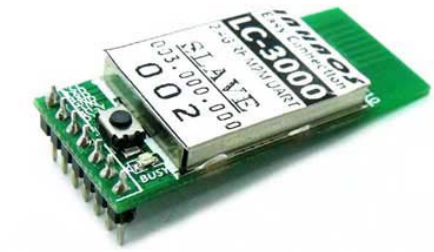
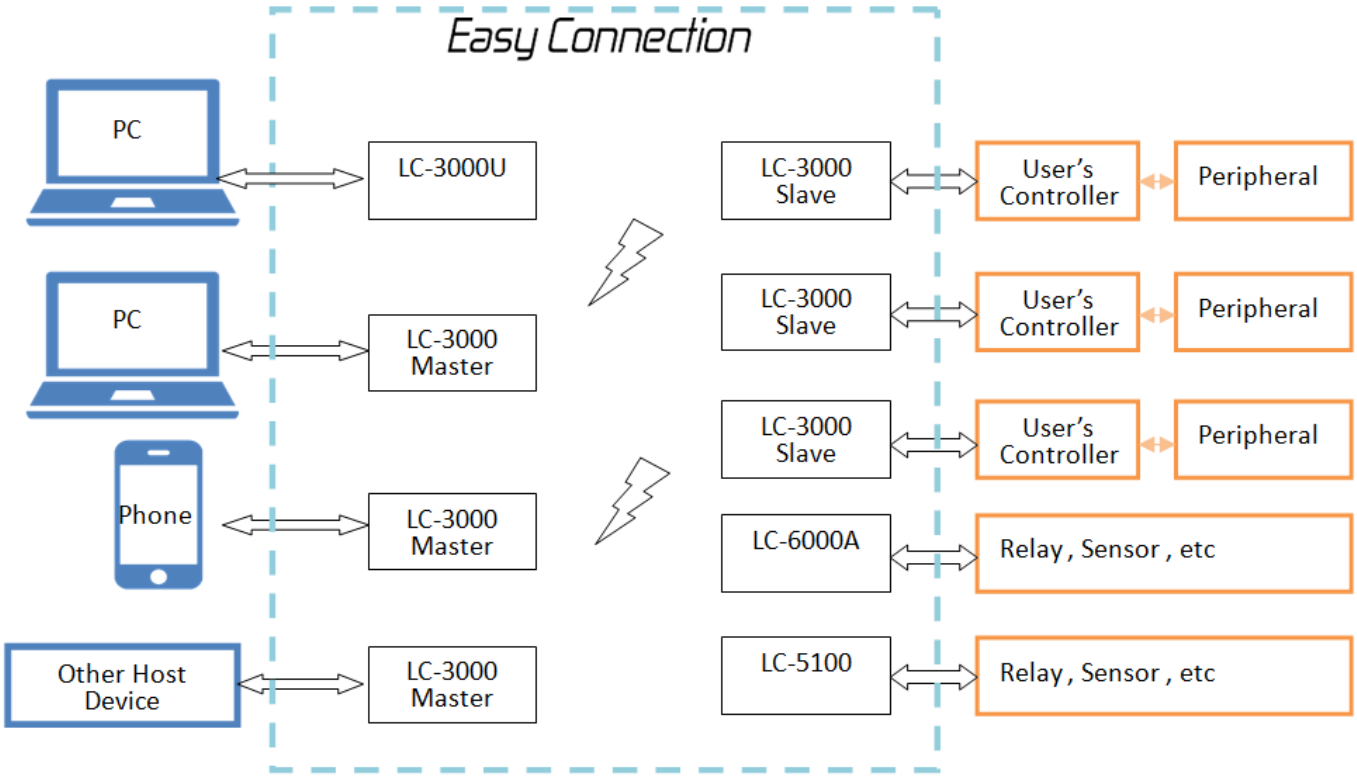
1. Features





A Simple Multi to Multi network based on 2.4G UART connection

The LC-3000(PA) serial 2.4GHz RF Multi-To-Multi (M2M) UART module is a new module release by INHAOS, It's combined a high performance and low power Cortex-M0 processor and 2.4G RF SOC, provide user a very easy way to implement 2.4G RF communication via simple UART programming.

- Low Cost , easy to use , **zero RF knowledge request for wireless communication design.**
- **Implement wireless communication over the wired UART programming.**
- Multi to Multi network mode communication (M2M), with Per to Per mode communication (P2P) supported.
- Full duplex communication, max support baud rate 512Kpbs.

- Max continuous bi-direct baud rate up to 115Kbps when in P2P mode.
- **DTR signal support used for Arduino for remote upload sketch when in P2P mode.**
- Power Save Mode (PSM) current consumption down to.
- Sleep mode current consumption down to 150uA.
- **LC-3000:** 0dBm, Low power , 20 to 30 meters outdoor open and clear distance

  LC-3000PA: 20dBm, 200 to 300 meters outdoor open and clear distance

The RF communication distance is very depended on it's working environment, the distance is out door clear environment and only for reference.

Important Notice:

The LC-3000 serial included Master and Slave , Master can not communication with Master , and Slave can not communication with Slave.

The Master and Slave need PAIRED before communication, one Master can be communication with max 126 Slaves , and one Slave can be communication with Max 126 Masters .

Each device has of an UID, the device use the last Byte of UID to identify the other side of the device in communication , so in a network it can not have the same last byte of UID

Easy Connection is a mark for LC-3000 and LC-5000/6000 serial modules, All Easy Connection marked device can be communication to each other.

2. Pin description

Pin No.	Pin Name	I/O	Function	Remark
1	VCC	P	Power supply, DC 3.3V	
2	TXD	O	UART TXD	
3	RXD	I	UART RXD	
4	GND	P	Power Ground	
5	FUN/CFG	I	PAIR/CONFIG	Active LOW, Pull LOW and hold over 3 sec to enter PAIRING MODE
6	BUSY	O	Busy Status Indication	Active LOW, Before sent data to TX pin , ensure BUSY pin is High
7	DTR	*	DTR signal, Used for Arduino Upload sketch	The direction is: MASTER->Slave **Master:** INPUT **Slave:** OUTPUT in P2P mode and INPUT in M2M mode

About the PAIR/CFG pin

One onboard tack switch is connected to this pin , when press the switch , the pin will be pull low. The functions for this pin are:

- **Config module parameter**

  When the module is set into P2P mode, all the data is transparent data. Only after the "PAIR/CFG" pin pulled low the LC-3000 will enter the CFG mode and receive any config command from UART. Please write the config command within 3000ms after the "PAIR/CFG" pin pulled low. And after the "PAIR/CFG" set to HIGH, it will back to normal.

- **Enter PARING mode**

  Long press the "PAIR/CFG" button and last more than 3000ms, the module will be entry PAIRING mode , in this mode , the LED will fast flash until paired or pairing mode timeout. And then the module will be exit PARING mode and back to normal.

- **Wake Up the module**

  Pull the "PAIR/CFG" pin to low will wake up the LC-3000 when it is in sleep mode or power save mode.

About the BUSY pin

The BUSY pin is a multi function pin. The onboard LED is indication the Busy Pin status, when Busy pin is high (3.3V) , the LED will be light.

- When user long press “PAIR” button (pull PAIR/CONFG low) , the module will be entry PARRING mode , in this case , the LED will be flash , and it will be turned off after paired. To paired two module , user need to make two module entry PAIRING mode .
- After user application wrote data into LC-3000, the LC-3000 will set the BUSY pin to LOW for indicating transmission busy. And till all the data sent out, the BUSY pin will set back to HIGH. So before write data into LC-3000, user application should check this BUSY pin firstly. Only when the BUSY pin is HIGH, user application can write new data in.

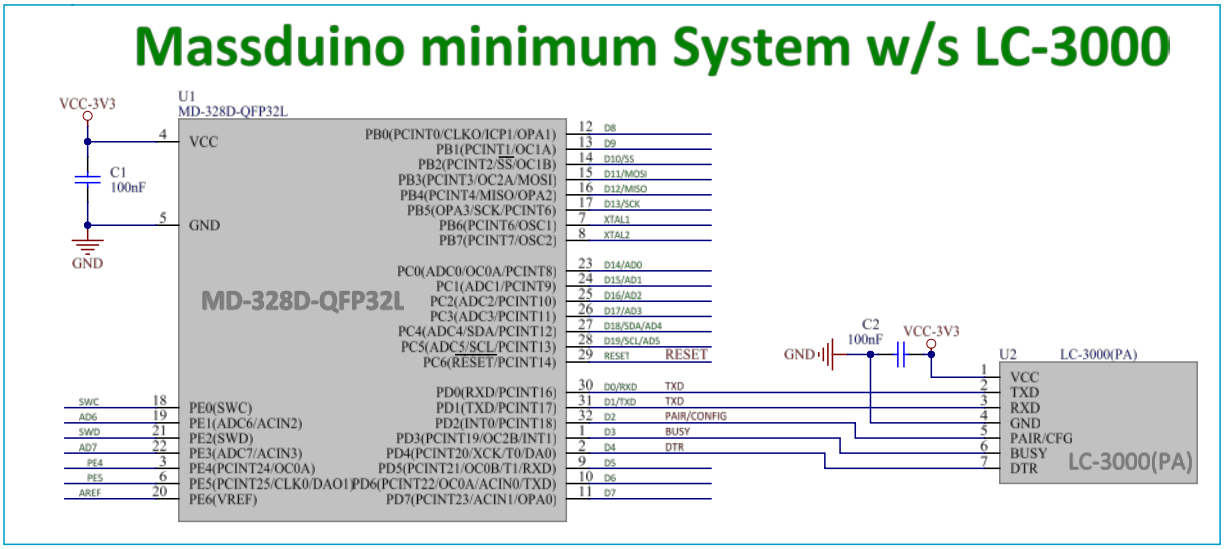
In the application, user can be read the BUSY pin status by a GPIO.

About the DTR pin

For the master module, the DTR pin is always input Hi-z state.

For the slave module, the DTR pin is input Hi-z state, until the module configured into P2P mode. When the module in P2P mode, the DTR pin of the slave module will become output, and follow the state of master module's DTR input state.

3. Hardware specification



LC-3000-Master/Slave Electrical Characteristics:

No.	Symbol	Parameter	Test Condition	Unit	Min	Typ.	Max	Remark
1	Vcc	Power Supply		V	2.7	3.3	3.5	
2	Icc_standby	Standby mode current	Vcc = 3.3V , RF Paired	mA				
3	Icc_sleep	Sleep mode current	Vcc = 3.3V , RF Paired , Sleep mode	uA				
4	Icc_PSM	PSM mode current	Vcc = 3.3V , RF Paired , Sleep mode	uA				
5	Iccpeak_TX	Peak current in TX mode	Vcc = 3.3V , CW output	dBm				
6	Power_RF	RF Output power	Vcc = 3.3V , CW output	dBm	18.5	20	22	
7	RXsens	RX Sensitivity	Vcc = 3.3V , 0.1% BER	dBm	-98	-102	-104.5	

LC-3000PA-Master/Slave Electrical Characteristics:

No.	Symbol	Parameter	Test Condition	Unit	Min	Typ.	Max	Remark
1	Vcc	Power Supply		V	2.7	3.3	3.5	
2	Icc_standby	Standby mode current	Vcc = 3.3V , RF Paired	mA				
3	Icc_sleep	Sleep mode current	Vcc = 3.3V , RF Paired , Sleep mode	uA				
4	Icc_PSM	PSM mode current	Vcc = 3.3V , RF Paired , Sleep mode	uA				
5	Iccpeak_TX	Peak current in TX mode	Vcc = 3.3V , CW output	dBm	25	27.5	29.5	
6	Power_RF	RF Output power	Vcc = 3.3V , CW output	dBm	-3	0	+5.5	
7	RXsens	RX Sensitivity	Vcc = 3.3V , 0.1% BER	dBm	-83	-88	-89.5	

4. LC-3000(PA) Network mode

The LC-3000(PA) supports two network modes: Multi to Multi network mode and Per to Per network mode.

4.1. Multi to Multi network mode (M2M):

In this mode, a Multi to Multi network communication topology can be set up via M master module and N slave module.

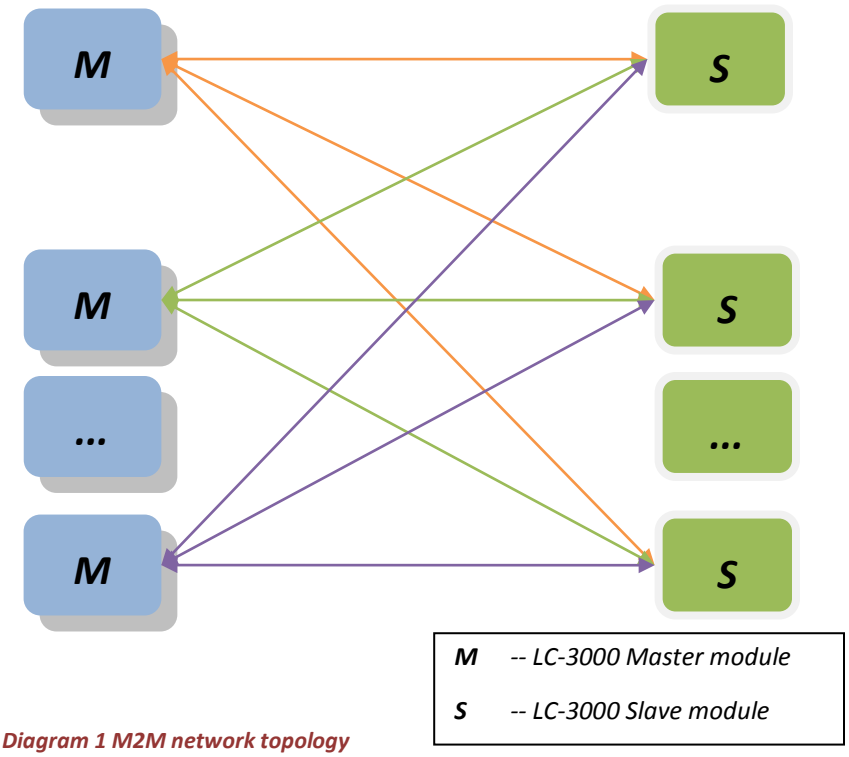


Diagram 1 M2M network topology

As it shows in diagram 1, the M2M network topology has features as follows:

- Any one of the master module can communicate with all slave modules, meanwhile any slave module can send and receive data to/from all master modules in this topology.
- Each of the modules no matter master or slave in this network, has its own unique local ID which is used for communication addressing.
- All communications operations were initiated by the master, meanwhile the slave only passive to receive. Master can do read (write) from (to) slave any time. But the data sending from slave to master will be delayed for some time which based on the polling period of the master. Each one of the masters must do the polling from all the slaves one after another periodically. And the polling period can be configured by user according to the specific applications.
- This network mode, theoretically can accommodate up to 126 master modules and 126 slave modules. But because there are competitions between masters while communicate with slave modules, therefore the max quantity of master is the smaller the better.

4.1.1. Data transmission in M2M mode

The data transmission in M2M mode is not transparent. The data packet must in according with the specified format which contains head bytes, receiver's LOC_ID, data length and data. Please reference 1.28 Data Command for details.

After LC-3000 received Data Command packet from user's application through UART interface, it will pull low BUSY pin at once, then start the transmission to the remote receiver (which marked by the LOC_ID section in Data Command). After all the packet sent out, it will stay in receiving state waiting for the ACK packet from the remote receiver. Once the ACK data packet received successfully, it will set the BUSY pin to HIGH, and then send a OK Response to the user's application. Otherwise, if the time for the transmission exceeds TX_MAX_TIME, the LC-3000 will stop the transmission and pull the BUSY pin HIGH, meanwhile an error response ("703") will send back to the user's application.

Any data packets received from the remote receiver will be sent to the user's application through UART interface. After received packet from UART, the user's application must response the ACK packet to the LC-3000 as soon as possible. Because the remote LC-3000 is pending and waiting for the ACK packet until it's exceed the TX_MAX_TIME. So how fast the user's application make the response to LC-3000 partly determines the speed of the transmission of LC-3000.

If the user's application write the data command packet to LC-3000 at the time another packet which wrote before is in the sending busy progress, it will response a buffer full error ("704").

The follows diagram shows the data transmission sequence of the LC-3000.

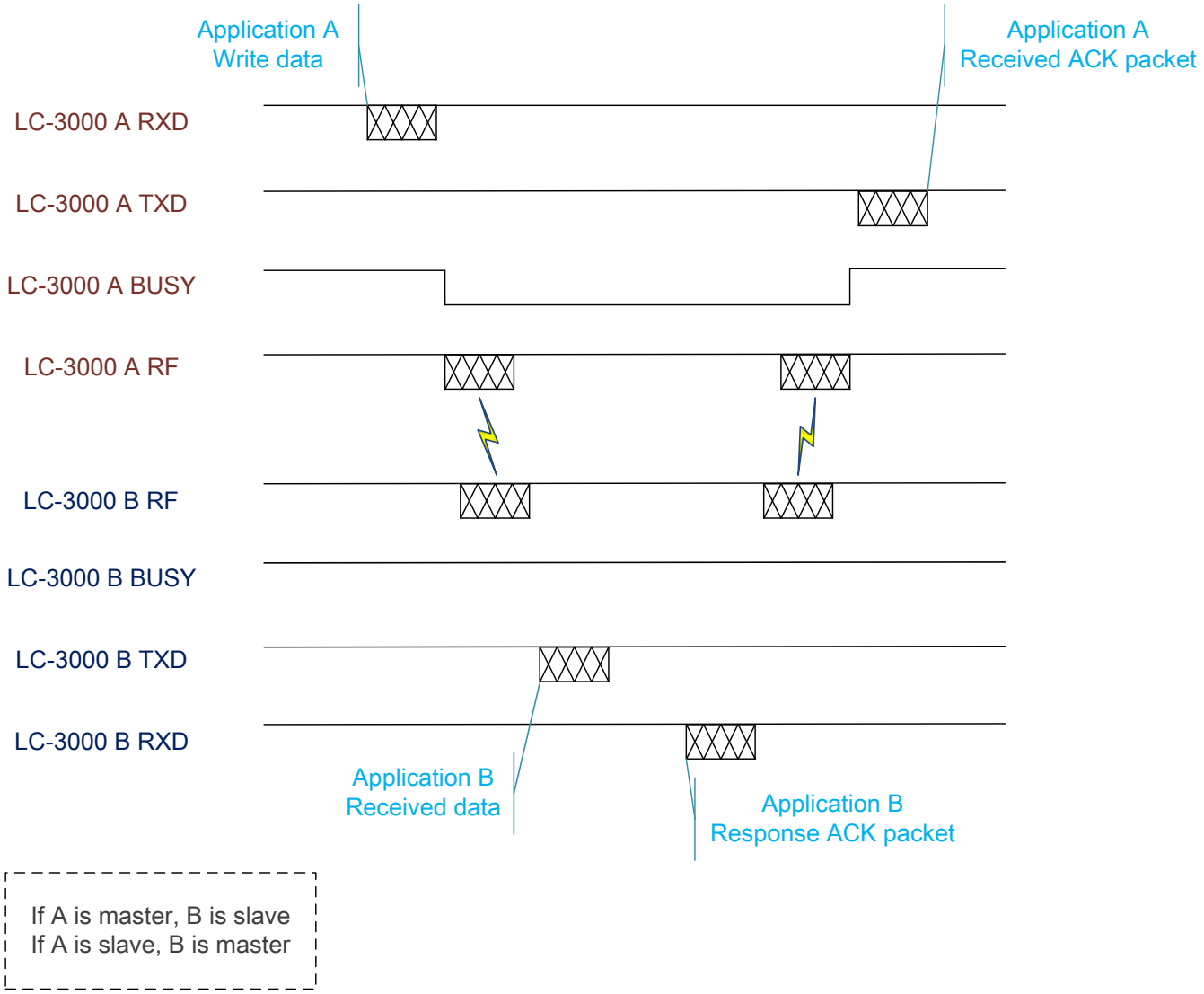


Diagram 2 M2M Transmitting Sequence Diagram

4.1.2. How to sending data using LC-3000 in M2M mode

There are two ways for sending data in M2M mode.

4.1.2.1. Sending with checking BUSY pin

This is most easy way to sending data using LC-3000 in M2M mode. User application directly check the BUSY pin to determine whether it send complete for previous data packet or not. Once the BUSY pin set to HIGH, a new data packet can be wrote in. The flow chart for this sending progress is as follows:

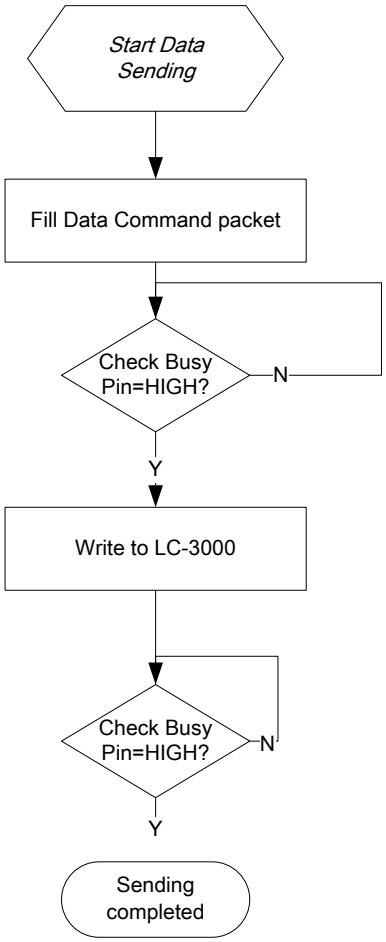


Diagram 3 Data Sending with checking BUSY pin Flow Chart in M2M mode

4.1.2.2. Sending with checking Response Packet

This is most rigorous way to sending data using LC-3000 in M2M mode.

User application waiting for the LC-3000's response after write a data packet to it. According the response packet the user application can determine whether the sending result for previous data packet is success or not. When the LC-3000 sending completed successfully it will response OK, otherwise it may response an error.

The flow chart for this sending progress is as follows:

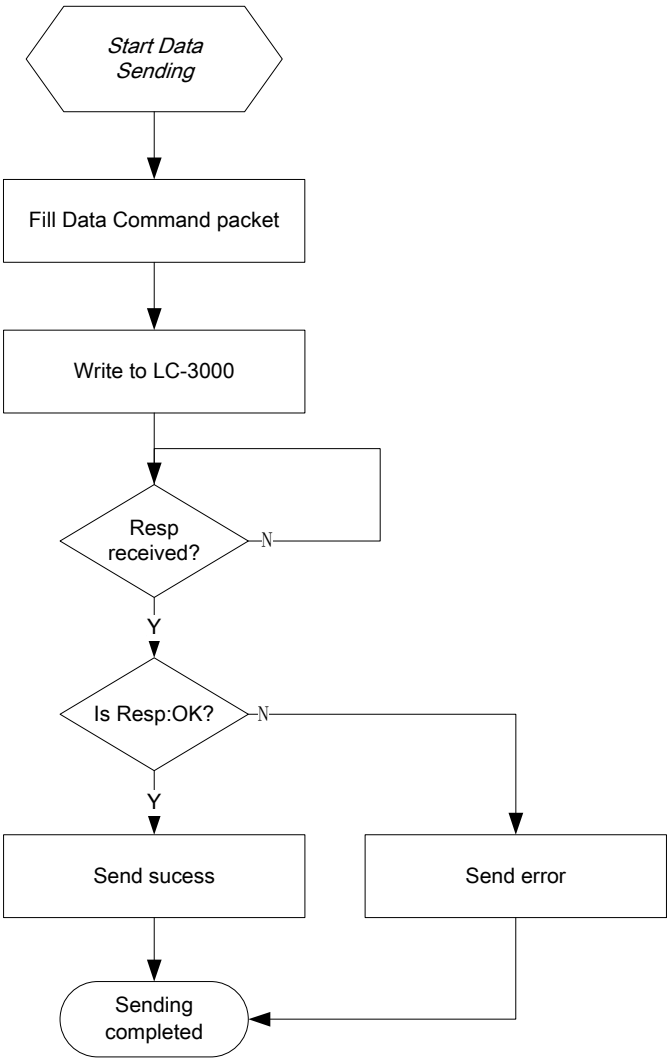


Diagram 4 Data Sending with checking response Flow Chart in M2M mode

4.1.3. How to receiving data using LC-3000 in M2M mode

The receiving data progress can be divided into two independent processes: UART interrupt process and the receiving data process.

4.1.3.1. UART interrupt process

For most of the applications using UART interrupt process to receiving the data sending from LC-3000. The application needs to read the receiving data byte by byte from UART and store them in a buffer, meanwhile the application also needs to search the "\r\n" flag to determine the end of one packet. The demonstration flow chart for UART interrupt process is as below.

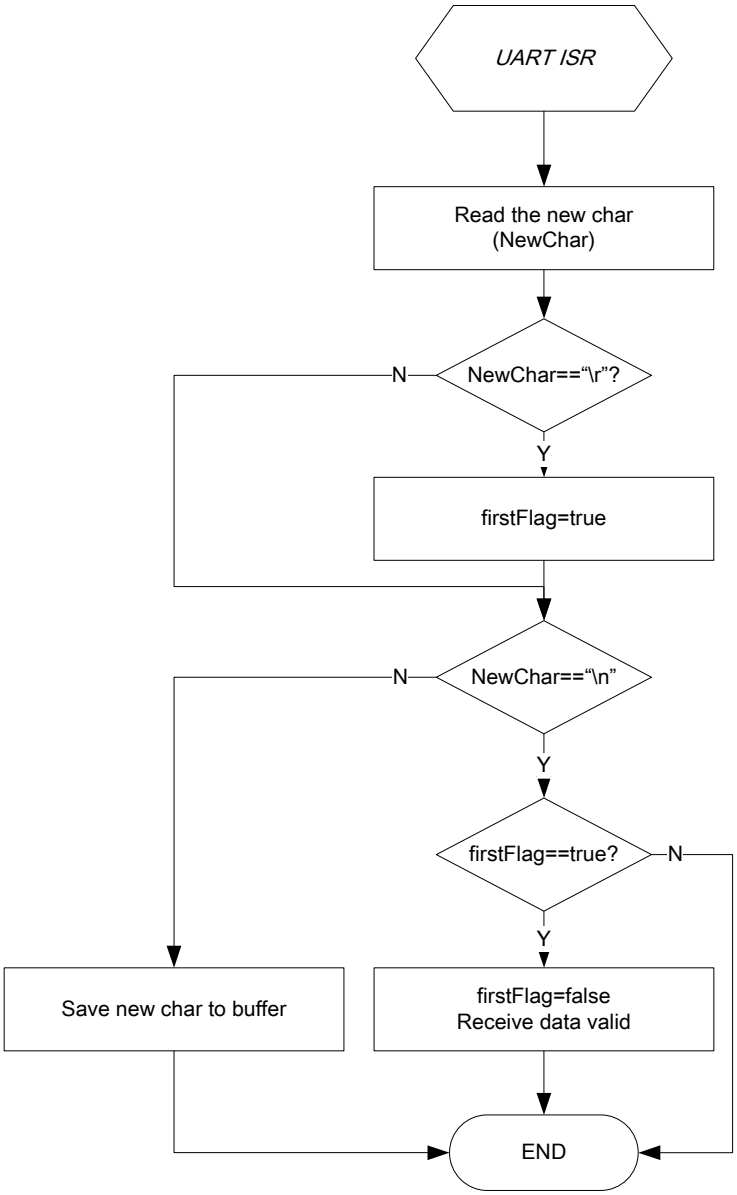


Diagram 5 UART interrupt process for data receiving in M2M

4.1.3.2. Receiving data process

After the application received data bytes via UART interrupt process. The user application need to do more receiving data process and then make the corresponding response to LC-3000. Otherwise the remote LC-3000's sending progress will blocking for waiting the response, which will significantly decrease the communication speed.

According to the head byte of the received data, user application can distinguish the data packet type. For example, if the head byte is "#", express the data packet is an response packet; and if the head byte is "B", express the data packet is a broadcast packet; and for the normal data packet, its begin with the head byte "=". Once the normal data packet received the user application must send a response packet back to LC-3000 as soon as possible. And it's no need any response for the response packet and the broadcast packet.

The receiving data process for LC-3000 that we suggested is as follow flow chart shows.

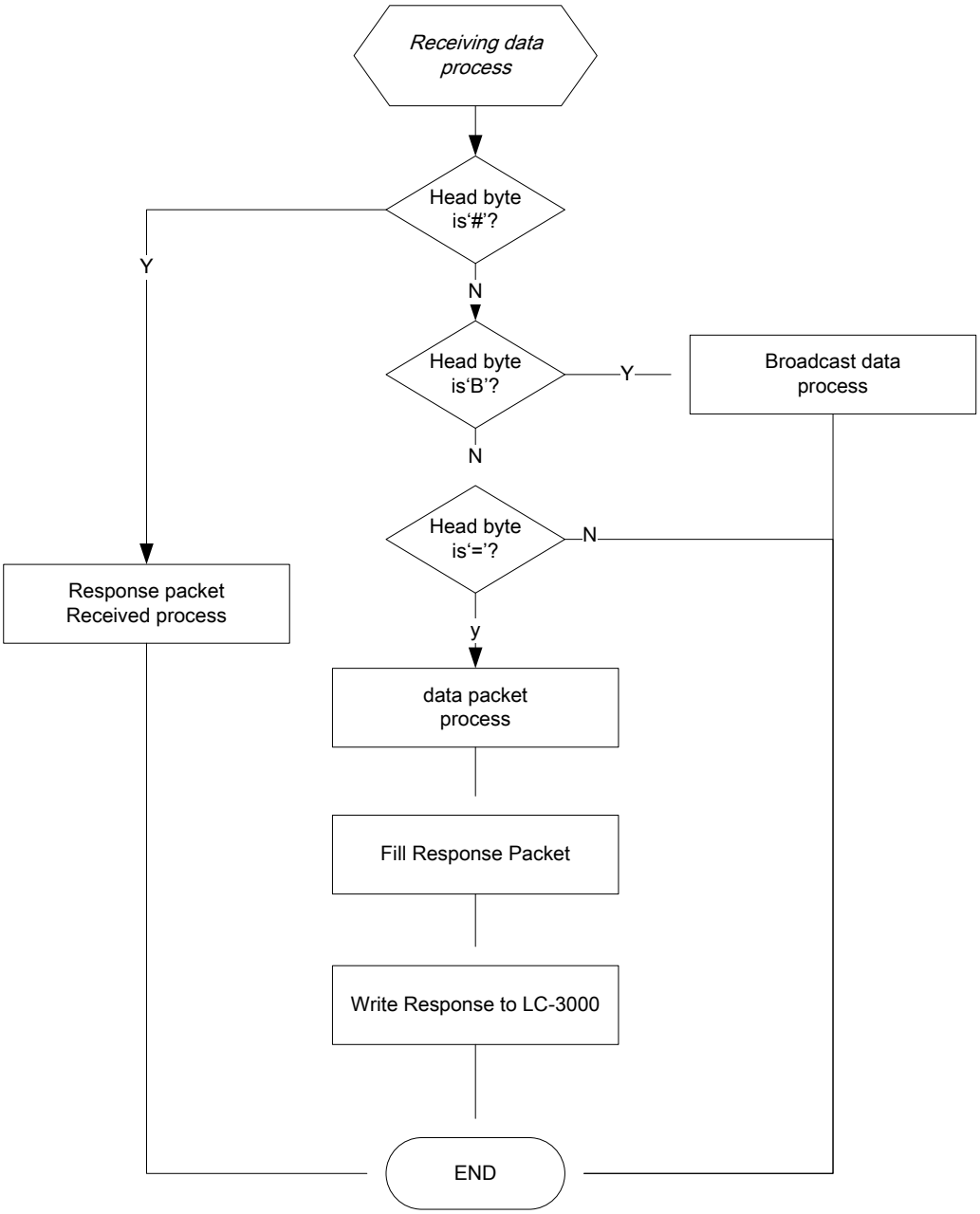


Diagram 6 Data receiving process in M2M

4.1.4. M2M Slave Priority Control

Due to one master module can connect to multiple slave modules, not all the data sending from the slave module can be received by the master simultaneously. So that the master LC-3000 is built-in a priority control mechanism, which has 4 priority levels: 0 to 3. Level 0 is the highest priority, and Level 3 is the lowest priority. When a new slave module pairing in, it's default priority is Level 2. The higher priority the slave module has, the faster speed for the slave's data transmit to master.

4.1.5. Broadcasting Function

The LC-3000 master module supports broadcasting function, and the slave module does not support this function.

The master side's user application can send data to all the slave modules at the same time using broadcasting function. It will save a massive time for data transmission to multiple slaves compared with one by one transmission. The broadcasting max time is determined by BROADCAST_TIME, which can be set via "Set Broadcast Timeout" command. Although the broadcasting function cannot guarantee all the slave can received the data, but the longer the BROADCAST_TIME the more reliable for each slave received the broadcast data. And another disadvantage for broadcasting function is that there are no response from the slave modules.

4.1.6. M2M's UART Baudrate

The LC-3000 will set to the default baud rate (defined by BAUDRATE_M2M) after power on into M2M mode, meanwhile the Auto-baud rate function is enabled, so that the user's application can read write to LC-3000 through UART with any baud rate.

The baud rate of the slave won't following the master's baud rate, so the two user's applications (local and remote) can using different baud rate for UART communication.

Please note, after the user's application changed its baud rate, it must write to the LC-3000 at least once for updating the LC-3000's baud rate to the new one.

4.2. Per to Per network mode (P2P):

P2P model is used for the temporarily mass data transmission, especially can be used for the remote upload sketch operation for Arduino. The features supported by P2P mode is as follows:

- The master will stop polling operations
- The data will transfer between the master and slave modules with extremely short delay
- It is a transparent transmission mode. The master or the slave will no longer care about the format of the data, unless the CFG button is pressed.
- The baud rate of the slave module will following the master's
- The DTR pin of the slave module will following the master's DTR input state
- The BUSY pin will indicate whether the transfer operation of data completed or not

4.2.1. Data transmission in P2P mode

After enter the P2P mode, all the user's application data write to LC-3000 through UART will be treated as transparent data. It means that if the user's application write any data to the master, and the data will be transfer to slave module with no data changing. So does the slave side.

When any data received by LC-3000 through the UART interface, it will set the BUSY pin to LOW firstly, then start the sending progress, and after the sending progress complete, the BUSY pin will set to HIGH immediately. And if the sending

progress takes more than 2 seconds, the LC-3000 will stop the sending , clear all the data, set the BUSY pin to HIGH and then exit the P2P mode back to the M2M mode.

The LC-3000 will keep alive the connection with the remote LC-3000 which in pairs . Once the communication blocking up to 2 seconds, the LC-3000 will quit the P2P mode and return back to M2M mode.

4.2.2. How to sending data using LC-3000 in P2P mode

For the P2P mode user's application can sending data to LC-3000 any time. And no matter any data wrote to the LC-3000, it will set the BUSY pin to LOW, and start sending. After sending complete, it will set the BUSY pin to HIGH. So user application can use this feature to detect whether the sending progress is completed. A demonstration flow chart is as below.

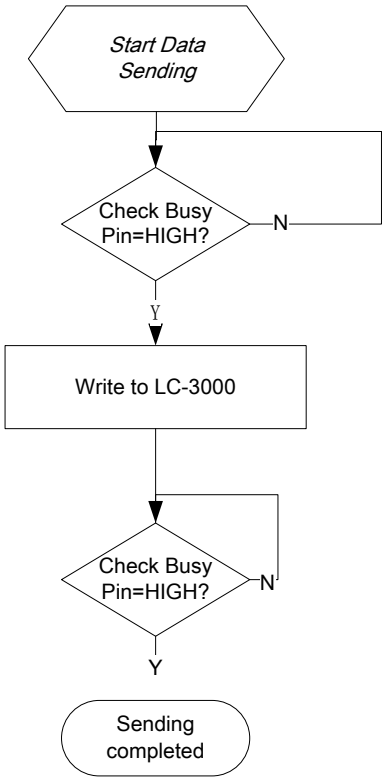


Diagram 7 Data Sending progress flow chart in P2P

4.2.3. How to receiving data using LC-3000 in P2P mode

It suggested that the user application using the UART interrupt to receiving the data bytes from LC-3000. And for the P2P mode, user application no need to make any response to LC-3000.

4.2.4. DTR function in P2P mode

In P2P mode, the DTR pin of the slave module will following the master's DTR input state. So the user's application can control the slave's DTR state via set the master's DTR state.

4.3. P2P's UART Baud rate

After switched to P2P mode, the master LC-3000 will set the baud rate to BAUDRATE_P2P, and the slave module's baud rate will follow the master side for each data packets. The master side's user application can change the BAUDRATE_P2P using "Set P2P baud rate" command. But the slave LC-3000's baud rate is not affected, even slave side's application send "Set P2P baud rate" command to change its baud rate.

Both the master and slave LC-3000 will disable the auto-baudrate function in the P2P mode. But when the CFG button pulled low it will enter the CONFIG mode, and re-enable the auto-baudrate again for user application to send configuration commands.

When LC-3000 exit P2P mode return to M2M mode, its baud rate will set back to the latest baud rate which saved in the M2M mode while auto-baudrate function is enabled.

4.4. Network mode transition diagram

The LC-3000 will stay in the M2M mode by default after powered on.

User's application can set the LC-3000's network mode via "Set Net Mode" command. Please note only the master module support this command, if application trying to send the "Set Net Mode" command to the slave module, will receive an error response.

Once received the "Set Net Mode" command, the master module will trying to set the slave's network mode through RF. After received the OK response from the slave, the master will set itself network mode to the new mode, otherwise, the master's mode keeps no changing and it will response an error to the application. The max time for master trying to change slave's network mode is TX_MAX_TIME.

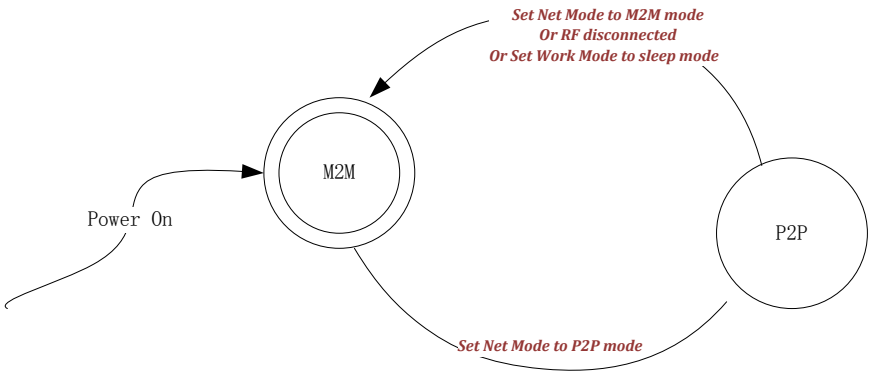


Diagram 8 Network mode transition diagram

4.4.1. Enter P2P mode

The only way to enter P2P mode is to send "Set Net Mode" command (with parameter NET_MODE=1) to the master. This command will force both the master and slave switch into P2P mode simultaneously.

4.4.2. Exit P2P mode

There are three ways to exit the P2P mode:

1. Using "Set Net Mode" command with parameter NET_MODE set to 0. This command will force the master and slave return to M2M mode simultaneously.
2. Master and slave communication timeout with no response, both the master and slave will exit the P2P mode return to M2M mode. But it does not guarantee that they can exit simultaneously. The following conditions may lead to the communication timeout.
   - Either the master module or the slave powered down with accident
   - The communication range exceed the range for the 2.4G RF communication can reach.
3. Either master or slave module is set into sleep mode by user's application. And the another side will exit the P2P mode automatically after communication timeout.

4. LC-3000(PA) Work mode

The LC-3000(PA) has 3 work modes: Full speed mode, Power save mode and Sleep mode.

4.1. System Work mode transition

User application can make work mode transition from one to another through "Set Work Mode" command. But this configure operation has some different between M2M mode and P2P mode. For the M2M mode, all the data write to LC-3000 follows the packet format, so the LC-3000 is able to identify any effective command packet no need application pull CFG pin LOW. But for the P2P mode, because all the data is treated as transparent data, so it's need the user application to set the CFG pin to LOW before any command packet wrote to the LC-3000.

The following diagram shows the system work mode transition.

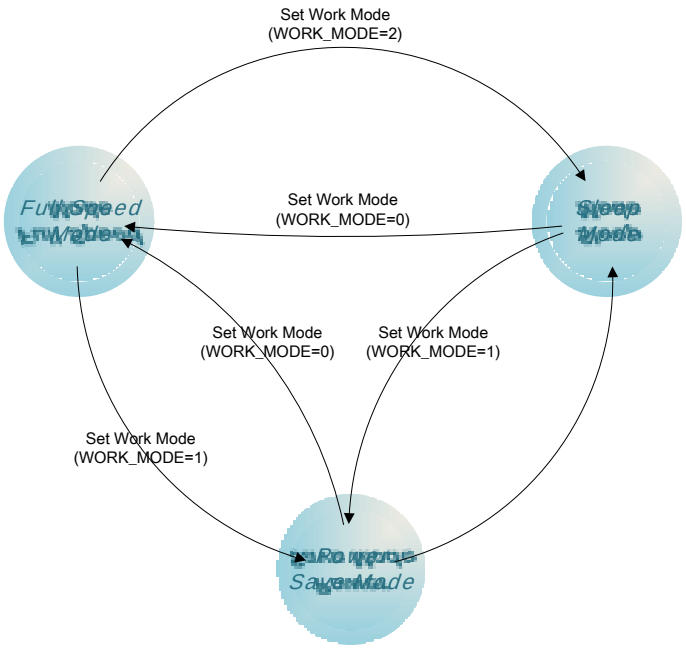


Diagram 9 System Work Mode transition diagram in M2M

4.2. Full speed mode:

In the full speed mode, the LC-3000 master module will do polling with each slave module in real time. And all the slave modules will be in receiving state. The slave module will response to the master while the polling packet from master module received. So the user application's data packet can be sent to destination via LC-3000 at any time. In the full speed mode, it has the shortest time delay for data transmission, but oppositely it will consume the highest power. So this mode can be used on the applications which have higher request for real-time data transmission.

4.3. Power Save Mode (PSM):

4.3.1. For the slave module

In the power save mode, the LC-3000 will do intermittent work with 10% proportion wake up time. The user application can also send data via LC-3000 to the remote any time, but it will take a little bit time delay for the data to be received by the remote LC-3000. The delay time may vary according to the work mode of the remote LC-3000. And the max transmit time is determined by the TX_MAX_TIME. Because of this feature, it will save 90% of power, on the basis of guarantee the data transmission with a little bit time delay.

4.3.2. For the master module

When the master module is set in the PSM mode, it will enter a mode which most like the sleep mode: keep in sleep state, no automatically wake up, and etc,. The only different between the PSM and sleep mode for master module is that: both the UART interrupt and CFG pin interrupt can wake up the LC-3000 in PSM mode, meanwhile in the sleep mode, it is only CFG pin interrupt can wake up the LC-3000.

4.4. Sleep Mode:

In this mode , the LC-3000 will stay in sleep mode , and stop any data transmission. User application can only wake it up through pull CFG pin to LOW, and after all the data transmission task is completed, it will go back to sleep state again. The sleep mode has minimum power consumption. But it's up to the user application for guarantee communication successful.

5. PAIRING

Before using the module , user need to pairing it , then the module can be communication to each other , to pairing the module , user need to notice the below things:

1, Only one MASTER and one SLAVE can be paired.

2, To pairing the module , press and hold the “PAIRING” button one the module , or pull the CFG pin low and hold it over 3sec , the module will be entry PAIRING mode , the LED will be flash to indication the status.

3, Do the same actions on other side, once both module entry PAIRING mode , they will be pair very quickly and the LED will be OFF , that mean the module is paired and ready to communication to each other.

4, Each of the MASTERs can be paired to multiple SLAVEs, and also each of the SLAVEs can be paired to multiple MASTERs.

If you sent configuration command after pull low the CFG pin , the module will entry CONFIG mode , other than PAIRING mode .

6. Configuration Command

In order to configuration the module , user need to pull “PAIR/CFG” pin low , and the module will be entry CONFIG mode , then sent configuration command via TXD , after configuration , pull “PAIR/CFG” pin high. The timing as below:

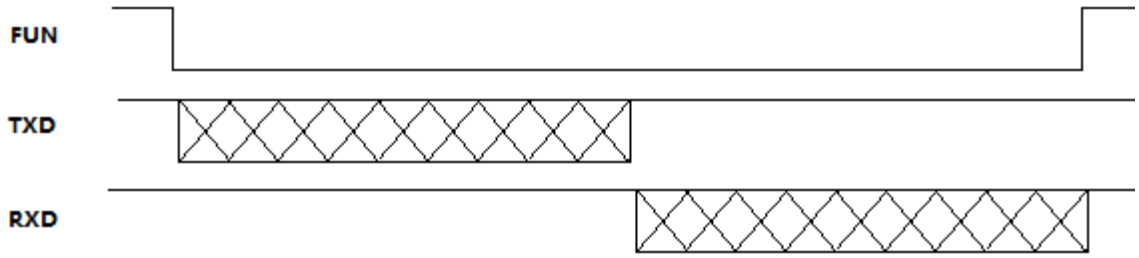


Diagram 10 Configuration timing chart

In CONFIG mode , user can sent command with any baud rate , the module will auto apply the baud rate , after exit the CONFIG mode , the module will apply baud rate as the configuration data (may be difference as the CONFIG mode).

The configuration command is full text mode, the format as below:

Fixed token: “=”

Command Value: 3 digitals with ASCII character (Range 000 to 999)

Fixed separators : “:”

Parameter: max 240 ASCII character

Fixed end symbol: “\r\n”

For the example:

MCU send command to LC-3000-P2P: =”**800:\r\n**” // Read local UID

LC-3000 response: =”**#800:100.000.000.001\r\n**” // This module’s UID is : 100.000.000.001

The detailed command list as below:

6.1. Set Loc ID

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"900"	"000"	":"	LOC_ID("AAA")	'\r\n'
Resp:OK	"#"	"900"	"000"	":"		'\r\n'
Resp:Error	Error Message					

Set the LC-3000 Loc ID. the LOC_ID is the UID number, value range: 000~125.

for example:

App Send: =900000:001\r\n， Set the LC-3000's UID to 001;

And LC-3000 will send back the response: =900000: when the set operation is done successfully. Otherwise, it will send back a error message, please check the Error Message List for detail.

This address id section combine with Loc id section form a four bytes address of the LC-3000. The address format:

Master/Slave	LC-3000 Address(DEV_ADDRESS)	
	3 bytes	1 byte
Master	ADDRESS_ID	LOC_ID
Slave	ADDRESS_ID	LOC_ID

Please note, the LOC_ID has already been written by INHAOS in factory. So user no need to changed it by default.

6.2. Set Address ID

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"901"	"000"	":"	ADDRESS_ID("AAA.AAA.AAA")	'\r\n'

Resp:OK	"#"	"901"	"000"	":"		'\r\n'
Resp:Error	Error Message					

Set the address id of the LC-3000, ADDRESS_ID is the 3 bytes address id value, the value range of each byte is: 000 to 255.

This address id section combine with Loc id section consist of a four bytes address of the LC-3000. The address bytes format:

Master/Slave	LC-3000 Address(DEV_ADDRESS)	
	3 bytes	1 byte
Master	ADDRESS_ID	LOC_ID
Slave	ADDRESS_ID	LOC_ID

6.3. Set Work Mode

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"902"	"000"	":"	WORK_MODE(A) "0": Full Speed Mode "1": Power Save Mode "2": Sleep Mode	'\r\n'
Resp:OK	"#"	"902"	"000"	":"		'\r\n'
Resp:Error	Error Message					

6.4. Set Net Mode

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"903"	"000"	":"	NET_MODE(A),LOC_ID(AAA)	'\r\n'
Resp:OK	"#"	"903"	"000"	":"		'\r\n'
Resp:Error	Error Message					

NET_MODE: 0 -- M2M Mode; 1 -- P2P Mode

LOC_ID: The loc id of the slave LC-3000, its value range is 000 to 125

Set the net mode of the LC-3000, and this command only available for master, and not for slave. If send it to a slave LC-3000, it will response a error message.

6.5. Clear FIFO

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"904"	"000"	":"		'\r\n'
Resp:OK	"#"	"904"	"000"	":"		'\r\n'
Resp:Error	Error Message					

Clear the TX FIFO of the LC-3000. After received this command, the LC-3000 will stop the RF sending process and clear all the waiting data in FIFO, then reset the BUSY pin to HIGH.

6.6. Set Query Priority

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"905"	"000"	":"	LOC_ID(AAA),PRIORITY(A)	'\r\n'
Resp:OK	"#"	"905"	"000"	":"		'\r\n'
Resp:Error	Error Message					

Set the query priority of the specific Loc id in master LC-3000. LOC_ID is the UID number, value range: 000~125. And the value range of PRIORITY is: 0 to 3.

6.7. Clear Device List

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"906"	"000"	":"		'\r\n'
Resp:OK	"#"	"906"	"000"	":"		'\r\n'
Resp:Error	Error Message					

Remove all device in device list.

6.8. Remove One from Device List

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"907"	"000"	":"	LOC_ID(XXX)	'\r\n'

Resp:OK	"#"	"907"	"000"	":"		'\r\n'
Resp:Error	Error Message					

6.9. Save Device to List

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"908"	"000"	":"	LOC_ID(AAA), DEV_ADDRESS(AAA.AAA.AAA.AAA)	'\r\n'
Resp:OK	"#"	"908"	"000"	":"		'\r\n'
Resp:Error	Error Message					

Save device address to device list directly, skip the pairing process. The LOC_ID Specifies the storage location of the device, and the DEV_ADDRESS is the 4 bytes device address of LC-3000.

Please note:

The LOC_ID and the LSB of DEV_ADDRESS Must follow the rules:

For master, the LOC_ID = LSB(DEV_ADDRESS) + 128;

For slave, the LOC_ID= LSB(DEV_ADDRESS)

6.10. Set pairing mode

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"909"	"000"	":"	Enable(A) "0": disable "1": enable	'\r\n'
Resp:OK	"#"	"909"	"000"	":"		'\r\n'
Resp:Error	Error Message					

6.11. Set P2P baudrate

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"910"	"000"	":"	BAUDRATE_P2P(XXXXXX)	'\r\n'
Resp:OK	"#"	"910"	"000"	":"		'\r\n'

Resp:Error	Error Message

Set the UART baudrate for P2P mode. And the BAUDRATE_P2P is the baudrate value, and its value range: "000000"~"256000".

6.12. Set M2M default baudrate

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"911"	"000"	":"	BAUDRATE_M2M(XXXXXX)	'\r\n'
Resp:OK	"#"	"911"	"000"	":"		'\r\n'
Resp:Error	Error Message					

Set the default UART baud rate for M2M mode which will be used after the device powered on and has no any APP command received yet. And the BAUDRATE_M2M is the baud rate value, and its value range: "000000"~"256000".

6.13. Set TX timeout

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"912"	"000"	":"	TX_MAX_TIME(XXXXXX)	'\r\n'
Resp:OK	"#"	"912"	"000"	":"		'\r\n'
Resp:Error	Error Message					

Set the maximum time for RF sending process after received data from APP. After then sending time exceeds TX_MAX_TIME, then the LC-3000 will stop sending and response a error message to APP. The value range of TX_MAX_TIME is: "0" to "65535".

6.14. Set Broadcast Timeout

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"913"	"000"	":"	BROADCAST_TIME("0"~"65535", Default: "1000")	'\r\n'
Resp:OK	"#"	"913"	"000"	":"		'\r\n'
Resp:Error	Error Message					

6.15. Set Assert Timeout

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"915"	"000"	":"	TX_ASSERT_TIME(XXXXXX)	'\r\n'
Resp:OK	"#"	"915"	"000"	":"		'\r\n'
Resp:Error	Error Message					

Set the maximum waiting time for the user application to make the response after LC-3000 sending data to it. After the waiting time exceeds the TX_ASSERT_TIME, the LC-3000 will send an error response to the remote LC-3000.

The value range of TX_ASSERT_TIME is: "0" to "65535".

6.16. Get Device Address

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"801"	"000"	":"		'\r\n'
Resp:OK	"#"	"801"	"000"	":"	DEV_ADDRESS ("AAA.AAA.AAA.AAA")	'\r\n'
Resp:Error	Error Message					

6.17. Get Work Mode

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"802"	"000"	":"		'\r\n'
Resp:OK	"#"	"802"	"000"	":"	WORK_MODE(A) "0": Full Speed Mode "1": Power Save Mode "2": Sleep Mode "3": Pairing Mode	'\r\n'
Resp:Error	Error Message					

6.18. Get Net Mode

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"803"	"000"	":"		'\r\n'

Resp:OK	"#"	"803"	"000"	":"	NET_MODE(A),LOC_ID(AAA)	'\r\n'
Resp:Error	Error Message					

NET_MODE: 0 -- M2M Mode; 1 -- P2P Mode

LOC_ID: The loc id of the slave LC-3000, its value range is 000 to 125

6.19. Get Device list changed count

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"804"	"000"	":"		'\r\n'
Resp:OK	"#"	"804"	"000"	":"	DEV_LIST_CHANGE_COUNT	'\r\n'
Resp:Error	Error Message					

This command return the device list changed count, and its value range: "00000"~"65535". Each time when the device list changed the DEV_LIST_CHANGE_COUNT will increase one. By query the DEV_LIST_CHANGE_COUNT, the APP can clearly know whether the device list has been modified or not.

6.20. Read one from Device list

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"805"	"000"	":"	LOC_ID ("0"~"125")	'\r\n'
Resp:OK	"#"	"805"	"000"	":"	ENABLE(A), DEV_ADDRESS (AAA.AAA.AAA.AAA), PRIORITY (A)	'\r\n'
Resp:Error	Error Message					

Read one device info from device list.

ENABLE indicates whether the device is valid or not. ENABLE=0, the device info is invalid; ENABLE=1, the device info is valid.

6.21. Get Device List Mask

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"806"	"000"	":"		'\r\n'
Resp:OK	"#"	"806"	"000"	":"	DEV_MASK("AA,AA,AA,AA,...")	'\r\n'
Resp:Error	Error Message					

Get the device list mask of LC-3000. The DEV_MASK consists of 16 bytes hex string and each bit of them mapping to one device of the device list. If the bit is set, indicates the corresponding device in the device list is valid; otherwise, the device info is invalid.

6.22. Get Version

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"807"	"000"	":"		'\r\n'
Resp:OK	"#"	"807"	"000"	":"	FW(AAA.AAA),HW(AAA.AAA),PT(AAA.AAA),M/S(A)	'\r\n'
Resp:Error	Error Message					

This command read the version information of the LC-3000.

The version information including:

- FW - Firmware version;
- HW - Hardware version;
- PT - Protocol version
- M/S-Master or slave, "M"-- Master; "S"--Slave

6.23. Get RF state

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"808"	"000"	":"		'\r\n'
Resp:OK	"#"	"808"	"000"	":"	RF_STATE(A) "0": Free "1": busy "2": success "3": fail	'\r\n'
Resp:Error	Error Message					

6.24. Broadcast

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"809"	"000"	":"	DATA(max 51Bytes)	'\r\n'

Response	"#"	"809"	"000"	":"		'\r\n'
Remote Received	"B"	"809"	"000"	":"	DATA(max 51Bytes)	'\r\n'

App can use this command to broadcast data to all valid devices in the device list. The broadcast time is determined by the BROADCAST_TIME. Please note: the head section is different between send packet and the receive packet and also the response packet. The user application need using the "=" as head in the broadcast packet write to LC-3000, and LC-3000 will response with "#" as the its response packet head, moreover for the slave modules will receive the broadcast packet start with "B" head. The reason to do so is that to facilitate the user application to distinguish those packets: with normal data, with broadcast data and the response data. And the most important is that user application needs response to LC-3000 whenever a normal data packet received.

6.25. Get P2P baudrate

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"810"	"000"	":"		'\r\n'
Resp:OK	"#"	"810"	"000"	":"	BAUDRATE_P2P(XXXXXX)	'\r\n'
Resp:Error	Error Message					

Get the UART baud rate for P2P mode. And the BAUDRATE_P2P is the baud rate value, and its value range: "000000"~"256000".

6.26. Get M2M default baudrate

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"811"	"000"	":"		'\r\n'
Resp:OK	"#"	"811"	"000"	":"	BAUDRATE_M2M(XXXXXX)	'\r\n'
Resp:Error	Error Message					

Get the default UART baud rate for M2M mode, and the BAUDRATE_M2M is the baud rate value, and its value range: "000000"~"256000".

6.27. Get TX timeout

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"812"	"000"	":"		'\r\n'
Resp:OK	"#"	"812"	"000"	":"	TX_MAX_TIME(XXXXXX)	'\r\n'

Resp:Error	Error Message

Get the maximum time for RF sending process after received data from APP. The value range of TX_MAX_TIME is: "0" to "65535".

6.28. Get Broadcast Timeout

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"813"	"000"	":"		'\r\n'
Resp:OK	"#"	"813"	"000"	":"	BROADCAST_TIME ("0"~"65535")	'\r\n'
Resp:Error	Error Message					

6.29. Get Assert timeout

Dir	Head	CMD	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"815"	"000"	":"		'\r\n'
Resp:OK	"#"	"815"	"000"	":"	TX_ASSERT_TIME(XXXXXX)	'\r\n'
Resp:Error	Error Message					

6.30. Data Command

Dir	Head	LOC_ID	Length	Separator	Data(max 240Bytes)	End
APP Send	"="	"000"~"125"	"001"~"240"	":"	DATA (max 240Bytes)	'\r\n'
Resp:OK	"#"	"000"~"125"	"000"	":"		'\r\n'
Resp:Error	Error Message					
Remote Receive	"="	"000"~"125"	"001"~"240"	":"	DATA (max 240Bytes)	'\r\n'

App can use this command for sending data to any valid devices in device list. The aim device is decided by the LOC_ID section. And the DATA section can use the ASCII string or any other coding format.

Please note that unlike the broadcast command, the data command's data packet's head section will not be modified when the remote LC-3000 module received it.

7. LC3000 Arduino Library

INHAOS provides a LC-3000 library on Arduino platform. You can download it from the download page in INHAOS websit: www.inhaos.com.

In order to compatible with various kind of usage, LC3000 library does not manage any detail serial port to communicate with LC-3000 module, so user application must processing the serial port function, such as: serial port write and serial port event handling. LC3000 library gives three callback function interface: LC3000_WriteFunc, LC3000_EventProc and LC3000_SerialListenFunc for interaction with the serial port in user application.

7.1. LC3000 write callback function

Prototype:

void LC3000_WriteFunc(uint8_t *wData, uint16_t len)

Function:

write a specific length of data to serial.

Parameter:

wData -- the data array bytes need to be wrote

len -- the length of the data bytes array

Example:

```
void LC3000_WriteFunc(unit8_t *wData, uint16_t len)
{
    Serial.write(wData,len); //Print data via Serial Port
}
```

7.2. LC3000 Serial listen callback function

Prototype:

void LC3000_SerialListenFunc()

Function:

When user application use a software serial to as a communicate interface with LC-3000 module, user application need to complete this function. This call function mainly used for LC3000 library call any synchronous functions, such as: setAddress(), sendData() .etc. For those synchronous function will block the loop progress of the arduino, so LC3000 cannot received the data from the software serial. After add the polling software serial code into the callback function, LC3000 will call this callback function to query the software serial data itself in synchronous functions.

Parameter:

None

Example:

```
void LC3000_SerialListenFunc()
{
    mySerial.listen();
    /* if use software serial , you should add the following codes */
    if (mySerial.aveilable())
    {
        inChar = mySerial.read();
        lc01.receiveByte(inChar);
    }
}
```

7.3. LC3000 Event callback function

Prototype:

bool LC3000_EventProc(uint8_t eventType, uint16_t cmdOrGroup, uint8_t *eventData, uint16_t eventDataLen)

Function:

called by LC3000 library when any event occur.

Parameter:

eventType -- The type of the event. There are three kinds of eventType:

LC_EVENT_DATA: Indicates a new data packet has been received from the remote LC-3000 module; The LOC_ID of the remote LC-3000 is store in cmdOrGroup; and you can read the detail data bytes from eventData pointer with the data length eventDataLen.

LC_EVENT_BROADCAST: Indicates a new broadcast data packet has been received from the remote LC-3000 module. And also the LOC_ID of the remote LC-3000 is store in cmdOrGroup; and you can read the detail data bytes from eventData pointer with the data length eventDataLen.

LC_EVENT_RESPONSE: Indicates a response packet received. Here the cmdOrGroup is the response cmd and the eventData and eventDataLen only valid when the response packet has data coming with.

cmdOrGroup -- The LOC_ID of the remote LC-3000 module when eventType is LC_EVENT_DATA or LC_EVENT_BROADCAST; And the command for eventType is LC_EVENT_RESPONSE

eventData -- The data array bytes pointer

eventDataLen -- The valid length in data array bytes

Example:

C++ Code

```
bool LC3000_EventProc(uint8_t eventType, uint16_t cmdOrGroup ,  uint8_t *eventData, uint16_t eventDataLen)
{
  switch (eventType)
  {
    case LC_EVENT_DATA:
        mySerial.print("data received from:");
        mySerial.println(cmdOrGroup);
        mySerial,print("data array:");
        mySerial.println(eventData, eventDataLen);
      break;
    case LC_EVENT_BROADCAST:
      break;
    case LC_EVENT_RESPONSE:
    if( cmdOrGroup == sendCMD )
    {
      dataValid = true;
      mySerial.print(cmdOrGroup);
      mySerial.println("Send OK");
    }
      break;
  }
  return true;
}
```

7.4. The steps for using the LC3000 on Arduino

7.4.1. Include the library into your Arduino Sketch.

```
#include <LC3000.h>
#include <LC3000_Config.h>
```

7.4.2. Declare the LC3000 object

```
LC3000 lc01(2, 3);    //configPin, busyPin
```

7.4.3. Initialize the serial port which connected to LC-3000

```
void setup() {
  // put your setup code here, to run once:
  delay(500);
  Serial.begin(9600);
  Serial.println(F("Serial inited"));
}
```

7.4.4. Call the begin function of the LC3000 object

```

void setup() {
  // put your setup code here, to run once:
  delay(500);
  Serial.begin(9600);
  Serial.println(F("Serial inited"));

  lc01.begin(LC3000_WriteFunc, LC3000_EventProc, LC3000_SerialListenFunc);
}
```

7.4.5. Call the doLoop() function of LC3000 object

```
void loop() {
  // put your main code here, to run repeatedly:
  lc01.doLoop();
```

7.4.6. Call the receiveByte() function in your serial port event function

```
void serialEvent() {
  while (Serial.available()) {
    // get the new byte:
    byte inChar = Serial.read();
    lc01.receiveByte(inChar);
  }
}
```

7.4.7. Add LC3000 write callback function

```
void LC3000_WriteFunc(uint8_t *wData, uint16_t len)
{
  Serial.write(wData, len);    //USE SERIAL1
  // Serialx.write(wData, len);    //USE SERIAL 2,3,4...
  // mySerial.write(wData, len);  //USE SOFTWARE SERIAL
}
```

7.4.8. Add Serial Listen callback function

If you are using the software serial, you need to do your software serial polling in the LC3000 SerialListenFunc() function. Otherwise, your software serial data cannot be delivery to LC3000 library.

```
void LC3000_SerialListenFunc()
{
    mySerial.listen();
    /* if use software serial , you should add the following codes */
    if (mySerial.aveilable())
    {
        inChar = mySerial.read();
        lc01.receiveByte(inChar);
    }
}
```

7.4.9. Add the LC3000_EventProc() function

```
bool LC3000_EventProc(uint8_t eventType, uint16_t cmdOrGroup ,  uint8_t *eventData, uint16_t eventDataLen)
{
  switch (eventType)
  {
    case LC_EVENT_DATA:
      break;
    case LC_EVENT_BROADCAST:
      break;
    case LC_EVENT_RESPONSE:
       break;
  }
  return true;
}
```

7.4.10. Call the set/read functions in the LC3000 library

You can send the string or data now, by adding code in the loop() function. But before any data sending, you must call the availableForWrite() function to check whether a new data packet can be wrote into LC-3000.

```
void loop() {
  // put your main code here, to run repeatedly:
  lc01.doLoop();

  if (lc01.availableForWrite())
  {
    if (lc01.sendString( 1, "Hello LC3000"))
    {
      mySerial.println(F("Send OK"));
    }
    else
    {
      mySerial.println(F("Send Fail"));
    }
    delay(1000);
  }
}
```

7.5. LC3000 Library Configure

The default send buffer and receive buffer size is 128 bytes. And the max buffer size LC3000 support is 240 bytes.

User application can change the send and receive buffer size in LC3000 library. You can open the file "LC3000_Config.h" in the LC3000 library and change the following macros:

```
//User define

//LC3000 Send Buffer size

#define LC_SEND_BUFFER_SIZE          128    //Range: 19 to 240

//LC3000 Receive Buffer size

#define LC_RECV_BUFFER_SIZE          128    //Range: 47 to 240
```

After changed this config file, you must save it and then you can rebuild your arduino sktech.

8. PC Tool LC-3000 Debugger Overview

In order to convenient for LC-3000 debugging, a PC tool: LC-3000 debugger is provided by INHAOS. The command page contains all the normal set/get command supported by LC-3000, and the device list page shows all the commands related to device list manage, include pairing, device list read, device list clear, etc. Please see the follows diagram for detail.

Your can download the LC-3000 Debugger from the download page in INHAOS website.

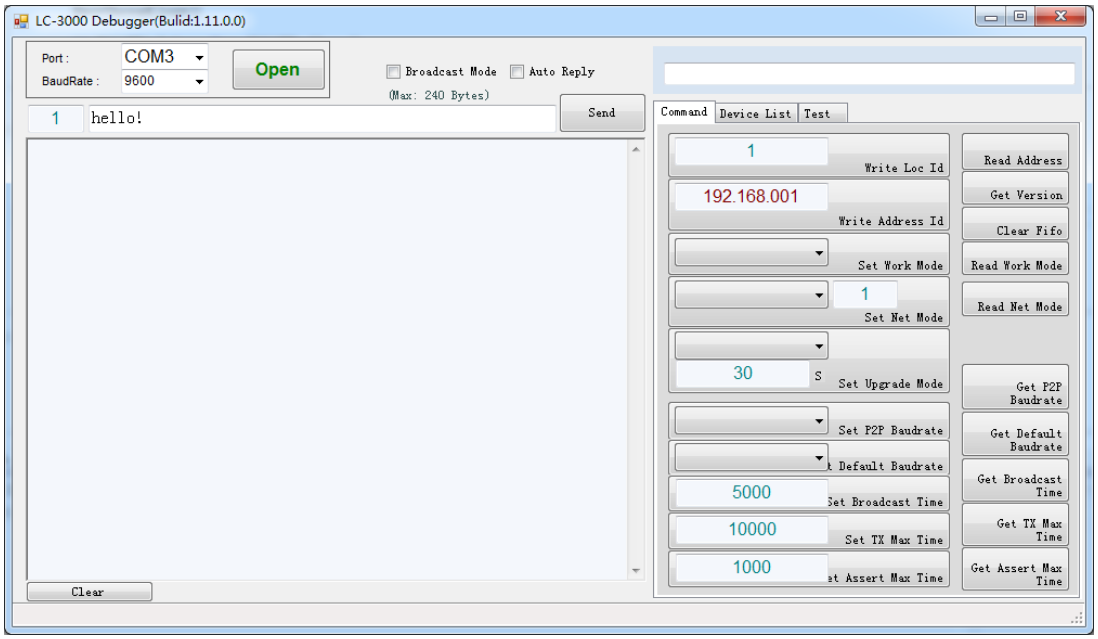


Diagram 11 LC-3000 Debugger Command Page

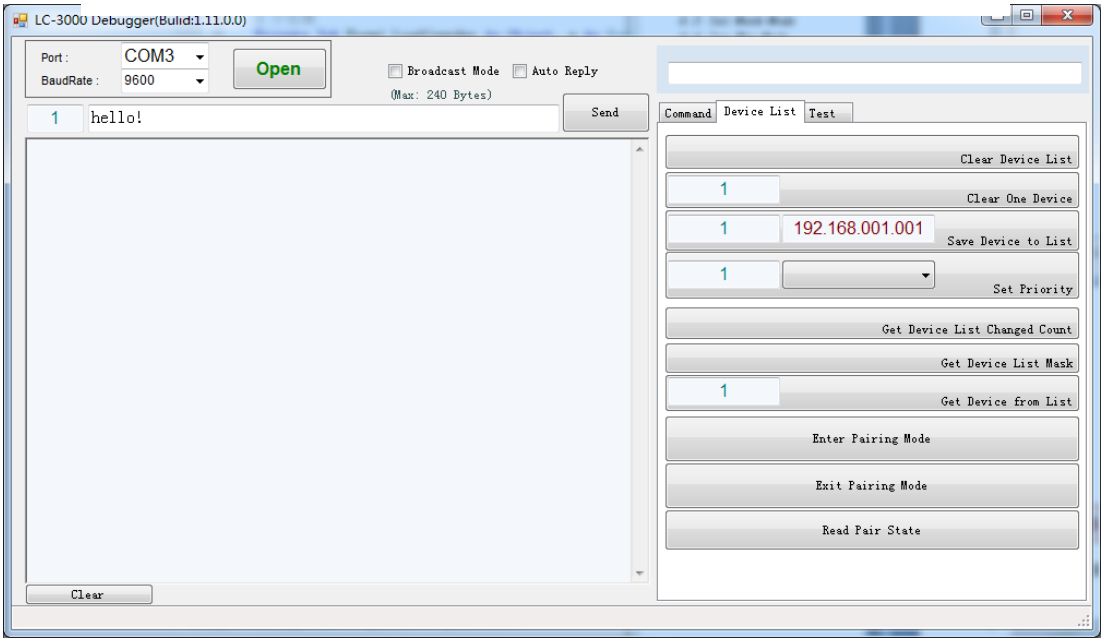


Diagram 12 LC-3000 Debugger Device List Page

9. CB-LC-2000 --- Connection Board for LC-3000(PA)

CB-LC-2000 provided a easy connection between LC-3000(PA) and the PC via UC-2102 USB to Serial convertor.

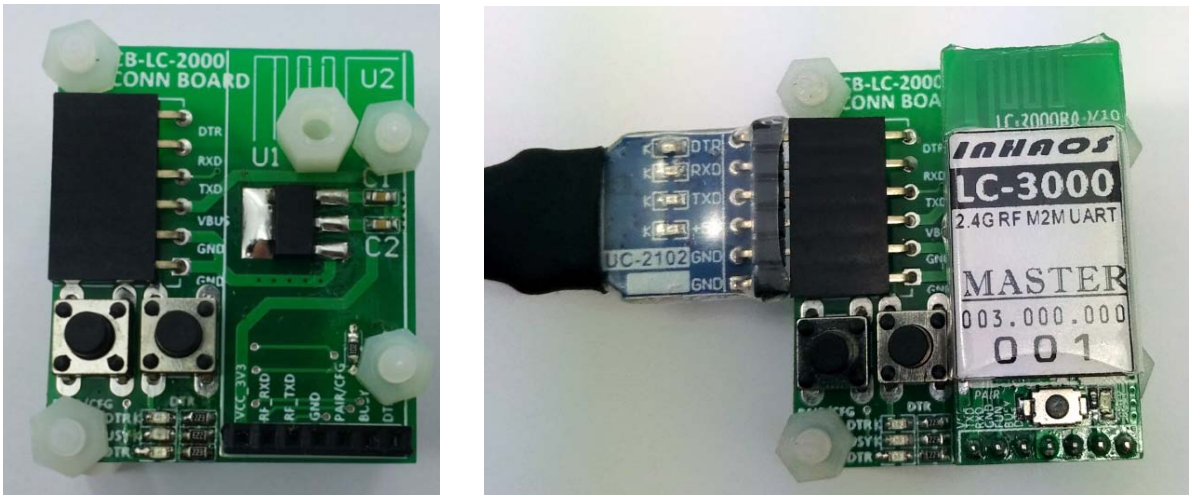


Diagram 14 CB-LC-2000 Pictures

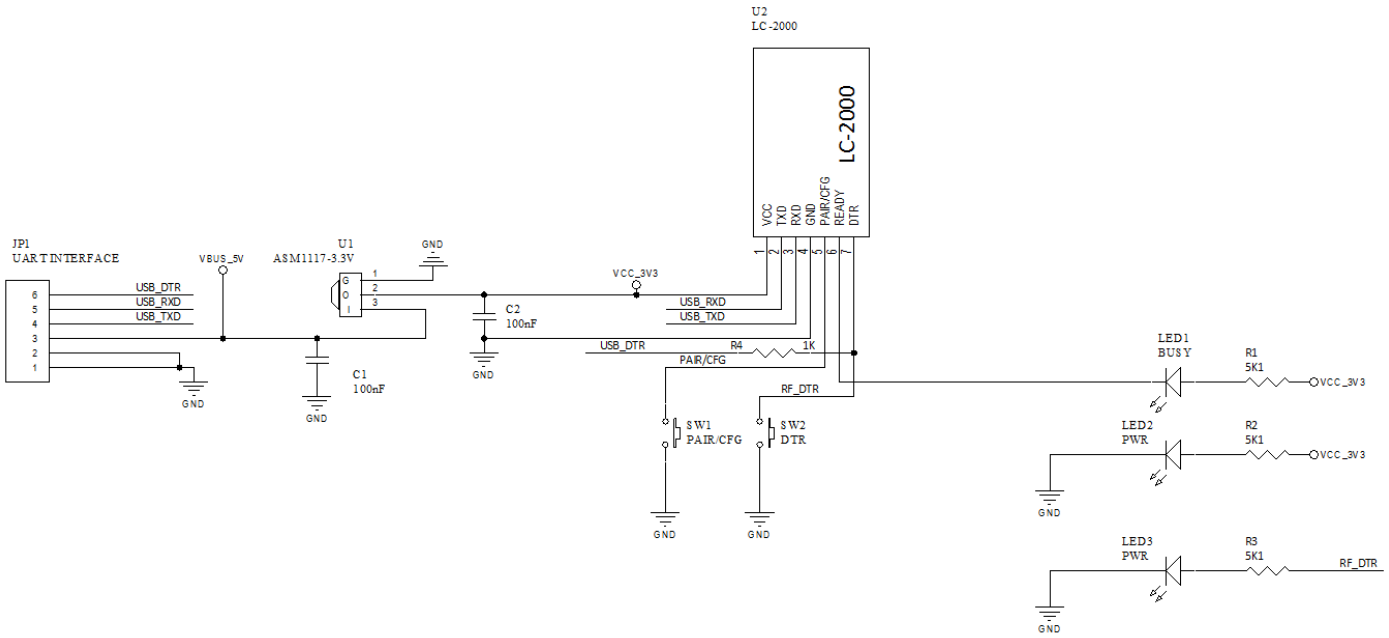


Diagram 13 CB-LC-2000 Schematic

One the CB-LC-3000 , here have two buttons and 3 LEDs , the function as below:

1, **DTR Button:** press this button to pull DTR pin low, in SLAVE side you should see DTR LED is OFF. When you connected SLAVE to a UNO R3 , you can reset the UNO by press this button.

2, **CFG button:** this button is connected to FUN pin , when you want sent configuration command from PC to Module , you can press and hold this button , and then sent command from the PC. Long press this button , the module will entry pairing mode.

3, **DTR LED:** indication DTR pin's status , when you press DTR button on MASTER side , the SLAVE side LED will be OFF.

4, **BUSY LED:** indication BUSY status, this LED light means module is busy.

5, **POWER LED:** when module powered the LED will be light.

10. Remote upload sketch for Arduino

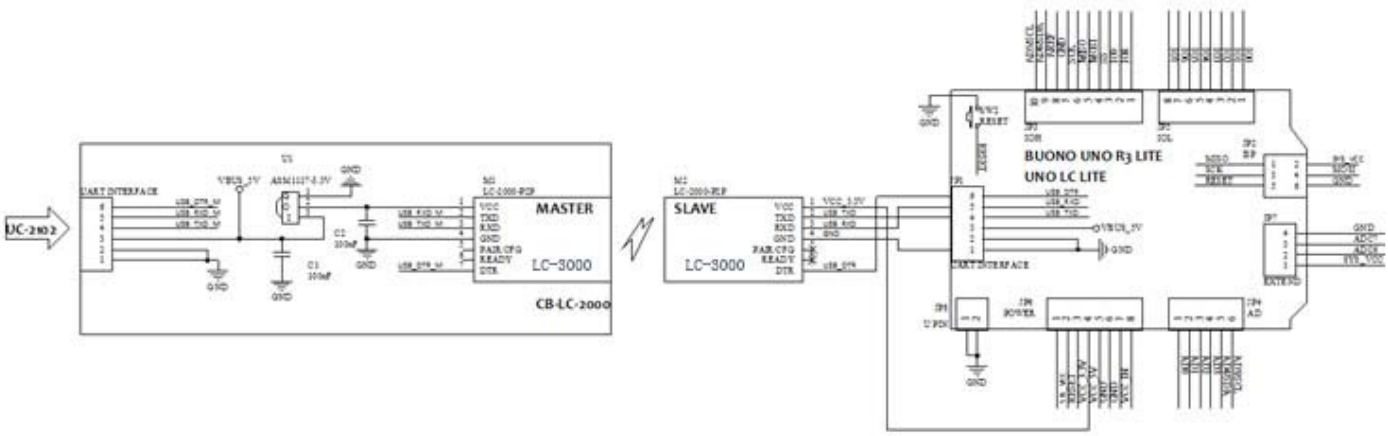
The remote upload sketch is a very special function of LC-3000(PA)-P2P, this feature support stand Arduino and the operation is very simple, in arduino system , the DTR pin is connected to RESET pin of the MCU via a capacitor , if user need to upload sketch , the Arduino IDE will be pull low the DTR pin , the RESET will get a negative pulse and the MCU will be reset and enter bootloader mode, then IDE will sent sync word to MCU , the MCU sent ACK and then Arduino IDE will start a data transmission.

The LC-3000(PA)-P2P will sent DTR signal from Master to Slave side , this will allow user to upload sketch by remote , most RF module like Bluetooth UART / WIFI UART does not support this function.

To implement this feature , user need to connected Master to PC side ,and Slave to Arduino side, and the periphery of the necessary parts in DTR to RESET patch is required.

INHAOS released some Arduino product without USB connect , we call it LITE version , for those module , it's very easy to connected to LC-3000(PA)-P2P, the module include: **BUONO UNO R3 LITE** and **BUONO UNO LC LITE** and **Mega2560-Core** and so on.

The Connection as below :

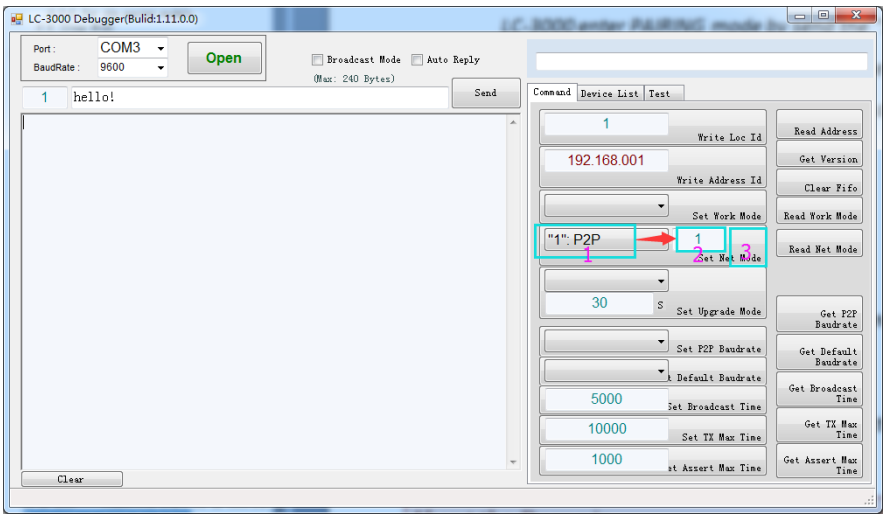


Please following the steps below to remote uploading Arduino sketch:

- Pairing the LC-3000 master and slave module

  You can pairing the LC-3000 before plug them into the board using two CB-LC-2000 board, or you can make the LC-3000 enter PAIRING mode by send the "Set Pairing Mode" command to it.
- Set the LC-3000 master and slave enter the P2P network mode

  Open the LC-3000 Debugger PC software, and using the "Set Net Mode" button to set the LC-3000 into P2P mode. Before click the "Set Net Mode" button please make sure the P2P mode selected and the Loc Id is the correctly the remote slave's Loc Id. After clicked the button, the LC-3000 will response the result of this operation, if it response OK, then you can goto the next step. Otherwise, please retry this step.



- Using Arduino to uploading the sketch

  Till this step, you can upload sketch, just the same as the wired connection uploading operation.

11. Order Information

Please notice the LC-3000(PA) have to working in multi pairs, multi Master and multi Slave , the main difference between Master and Slave are in below:

1, Baud Rate: In P2P mode , The master side can be config baud rate by config command (=911) ,and the slave side's baud rate will be follow the master side , for the example , if master set baud rate to 115Kbps , the slave side will be change to 115Kbps automatic. So in the application , user need to notes that have to keep slave side's MCU's baud rate setting same as the master's side's MCU baud rate setting.

2, DTR signal direction: In P2P mode , In the master side , the DTR pin is a input pin . In slave side , the DTR pin is a output pin while in P2P mode, the slave side's DTR signal will be follow the master side's DTR signal .

Please notice :

If you only use LC-3000(PA) for normal UART communication , it no matter who is master and who is slave ,

If user use it with Arduino and want to upload sketch by remote , the PC side must use Master and Arduino side must use Slave.

For LC-3000(PA) , we provide below purchase option:

No.	Purchase code	Package List	RF Power	Reference outdoor distance	Remark
1	LC-3000	LC-3000 Master * 1 LC-3000 Slave * 1	0dBm	20 to 30m	
2	LC-3000PA	LC-3000PA Master * 1 LC-3000PA Slave * 1	20dBm	200 to 300m	
3	CB-LC-3000	CB-LC-3000 * 1 UC-2102 * 1	---	---	Used for connect LC-3000(PA) to PC

INHAOS Headquarter：

1111 Oakmont Drive #C, San Jose, CA 95117

INHAOS China office：

No.6 Building,Songke Estate,Songshan Lake National Hi-tech Industrial Development Zone,Dongguan,Guangdong Province，523808,China