, selectable SPI functionality (Wisol)	I/O
GPIO8, GPIO4	General purpose IO (Wisol)	I/O
PE6, PB4, PB5, PD5	General purpose IO (Microcontroller)	I/O
MISO, MOSI, SCK (Grey)	General purpose IO, selectable SPI functionality (Microcontroller)	I/O
AREF	Analog reference (Microcontroller)	Input reference voltage when required.
PF0, PF1, PF4, PF5	General purpose IO, selectable ADC functionality (Microcontroller)	I/O
GPIO0, GPIO1	General purpose IO, selectable ADC functionality (Wisol)	I/O
RESET	Microcontroller reset	I
+3v3, +5V	3.3V and 5V voltage outputs	O

Please note, the pins PB6 can be used as SoftwareSerial RX and PB7 can be used as SoftwareSerial TX. PD2 (RXD1) and PD3 (TXD1) of the Microcontroller are directly connected to the TX and RX of the Wisol module respectively on the board, to enable serial communication from the microcontroller. From the IDE, this is done using serial class called **Serial1**. You use it the same way as the **Serial** class. The **Serial** class is used to communicate with the USB connection to the PC.

*For the Leonardo, it is necessary to execute **while**(!Serial) before proceeding with carrying out any serial communication with the PC.*

7.1. Microcontroller and Wisol module internal connections

Microcontroller	Wisol Module	Remarks
TXD1 (Serial1 on IDE)	RX	Send AT commands to Wisol module.
RXD1 (Serial1 on IDE)	TX	Receive responses from Wisol module
PD6 (Digital pin 12 on IDE)	GPIO9	Toggle this pin to wake up Wisol module from deep sleep. (See notes below for deep sleep)

7.2. Wisol Digital Inputs and Outputs

All digital inputs are Schmitt trigger inputs, digital input and output levels are LVCMOS/LVTTL compatible. All GPIO pins and UARTRX start up as input with pull–up. For explanations on how to use the GPIO pins, see chapter AT Commands List.

8. Absolute Maximum Ratings

Stresses exceeding those listed in the Maximum Ratings table may damage the device. If any of these limits are exceeded, device functionality should not be assumed, damage may occur and reliability may be affected.

Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

8.1. Wisol Module

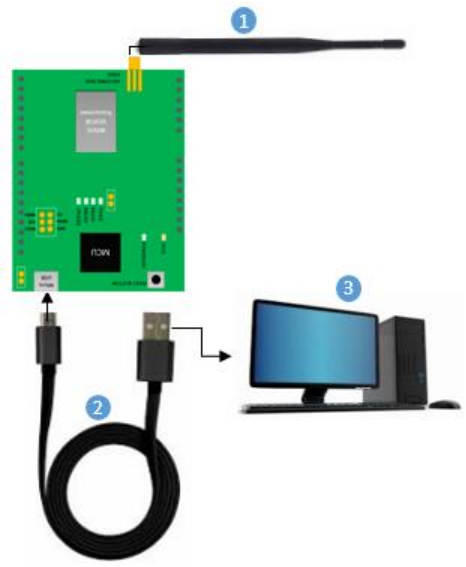
Symbol	Description	Condition	Min	Max	Units
VDD_IO	Supply voltage		-0.5	5.5	V
IDD	Supply current			200	mA
Ptot	Total power consumption			800	mW
Ii1	DC current into any pin except ANTP, ANTN, ANTP1		-10	10	mA
Ii2	DC current into pins ANTP, ANTN, ANTP1		-100	100	mA
IO	Output Current			40	mA
Via	Input voltage ANTP, ANTN, ANTP1 pins		-0.5	5.5	V
	Input voltage digital pins*		-0.5	5.5	V
Ves	Electrostatic handling	HBM	-2000	2000	V
Tamb	Operating temperature		-30	85	°C
Tstg	Storage temperature		-40	125	°C
* Pins GPIO[3:0] must not be driven above 3.3V, all other digital inputs are 5 V tolerant.					

8.2. Microcontroller

Description	Min	Max	Units
DC Current VCC and GND Pins		200	mA
DC Current per I/O Pin		40	mA
Maximum Operating Voltage		6	V
Voltage on RESET with respect to Ground	-0.5	13	V
Voltage on any Pin except RESET and VBUS with respect to Ground	-0.5	5.5	V
Voltage on VBUS with respect to Ground	-0.5	6	V
Operating temperature	-40	85	°C
Storage temperature	-65	150	°C

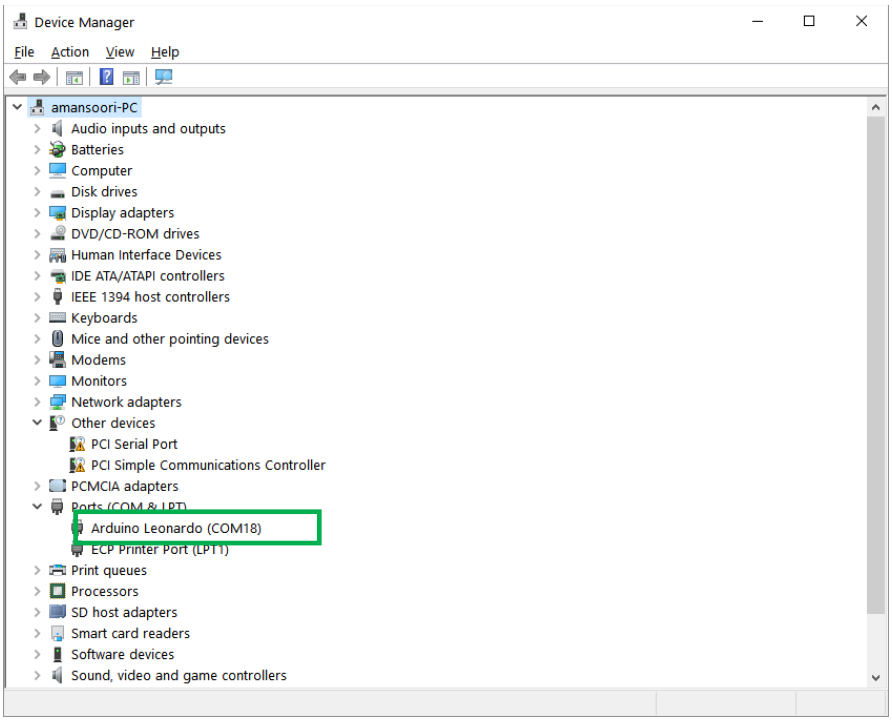
9. Hardware Setup

1. Connect the board to a PC (3) with the USB cable (2).
2. Mount the antenna (1) to the SMA connector on the board.



(1) SMA Antenna
(2) Micro USB cable
(3) PC

Once connected to the PC the USB driver will auto install. Please refer to the driver installation information in this section. Once the driver is installed, you will get a COM port number. To know this COM port, run device manager and click on Ports (COM & LPT) as below. In the example below COM18 is the serial port allocated for the board.



The POWERLED and CPULED will turn on.

10. Current Measurement Jumper JP1

The evaluation board provides a current sensing header JP1 for power consumption measurement of the prototyping board.

To measure current consumption, use a multimeter or other precision current measurement device in series with the two jumper pins.

When not measuring current, use the jumper to close the connection between the two pins.

There is another jumper JP2 to connect/disconnect the LEDs. This is useful when the total current drawn by the prototyping board is to be measured discounting the current drawn by the LEDs.

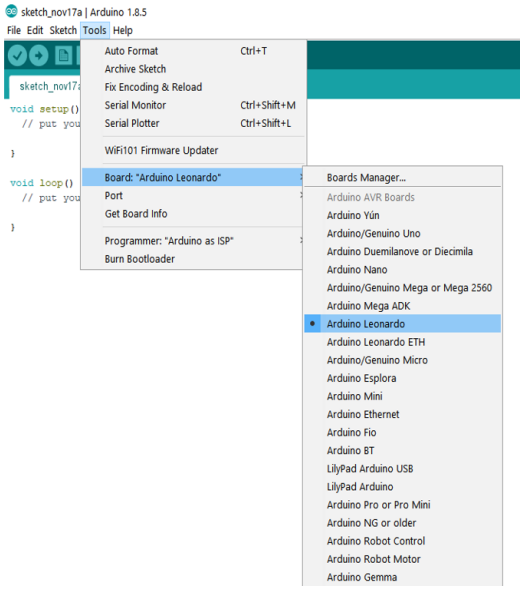
11. Registering the device to the Sigfox Backend

To be able to send data over the Sigfox network to the backend, you will need to associate the device with a valid subscription to the Sigfox service.

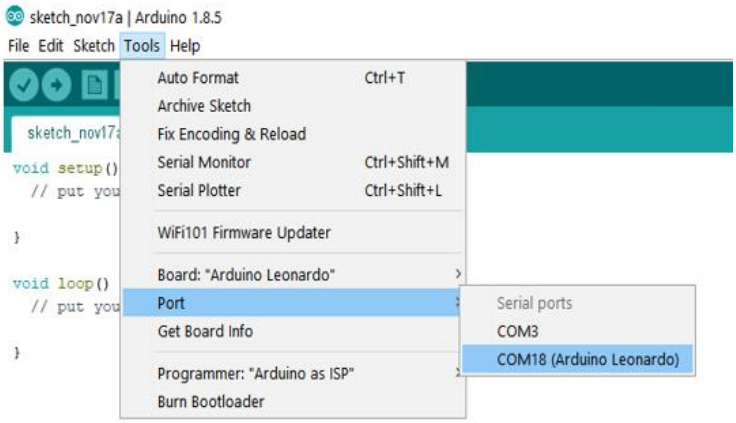
Please follow the steps given at this link to follow through the process.

12. Running your first program

1. After plugging in the device via the USB cable to the PC, start up the Arduino IDE.
2. Click on Tools, select Board and then select Arduino Leonardo, as shown below.



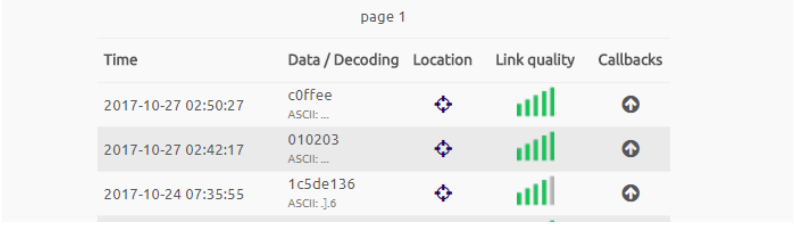
3. Click on Tools, select Port and then select the COM port, as shown below.



4. Initiate Serial communication on the Microcontroller to be able to talk to the Wisol module. It uses a bit rate of 9600 baud, no parity, 8 data bits and one stop bit. Example code provided below,

```
Serial1.begin (9600);
while (!Serial1)
 {
        digitalWrite(LED_BUILTIN, HIGH);
        delay (25);
        digitalWrite(LED_BUILTIN, LOW);
        delay (975);
 }
```

5. Reset the software by sending the command **AT$P=0** over Serial1
6. Get device ID by sending the command **AT$I=10** over Serial1
7. Get device PAC by sending the command **AT$I=11** over Serial1
8. Send the command **AT$CB=-1,0** over Serial1
9. Send the command **AT$RC** over Serial1
10. Send the command **AT$SF=0C000f0f0e0e** over Serial1. This command will send the message in hexadecimal to the Sigfox backend.
11. Log on to the Sigfox backend to confirm message receipt.



Deep Sleep

To put the Wisol module to deep sleep and wake up from deep sleep, perform the following steps,

Put to deep sleep

Write all settings to flash by sending command AT$WR over Serial1
Send the command AT$P=2 over Serial1

Wake up from deep sleep

Put digital pin 12 in output mode (pinMode(12, OUTPUT))
Delay 50 milliseconds
Put digital pin 12 in input mode (pinMode(12, INPUT))

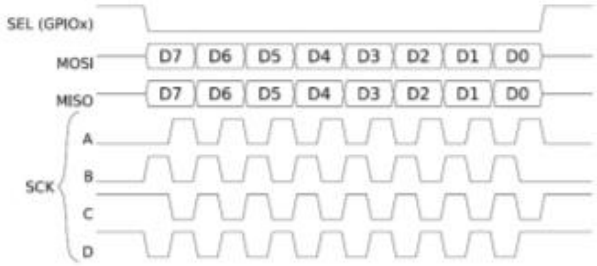
Important Note – It is recommended to wait for an "OK" response from the Wisol module for commands sent to it.

For a complete list of AT commands please refer to this section.

13. AT Commands list

Command	Name	Description
AT	Dummy Command	Just returns 'OK' and does nothing else. Can be used to check communication.
AT$SB=bit[,bit]	Send Bit	Send a bit status (0 or 1). Optional bit flag indicates if AX-SFEU should receive a downlink frame.
AT$SF=frame[,bit]	Send Frame	Send payload data, 1 to 12 bytes. Optional bit flag indicates if AX-SFEU should receive a downlink frame.
AT$SO	Manually send out of band message	Send the out-of-band message.
AT$TR?	Get the transmit repeat	Returns the number of transmit repeats. Default: 2
AT$TR=?	Get transmit range	Returns the allowed range of transmit repeats.
AT$TR=uint	Get transmit repeat	Sets the transmit repeat.
ATSuint?	Get Register	Query a specific configuration register's value. See chapter "Registers" for a list of registers.
ATSuint=uint	Set Register	Change a configuration register.
ATSuint=?	Get Register Range	Returns the allowed range of transmit repeats.
AT$IF=uint	Set TX Frequency	Set the output carrier macro channel for Sigfox frames.
AT$IF?	Get TX Frequency	Get the currently chosen TX frequency.
AT$DR=uint	Set RX Frequency	Set the reception carrier macro channel for Sigfox frames.
AT$DR?	Get RX Frequency	Get the currently chosen RX frequency.
AT$CW=uint,bit[,uint_opt]	Continuous Wave	To run emission tests for Sigfox certification it is necessary to send a continuous wave, i.e. just the base frequency without any modulation. Parameters: **Name** / **Range** / **Description** Frequency / 800000000–999999999, 0 / Continuous wave frequency in Hz. Use 868130000 for Sigfox or 0 to keep previous frequency. Mode / 0, 1 / Enable or disable carrier wave. Power / 0–14 / dBm of signal | Default: 14
AT$CB=uint_opt,bit	Test Mode: TX constant byte	For emission testing it is useful to send a specific bit pattern. The first parameter specifies the byte to send. Use '–1' for a (pseudo–)random pattern. Parameters: **Name** / **Range** / **Description** Pattern / 0–255, –1 / Byte to send. Use '–1' for a (pseudo–)random pattern. Mode / 0, 1 / Enable or disable pattern test mode.
AT$T?	Get Temperature	Measure internal temperature and return it in 1/10th of a degree Celsius.
AT$V?	Get Voltages	Return current voltage and voltage measured during the last transmission in mV.

Command	Name	Description
AT$I=uint	Information	Display various product information: 0: Software Name & Version Example Response: AX-SFEU 1.0.6-ETSI 1: Contact Details Example Response: support@axsem.com 2: Silicon revision lower byte Example Response: 8F 3: Silicon revision upper byte Example Response: 00 4: Major Firmware Version Example Response: 1 5: Minor Firmware Version Example Response: 0 7: Firmware Variant (Frequency Band etc. (EU/US)) Example Response: ETSI 8: Firmware VCS Version Example Response: v1.0.2-36 9: SIGFOX Library Version Example Response: DL0-1.4 10: Device ID Example Response: 00012345 11: PAC Example Response: 0123456789ABCDEF
AT$P=uint	Set Power Mode	To conserve power, the AX-SFEU can be put to sleep manually. Depending on power mode, you will be responsible for waking up the AX-SFEU again! 0: software reset (settings will be reset to values in flash) 1: sleep (send a break to wake up) 2: deep sleep (toggle GPIO9 or RESET_N pin to wake up; the AX-SFEU is not running and all settings will be reset!)
AT$WR	Save Config	Write all settings to flash (RX/TX frequencies, registers) so they survive reset/deep sleep or loss of power. Use AT$P=0 to reset the AX-SFEU and load settings from flash.
AT:Pn?	Get GPIO Pin	Return the setting of the GPIO Pin *n*; *n* can range from 0 to 9. A character string is returned describing the mode of the pin, followed by the actual value. If the pin is configured as analog pin, then the voltage (range 0…1 V) is returned. The mode characters have the following meaning: **Mode** **Description** 0 Pin drives low 1 Pin drives high Z Pin is high impedance input U Pin is input with pull-up A Pin is analog input (GPIO pin 0…3 only) T Pin is driven by clock or DAC (GPIO pin 0 and 4 only) The default mode after exiting reset is U on all GPIO pins.
AT:Pn=?	Get GPIO Pin Range	Print a list of possible modes for a pin. The table below lists the response. **Pin** **Modes** P0 0, 1, Z, U, A, T P1 0, 1, Z, U, A P2 0, 1, Z, U, A P3 0, 1, Z, U, A P4 0, 1, Z, U, T P5 0, 1, Z, U P6 0, 1, Z, U P7 0, 1, Z, U P8 0, 1, Z, U P9 0, 1, Z, U
AT:Pn=mode	Set GPIO Pin	Set the GPIO pin mode. For a list of the modes see the command AT:Pn?

Command	Name	Description
AT:ADC Pn[-Pn[(1V|10V)]]?	Get GPIO Pin Analog Voltage	Measure the voltage applied to a GPIO pin. The command also allows measurement of the voltage difference across two GPIO pins. In differential mode, the full scale range may also be specified as 1 V or 10 V. Note however that the pin input voltages must not exceed the range 0..VDD_IO. The command returns the result as fraction of the full scale range (1 V if none is specified). The GPIO pins referenced should be initialized to analog mode before issuing this command.
AT:SPI[(A|B|C|D)]=bytes	SPI Transaction	This command clocks out *bytes* on the SPI port. The clock frequency is 312.5 kHz. The command returns the bytes read on MISO during output. Optionally the clocking mode may be specified (default is A): **Mode** **Clock Inversion** **Clock Phase** A normal normal B normal alternate C inverted normal D inverted alternate  Note that SEL, if needed, is not generated by this command, and must instead be driven using standard GPIO commands (AT:Pn=0|1).
AT:CLK=freq,reffreq	Set Clock Generator	Output a square wave on the pin(s) set to T mode. The frequency of the square wave is (freq / 2^{16}) × reffreq. Possible values for reffreq are 20000000, 10000000, 5000000, 2500000, 1250000, 625000, 312500, 156250. Possible values if freq are 0…65535.
AT:CLK=OFF	Turn off Clock Generator	Switch off the clock generator
AT:CLK?	Get Clock Generator	Return the settings of the clock generator. Two numbers are returned, freq and reffreq.
AT:DAC=value	Set ΣΔ DAC	Output a ΣΔ DAC value on the pin(s) set to T mode. Parameter value may be in the range –32768…32767. The average output voltage is (1/2 + value / 2^{17}) × VDD. An external low pass filter is needed to get smooth output voltages. The modulation frequency is 20 MHz. A possible low pass filter choice is a simple RC low pass filter with R = 10 kΩ and C = 1 µF.
AT:DAC=OFF	Turn off ΣΔ DAC	Switch off the DAC
AT:DAC?	Get ΣΔ DAC	Return the DAC value

Command	Name	Description
AT$TM=mode,config	Activates the Sigfox Testmode	Available test modes: 0. TX BPSK Send only BPSK with Synchro Bit + Synchro frame + PN sequence: No hopping centered on the TX_frequency. Config bits 0 to 6 define the number of repetitions. Bit 7 of config defines if a delay is applied of not in the loop 1. TX Protocol: Tx mode with full protocol with Sigfox key: Send Sigfox protocol frames with initiate downlink flag = True. Config defines the number of repetitions. 2. RX Protocol: This mode tests the complete downlink protocol in Downlink only. Config defines the number of repetitions. 3. RX GFSK: RX mode with known pattern with SB + SF + Pattern on RX_frequency (internal comparison with received frame ⇔ known pattern = AA AA B2 27 1F 20 41 84 32 68 C5 BA AE 79 E7 F6 DD 9B. Config defines the number of repetitions. Config defines the number of repetitions. 4. RX Sensitivity: Does uplink + downlink frame with Sigfox key and specific timings. This test is specific to SIGFOX's test equipments & softwares. 5. TX Synthesis: Does one uplink frame on each Sigfox channel to measure frequency synthesis step
AT$SE	Starts AT$TM=3,255 indefinitely	Convenience command for sensitivity tests
AT$SL[=frame]	Send local loop	Sends a local loop frame with optional payload of 1 to 12 bytes. Default payload: 0x84, 0x32, 0x68, 0xC5, 0xBA, 0x59, 0xAE, 0x79, 0xE7, 0xF6, 0xDD, 0x9B.
AT$RL	Receive local loop	Starts listening for a local loop.