Wi07c

From ElectroDragon

Contents

Specification

- Module power 3.3V, regular current consumption at 70ma, peak current at 240mA (300mA must be able to provided)
- +20Dbm power, 100M max transmitting distance on ideal circumstance.
- It is common and correct to see some random error data when module is power up, and end up with "ready" (Turn baud rate to 115200 can see this actual debug data, this is used for firmware updating)

IC Features

- 802.11 b / g / n
- WIFI @ 2.4 GHz, supports WPA / WPA2 security mode
- Ultra-small size module 11.5mm * 11.5mm
- Built-in 10 bit precision ADC
- Built-in TCP / IP protocol stack
- Built-in TR switch, balun, LNA, power amplifier and matching network
- Built-in PLL, voltage regulator and power management components
- 802.11b mode + 19.5dBm output power
- Supports antenna diversity
- Off leakage current is less than 10uA
- Built-in low-power 32-bit CPU: can double as an application processor
- SDIO 2.0, SPI, UART
- STBC, 1x1 MIMO, 2x1 MIMO
- The guard interval A-MPDU, the polymerization of the A-MSDU and 0.4 s of
- Within 2ms of the wake, connect and transfer data packets
- Standby power consumption is less than 1.0mW (DTIM3)
- Operating temperature range -40 ~ 125 ℃

AT Commands

Format

- Baud rate at 57600, 115200 (new line) use option "send new line" or 'carriage return' for each command
- x is the commands

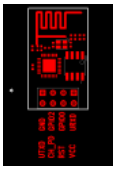
Set	Inquiry	Test	Execute
AT+<x>=<…>	AT+<x>?	AT+<x>=?	AT+<x>
AT+CWMODE=<mode>	AT+CWMODE?	AT+CWMODE=?	-
Set the network mode	Check current mode	Return which modes supported	-

Commands

- carefully there are must be no any spaces between the " and IP address or port

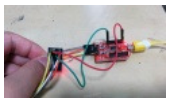
Commands	Description	Type	Set/Execute	Inquiry	test	Parameters and Examples
AT	general test	basic	-	-	-	-
AT+RST	restart the module	basic	-	-	-	-
AT+GMR	check firmware version	basic	-	-	-	-
AT+CWMODE	wifi mode	wifi	AT+CWMODE=<mode>	AT+CWMODE?	AT+CWMODE=?	1= Sta, 2= AP, 3=both, Sta is the default mode of router, AP is a normal mode for devices
AT+CWJAP	join the AP	wifi	AT+ CWJAP =<ssid>,< pwd >	AT+ CWJAP?	-	ssid = ssid, pwd = wifi password
AT+CWLAP	list the AP	wifi	AT+CWLAP			
AT+CWQAP	quit the AP	wifi	AT+CWQAP	-	AT+CWQAP=?	
AT+ CWSAP	set the parameters of AP	wifi	AT+ CWSAP= <ssid>, <pwd>,<chl>, <ecn>	AT+ CWSAP?		ssid, pwd, chl = channel, ecn = encryption; eg. Connect to your router: AT+CWJAP="www.electrodragon.com","helloworld"; and check if connected: AT+CWJAP?
AT+CWLIF	check join devices' IP	wifi	AT+CWLIF	-	-	
AT+ CIPSTATUS	get the connection status	TCP/IP	AT+ CIPSTATUS			<id>,<type>,<addr>,<port>,<tetype>= client or server mode
AT+CIPSTART	set up TCP or UDP connection	TCP/IP	1)single connection (+CIPMUX=0) AT+CIPSTART= <type>, <addr>,<port>; 2) multiple connection (+CIPMUX=1) AT+CIPSTART= <id> <type>,<addr>, <port>	-	AT+CIPSTART=?	id = 0-4, type = TCP/UDP, addr = IP address, port= port; eg. Connect to another TCP server, set multiple connection first: AT+CIPMUX=1; connect: AT+CIPSTART=4,"TCP","X1.X2.X3.X4",9999
AT+CIPMODE	set data transmission mode	TCP/IP	AT+CIPMODE=<mode>	AT+CIPSEND?		0 not data mode, 1 data mode; return "Link is builded"
AT+CIPSEND	send data	TCP/IP	1)single connection(+CIPMUX=0) AT+CIPSEND=<length>; 2) multiple connection (+CIPMUX=1) AT+CIPSEND= <id>, <length>		AT+CIPSEND=?	eg. send data: AT+CIPSEND=4,15 and then enter the data.
AT+CIPCLOSE	close TCP or UDP connection	TCP/IP	AT+CIPCLOSE=<id> or AT+CIPCLOSE		AT+CIPCLOSE=?	
AT+CIFSR	Get IP address	TCP/IP	AT+CIFSR		AT+ CIFSR=?	
AT+ CIPMUX	set mutiple connection	TCP/IP	AT+ CIPMUX=<mode>	AT+ CIPMUX?		0 for single connection 1 for multiple connection
AT+ CIPSERVER	set as server	TCP/IP	AT+ CIPSERVER= <mode>[,<port>]			mode 0 to close server mode, mode 1 to open; port = port; eg. turn on as a TCP server: AT+CIPSERVER=1,8888, check the self server IP address: AT+CIFSR=?
AT+ CIPSTO	Set the server timeout	AT+CIPSTO=	AT+CIPSTO?		< time>0~28800 in second	
+IPD	received data					For Single Connection mode(CIPMUX=0): + IPD, <len>: For Multi Connection mode(CIPMUX=1): + IPD, <id>, <len>: <data>

Pin Wiring (V090)

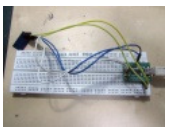


The pin definition

use a standard battery is the most optimized case



Connect 3V3, GND, TXD, RXD, CH_PD to 3V3 (red), GPIO0 to GND (green, **ONLY** connect when update firmware)



Use other FTDI FT232RL standard board from our store, two yellow wires to GND include GPIO0 for update

- Use FT232RL can supply enough power, must be genius IC of course
- Swap the uart pins if no data show up on the monitor
- There are two leds on the board, one is power led (RED), another one is status LED(BLUE), when power up, pwr led keeps on and status led will blink once.
- baud rate may work at 9600 (**seems the latest correct one**), 115200 or 57600

Module Pin Description

Pin	High Status	Low Status	Note
VCC, GND			Use standalone power source, or large capacitor, all power of this module from external
TXD, RXD			The serial port, swap these two pins if no data come up. this is very easily go wrong. (TX to RX, and RX to TX, not TX to TX and RX to RX)
RST	-	restart	
CH_PD	Flash boot and Update Mode	-	chip enable, so always connect to high status with VCC
GPIO0	-	Update mode	
GPIO 15 (when avaialble)	-	Flash boot/working Mode	Only for a few version, Wi07-3, normally already to GND when NA
GPIO 16 (RST)	Normal working mode	Hardware RST	
GPIO 2	should be high on booting, normally already set to high on default, don't pull it low on booting		

- No need any pull-up

Old version (V080)

The old version

Setup Verification

- check two LEDS status when boot up, if this is not working, double check your wiring first then continue, don't forget CHPD to VCC
- check if your devices (phone) can find a wifi spot named like "ESP_98529F" or similar, the later number part is the mac ID, if you can see this wifi spot, it means your module boot up sucessfully
- swap RX and TX pins if you can not get AT commands response
- Tick "new line" option on SSCOM32 serial port monitor tool
- Try baudrate 9600 or 115200 normally should be these two, old version is 115200

"ESP_990B15" is the module wifi spot

See the final "ready" when boot up successfully

- Don't forget to connect GPIO15 to GND if you are using the SMD model

First time use guide

Using arduino as serial port montior

- Connect VCC and GND of module to 3v3 and GND of arduino, RXD to TXD of arduino, and TXD to RXD of arduino (should add resistors or logic level shifter for logic level and protect IOs)
- Simply upload blink sketch to arduino, to ensure MCU won't use serial port
- User any other serial port monitor like SSCOM32 we used here, available here UART
- In the serial port, you should see "ready" in the end of the random data after powered up.
- Send AT (commands, with "newline option")will receive OK in return.

Steps and note

- **AT+RST** restart the module, received some strange data, and "ready"
- **AT+CWMODe=3** change the working mode to 3, AP+STA, only use the most versatile mode 3 (AT+RST may be necessary when this is done.)

Join Router

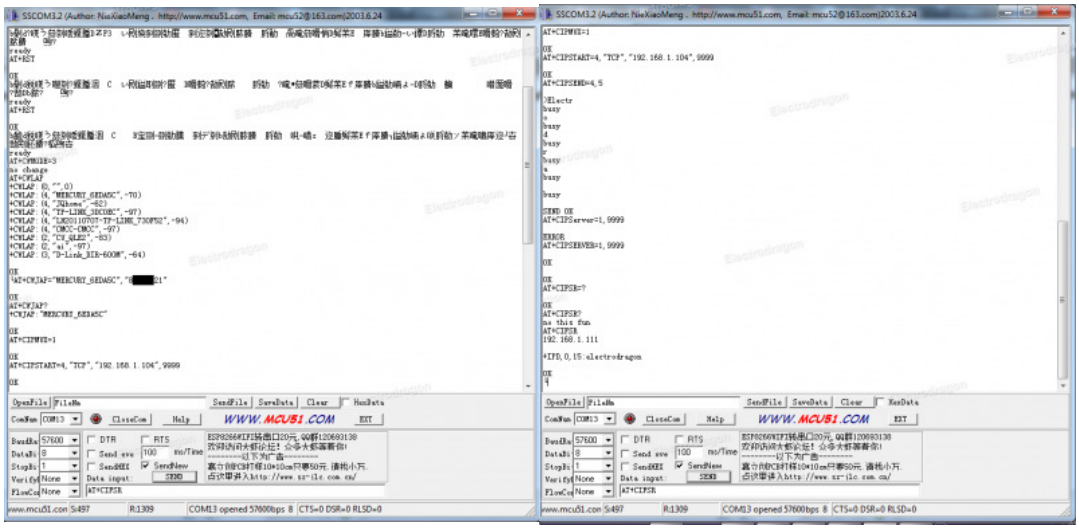
- **AT+CWLAP** search available wifi spot
- **AT+CWJAP=“you ssid”, “password”** join my mercury router spot (ops, the wifi password is here :))
- **AT+CWJAP=?** check if connected successfully, or use AT+CWJAP?

TCP Client

- **AT+CIPMUX=1** turn on multiple connection
- **AT+CIPSTART=4,"TCP","192,168.1.104",9999** connect to remote TCP server 192.168.1.104 (the PC)
- **AT+CIPMODE=1** optionally enter into data transmission mode
- **AT+CIPSEND=4,5** send data via channel 4, 5 bytes length (see socket test result below, only "elect" received), link will be "unlink" when no data go through

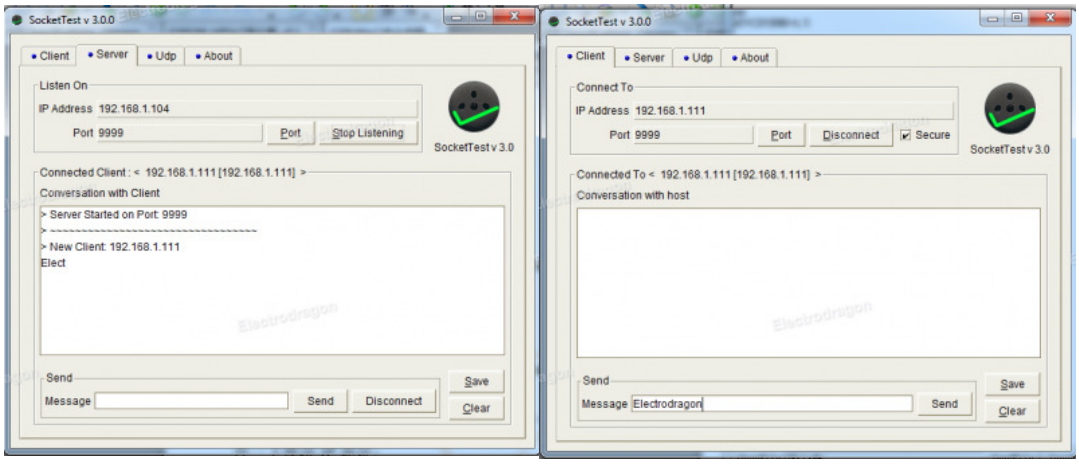
TCP Server

- **AT+CIPSERVER=1,9999** setup TCP server, on port 9999, 1 means enable
- **AT+CIFSR** check module IP address
- PC as a TCP client connect to module using socket test, send data



Socket test running result

- In the sockettest, do not tick the "secure" in TCP client, it causes unstable



Other Arduino Demo Code

- In this case, the wifi module still connect to hardware serial (software serial port can not higher than 19200 baud rate), and another software serial port should be created on arduino and print out via another serial port
- So the connection should be

```
Wifi's uart to arduino hardware uart;
arduino's sotware UART to another serial port device, for example like FTDI basic, CP2102 breakout, etc, and this serial port device can connect to PC to read data
```

- change the SSID and password in code for your wifi router

```
#include <SoftwareSerial.h>
#define SSID        "xxxxxxxx"
#define PASS        "xxxxxxxx"
#define DST_IP      "220.181.111.85"    //baidu.com
SoftwareSerial dbgSerial(10, 11); // RX, TX
void setup()
{
  // Open serial communications and wait for port to open:
  Serial.begin(57600);
  Serial.setTimeout(5000);
  dbgSerial.begin(9600);  //can't be faster than 19200 for softserial
  dbgSerial.println("ESP8266 Demo");
  //test if the module is ready
  Serial.println("AT+RST");
  delay(1000);
  if (Serial.find("ready"))
  {
    dbgSerial.println("Module is ready");
  }
  else
  {
    dbgSerial.println("Module have no response.");
    while (1);
  }
  delay(1000);
  //connect to the wifi
  boolean connected = false;
  for (int i = 0; i < 5; i++)
  {
    if (connectWiFi())
    {
      connected = true;
      break;
    }
  }
  if (!connected) {
    while (1);
  }
  delay(5000);
  //print the ip addr
  /*Serial.println("AT+CIFSR");
  dbgSerial.println("ip address:");
  while (Serial.available())
    dbgSerial.write(Serial.read());*/
  //set the single connection mode
  Serial.println("AT+CIPMUX=0");
}
void loop()
{
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += DST_IP;
  cmd += "\",80";
  Serial.println(cmd);
  dbgSerial.println(cmd);
  if (Serial.find("Error")) return;
  cmd = "GET / HTTP/1.0\r\n\r\n";
  Serial.print("AT+CIPSEND=");
  Serial.println(cmd.length());
  if (Serial.find(">"))
  {
    dbgSerial.print(">");
  } else
  {
    Serial.println("AT+CIPCLOSE");
    dbgSerial.println("connect timeout");
    delay(1000);
    return;
  }
```

```
  Serial.print(cmd);
  delay(2000);
  //Serial.find("+IPD");
  while (Serial.available())
  {
    char c = Serial.read();
    dbgSerial.write(c);
    if (c == '\r') dbgSerial.print('\n');
  }
  dbgSerial.println("====");
  delay(1000);
}
boolean connectWiFi()
{
  Serial.println("AT+CWMODE=1");
  String cmd = "AT+CWJAP=\"";
  cmd += SSID;
  cmd += "\",\"";
  cmd += PASS;
  cmd += "\"";
  dbgSerial.println(cmd);
  Serial.println(cmd);
  delay(2000);
  if (Serial.find("OK"))
  {
    dbgSerial.println("OK, Connected to WiFi.");
    return true;
  } else
  {
    dbgSerial.println("Can not connect to the WiFi.");
    return false;
  }
}
```

Debug and Note

- Better use standalone power source, not using power from USB-TTL module, it may not able to provide sufficient current.
- Module will disconnect "unlink" TCP/UDP when no data go through
- Wait AT commands feedback and continue, otherwise will return "busy"
- Potential cause for "error" : password length must be more than 8 bytes, use multiple connection and mode three, try disconnect current connection before try "AT+CWLAP" (module will reconnect after restart), re-flash firmware.
- mac address please check in your router page or use arp to check.
- (1st Oct.) change CR to CR/LF (\r\n in coding), which means "carriage return and line feed" for new firmware version 0.92

Firmware

You can find all the tools on our documents link at the end of this page. two types of firmware examples available: AT (Commands, normally only UART pins used) and IO (Control, full IOs version)

ESP flash download tool, image show the setting of updating

ESP8266 flasher, interface in chinese, but one click to done

less friendly, a little complex to use, only COM1-6

- ESPTool (Python) in command line (https://github.com/themadinventor/esptool/)

How to use for XTCOM

- Download the bin file
- Set the module to update mode, connect the module : choose "tools" - "configure device"
- Upload bin file: API Test - Flash image download
- upload the bin file eagle.app.v6.flash.bin at 0x00000 (app)
- upload the second bin file eagle.app.v6.irom0text at 0x40000 (this one is optionally, libraries)

Updates

- Latest update can find it on espressif community (http://bbs.espressif.com/viewforum.php?f=5&sid=9d6b977753163ccd7b338340010e0862)

Main update (Cloud update to V0.9.2.0): See the post here about cloud update (http://blog.electrodragon.com/cloud-updating-your-wi07c-esp8266-now/) (From 00160900 to 00170901 and 00180902), or find the tool and firmware 00170901 here (https://drive.google.com/open?id=0B3dUKfqzZnlwVGc1YnFyUjgxelE&authuser=0) .

- Find most upated firmware on this google link. (https://drive.google.com/open?id=0B3dUKfqzZnlwRjFaNTUzZFptbzg&authuser=0) , and check the update log below
- V0.9.2.2 (https://drive.google.com/open?id=0B3dUKfqzZnlwdUJUc2hkZDUyVjA&authuser=0) - support to change baudrate, default baudrate is 9600, More stable version than the cloud updated version, few bugs fixed.

AT command:

Commands	Description	Note
AT+CIOBAUD	AT+CIOBAUD=? (Inquiry), AT+CIOBAUD? (check), AT+CIOBAUD=9600 (Set)	supported 9600, 19200, 38400, 57600, 74880, 115200, 230400,460800, 921600
AT+CSYSWDTENABLE	watchdog, auto restart when program have errors occurred, enable	-
AT+CSYSWDTDISABLE	watchdog, auto restart when program have errors occurred. disable	-

Custom firmware or similar

- LUA based firmware, easy to use (https://github.com/funshine/nodemcu-firmware)
- GPIO Control (https://drive.google.com/open?id=0B3dUKfqzZnlwZ3IzS21yVnFEaE0&authuser=0) . See explanation post here. (http://blog.electrodragon.com/esp8266-gpio-test-edited-firmware/)

Firmware Details

Boot Process

- Reset vector is 0x40000080.
- Boots into Espressif code in IROM0.
- Loads SPI ROM data.
- Starts executing ESP SDK-code shadowed SPI ROM (unconfirmed).

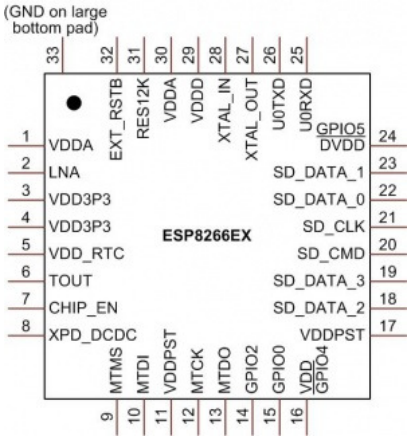
SPI Flash ROM Layout

Address	Size	Name	Description
00000h	248k	app.v6.flash.bin	User application
3E000h	8k	master_device_key.bin	OTA device key
40000h	240K	app.v6.irom0text.bin	SDK libraries
7C000h	8k	esp_init_data_default.bin	Default configuration
7E000h	8k	blank.bin	Filled with FFh. May be WiFi configuration.

SDK

- Official SDK link, maybe the most easist to use on ubuntu (https://drive.google.com/folderview?id=0B5bwBE9A5dBXaExvdDExVFNrUXM&usp=sharing)
- GCC Linux toolchain nurdspace First setup guide - GCC https://nurdspace.nl/ESP8266/First_setup, not yet tested to work with 0.92 firmware, work with 0.91 (seems different settings, need update)
- SDK source code and more tools please find on the drive for now, examples folder includes AT (default UART commands flashed in the module) and IoT demo (simple webserver, socket, light control, etc)

IC Pin Defintion



Pin	Name	Type	GPIO	Function	UART, IIC, 3Bits SDIO, LED	Plug/PWM light/Sensor	Boot
1	VDDA	P		Analog Power 3.0 ~3.6V			
2	LNA	IO		RF Antenna Interface,Chip Output Impedance=50 Ω Recommend that the π- type matching network is retained.			
3	VDD3P3	P		Amplifier Power 3.0~3.6V			
4	VDD3P3	P		Amplifier Power 3.0~3.6V			
5	VDD_RTC	P		NC(1.1V)			
6	TOUT	I		ADC Pin			
7	CHIP_EN	I		Chip Enable. High: On, chip works properly; Low: off power supply, minimum current			High for booting
8	XPD_DCDC	IO	GPIO16	Deep-Sleep Wakeup, external RST, low to reset, high for working mode			
9	MTMS	IO	GPIO14	HSPICLK	I2C SCL	IR recv	
10	MTDI	IO	GPIO12	HSPIQ	Communication status LED	(Light)PWM red LED control	
11	VDDPST	P		Digital/IO Power Supply (1.8V~3.3V)			
12	MTCK	IO	GPIO13	HSPID	RST Key	(Light)PWM blue LED control; (plug) RST Key	
13	MTDO	IO	GPIO15	HSPICS	3bits SDIO	(Light)PWM green LED control; (Plug) relay control, H/L TTL	Low for booting
14	GPIO2	IO	GPIO2	UART Tx during flash progamming (?); must be high when booting, already set to high on default	3bits SDIO; UART1 system info print (TX); I2C SDA		
15	GPIO0	IO	GPIO0	SPICS2	3bits SDIO; Status LED	(plug) wifi status LED	Low for flashing mode
16	GPIO4	IO	GPIO4				
17	VDDPST	P		Digital/IO Power Supply (1.8V~3.3V)			
18	SDIO_DATA_2	IO		Connect to SD_D2 (Series R 200Ω);SPIHD; HSPIHD			
19	SDIO_DATA_3	IO		Connect to SD_D3 (Series R 200Ω); SPIWP; HSPIWP			
20	SDIO_CMD	IO		Connect to SD_CMD(Series R 200Ω); SPICS0			
21	SDIO_CLK	IO		Connect to SD_CLK (Series R 200Ω); SPICLK			
22	SDIO_DATA_0	IO		Connect to SD_D0 (Series R 200Ω); SPIQ			
23	SDIO_DATA_1	IO		Connect to SD_D1 (Series R 200Ω); SPID			
24	GPIO5	P	GPIO5				
25	U0RXD	IO	GPIO3	UART Rx during flash progamming (?)	UART0 for user		
26	U0TXD	IO	GPIO1	GPIO1; SPICS1 (?)	UART0 for user, do not allow low when booting		
27	XTAL_OUT	IO		Connect to crystal output, can be used to provide BT clock input			
28	XTAL_IN	IO		Connect to crystal input			
29	VDDD	P		Analog Power 3.0~3.6V			
30	VDDA	P		Analog Power 3.0~3.6V			
31	RES12K	I		Connect to series R 12kΩ to ground			
32	EXT_RSTB	I		External reset signal (Low: Active)			

Documents

- Our documents link here, will keep updating from time to time. (https://drive.google.com/folderview?id=0B3dUKfqzZnlwRXhBTmlhaTROTmM&usp=sharing)
- ESP8266 wiki page, very helpful (https://github.com/esp8266/esp8266-wiki/wiki/Examples)
- ESP8266 IoT Standard example (in Progress)

Categories: Ethernet, Wifi | ESP8266
