Features

- Fully integrated, single-chip RF transceiver (SIGFOX compliant)
- Based on WISOL SFM10R1 chip
- Small dimensions 24.31 x 14.97 mm
- Controlled by simple AT commands
- Only 5 wire connection
- U.FL and DuPont compatible
- System-on-chip solution including SIGFOX related protocol handling for modem operation
- ON® microcontroller core with embedded firmware, SIGFOX, protocol stack and ID/PAC
- Supports up- and downlink operation, i.e., transmit and receive of data telegrams with SIGFOX base stations in EU
- Typical operating frequency uplink 868.130MHz, downlink 869.525MHz
- Low current consumption 65mA during transmit and 15mA during receive operation
- Typical sleep mode current 2µA at VCC +3.3V and +25°C
- UART interface for data access and transceiver configuration and control
- Supply voltage ranges from 1.8V to 3.6V
- Temperature range –30°C to +85°C

Application

Applications

SIGFOX™ compatible modem for long-range, low-power and low-cost applications using the SIGFOX network

- Home and building automation
- Alarm and security systems
- Smart environment and industrial
- Smart parking
- Tracking
- Metering

1. General Description

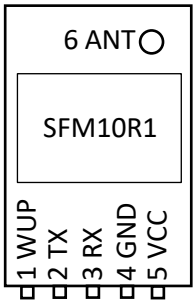
1.1. Introduction

The Iottoall Sigfox Devkit 868 is a highly integrated, low-power RF transceiver with an integrated ON® microcontroller for applications using the wide area SIGFOX™ network.

The Iottoall Sigfox Devkit 868 is partitioned into three sections: an RF front end, a digital baseband and the low power microcontroller. The product is designed for the EU ISM frequency band in the range of 868.0MHz to 868.6MHz and 869.4MHz to 869.65MHz. The external part count is kept to a minimum due to the very high level of integration in this device. By combining outstanding RF performance with highly sophisticated baseband signal processing, robust wireless communication can be easily achieved.

The UART interface enables external control and device configuration.

1.2. Pinning



Pin No.	Pin Name	Description
1	**WUP**	**Wake Up (GPIO9)**
2	**TX**	**UART TX output**
3	**RX**	**UART RX input**
4	**GND**	**Power ground**
5	**VCC**	**Power VCC**
6	**ANT**	**Antenna input and output**

UART configuration is 9600baud, 8 data bits, 1 stop bit, no parity, and no flow control.

1.3. Applications

This section provides application examples for the Iottoall Sigfox Devkit device.

1.3.1. Example A

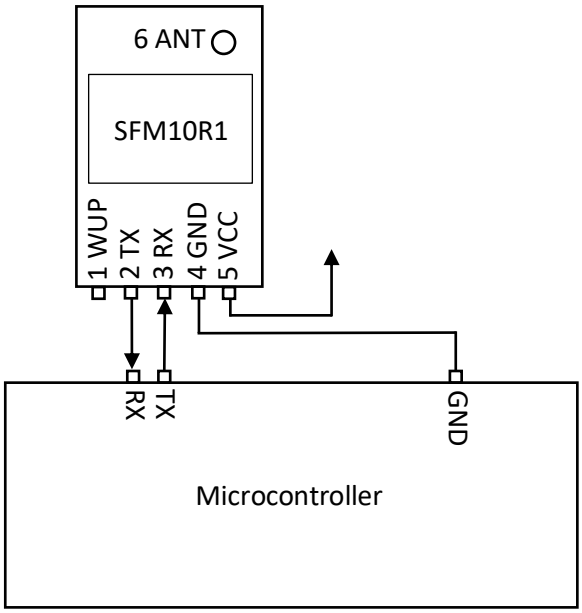
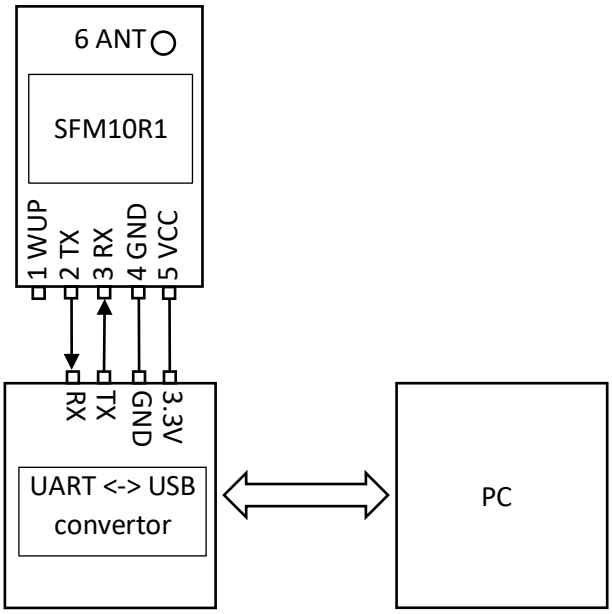


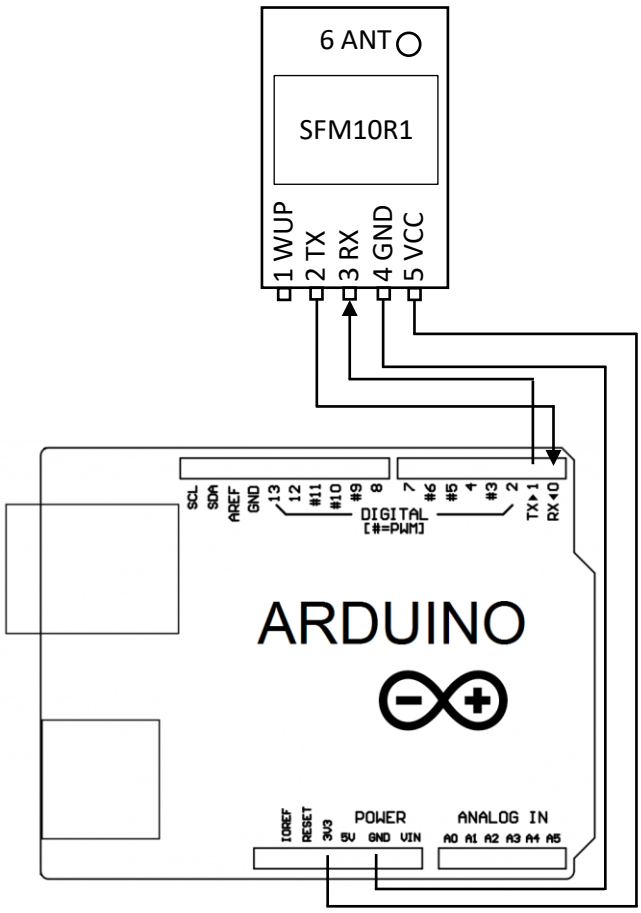
Figure shows basic Iottoall Sigfox Devkit connection to generic microcontroller. In this case the microcontroller sends AT commands to devkit directly thru the UART interface (9600baud). Is recommended to use full duplex UART. In case of using **half duplex**, AT commands has to be ended **only** with **one** of **'\r' or '\n' not both**. Because if you send "AT\r\n" the Sigfox devkit starts sending "OK" instantly after it receive '\r', but microcontroller is still sending byte '\n'.

1.3.2. Example B



This example shows connection between Iottoall Sigfox Devkit and computer. In this case is used UART to USB convertor, whose driver creates virtual COM port in computer operating system. Thru this port is possible to send AT commands to the Sigfox Devkit. Communication speed is 9600baud. **AT** commands has to be written in **upper case**.

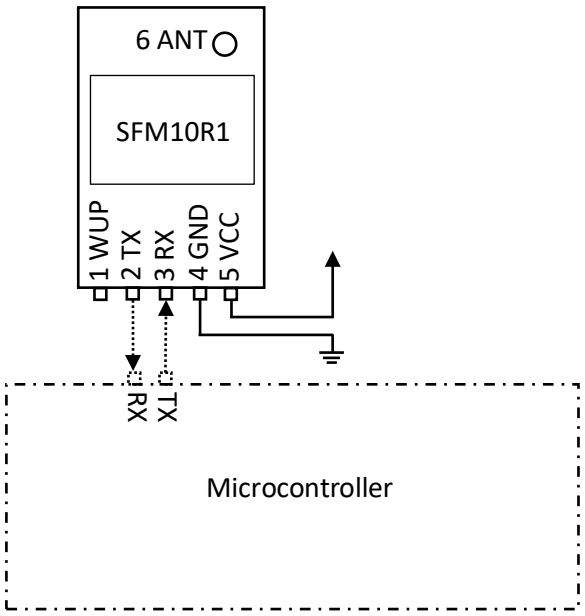
1.3.3. Example C



This is the simplest connection with the shortest code that is needed. After power supply connected to Arduino board message 0x01020304 will sent to Sigfox network. Iottoall Sigfox Devkit (RX and TX pins) must be **disconnected when you are downloading program to Arduino** in this example.

```
void setup(void){
 Serial.begin(9600);
 Serial.println("AT$SF=01020304");
}
void loop(void){
}
```

1.3.4. Example D



This example shows how to use Iottoall Sigfox Devkit to automatically send out of band messages. In this case microcontroller is not required (only for configuration). This messages contains temperature of the module, voltage on VCC in standby mode and voltage on VCC in TX mode. In default configuration Iottoall Sigfox Devkit sends this out of band message each 24 hours. Is possible to modify this interval to another value in interval 0 to 24. Changing this value is possible by following AT commands:

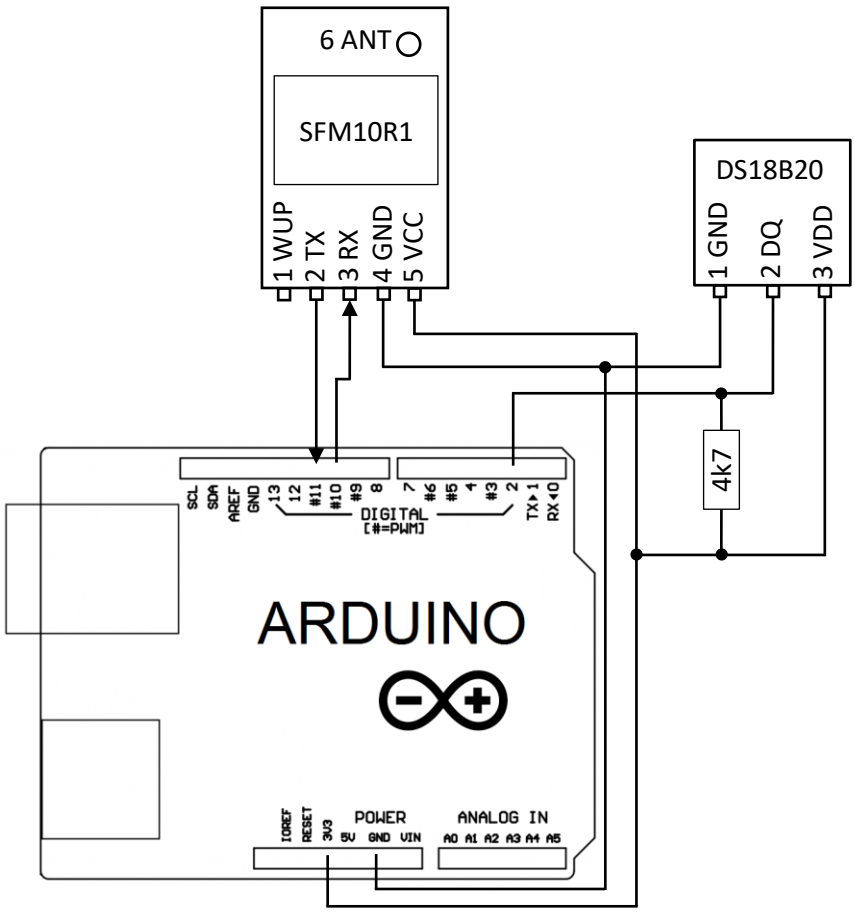
AT command	Description
ATS300=1	Change interval of sending out of band message to one hour. Set to 0 to disable sending of this message.
AT$WR	Save modified configuration to flash.

Arduino program for this example. It sends AT command for change interval.

```
void setup(void){
 Serial.begin(9600);
 Serial.println("ATS300=1");
 Serial.println("AT$WR");
}
void loop(void){
}
```

After program is downloaded to the microcontroller, you can disconnect this microcontroller. After download Iottoall Sigfox Devkit need only power source. Please keep in mind that sleep mode is not used in this case.

1.3.5. Example E



This image depicts connection of Iottoall Sigfox Devkit to the Arduino for measuring temperature by sensor DS18B20. Communication between Arduino and Sigfox Devkit is achieved by SoftwareSerial library (pins D10 and D11), therefore is hardware UART (pins D0 and D1) free for communication between Arduino and computer. Temperature sensor DS18B20 use OneWire bus that is initialized at pin D2.

The following code for Arduino reads temperature every 11 minutes and send it to the Sigfox network.

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <SoftwareSerial.h>

// Data wire is plugged into port 2 on the Arduino
#define ONE_WIRE_BUS 2

// Setup a oneWire instance to communicate with any OneWire devices
OneWire oneWire(ONE_WIRE_BUS);
// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);
```

```
SoftwareSerial mySerial(10, 11); // RX, TX

void measure(){
 //Send the command to get temperatures
 sensors.requestTemperatures();
 char str[20];
 float t = sensors.getTempCByIndex(0);
 int ti = (int)t;
 int td = (((int)(t*100))%100);
 sprintf(str, "AT$SF=%02X%02X\n", ti, td);
 Serial.print(str);
 mySerial.print(str);
}

void setup(void){
 // start serial port
 Serial.begin(9600);
 Serial.setTimeout(10);

 mySerial.begin(9600);
 mySerial.setTimeout(10);

 // Start up the library
 sensors.begin();

 measure();
}

void loop(void){
 if(mySerial.available()){
  Serial.print(mySerial.readString());
 }

 if(Serial.available()){
  String cmd = Serial.readString();
  cmd.trim();
  if(cmd == "measure"){
   measure();
  }else{
   //do not use println because it sends \r\n and while
   //sending \n the sigfox module is already sending response
   //and software serial has only half duplex
   mySerial.print(cmd);
   mySerial.print("\n");
  }
 }
```

```
 static unsigned long last = 0;
 if((millis() - last) > 660000){
   last = millis();
   measure();
  }
}

 static unsigned long last = 0;
 if((millis() - last) > 660000){
   last = millis();
   measure();
  }
}
```

2. System Functional Description

2.1. AT command examples

This section explain basic AT commands.

2.1.1. Reading ID and PAC example

AT command	Description
AT$I=10	Return device ID
AT$I=11	Return PAC

2.1.2. Sending data example

AT command	Description
AT$SF=10AA	Send value 0x10AA to Sigfox network. Returns “OK”.
AT$SF=10AA,1	Send 0x10AA with downlink request. Returns “OK” and “RX=00 00 00 00 00 00 00 00”, where “00” represents received data.

2.1.3. Measuring

AT command	Description
AT$V?	Return current volatage and voltage measured during the last transmission in mV.
AT$T?	Get internal temperature in 1/10th of a degree Celsius.

2.1.4. Sleep mode

AT command	Description
AT$P=1	Enter sleep mode. Send a break (‘\n’) to wake up.
AT$P=2	Enter deep sleep mode. Make power reset module to wake up.

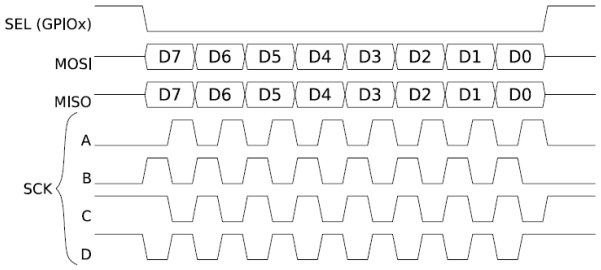
2.2. UART AT Command Interface

The UART AT command interface provides a set of commands to control the operation of the Iottoall Sigfox Devkit.

<table>
<tr><th>AT command</th><th>Name</th><th>description</th></tr>
<tr><td>AT</td><td>Dummy Command</td><td>Just return ‘OK’ and does nothing else. Can be used to check communication.</td></tr>
<tr><td>AT$SB=bit[,bit]</td><td>Send Bit</td><td>Send a bit status (0 or 1). Optional bit flag indicates if AX-SFEU should receive a downlink frame.</td></tr>
<tr><td>AT$SF=frame[,bit]</td><td>Send Frame</td><td>Send payload data, 1 to 12 bytes. Optional bit flag indicates if AX-SFEU should receive a downlink frame.</td></tr>
<tr><td>AT$SO</td><td>Manually send out of band message</td><td>Send the out-of-band message.</td></tr>
<tr><td>AT$TR?</td><td>Get the transmit repeat</td><td>Returns the number of transmit repeats.</td></tr>
<tr><td>AT$TR=uint</td><td>Set transmit repeat</td><td>Sets the transmit repeat.</td></tr>
<tr><td>ATSuint?</td><td>Get Register</td><td>Query a specific configuration register’s value. See chapter “Registers” for a list of registers.</td></tr>
<tr><td>ATSuint=uint</td><td>Set Register</td><td>Change a configuration register.</td></tr>
<tr><td>ATSuint=?</td><td>Get Register Range</td><td>Returns the allowed range of registers.</td></tr>
<tr><td>AT$IF=uint</td><td>Set TX Frequency</td><td>Set the output carrier macro channel for Sigfox frames.</td></tr>
<tr><td>AT$IF?</td><td>Get TX Frequency</td><td>Get the currently chosen TX frequency.</td></tr>
<tr><td>AT$DR=uint</td><td>Set RX Frequency</td><td>Set the reception carrier macro channel for Sigfox frames.</td></tr>
<tr><td>AT$DR?</td><td>Get RX Frequency</td><td>Get the currently chosen RX frequency.</td></tr>
<tr><td>AT$CW=uint,bit[, uint_opt]</td><td>Continuous Wave</td><td>The run emission tests for Sigfox certification it is necessary to send a continuous wave, i.e. just the base frequency without any modulation. Parameters:
<table>
<tr><th>Name</th><th>Range</th><th>Description</th></tr>
<tr><td>Frequency</td><td>800000000-999999999, 0</td><td>Continuous wave frequency in Hz. Use 868130000 for Sigfox or 0 to keep previous frequency.</td></tr>
<tr><td>Mode</td><td>0, 1</td><td>Enable or disable carrier wave.</td></tr>
<tr><td>Power</td><td>0-14</td><td>dBm of signal | Default: 14</td></tr>
</table></td></tr>
<tr><td>AT$CB=uint_opt, bit</td><td>Test Mode: TX constant byte</td><td>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:</td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Range</th><th>Description</th></tr>
<tr><td>Pattern</td><td>0-255, -1</td><td>Byte to send. Use ‘-1’ for a (pseudo-)random pattern.</td></tr>
<tr><td>Mode</td><td>0, 1</td><td>Enable or disable pattern test mode.</td></tr>
</table>
</td></tr>
<tr><td>AT$T?</td><td>Get Temperature</td><td>Measure internal temperature and return it in 1/10th of a degree Celsius.</td></tr>
<tr><td>AT$V?</td><td>Get Voltages</td><td>Return current voltage and voltage measured during the last transmission in mV.</td></tr>
<tr><td>AT$I=uint</td><td>Information</td><td>Display various product information:
<table>
<tr><td>0:</td><td>Software Name & Version
Example Response: AX-SFEU 1.0.6-ETSI</td></tr>
<tr><td>1:</td><td>Contact Detail
Example Response: info@lpwan.cz</td></tr>
<tr><td>2:</td><td>Silicon revision lower byte
Example Response: 8F</td></tr>
<tr><td>3:</td><td>Silicon revision upper byte
Example Response: 00</td></tr>
<tr><td>4:</td><td>Major Firmware Version
Example Response: 1</td></tr>
<tr><td>5:</td><td>Minor Firmware Version
Example Response: 0</td></tr>
<tr><td>7:</td><td>Firmware Variant (Frequency Band etc. (EU/US))
Example Response: ETSI</td></tr>
<tr><td>8:</td><td>Firmware VCS Version
Example Response: V1.0.2-36</td></tr>
<tr><td>9:</td><td>SIGFOX Library Version
Example Response: DL0-1.4</td></tr>
<tr><td>10:</td><td>Device ID
Example Response: 00012345</td></tr>
<tr><td>11:</td><td>PAC
Example Response: 0123456789ABCDEF</td></tr>
</table>
</td></tr>
<tr><td>AT$P=uint</td><td>Set Power Mode</td><td>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!
<table>
<tr><td>0:</td><td>Software reset (settings will be reset to values in flash)</td></tr>
<tr><td>1:</td><td>Sleep (send a break to wake up)</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td></td><td><table><tr><td>2:</td><td>Deep sleep (toggle WUP (GPIO9) or RESET_N pin to wake up; the AX-SFEU is not running and all settings will be reset!)</td></tr></table></td></tr>
<tr><td>AT$WR</td><td>Save Config</td><td>Write all settings to flash (RX/TX frequencies, registers) so they survive reset/deep sleep or loss of power.
Use AT$P=0 to reset the AX-SFEU and load settings from flash.</td></tr>
<tr><td>AT:Pn?</td><td>Get GPIO Pin*</td><td>Return the settings 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:<table><tr><td>Mode</td><td>Description</td></tr><tr><td>0</td><td>Pin drives low</td></tr><tr><td>1</td><td>Pin drives high</td></tr><tr><td>Z</td><td>Pin is high impedance input</td></tr><tr><td>U</td><td>Pin is input with pull-up</td></tr><tr><td>A</td><td>Pin is analog input (GPIO pin 0…3 only)</td></tr><tr><td>T</td><td>Pin is driven by clock or DAC (GPIO pin 0 and 4 only)</td></tr></table>The default mode after exiting reset is U on all GPIO pins.</td></tr>
<tr><td>AT:Pn=?</td><td>Get GPIO Pin Range*</td><td>Print a list of possible modes for a pin. The table below lists the response.<table><tr><td>Pin</td><td>Mode</td></tr><tr><td>P0</td><td>0, 1, Z, U, A, T</td></tr><tr><td>P1</td><td>0, 1, Z, U, A</td></tr><tr><td>P2</td><td>0, 1, Z, U, A</td></tr><tr><td>P3</td><td>0, 1, Z, U, A</td></tr><tr><td>P4</td><td>0, 1, Z, U, T</td></tr><tr><td>P5</td><td>0, 1, Z, U</td></tr><tr><td>P6</td><td>0, 1, Z, U</td></tr><tr><td>P7</td><td>0, 1, Z, U</td></tr><tr><td>P8</td><td>0, 1, Z, U</td></tr><tr><td>P9</td><td>0, 1, Z, U</td></tr></table></td></tr>
<tr><td>AT:Pn=mode</td><td>Set GPIO Pin*</td><td>Set the GPIO pin mode.
For a list of the modes see the command AT:Pn?</td></tr>
<tr><td>AT:ADC Pn[–Pn [(1V|10V)]]?</td><td>Get GPIO Pin Analog Voltage*</td><td>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</td></tr>
</table>

<table>
<tr><td></td><td></td><td>specified). The GPIO pins referenced should be initialized to analog mode before issuing this command.</td></tr>
<tr><td>AT:SPI[(A|B|C|D)]=bytes</td><td>SPI Transaction*</td><td>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):
<table>
<tr><th>Mode</th><th>Clock Inversion</th><th>Clock Phase</th></tr>
<tr><td>A</td><td>Normal</td><td>Normal</td></tr>
<tr><td>B</td><td>Normal</td><td>Inverted</td></tr>
<tr><td>C</td><td>Inverted</td><td>Normal</td></tr>
<tr><td>D</td><td>inverted</td><td>Inverted</td></tr>
</table>


Note that SEL, if needed, is not generated by this command, and must instead be driven using standard GPIO commands (AT:Pn=0|1).</td></tr>
<tr><td>AT:CLK=freq,reffreq</td><td>Set Clock Generator*</td><td>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.</td></tr>
<tr><td>AT:CLK=OFF</td><td>Turn off Clock Generator*</td><td>Switch off the clock generator</td></tr>
<tr><td>AT:CLK?</td><td>Get Clock Generator*</td><td>Return the settings of the clock generator. Two numbers are returned, freq and reffreq.</td></tr>
<tr><td>AT:DAC=value</td><td>Set ΣΔ DAC*</td><td>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.</td></tr>
<tr><td>AT:DAC=OFF</td><td>Turn off ΣΔ DAC*</td><td>Switch off the DAC</td></tr>
<tr><td>AT:DAC?</td><td>Get ΣΔ DAC*</td><td>Return the DAC value</td></tr>
<tr><td>AT$TM=mode,config</td><td>Activates the Sigfox Testmode</td><td>Available test modes:
<table>
<tr><td>0.</td><td>TX BPSK</td></tr>
</table></td></tr>
</table>

		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.
		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$SL[=frame]	Send local loop	Sends a local loop frame with optional payload of 1 to 12 bytes. Default payload: 0x84, 0x32, 0x68, 0xC5, 0xBA, 0x53, 0xAE, 0x79, 0xE7, 0xF6, 0xDD, 0x9B.
AT$RL	Receive local loop	Starts listening for a local loop.

* not applicable on Iottoall Sigfox Devkit, there is no GPIO pins connected

Registers

Number	Name	Description	Default	Range	Units
300	Out Of Band Period	AX−SFEU sends periodic static messages to indicate that they are alive. Set to 0 to disable.	24	0-24	Hours
302	Power Level	The output power of the radio.	14	0-14	dBm

3. ELECTRICAL CHARACTERISTIC

2.3. Absolute Maximum Ratings

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Symbol	Parameter	Rating	Unit
VCC	Module input voltage	-0.5 to 5.5	V
OT	Operating Temperature	-30 to +85	°C
ST	Storage Temperature	-40 to +125	°C

2.4. DC Characteristics

Symbol	Parameter	Min	Typ.	Max	Unit
VCC	Module input voltage	1.8	3.3	3.6	V
Current	Tx Current (@"15" setting, CW)	-	65	-	mA
	Tx Current (@"14" setting, CW)	-	54	-	mA
	Rx Current	-	15	-	mA
	Sleep Current	-	2	-	µA
	Standby current	-	500	-	µA

2.5. I/O Specifications

Symbol	Parameter	Min	Typ.	Max	Unit
VIH	High level input voltage @VCC=3.3V	2	-	-	V
VIL	High level input voltage @VCC=3.3V	-	-	0.8	V

2.6. RF Specifications

Conditions: VCC=3.3V, Temp=25°C

Parameter	Min	Typ.	Max	Unit
RF Frequency TX		868.130		MHz
RF Frequency RX		869.525		MHz
Tx output power (at "15" setting)	12.5	13.5	15.5	dBm
Tx output power (at "14" setting)	11.5	12.5	14.5	dBm
Frequency Error Tolerance (+25°C)	-2.5	-	+2.5	ppm
2nd Harmonics (conducted)	-	-37	-35	dBm
3nd Harmonics (conducted)	-	-41	-35	dBm
Rx Sensitivity (@600bps, GFSK)	-127	-		dBm
Rx Spurious Emission (30MHz to 12.75GHz)			-54	dBm

4. DRAWING