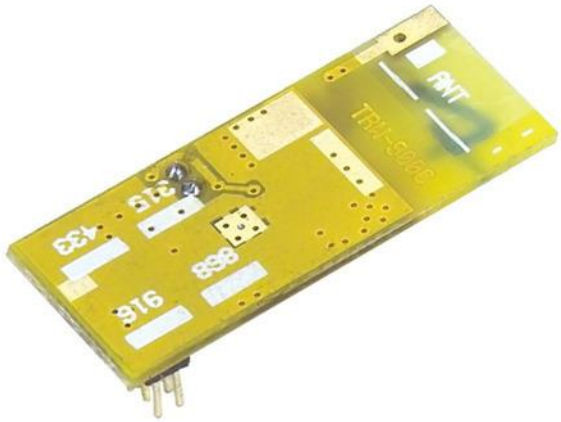
Wireless Low Cost 900MHz RF Transceiver Module



Version History

Version	Date	Changes
V1.01	Jun. 12, 2007	1st. Edition
V1.02	Aug.20,2007	2nd. Edition
V1.03	Mar.15,2008	3rd. Edition
V1.04	Apr.20,2008	4th. Edition
V1.05	Jun.18,2008	5th. Edition

Specification

● Remote Control Systems	● Wireless Data Transceiver
● 300MHz~928MHz ISM/SRD Band	● Remote Metering
● Wireless Security Systems	● Automatic Meter Reading
● Application Range : Remote Metering、Wireless Security Systems 、Automatic Meter 、Reading、Home Automation	

Application

The RF transceiver is integrated with a highly configurable baseband modem. The modem supports various modulation formats and has a configurable data rate up to 250 Kbps. The communication range can be increased by enabling a Forward Error Correction option, which is integrated in the modem.

TRW-900C provides extensive hardware support for packet handling, data buffering, burst transmissions, clear channel assessment, link quality indication and wake-on-radio.

The main operating parameters and the 64-byte transmit/receive FIFOs of TRW-900C can be controlled via an SPI interface. In a typical system, the TRW-900C will be used together with a microcontroller and a few additional passive components.

Key Feature

- Small size
- Separate 64-byte RX and TX data FIFOs
- Efficient SPI interface: All registers can be programmed with one "burst" transfer
- Programmable output power up to +10dBm
- High sensitivity (-111 dBm at 1.2 Kbps, 1% packet error rate)

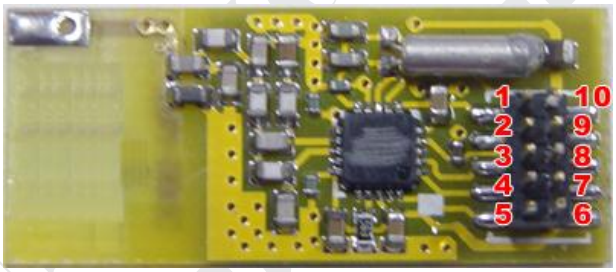
Absolute Maximum Rating

Parameter	Min	Max	Unit	Condition/Note
Supply Voltage	2.7V	3.3V	V	
Voltage on Any Digital Pin	-0.3	VDD+0.3	V	
Input RF Level		+10	dBm	
Storage Temperature Range	-50	150	℃	
ESD		<500	V	According to JEDEC STD 22, method A114, Human Body Model

General Characteristic and Electrical Specification

Parameter	Min	Type	Max	Unit	Condition/Note
Frequency Range	300		928	MHz	
Data Rate	1.2		250	Kbps	
Power Down Modes		400		nA	Voltage regulator to digital part off, register values retained (sleep state)
Current Consumption, RX		16		mA	
Current Consumption, TX		30		mA	Transmit mode, +10dBm output power
Receiver Sensitivity	1.2k data rate -111dBm				
Transmit Output Power			+10	dBm	Transmit mode

Pin Assignment

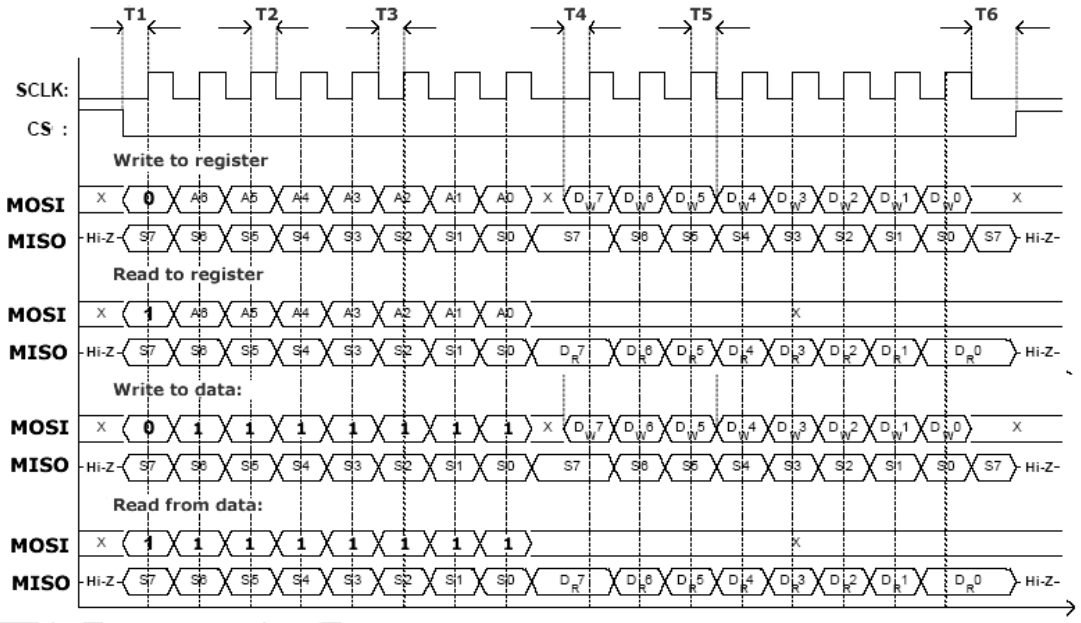


No.	Name	Type	Description
1	DR	Output	Normal state is 0. When it is 1, it expresses RF are sending or receiving data.
2	CS	Input	Low level is effective and it is selection module.
3	MISO	Output	The data which is read from module includes data and configuration information.
4	SCLK	Output	It provides clock for RF read and write.
5	MOSI	Output	The data which is written includes data and configuration information.
6,9,10	VDD		Power cathode.
7,8	GND		Power negative.

Timing Note

Data sending way of TRW-900C is taking SPI. When you want to send, please add the address, and the address is the direction of bit7. When it is 0, it is under writing information. When it is 1, it is under reading data, and purpose of reading on configuration is in order to verify the correctness of writing.

1. Configuration timing description: format: address + data
2. Data sending/ receive timing description: it's timing is the same as configuration and reading the written address is with different information.
   Sending format: 0 x7F + N of Byte (N <65)
   Receiving format: 0xFF+ N of Byte（N<61）
3. 0xFF+N of BYTE（N<61）, maximum Byte to write the information to the RF is 64 one time.
   Read out bits of information received and the address is 0xFF. Maximum reading Byte number is 64 one time.



Fsclk	<	6.5MHz			
T1	>	200us	T2	>	50ns
T3	>	50ns	T4	>	80ns
T5	>	20ns	T6	>	20ns

Order Value description

Order Value	Function Description
0x30	Reset RF module
0x32	Close OSC of RF
0x33	Automatically locked, and time is less than 1 ms.
0x34	Clean reception buffer zone.
0x35	Clean sending buffer zone.
0x36	Module is into statistic mode.
0x39	Module is into low consumption mode.
0x3A	Module is into receiving mode.
0x3B	Module is into sending mode.
0x7E	Transmission power set mode.
0x7F/0xFF	Writing sending/reading receiving buffer zone

Register Note

Address	1.2K	2.4K	4.8K	10K	38.4	76.8	100	250K	Function Note
	FSK	FSK	FSK	FSK	FSK	FSK	FSK	MSK	Modulation method.
0x0D	0x1E	0x1E	0x1E	0x1E	0x1E	0x1E	0x1E	0x1E	TRW-900C set up the fundamental frequency, and it is work-for-868 MHz.
0x0E	0xC4	0xC4	0xC4	0xC4	0xC4	0xC4	0xC4	0xC4	
0x0F	0xEC	0xEC	0xEC	0xEC	0xEC	0xEC	0xEC	0xEC	
0x13	0x23	0x23	0x23	0x23	0x23	0x23	0x23	0x23	13H[6:4] is the number of leader.
0x14	0x3B	0x3B	0x3B	0x3B	0x3B	0x3B	0x3B	0x3B	13H[1:0]14H is the fundamental frequency spacing.
0x0A	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	To set the channel.
0x0B	0x08	0x08	0x06	0x06	0x06	0x06	0x06	0x0B	0BH[3:0]IF setting.
0x0C	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	Constant.

0x10	0xF5	0xF6	0xC7	0xC8	0xCA	0x7B	0x5B	0x2D	Receive bandwidth set module rate set
0x11	0x83	0x83	0x83	0x93	0x83	0x83	0xF3	0x3B	
0x12	0x03	0x03	0x03	0x03	0x03	0x03	0x03	0x73	12H[7:4] modulation choice way.
0x15	0x15	0x15	0x15	0x34	0x34	0x42	0x47	0x00	Modulation setting.
0x22	0x10	0x10	0x10	0x10	0x10	0x10	0x10	0x10	The corresponding number of fixed rates are as follows. When frequency is less then 430.5MHz, the value of 0x0A.
0x21	0x56	0x56	0x56	0x56	0x56	0xB6	0xB6	0xB6	
0x18	0x08	0x08	0x08	0x08	0x08	0x08	0x08	0x08	
0x19	0x16	0x16	0x16	0x16	0x16	0x1D	0x1D	0x1D	
0x1A	0x6C	0x6C	0x6C	0x6C	0x6C	0x1C	0x1C	0x1C	
0x1B	0x03	0x03	0x43	0x43	0x43	0xC7	0xC7	0xC7	
0x1C	0x40	0x40	0x40	0x40	0x40	0x00	0x00	0x00	
0x1D	0x91	0x91	0x91	0x91	0x91	0xB2	0xB2	0xB2	
0x23	0xA9	0xA9	0xA9	0xA9	0xA9	0xEA	0xEA	0xEA	
0x24	0x2A	0x2A	0x2A	0x2A	0x2A	0x2A	0x2A	0x2A	
0x25	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	
0x26	0x11	0x11	0x11	0x11	0x11	0x11	0x11	0x11	
0x29	0x59	0x59	0x59	0x59	0x59	0x59	0x59	0x59	
0x2C	0x81	0x81	0x81	0x81	0x81	0x88	0x88	0x88	
0x2D	0x35	0x35	0x35	0x35	0x35	0x31	0x31	0x31	
0x2E	0x0B	0x0B	0x0B	0x0B	0x0B	0x0B	0x0B	0x0B	
0x08	0x05	0x05	0x05	0x05	0x05	0x05	0x05	0x05	Packet control.
0x02	0x06	0x06	0x06	0x06	0x06	0x06	0x06	0x06	Constant
0x00	0x06	0x06	0x06	0x1B	0x1B	0x1B	0x1B	0x1B	
0x06	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	Packet length setting and value is 0XFF when it is variable

This is table is based on fundamental frequency on 868MHz, frequency spacing is 250K，select the channel 0　and three is some reference rate in different channels.

1. **TRW-900C fundamental frequency setting：0DH，0EH，0FH**

   FREQ[23:0] = Fcarrier*2^16/26000000

   EX 1：frequency is 868MHz， FREQ[23:0] = 868000000*2^16/26000000 = 2187894

= 0x216276

And 0DH = 0x21　　0EH = 0x62　　0FH = 0x76

EX 2: frequency is 878.5MHz，FREQ[23:0] = 878500000*2^16/26000000 =2214360 = 0x21C9D8

And 0DH = 0x21　　0EH = 0xC9　　0FH = 0xD8

2. **Frequency space setting between the channel: 13H[1:0]14H**

$f_{CHANNEL}$ = 26000000*(256+11H[7:0])*2^10[1:0]/2^18

If you want set the frequency space of channel to 250K, the 14 = 0x3B,13[1:0]=3

3. **Selection of channel space:0x0A**

$f_{carrier} = f_{BASE} + f_{CHANNEL}$*0AH[7:0]

If fundamental frequency is 800MHz，f_{CHANNE} = 250K，0x0A=00，frequency will be 800MHz

If fundamental frequency is 800MHz ，f_{CHANNE} = 250K，0x0A=04，frequency will be 801MHz

4. **Leader code setting：13H[6:4]**

13H[6:4] = 0 is the second Byte leader code.

13H[6:4] = 1 is the third Byte leader code.

13H[6:4] = 2 is the forth Byte leader code.

13H[6:4] = 3 is the sixth Byte leader code.

13H[6:4] = 4 is the eighth leader code

13H[6:4] = 5 is the twelve Byte leader code

13H[6:4] = 6 is the sixteenth Byte leader code.

13H[6:4] = 7 則爲 24 個 Byte 的前導 is the twenty-forth Byte leader code.

5. **Reception bandwidth setting：10H[7:4]，the corresponding value is the corresponding frequency as bellows:**

10H[7:4]	= 00	812K	10H[7:4]	= 01	625K
10H[7:4]	= 02	541K	10H[7:4]	= 03	464K
10H[7:4]	= 04	406K	10H[7:4]	= 05	325K
10H[7:4]	= 06	270K	10H[7:4]	= 07	232K
10H[7:4]	= 08	203K	10H[7:4]	= 09	162K
10H[7:4]	= 10	135K	10H[7:4]	= 11	116K
10H[7:4]	= 12	102K	10H[7:4]	= 13	81K
10H[7:4]	= 14	68K	10H[7:4]	= 15	58K

6. **IF frequency setting：f_{IF} = 25390.625*0BH[3:0]**

Ex: 2.4K working rate, 0BH[3:0]=8, and f_{IF} = 203.125KHz

7. **Rate setting: 10H[3:0]，11H[7:0]，try to find the nearest value.**

RATE = (256+11H[7:0])*2^10H[3:0]*26000000/2^28

EX: rate is 2.4K and 11H = 0x83,10H[3:0] = 0xx6

EX: rate is 1.2K and 11H = 0x83,10H[3:0] = 0xx5

8. **Modulation : 12H[7:4]**

12H[7:4] = 0，it expresses selected 2-FSK mode.

12H[7:4] = 1，it expresses selected MSK mode.

9. Modulation set: 15H

a. Value will be 1 on MSK.

b. Modulation formula is as below when it is on 2-FSK and try to find the nearest value.

f_{dev} =（8+15H[2:0]*2^15H[6:4]*198

10. Packet control 08H

08H = 0x04，the packet length is fixed, the value of its length is in the 06H.

EX: When the 06 H in the value of 0 x70, packet in the launching and receiving is for 112 Byte. During transmit and receive, it need to subparagraph send and receive, and just can finish data receiving.

Transmit format: 0 x7F + N-Byte (N = 06H in value)

08H = 0x05，The length of the packet is variable, and the length of the value of transmitting is the first Byte information, 06H is in the value of 0 xFF.

Transmit format: 0 x7F + N-Byte (N = packet length)

Remark:

a. When the state of the transmitted, if the packet length is less or equal to 64, the information can written into TRW-900C module. If the length of packet is over 64 , you can write the data into TRW-9064 for one time only and paragraph to send the data to TRW-900C because TRW-900C only has one 64 Bytes buffer zone.

Paragraph way is as below:

10.1.1 Write 64 Bytes to TRW-900C at first time.

10.1.2 Wait for DR become to high.

10.1.3 Read 0XFA value. If the value is 0x10, it is express there are still 16 Bytes is not finish sending on buffer zone. If the value is 0x20, it is expresses there are 32 Bytes still not finish sending.

10.1.4 According the value of 0XFA which it read, the user can decided the time which he want to sent the data to TRW-900C, but use can’ t wait the value to 0, it will be wrong then.

10.1.5 If send the data again and it is still not finish sending, please repeat 10.1.13 and 10.1.14.

b. When it is under receiving, please wait the DR line to high. If packet length is less then 61, please wait DR line lower and read more two Bytes data. For those two more data, first one is RSSI value, another is CRC value. When CRC value bit7 is 1, it is expresses packet is correct, otherwise, it is wrong. RSSI value is with sign. If the value which it read is more big, it expresses receive single is better.

*Attention: 0XFF is -1 and it is less then 0. If packet length is over 60, the read method is as below:

10.1.6 Wait DR line to be 1

10.1.7 If 0xFB value is 51, please read 50 BYTE data from TRW-900C. (51/50 can define by user, and please note the value of 0XFB just need under 1.)

10.1.8 Repeat 10.2.2. During it is reading 0XFB, if DR line become to low, it expresses data is less then 50 Bytes. At this time, please withdraw from reading 0XFB status and read the rest data.

10.1.9 (Attention: read data are more then actual for two)

10.1.10 After DR line become low, the way to know who many data is still need to read:

10.1.11 Under constant packet status, value of 0x06 is 502 of N.

10.1.12 Under Variable packet status, the first read BYTE value is 50+2 of N.

11. Power set：

Frequency	Power Data Rate（dBm）													
	+10	+8	+6	+4	+2	+0	-2	-4	-6	-8	-10	-12	-15	-52
300~400MHz	C2	C8	CE	89	8D	3F	54	57	6B	35	34	33	1D	00
400~750MHz	C0	C5	81	88	8D	3F	53	57	2A	35	34	33	1C	00
750~999MHz	C3	C9	CE	87	8C	3F	54	57	2B	28	34	33	23	00

Address of setting power is 0x7E plus 8 BYTES data, please set those 8 Bytes to the same and select power value.

The setting power of read is 0Xfe plus 8 BYTES data, please check those 8 Bytes are the same of setting.

Question and Answer

1. Q: Power address of TRW-900C is any of 0x7E/7F/80/81/82/83/84/85 as deposit data?

   A: No, 0X7E address has 8 BYTE of data buffer (please just think about this is ok.), not any of 0x7E/7F/80/81/82/83/84/85.

2. Q: MISO will be equal to 0 when CS is equal to 0?

   A: Yes, when CS is 0, MISO will be to low. (But prerequisite must be MCI had been put this pin as input, otherwise, it will be affected.), and then put the data into module.

Application Example

1．Reset TRW-900C Secondary program： Reset-TRW-900C

2．Collocation TRW-900C Secondary program： Config-TRW-900C

3．Selection working mode：

Transmit mode： Secondary program： TRW-900C-TxMode

Receive mode： Secondary program： TRW-900C-RxMode

Available mode： Secondary program： TRW-900C-TdleMode

4．Data transfer：

Sending data Secondary program： TRW-900C-Send-Data

Receive data Secondary program： TRW-900C-Receive-Data

5．Other mode：

Set power Secondary program： TRW-900C-Set-Power

Low-power mode Secondary program：TRW-900C-LowPowerMode

```
/* ********************************************************************
/* Company name ：WENSHING ELECTRONIC CO., LTD.
/* Module name：TRW-900C
/* Create person ：LHX Date ：2006-11-03
/* Adjust person ：LHX Date ：2006-11-03
/* Function description ：How to use TRW-900C
/* Version ：Ver1.00
/********************************************************************
#include <C8051F320.H>
/********************************************************************
/********************************************************************
/* ********************************************************************
const unsigned char code TRW-900C_Table[68];
/* ********************************************************************
sbit CLK = P1^1;
sbit MISO = P1^2;
sbit MOSI = P1^3;
sbit CS = P1^6;
sbit DR = P1^4;
sbit TX_LED = P2^1;
/********************************************************************
/* Function name ：W-TRW-900C- Byte */
/* Function description ：Write a Byte data into TRW-900C */
/* Input ：x */
/* Back ：No */
/********************************************************************
void W-TRW-900C_Byte(char x)
{
{
unsigned char i;
for(i=0;i<8;i++)
{
CLK = 0;
MOSI = 0;
if(x&0x80)
MOSI = 1;
x<<=1;
CLK = 1;
}
CLK = 0;
```

```
}
/*********************************************************************
/* Function name  ：R-TRW-900C- Byte   */
/* Function description  ：Read the data from TRW-900C */
/* Input  ：No */
/* Back  ：x */
/*********************************************************************
char R-TRW-900C-Byte(void)
{
unsigned char i,x;
for(i=0;i<8;i++)
{
CLK = 0;
x <<= 1;
x &= 0xFE;
if(MISO)
x|= 0x01;
CLK = 1;
}
CLK = 0;
return(x);
}
/*********************************************************************
/* Function name  ：Config-TRW-900C_ Byte */
/* Function description  ：Allocate Byte to TRW-900C */
/* Calling function  ：W-TRW-900C- Byte */
/* R-TRW-900C- Byte */
/* Function description  ：When the seventh place of address is 1，please read */
/* Use the value to see the input data is correct or not.
/* Input  ：address */
/* Back  ：c_data */
/* Function description  ：Collocate one Byte to TRW-900C */
*********************************************************************************************************
char Config-TRW-900C-Byte(char address,char c_data)
{
CS = 0;
while(MISO); // Determine whether low-level, or else wait for a low after the withdrawal
W-TRW-900C-Byte(address);
if(address&0x80)
c_data = R-TRW-900C-Byte();
else
W-TRW-900C-Byte(c_data);
CS = 1;
return(c_data);
```

```
}
/*********************************************************************
/* Function name  ：Reset-TRW-900C*/
/* Function description  ：Rest TRW-900C*/
/* Calling function  ：W-TRW-900C- Byte */
/* Input  ：No*/
/* Back  ：No*/
/*********************************************************************
void Reset-TRW-900C(void)
{
unsigned char i;
CS = 1;
for(i=0;i<100;i++);
CS = 0;
for(i=0;i<200;i++);
CS = 1;
for(i=0;i<200;i++);
CS = 0;
while(MISO); // Please determine MISO is low level or not, or you can wait it become to low-level and r wait for a low after the withdrawal
W_TRW-900C_Byte(0x30);
CS = 1;
}
/*********************************************************************
/* Function name  ：TRW-900C-Set-Power*/
/* Function description  ：Set TRW-900C power*/
/* Calling function  ：W-TRW-900C- Byte */
/* Function description  ：Write 8 Byte data one time*/
/* Input  ：x */
/* Output  ：No*/
/*********************************************************************
void TRW-900C_Set_Power(char x)
{
unsigned char i;
CS = 0;
while(MISO);
W_TRW-900C_Byte(0x7E);
for(i=0;i<8;i++)
W_TRW-900C_Byte(x);
CS = 1;
}
/*********************************************************************
/* Function name  ：Config_TRW-900C*/
```

```
/* Function collocation  ：Allocate TRW-900C*/
/* Calling function  ：Config_TRW-900C_ Byte */
/* : W_TRW-900C_ Byte */
/* Input  ：No*/
/* Output  ：No*/
/*********************************************************************
void Config_TRW-900C(void)
{
unsigned char i,x;
do
{
for(i=0;i<68;)
Config_TRW-900C_Byte(TRW-900C_Table[i++],TRW-900C_Table[i++]);
TRW-900C_Set_Power(0xC3); // Set the maximum transmit power
CS = 0;
while(MISO);
W_TRW-900C_Byte(0x33);
CS = 1;
CS = 0;
while(MISO);
W_TRW-900C_Byte(0x37);
CS = 1;
for(i=0;i<200;i++);
i = Config_TRW-900C_Byte(0x80,x);
}while(i!=0x1B);
}
/*************************************************************************************
/* Function name：TRW-900C_Tx Mode*/
/* Function description  ：Allocate TRW-900C working on transmitter mode */
/* Function description  ：This function will enable TRW-900C model or directly from the
receiver cut back to idle mode when it is in transmit mode. */
/* Calling function  ：W_TRW-900C_ Byte */
/* Input  ：No*/
/* Output  ：No*/
/*************************************************************************************
void TRW-900C_TxMode(void)
{
CS = 0;
while(MISO);
W_TRW-900C_Byte(0x36);
CS = 1;
CS = 0;
while(MISO);
```

```
W_TRW-900C_Byte(0x3B);
CS = 1;
CS = 0;
while(MISO);
W_TRW-900C_Byte(0x35);
CS = 1;
}
/*********************************************************************
/* Function name  ：TRW-900C_Rx Mode*/
/* Function description  ：Allocate TRW-900C working on receive mode */
/* Function description  ：This function will enable TRW-900C model or directly from the receiver cut back to idle mode when it is in transmit mode.*/
/* Calling function  ：W_TRW-900C_Byte*/
/* Input  ：No*/
/* Back  ：No*/
*********************************************************************/
void TRW-900C_Rx Mode(void)
{
CS = 0;
while(MISO);
W_TRW-900C_Byte(0x36);
CS = 1;
CS = 0;
while(MISO);
W_TRW-900C_Byte(0x3A);
CS = 1;
CS = 0;
while(MISO);

W_TRW-900C_Byte(0x34);
CS = 1;
}
/*********************************************************************
/* Function name  ：TRW-900C_Idle Mode*/
/* Function description  ：TRW-900C allocation of work in the idle mode */
/* Function description  ：This function can let TRW-900C transmit directly.
*/);Model or receive mode cut back to idle mode; application do not want to receive, but requires quick return to transmit or receive stats.*/
/* Calling function：W_TRW-900C_Byte*/
/* Input  ：No*/
/* Back  ：No*/
*********************************************************************/
void TRW-900C_Idle Mode(void)
```

```
{
CS = 0;
while(MISO);
W_TRW-900C_Byte(0x36);
CS = 1;
}
/*********************************************************************
void Send_FIFO_Pointer(void)
{
unsigned char i;
while(Config_TRW-900C_Byte(0xFA,0x00)>0x10)
for(i=0;i<200;i++);
}
/*****************************************************************************************************
/* Function name：TRW-900C_Send_Data */
/* Function description: Let TRW-900C send data/ */
/* Function description：Fixed send data 88H.During sending process, please avoid one-time information for a number of Byte 0x00,0XFF. Otherwise, it will cause receive wrong or can’t receive. */
/* Calling function：W_TRW-900C_Byte */
/* Input ：x is Byte number of a packet sending. Variable packet as an example, but a packet is less then 110 Byte.*/
/* Back：No*/
/*****************************************************************************************************
void TRW-900C_Send_Data(unsigned char x)
{
unsigned char i;
TRW-900C_TxMode();
CS = 0;
while(MISO);
W_TRW-900C_Byte(0x7F);
W_TRW-900C_Byte(x);
if(x<65)
{
for(i=0;i<x;i++)
W_TRW-900C_Byte(0x88); // Sending 88H data
CS = 1;
}
else
{
for(i=0;i<60;i++)
W_TRW-900C_Byte(0x88); // Sending 88H data
CS = 1;
```

```
while(!DR);
Send_FIFO_Pointer();
CS = 0;
while(MISO);
W_TRW-900C_Byte(0x7F);
for(i=0;i<(x-60);i++)
W_TRW-900C_Byte(0x88); // Sending 88H data
CS = 1;
}
while(!DR); // Wait transmit finish. After transmit finish, it will automatic into IDLE mode. If finish transmitted and want to be receive mode, please write TRW-900C_Rx Mode() function.
}
/*********************************************************************
/* Function name：Receive_FIFO_Pointer */
/* Function description：Read receiving data index from TRW-900C. When it receives the fifty-one data, please quit. */
/* Calling function：W_TRW-900C_Byte */
/* Input：No */
/* Back：0 , it expresses there are less then 50 Bytes data and receive will be finish. */
/* Back：1 ,it expresses that it had received 50 Bytes data. */
/*********************************************************************
unsigned char Receive_FIFO_Pointer(void)
{
unsigned char i = 0;
while(Config_TRW-900C_Byte(0xFB,0x00)<51)
{
TX_LED = ~TX_LED;
for(i=0;i<200;i++)
if(!DR)
i = 210;
if(i==211)
break; //Please back while sentence.
}
if(i==211)
return(0);
else
return(1);
}
/*********************************************************************
/* Function name：TRW-900C_Receive_Data */
/* Function description：Let TRW-900C receive data.*/
/* Function description：Variable packet as an example.*/
/* Calling function：W_TRW-900C_Byte */
/* Input：No */
```

```
/* Back：0,it expresses receive no data.*/
/* Back：1 ,it expresses received data already.*/
/* Back：2 ,it expresses received correct data.*/
/*********************************************************************
char TRW-900C_Receive_Data(void)
{
unsigned char i = 0, x = 0,y = 0,z,CRC_Value;
unsigned char xdata RF_Buffer[100],RF_Pointer = 0;
while(DR)
{
if(Receive_FIFO_Pointer())
{
x++;
CS = 0;
while(MISO);
W_TRW-900C_Byte(0xFF);
i = 0;
if(x==0)
{
RF_Pointer = 0;
y = R_TRW-900C_Byte();
y ++ ; // When it is receiving data, it will have extra two more then actual transmitted. First is
RSSI, another is BIT7 and value is CRC. If CRC is correct, IBT7 will be equal to 1, otherwise, it
is equal to 0.
RF_Buffer[RF_Pointer++] = y-1;
}
for(;i<50;i++)
RF_Buffer[RF_Pointer++] = R_TRW-900C_Byte();
CS = 1;
z = y-50;
}
else
{
CS = 0;
while(MISO);
W_TRW-900C_Byte(0xFF);
if(x==0)
{
RF_Pointer = 0;
y = R_TRW-900C_Byte();
y ++;
RF_Buffer[RF_Pointer++] = y-1;
for(i=0;i<y;i++)
RF_Buffer[RF_Pointer++] = R_TRW-900C_Byte);I
```

```
}
else
{
for(i=0;i<z;i++)
RF_Buffer[RF_Pointer++] = R_TRW-900C_Byte();
}
CRC_Value = R_TRW-900C_Byte();
RF_Buffer[RF_Pointer++] = CRC_Value;
CS = 1;
z = 0x7F;
}
}
if(z == 0x7F)
{
if(CRC_Value&0x80)
return(2);
else
return(1);
}
else
return(0);
}
/*********************************************************************
/* Function name：TRW-900C_Low Power Mode */
/* Function description：Let TRW-900C is under lower consumption status. */
/* Calling function：W_TRW-900C_Byte */
/* Input：No */
/* Back：No*/
/*********************************************************************
void TRW-900C_Low Power Mode(void)
{
CS = 0;
while(MISO);
W_TRW-900C_Byte(0x36);
CS = 1;
CS = 0;
while(MISO);
W_TRW-900C_Byte(0x39);
CS = 1;
CS = 0;
while(MISO);
W_TRW-900C_Byte(0x32);
CS = 1;
}
```

```
/*********************************************************************
void Init_MCU_Status (void)
{
P1MDOUT = 0x4B;
P2MDOUT = 0x01;
P1 = 0xF4;
XBR1 = 0x40;
OSCICN |= 0x03; // Frequency is 24MHz
}

/*********************************************************************
/* Function name：Main function*/
/* MCU Body：C8051F32x */
/* Input：No */
/* Back：No */
/*********************************************************************
void main(void)
{
unsigned int x;
unsigned char i;
PCA0MD = 0x00;
for(x=0;x<30000;x++);
Init_MCU_Status();
Reset_TRW-900C();
Config_TRW-900C();
for(x=0;x<1000;x++);
while(1)
{
TRW-900C_Send_Data(90);
TRW-900C_RxMode();
for(x=0;x<60000;x++)
TRW-900C_Receive_Data();
}
}
const unsigned char code TRW-900C_Table[68] =
{
0x0D,0x21, // 1.2K 2-FSK 868M
0x0E,0x62,
0x0F,0x76,
0x0B,0x08,
0x0C,0x00,
0x10,0xF5,
0x11,0x83,
0x12,0x03,
```

```
0x13,0x43,
0x14,0x3B,
0x0A,0x00,
0x15,0x15,
0x22,0x10,
0x21,0x56,
0x18,0x08,
0x19,0x16,
0x1A,0x6C,
0x1B,0x03,
0x1C,0x40,
0x1D,0x91,
0x23,0xA9,
0x24,0x2A,
0x25,0x00,
0x26,0x11,
0x29,0x59,
0x2C,0x81,
0x2D,0x35,
0x2E,0x0B,
0x08,0x05,
0x07,0x04,
0x02,0x06,
0x00,0x1B,
0x09,0x00,
0x06,0xFF
/* 0x0D,0x21, ; 250K MSK 868
0x0E,0x62,
0x0F,0x76,
0x0B,0x0B,
0x0C,0x00,
0x10,0x2D,
0x11,0x3B,
0x12,0x73,
0x13,0x43,
0x14,0x3B,
0x0A,0x00,
0x15,0x00,
0x22,0x10,
0x21,0xB6,
0x18,0x08,
0x19,0x1D,
0x1A,0x1C,
0x1B,0xC7,
```

```
0x1C,0x00,
0x1D,0xB2,
0x23,0xEA,
0x24,0x2A,
0x25,0x00,
0x26,0x11,
0x29,0x59,
0x2C,0x88,
0x2D,0x31,
0x2E,0x0B,
0x08,0x05,
0x07,0x04,
0x02,0x06,
0x00,0x06,
0x09,0x00,
0x06,0xFF
*/};

/* ********************************************************************************** */
// This program can be used for TWS-900C and TRW-900C and TRW-400 with the use of the reference program.
/* ********************************************************************************** */
void main (void)
{
Delayms(200)
Init_MCU ();
Config_TWS-900C ();
while(1)
{
TRW-400_Send_Data ();
}
}
/* ********************************************************** */
void Write_Word_TRW-400(char x,char y,char z)
{
unsigned char i;
do
{
for(i=0;i<8;i++)
{
CLK = 0;
WR = 0;
```

```
if(y&0x80)
WR = 1;
CLK = 1;
y<<=1;
}
y=z;
}while(x-->0)
}
/* ************************************************************** */
void RESET_TRW-400 (void)
{
unsigned char i;
CE = 1;
for(i=0;i<10;i++);
CE = 0;
for(i=0;i<10;i++);
CE = 1;
for(i=0;i<100;i++);
CE = 0;
while(RD);
Write_Word_TRW-400(1,0x30,0x00);
while(!RD);
CE = 1;
}
/* ************************************************************** */
void Config_TWS-900C(void)
{
unsigned char i;
RESET_TRW-400 ();
for(i=0;i<50;)
{
CE = 0;
while(RD);
Write_Word_TRW-400(2,Config_Table[i++],Config_Table[i++]);
CE = 1;
}
Write_Word_TRW-400 (9,0x7E,0xC3);
// 0xC3 +10dBm Input
// 0xC6 + 9dBm Input
// 0xC9 + 8dBm Input
// 0xCC + 7dBm Input
// 0xCE + 6dBm Input
// 0x86 + 5dBm Input
// 0x89 + 4dBm Input
```

```
// 0x8C + 3dBm Input
// 0x8D + 2dBm Input
// 0x3F + 0dBm Input
CE = 0;
while(RD);
Write_Word_TRW-400(1,0x37);
CE = 1;
Delayms(1);
CE = 0;
while(RD);
Write_Word_TRW-400(1,0x33);
CE = 1;
Delayms(5);
}
/* ************************************************************ */
void TRW-400_Send_Data(void)
{
CE = 0;
while(RD);
Write_Word_TRW-400(1,0x3B,0x00);
CE = 1;
Delayms(1);
CE = 0;
while(RD);
Write_Word_TRW-400(1,0x35,0x00);
CE = 1;
CE = 0;
while(RD);
Write_Word_TRW-400(14,0x7F,Send_Table[i]);
CE = 1;
while(!FLAG);
while(FLAG);
Delayms(1);
}
/* ************************************************************ */
unsigned char code Send_Table[] =
{
0x12,0x34,0x56,0x78,
0x01,0x02,0x03,0x04,
0x05,0x06,0x07,0x5D,0x5D
}
/* ************************************************************ */
// Allocation description：
// Frequency：434MHz
```

```
// Sending rate：4.8K
// Packet BYTE：Address number + data number+ CRC number
//  ：4+7+2 = 13
/* ************************************************************ */
unsigned char code Config_Table[] =
{
0x02,0x06, //
0x04,0x55, //
0x05,0x55, //
0x06,0x0E, // 14
0x08,0x00, //
0x0A,0x00, //
0x0D,0x10, //
0x0E,0xB1, //
0x0F,0x3B, //
0x10,0x87, //
0x11,0x83, //
0x12,0x03, //
0x13,0x02, //
0x14,0xF8, //
0x15,0x04, //
0x18,0x08, //
0x22,0x10, //
0x23,0xA9, //
0x24,0x2A, //
0x25,0x00, //
0x26,0x11, //
0x29,0x59, //
0x2C,0x81, //
0x2D,0x35, //
0x2E,0x0B //
}
```