ABIT CORPORATION

AL-050

User's Manual

V.1.5

1. Introduction 1
1.1. Hardware Characteristics 1
1.2. How to Use AL-050 (Arduino Shield) 4
2. Command Reference 6
2.1. Command Format 6
2.2. Module Commands 6
2.2.1. mod factory_reset 6
2.2.2. mod get_ver 6
2.2.3. mod get_hw_model 6
2.2.4. mod sleep <Deep> <INTwake> <Period> 7
2.2.5. mod set_baudrate <Baudrate> 7
2.2.6. mod set_echo <Status> 7
2.2.7. mod reset 8
2.2.8. mod save 8
2.3. LoRaWAN Commands 9
2.3.1. lorawan join <Mode> 10
2.3.2. lorawan get_join_status 10
2.3.3. lorawan tx <Type> <PortNum> <Data> 11
2.3.4. lorawan set_dr <DataRate> 11
2.3.5. lorawan set_adr <State> 11
2.3.6. lorawan set_txretry <RetryCount> 12
2.3.7. lorawan set_battery <BatteryLevel> 12
2.3.8. lorawan save 12
2.3.9. lorawan get_devaddr 12
2.3.10. lorawan get_deveui 12
2.3.11. lorawan get_dr 12
2.3.12. lorawan get_pwr 12
2.3.13. lorawan get_adr 13
2.3.14. lorawan get_txretry 13
2.3.15. lorawan get_rxdelay 14
2.3.16. lorawan get_rx2 14
2.3.17. lorawan get_ch_para <ChannelId> 14
2.3.18. lorawan get_ch_status <ChannelId> 14
2.3.19. lorawan get_dc_ctl 14

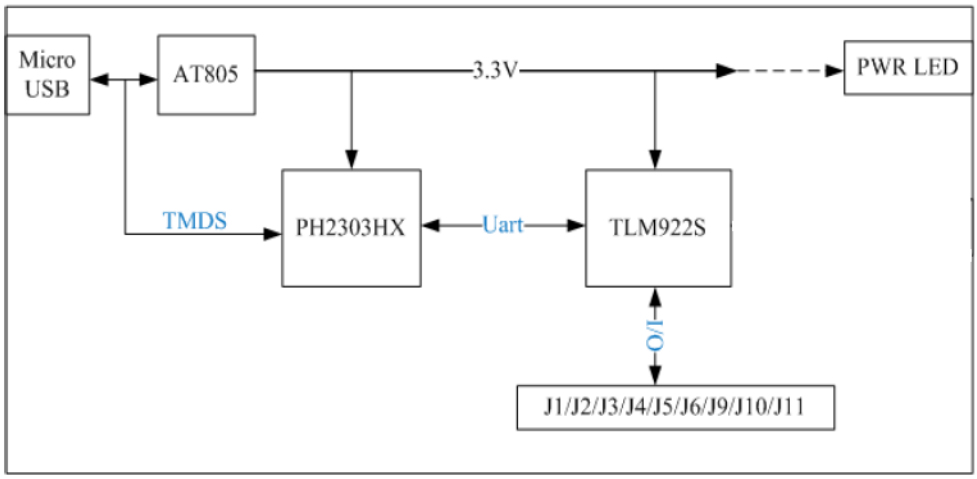
2.3.20. lorawan get_join_ch 15
2.3.21. lorawan get_upcnt 15
2.3.22. lorawan get_downcnt 15
2.3.23. lorawan get_class 15
2.3.24. lorawan get_pwr_table <Index> 15
2.3.25. lorawan get_dl_freq <ChannelID> 16
2.3.26. lorawan get_battery 16
3. Example 17
3.1. Arduino Sample Program 17
3.2. LoRaWAN 17
3.2.1. ABP and Uplink (*Currently, ABP is not available) 17
3.2.2. OTA and Uplink 17
3.2.3. Confirmed Uplink and Downlink 18

1. Introduction

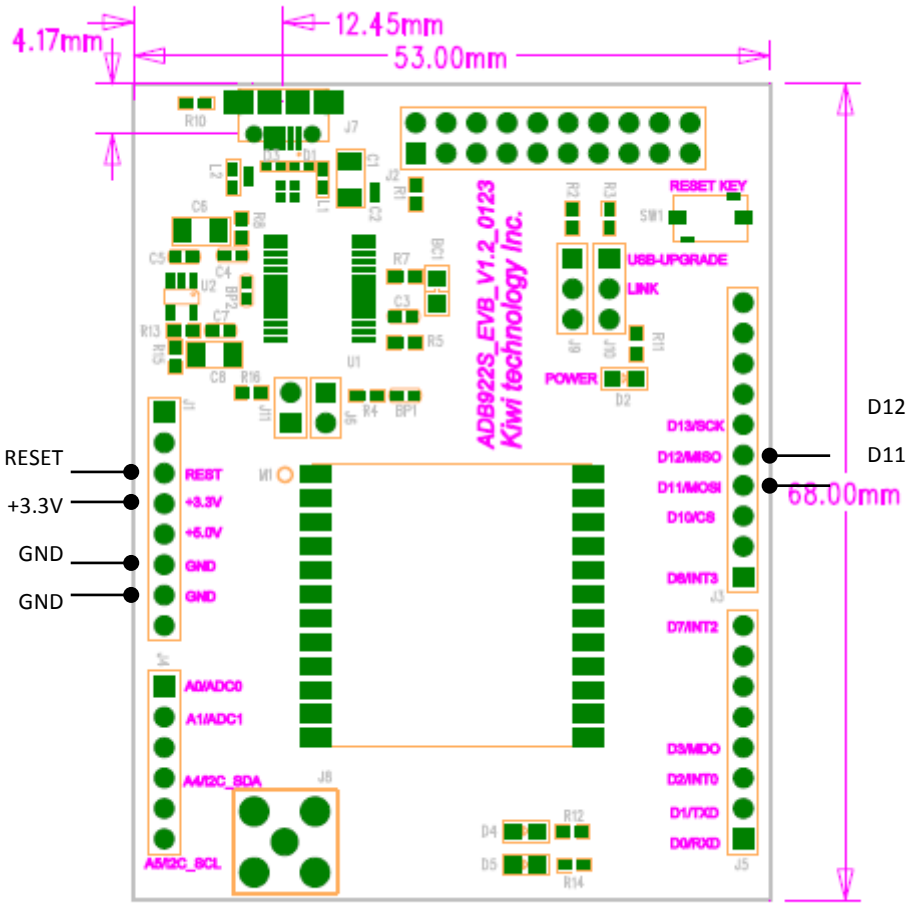
ABiT AL-050 is an Arduino shield, with Kiwitec TLM922S, which you can get started quickly to develop LoRaWAN nodes. TLM922S is a small size and ultra-low power UHF wireless module based on LoRa technology of Semtech. TLM922S module integrates a 32bit MCU, Cypress S6E1C31, and a single-chip radio transmitter, Semtech SX1272, designed for high performance at very low-power and low-voltage operation in cost-effective wireless systems. All filters are integrated, thus removing the need for costly external SAW and IF filters. The device is mainly intended for the ISM (**I**ndustrial, **S**cientific, and **M**edical) frequency bands at 862-932 MHz. The module integrated many RF functions and PA to make the maximum output power up to +20dBm and signal coverage can reach up 10km. (These frequency, power and coverage may vary by region or country.)

1.1. Hardware Characteristics

PCB Block Diagram



PCB Description



Pin Configuration

PIN NAME	Description
GND	System ground
GND	System ground
+3.3V	3.3V Power Input
D11	UART interface, UART_TX
D12	UART interface, UART_RX
RESET	External Reset Input pin with internal RC timing control circuit, active low

Note:

1. The module power supply voltage range is DC 3.0～3.6V, above DC 3.6V, the module will damage. It is recommended work at DC 3.3 V.
2. Other pin headers are open and not used.

General Characteristics

Parameter	Typical	Condition/Note
Operating supply voltage	DC 2.2 ~ 3.6V	
Frequency	920.6MHz~928MHz	
Frequency accuracy	±10KHz	
Modulation	LoRa/GFSK/FSK/OOK/MSK	Programmable
Transmit power	+2 ~ +13+-0.5dBm	Output power programmable
TX current consumption	< 110mA	Po= 13dBm (JP) (TLMS922S's current consumption < 80mA)
Deep Sleep State current consumption	< 20mA	Refer to IC operation states (TLMS922S's current consumption < 3uA)
Data rate	0.244～18.2Kbps(LoRa) 300Kbps(FSK)	Programmable
Spurious emissions and harmonics	< -30dBm	TX power +20dBm (US)
Communication distance	10Km	0.244Kbps Baud data rata, BW=125K Output power = +20dBm. (US)
Antenna impedance	50ohm	
Operating temperature	0°C ~ +40 °C	
Storage temperature range	-10°C ~ +50°C	
Dimension	68mm×53mmx22.8mm	

Test operating conditions：Ta=25°C, VCC=3.3V if nothing else stated.

Note：

1. The module transmission data rate will affect transmission distance, the higher the data rate, the closer the distance, and the lower the receiving sensitivity.
2. The supply voltage to the module will affect TX power, in the operating supply voltage range, the lower the voltage to get the lower the TX power.
3. Please connect the included antenna correctly. Modifying or replacing the antenna is prohibited. Do not attempt to modify the shield, the module and the firmware.

1.2. How to Use AL-050 (Arduino Shield)

AL-050 Arduino Shield equips with a USB-UART bridge and can be powered by USB. Figure 1 and following table depict this module and its connector.

1. USB Device Connector	4. TLM922S
2. Reset Button	5. Antenna Connector
3. UART TX/RX LED (D4, D5)	

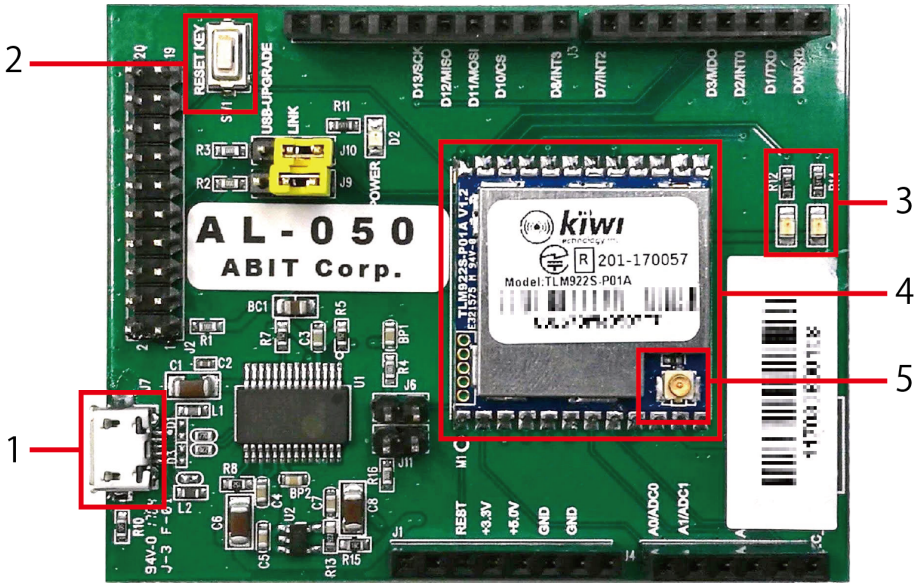


Figure 1 AL-050 Arduino Shield

To quickly start to use AL-050 Arduino shield, the first step is using USB cable to connect shield to PC via micro-USB connector. Then wire the D0 to D12 and D1 to D11, as figure 3 shown.

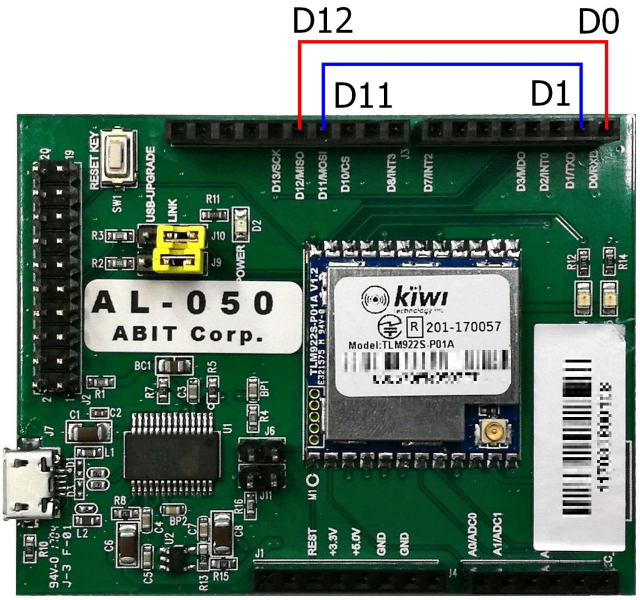


Figure 3 Wire Connection for Quick Start

Next step is checking whether the USB bridge driver is properly installed on your PC. If PC cannot recognize the device, the driver can be downloaded from Prolific's website[1] (the model name of USB bridge model is PL2303HXD). After successful installation of USB driver, you can use any terminal program to connect to shield and the default setting is **9600-8N1**. Then, the command interface can be used through the terminal program.

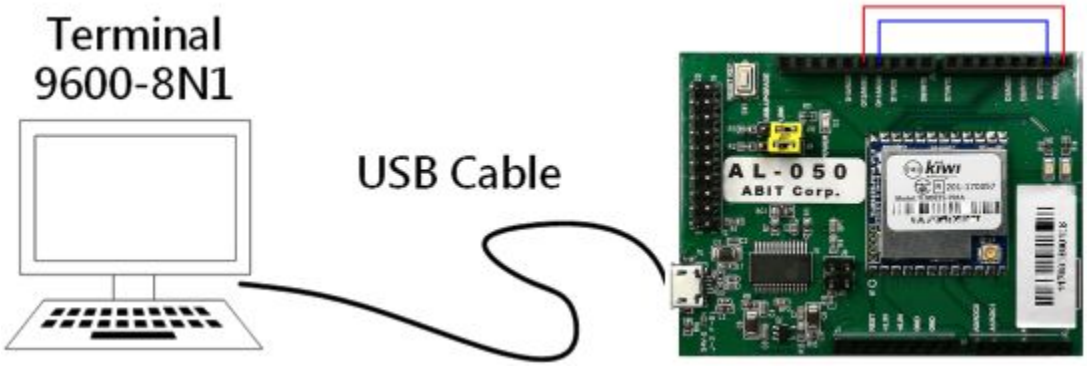


Figure 4 Connection with PC

The usual use of this shield is mounted on Arduino, as figure 5 shown. Figure 6 illustrates the pinout of Arduino shield. There is a sample Arduino program at 3.1.

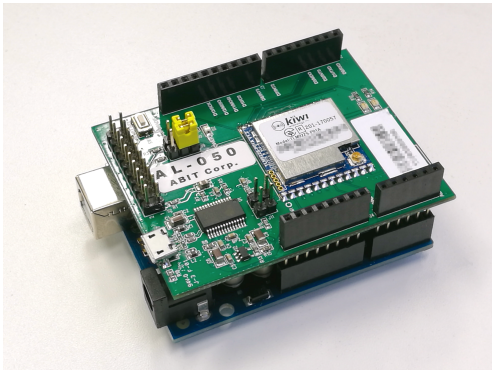


Figure 5 Mount on Arduino Uno

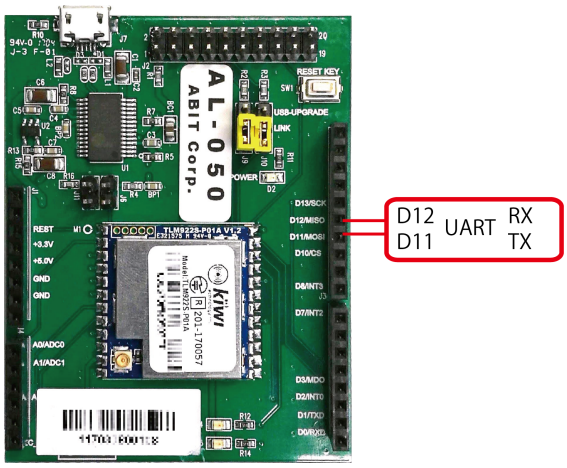


Figure 6 Pinout

[1] http://www.prolific.com.tw/US/ShowProduct.aspx?p_id=225&pcid=41

2. Command Reference

2.1. Command Format

AL-050 Arduino shield's commands can be categorized into 2 types: module command and LoRaWAN™ command. Module commands are control commands not related to radio transmission. LoRaWAN™ commands are used to utilize LoRaWAN™ protocol.

The command interface is readable ASCII string. AL-050 Arduino shield starts to accept command after outputting a '**>**' character, and the response string from AL-050 Arduino shield starts with two '**>**' characters. AL-050 Arduino shield will output message"> TLM922S <" when boot module. The first line of following example is a module command, and the second line is the response from AL-050 Arduino shield.

Example:

```
> mod get_ver
>> V1.3.1-Jan 20 2017-11:41:14
```

Note that the input command must end with a **CR**.

2.2. Module Commands

2.2.1. mod factory_reset

Response: Ok.

All LoRaWAN, radio and module configuration parameters will be set to default value.

Example:

```
> mod factory_reset
>> Ok
```

2.2.2. mod get_ver

Response: a string representing firmware version, bulid date and bulid time.

Get current firmware version.

Example:

```
> mod get_ver
>> V1.3.1-ar-Mar  3 2017-10:24:32
```

2.2.3. mod get_hw_model

Response: a string representing hardware model and UUID.

Get hardware model name.

Example:

```
> mod get_hw_model
>> TLM922S-P01A-117012000001
```

2.2.4. mod sleep <Deep> <INTwake> <Period>

<Period>: a string representing sleep time in second, can be from **0** to **65535** (**0** means infinite sleep, wakeup by P21).

<INTWake>: enable wakeup by P21 or not, can be **0** (disable) or **1** (enable).

<Deep>: deep sleep mode or not, **0** is normal sleep mode, and **1** is deep sleep mode.

Response: **Ok,** if <Period> and <Deep> string is valid

Invalid, if <Period> and <Deep> string is not valid.

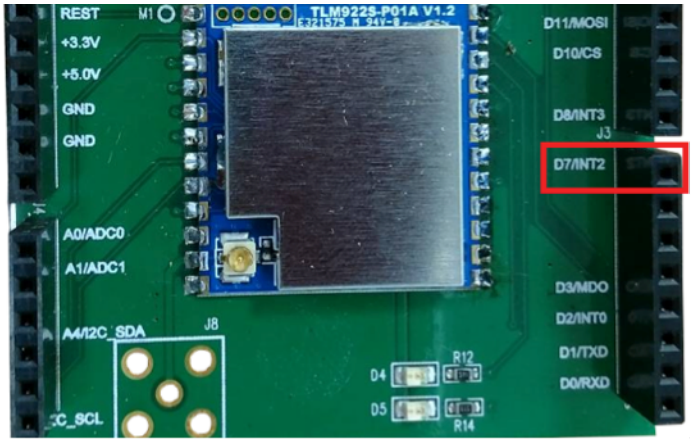
Module will be in sleep mode for <Period> seconds.

If <INTWake> is enabled, module also returns from sleep while P21 is triggered. To use P21 as a wakeup pin in sleep, P21 needs to set LOW before entering sleep. Once P21 changes from LOW to HIGH, module would return from sleep.

Note: P21 is mapped to D7/INT2 slot on Arduino Shield, as following figure shows.

Example:

```
> mod sleep 0 0 10
>> Ok
```



2.2.5. mod set_baudrate <Baudrate>

<Baudrate>: a string representing UART barudrate used for command interface, can be from **9600**, **19200**, **57600**, **115200**.

Response: **Ok,** if <Baudrate> string is valid

Invalid, if <Baudrate> string is not valid.

Module will use <Baudrarte> for command interface.

Example:

```
> mod set_baudrate 9600
>> Ok
```

2.2.6. mod set_echo <Status>

<Status>: a string representing echo status, can be **on** or **off**.

Response: **Ok,** if <Status> string is valid.

Invalid, if <Status> string is not valid.

Enable or disable UART echo mode.

Example:

```
> mod set_echo on
>> Ok
```

2.2.7. mod reset

Response: no response.

Reset TLM922S.

Example:

```
> mod reset
```

2.2.8. mod save

Response: **Ok**

Save module configuration (baudrate and echo status) to flash.

Example:

```
> mod save
>> Ok
```

2.3. LoRaWAN Commands

AL-050 Arduino shield Command Interface has several built-in configuration tables that can be used to customize for fulfilling different channel requirements. Before diving into these tables, data rate settings must be introduced first. Table 1 shows AL-050 Arduino shield Command Interface's data rate settings which are originally defined in LoRaWAN standard. The DR7, using FSK, is not supported by AL-050 Arduino shield. Thus the DR number can be used in AL-050 Arduino shield Command Interface is from **0** to **6**.

Table 1 Data Rate Settings

DR Setting	Configuration	Indicative bit rate (bit/s)	Supported by AL-050
DR0	SF12 BW125KHz	250	Yes
DR1	SF11 BW125KHz	440	Yes
DR2	SF10 BW125KHz	980	Yes
DR3	SF9 BW125KHz	1760	Yes
DR4	SF8 BW125KHz	3125	Yes
DR5	SF7 BW125KHz	5470	Yes
DR6	SF7 BW250KHz	11000	Yes
DR7	FSK 50bps	50000	No

The first configurable table is frequency table that sets frequencies used by AL-050 Arduino shield. There are 16 channels allowed to set. There are plenty of settings can be set for each channel as shown at Table 2. The **Frequency** of each channel can be added and set by joining to the Network Server. **Max. DR** and **Min. DR** are the maximum and minimum data rate supported in this channel. **Enable** indicates whether this channel is enabled or disabled. **Join Channel** uses to determine does this channel can be used in OTA. **Band ID** is which band in duty cycle table belongs to. All settings in frequency table except **Band ID** are can be configured by commands. **Band ID** is automatically determined from duty cycle table.

Table 2 Frequency Table

Index	Frequency	Max. DR	Min. DR	Enable	Join Channel	Band ID
0	923200000	5	0	On	Yes	0
1	923400000	5	0	On	Yes	0
2	0	0	0	Off	No	-
…	0	0	0	Off	No	-
15	0	0	0	Off	No	-

Duty cycle table, Table 3, defines transmission duty cycle for different frequency bands. AL-050 allows to configure 16 different bands.

Table 3 Duty Cycle Table

Band ID	Max Frequency	Min Frequency	Duty Cycle
0	920600000	928000000	11
1	0	0	1
2	0	0	1
...	0	0	1
15	0	0	1

Table 4 shows the TX power table used in AL-050 Arduino shield. The TX power index is used within LoRaWAN protocol for referring to different transmission power. AL-050 Arduino shield only supports 6 different TX power settings. Table 5 is the maximum payload size for each DR which defines at Table 1.

Table 4 TX Power Table

Index	TX Power (dBm)
0	13
1	11
2	9
3	7
4	5
5	3

Table 5 Max. Payload Size Table

Index	DR	Max. Payload Size (bytes)
0	DR0	0
1	DR1	0
2	DR2	11
3	DR3	53
4	DR4	125
5	DR5	242
6	DR6	242

2.3.1. lorawan join <Mode>

<Mode>: a string representing join mode of LoRaWAN, can be **otaa** (over-the-air activation) or **abp** (activation by personalization). (*Currently, only "**otaa**" is supported.)

Response: there is two responses after entering this command. The first response, used to indicate that whether command is valid or parameters is set appropriately, will be received after entering command. The second response will be received after join procedure.

First response: **Ok**, if <Mode> string is valid.
Invalid, if <Mode> string is not valid.
keys_not_init, keys are not configured.
no_free_ch, no channels are available.
busy, internal state is busy.

Second response: **accepted**, successfully join LoRaWAN.
unsuccess, join procedure is unsuccessful.

Star join procedure of LoRaWAN.

Example:

```
> lorawan join otaa
>> Ok
>> accepted
```

2.3.2. lorawan get_join_status

Response: a string representing whether module is joined successfully.

Returned string can be: **joined**, **unjoined**.

Get join status of LoRaWAN.

Example:

```
> lorawan get_join_status
>> joined
```

2.3.3. lorawan tx <Type> <PortNum> <Data>

<Type>: a string representing type of transmitting message, can be **cnf** (confirmed) or **ucnf** (unconfirmed).

<PortNum>: a decimal string representing port number used for transmission, can be from **1** to **233**.

<Data>: a hexadecimal string representing data to be transmitted.

Response: there is two responses after entering this command. The first response will be received after entering command. The second response will be received after transmission.

First response: **Ok**, if <Type>, <PortNum> and <Data> strings are valid.

Invalid, if <Type>, <PortNum> and <Data> strings are not valid.

not_joined, module is not joined LoRaWAN.

no_free_ch, no channels are available.

busy, internal state is busy.

invalid_data_length, data length is greater than allowed data length.

Second response: **tx_ok**, successfully transmit data.

rx <portnum> <data>, there is downlink data.

<portnum> - a decimal string representing receiving port

<data> - a hexadecimal string representing received data.

err, acknowledgement is not received, if confirmed message is used.

Star transmission LoRaWAN.

Example:

```
> lorawan tx ucnf 15 98ba34fd
>> Ok
>> tx_ok
> lorawan tx ucnf 15 6805
>> Ok
>> rx 15 6432
```

If the device class is set to class C, a downlink data would be received at any time. The downlink data of class C is outputted by TLM922S in **rx <portnum> <data>** format.

Example:

```
>
>> rx 15 6432
```

2.3.4. lorawan set_dr <DataRate>

<DataRate>: a decimal string representing data rate used for LoRaWAN, can be from **0** to **6**.

Response: **Ok,** if <DataRate> string is valid

Invalid, if <DataRate> string is not valid.

Set data rate used for LoRaWAN.

Example:

```
> lorawan set_dr 0
>> Ok
```

2.3.5. lorawan set_adr <State>

<State>: a string representing whether ADR is **on** or **off**.

Response: **Ok,** if <State> string is valid

Invalid, if <State> string is not valid.

Set the state of ADR.

Example:

```
> lorawan set_adr on
>> Ok
```

2.3.6. lorawan set_txretry <RetryCount>

<RetryCount>: a decimal string representing retry number of transmission, can be from **0** to **255**.

Response: **Ok,** if <RetryCount> string is valid

Invalid, if <RetryCount> string is not valid.

Set retry number of transmission.

Example:

```
> lorawan set_txretry 8
>> Ok
```

2.3.7. lorawan set_battery <BatteryLevel>

<BatteryLevel>: a decimal string representing battery level, can be from **0** to **255**.

Response: **Ok,** if <BatteryLevel> is valid.

Invalid, if <BatteryLevel> is not valid.

Setup module battery level, battery level can be get by lorawan server(use DevStatusReq command).

Example:

```
> lorawan set_battery 0
>> Ok
```

2.3.8. lorawan save

Response: **Ok**

Save LoRaWAN configuration parameters to flash.

Example:

```
> lorawan save
>> Ok
```

2.3.9. lorawan get_devaddr

Response: a hexadecimal string representing Device Address used for LoRaWAN.

Return Device Address used for LoRaWAN.

Example:

```
> lorawan get_devaddr
>> 12345678
```

2.3.10. lorawan get_deveui

Response: a hexadecimal string representing Device EUI used for LoRaWAN.

Return Device EUI used for LoRaWAN.

Example:

```
> lorawan get_deveui
>> 000b78ffff000000
```

2.3.11. lorawan get_dr

Response: a decimal string representing data rate used for LoRaWAN, can be from **0** to **6**.

Return data rate used for LoRaWAN.

Example:

```
> lorawan get_dr
>> 2
```

2.3.12. lorawan get_pwr

Response: a decimal string representing index of transmitting power in TX power Table, can be from **0** to **5**.

Return transmitting power index.

Example:

```
> lorawan get_power
>> 0
```

2.3.13. lorawan get_adr

Response: a string representing whether ADR is **on** or **off**.

Return the state of ADR.

Returned string can be: **on**, **off**.

Example:

```
> lorawan get_adr
>> on
```

2.3.14. lorawan get_txretry

Response: a decimal string representing retry number of transmission, can be from **0** to **255**.

Get retry number of transmission.

Example:

```
> lorawan get_txretry
>> 8
```

2.3.15. lorawan get_rxdelay

Response: **<rxdelay1> <rxdelay2>**

<rxdelay1> - delay interval in milliseconds used for receive window 1, can be from **0** to **65535**.

<rxdelay2> - delay interval in milliseconds used for receive window 2, can be from **0** to **65535**.

Get delay interval of receive window 1 and receive window 2.

Example:

```
> lorawan get_rxdelay
>> 1000 2000
```

2.3.16. lorawan get_rx2

Response: **<DR> <freq>**

<DR> - data rate of second receive window, can be **0** to **6**.

<freq> - operation frequency of second receive window in Hz, can be from **862000000** to **932000000**.

Get data rate and operation frequency used for second receive window.

Example:

```
> lorawan get_rx2
>> 2 923200000
```

2.3.17. lorawan get_ch_para <ChannelId>

<ChannelId>: a decimal string representing channel number, can be from **0** to **15**.

Response: **<freq> <minimum DR> <maximum DR>**, if <ChannelId> is valid.

<freq> - operation frequency of specified channel in Hz, can be from **862000000** to **932000000**.

<minimum DR> - minimum DR can be used, can be from **0** to **6**.

<maximum DR> - maximum DR can be used, can be from **0** to **6**.

Invalid, if <ChannelId> string is not valid.

Get operation frequency, maximum DR and minimum DR of specified channel.

Example:

```
> lorawan get_ch_para 0
>> 923200000 0 5 0
```

2.3.18. lorawan get_ch_status <ChannelId>

<ChannelId>: a decimal string representing channel number, can be from **0** to **15**.

Response: **on** or **off,** state of specified channel.

Invalid, if <ChannelId> string is not valid.

Get state of specified channel. **on** means the channel is enabled, and **off** means the channel is disabled.

Example:

```
> lorawan get_ch_status 0
>> on
```

2.3.19. lorawan get_dc_ctl

Response: state of duty cycle, **on** or **off**.

Default: **on**

Get state of duty cycle checking. **on** means the checking is enabled, and **off** means the checking is disabled.

Example:

```
> lorawan get_dc_ctl
>> on
```

2.3.20. lorawan get_join_ch

Response: a list of channel ID for join request.

Default: **0**, **1**, **2**

Get frequency channel ID for join request. The default channel ID for join request is 0, 1, 2.

Example:

```
> lorawan get_join_ch
>> 0 1
```

2.3.21. lorawan get_upcnt

Response: uplink counter that will be used at next transmission.

Default: **1**

Get uplink counter that will be used at next transmission.

Example:

```
> lorawan get_upcnt
>> 9
```

2.3.22. lorawan get_downcnt

Response: downlink counter that will be used at next transmission.

Default: **0**

Get downlink counter that will be used at next transmission.

Example:

```
> lorawan get_downcnt
>> 5
```

2.3.23. lorawan get_class

Response: class type of LoRaWAN, can be **A** or **C**.

Default: **A**

Get class type of LoRaWAN.

Example:

```
> lorawan get_class
>> A
```

2.3.24. lorawan get_pwr_table <Index>

<Index>: **0** to **5**.

Response: a decimal string representing TX power of given index.

Default: following is the default TX power table:

Index	TX Power (dBm)
0	13
1	11
2	9
3	7
4	5
5	3

Get TX power in dBm of given index.

Example:

```
> lorawan get_pwr_table 0
>> 13
```

2.3.25. lorawan get_dl_freq <ChannelID>

<ChannelID>: **0** to **15**.

Response: a frequency in Hz representing ChannelID's downlink frequency.

Get downlink frequency of given ChannelID.

Example:

```
> lorawan get_dl_freq 0
>> 0
```

2.3.26. lorawan get_battery

Response: a decimal string representing battery level.

Default: **0**

Get module battery level.

Example:

```
> lorawan get_battery
>> 0
```

3. Example

This section gives several complete examples on how to use AL-050 Arduino shield command interface. All examples include many comments followed by double slash. This comments are for clearly explanation and should not be inputted to AL-050 Arduino shield through command interface.

3.1. Arduino Sample Program

This is a simple program to get input from Arduino's UART and send the input to AL-050 Arduino shield.

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(11, 12); //Rx Tx

void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);
}

void loop() {
  if (Serial.available()) {
    mySerial.write(Serial.read());
  }
  if (mySerial.available()) {
    Serial.write(mySerial.read());
  }
}
```

3.2. LoRaWAN

3.2.1. ABP and Uplink (*Currently, ABP is not available)

```
// Activation by Personalization
> lorawan join abp
>> Ok
>> accepted

// Send unconfirmed uplink on port 15
> lorawan tx ucnf 15 1234
>> Ok
>> tx_ok
```

3.2.2. OTA and Uplink

```
// Over-the-Air Activation
> lorawan join otaa
>> Ok
>> accepted

// Send unconfirmed uplink on port 15
> lorawan tx ucnf 15 1234
>> Ok
>> tx_ok
```

3.2.3. Confirmed Uplink and Downlink

Before starting this example, make sure AL-050 Arduino shield has been activated, ABP or OTA, as above sections described.

```
// Send confirmed uplink on port 15
> lorawan tx cnf 15 1234
>> Ok
>> tx_ok

> lorawan tx cnf 15 1234
>> Ok
>> err                      // Fail to get confirm from server

> lorawan tx cnf 15 1234
>> Ok
>> rx 15 6432               // Receive downlink from server on port 15
```