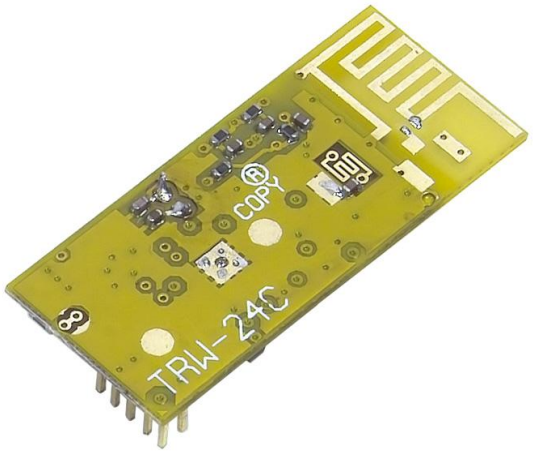
TRW-24C RF MODULE

Wireless Low Cost 2.4GHz Hi Power +19dBm RF Transceiver Module



Version History

Version	Date	Changes
V1.01	Jun 14, 2007	1st. Edition
V1.02	Aug 20, 2007	2nd. Edition
V1.03	Mar 15, 2008	3rd. Edition
V1.04	Jan 4, 2009	4th. Edition
V1.05	Jan 7, 2010	5th. Edition

Specification

● Remote Control Systems	● Wireless Data Transceiver
● 2.4GHz~2.483GHz 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-24C 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 64bytes transmit/receive FIFOs of TRW-24C can be controlled via an SPI interface. In a typical system, the TRW-24C will be used together with a microcontroller and a few additional passive components.

Key Feature

- Small size
- Separate 64bytes RX and TX data FIFOs
- Efficient SPI interface: All registers can be programmed with one "burst" transfer
- Programmable output power up to +19dBm
- High sensitivity (-106dBm at 10kbps, 1% packet error rate)

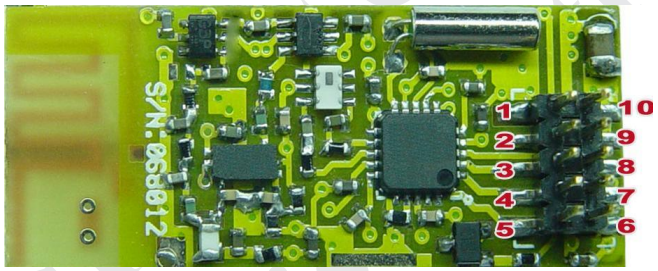
Absolute Maximum Rating

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

General Characteristics and Electrical Specification

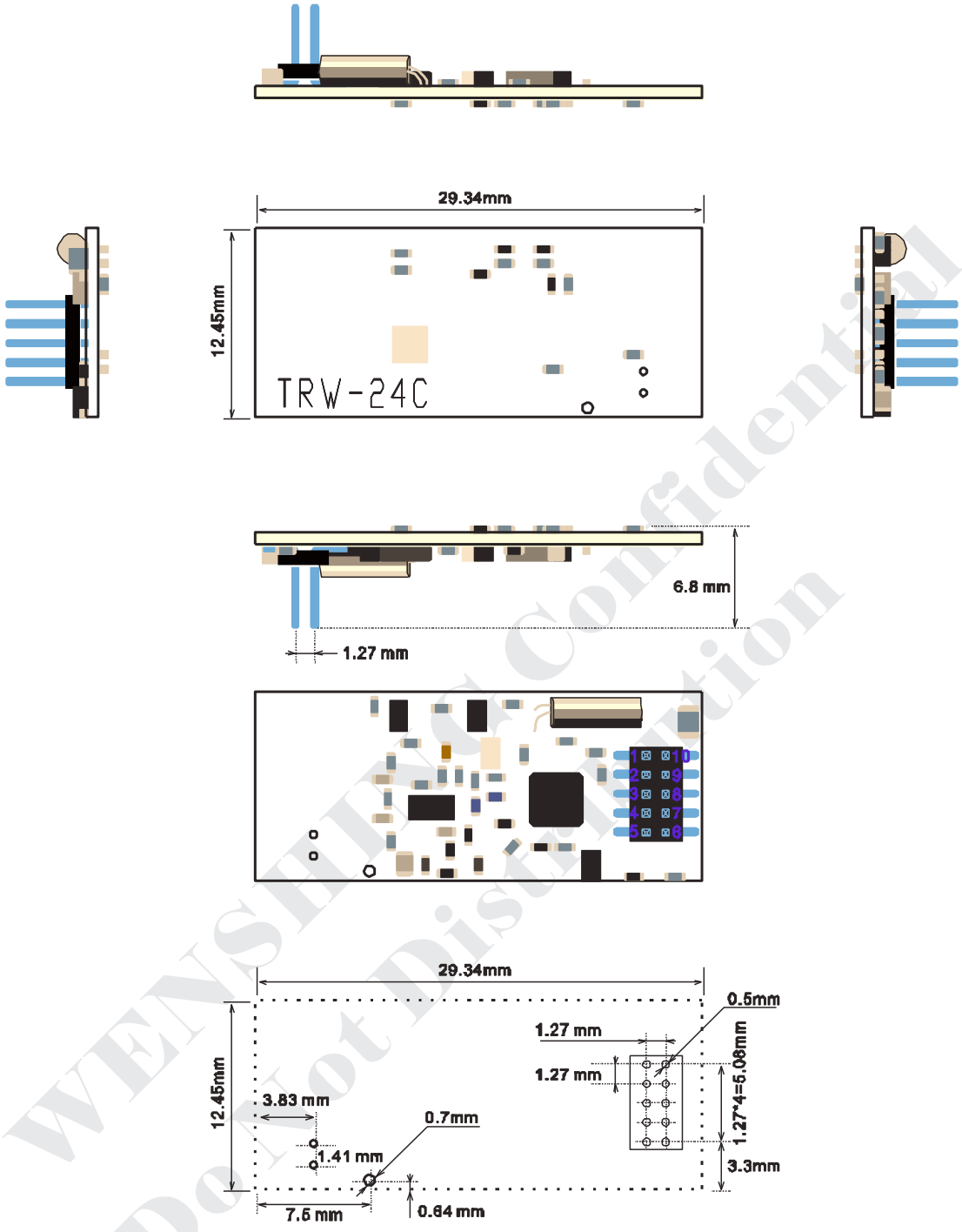
Parameter	Min	Type	Max	Unit	Condition/Note
Frequency Range	2400		2483.5	MHz	
Data Rate	1.2K		250	bps	
Power Down Modes		400		nA	Voltage regulator to digital part off, register values retained (sleep state)
Current Consumption, RX		19		mA	
Current Consumption, TX		90		mA	Transmit mode, +19dBm output power
Receiver Sensitivity	-106			dBm	
Transmit Output Power			+19	dBm	Transmit mode

Pin Assignment



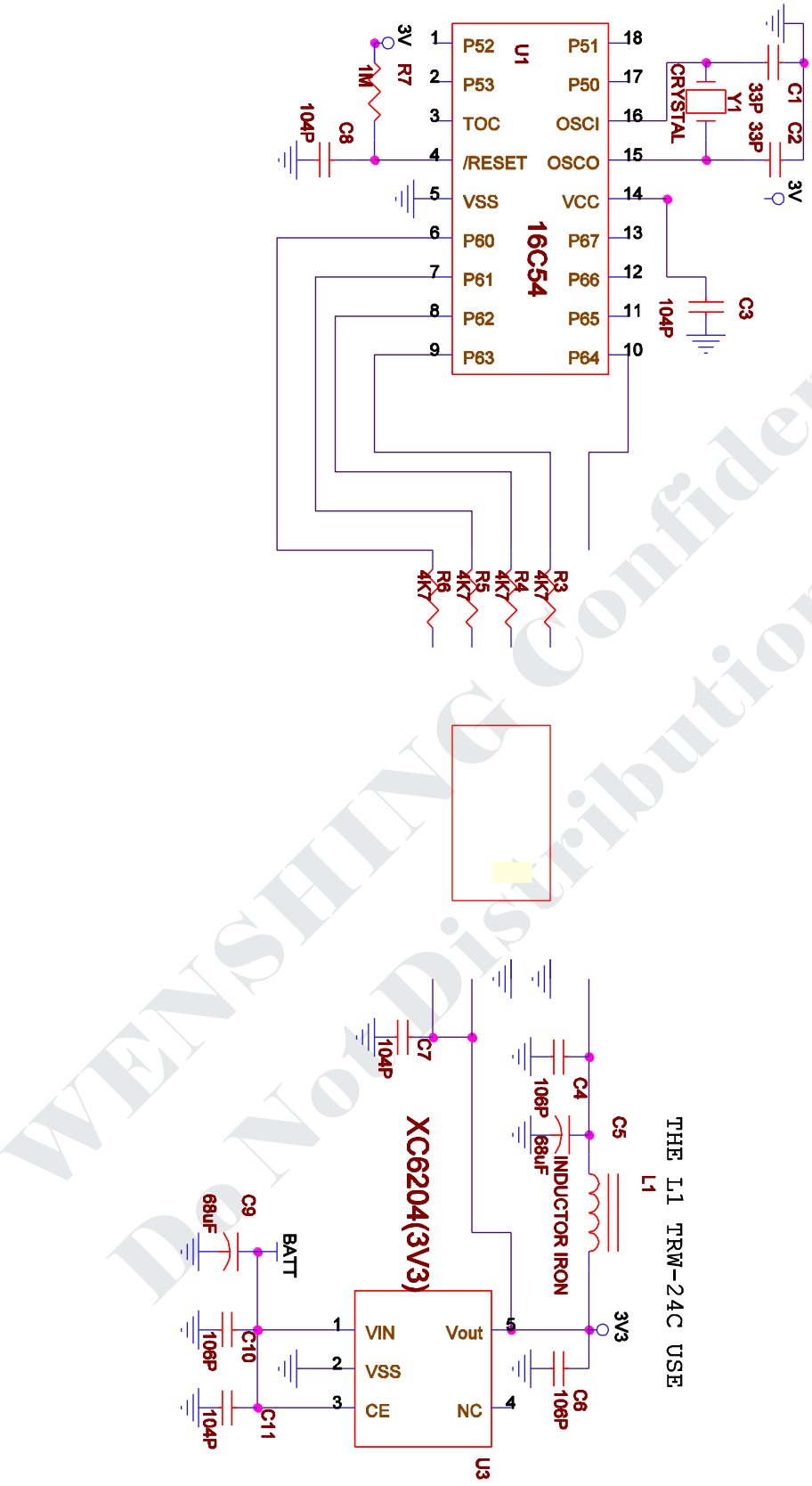
Pin	Function	Type	Description
1	DR	Output	Ordinary condition is for “0”, as for c “1”, is to mean RF data delivers/ Receives
2	CS	Input	Effective Low-powered , is selective module
3	MISO	Output	Read from module group contains data and configure info
4	SCLK	Input	Supply clock for RF reading / writing
5	MOSI	Input	To write in module group contains data and configure info
6, 9, 10	VDD		Power supply is anode
7, 8	GND		Power supply is cathode

Size



Reference hole position for PCB mounting(Bottom view)

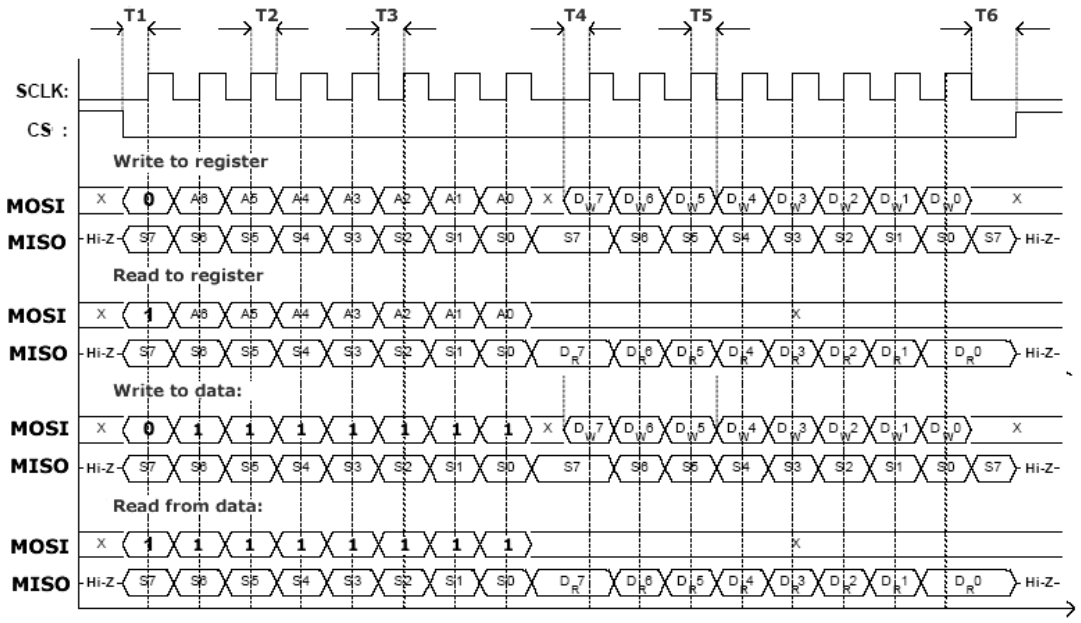
Demo Circuit



Time sequence explanation

TRW- 24C adopts SPI delivered style, all must add address each time for transmitting. Address bit 8 is direction, When is 0, then is read in data. When is 1, then is read out data, The goal is in order to confirm the accuracy which reads in when disposition reads out.

1. Configures time sequence explanation: Form: address+ data.
2. Data delivers/ Receives time sequence explanation: Meantime is same with the disposition, only the address data and write/ read data are different.
   Sends form: 0x7F+ N bytes (N< 65) .
   Receives form: 0xFF+ N bytes (N< 61) .
3. The address of data which transmits to read in is for 0x7F, at most for once can read in 64 bytes data to RF. Address for read out receive data is 0xFF, at most once can read out from RF is 64 bytes.



Fsclk < 6.5MHz

T1	>	200us	T2	>	50ns
T3	>	50ns	T4	>	80ns
T5	>	20ns	T6	>	20ns

Temporary saving utensil explanation

This watch as base frequently for 2410MHz, 250K channel alternation, chooses 0 channels work 2.4/10K/ 250K some reference under working speed.

Address	2.4K	10K	250K	Description
component	FSK	FSK	MSK	
0x0D	0x5C	0x5C	0x5C	Set up TRW- 24C base frequently. The base frequently is for 2410M.
0x0E	0xB1	0xB1	0xB1	
0x0F	0x3B	0x3B	0x3B	
0x13	0x43	0x43	0x43	13H [6:4] is precede code
0x14	0x3B	0x3B	0x3B	13H [1:0] 14H for channel alternation base frequently
0x0A	0x00	0x00	0x00	Sets up for appointed channels
0x0B	0x08	0x06	0x09	0BH [3:0] IF frequency's enactment
0x0C	0x00	0x00	0x00	Fixation
0x10	0x86	0x78	0x2D	10H [7:4] Receives bandwidth enactment
0x11	0x83	0x93	0x3B	Set up modular working speed
0x12	0x03	0x03	0x73	12H[7:4] Choice modulation way
0x15	0x44	0x44	0x01	Modulation percentage hypothesis
0x22	0x10	0x10	0x10	The corresponding speed fixed is as follows
0x21	0x56	0x56	0xB6	
0x18	0x08	0x08	0x08	
0x19	0x16	0x16	0x1D	
0x1A	0x6C	0x6C	0x1C	
0x1B	0x03	0x43	0xC7	
0x1C	0x40	0x40	0x00	
0x1D	0x91	0x91	0xB2	
0x23	0xA9	0xA9	0xEA	
0x24	0x0A	0x0A	0x0A	
0x25	0x00	0x00	0x00	
0x26	0x11	0x11	0x11	
0x29	0x59	0x59	0x59	
0x2C	0x81	0x81	0x88	
0x2D	0x35	0x35	0x31	
0x2E	0x0B	0x0B	0x0B	
0x08	0x05	0x05	0x05	Envelop wrap control
0x02	0x06	0x06	0x06	Fixation

0x00	0x1B	0x1B	0x1B	
0x06	0xFF	0xFF	0xFF	When setup the packet length, it is in the invariable situation ,the value is 0xFF

Command Value Illustration

Command Value	Function Description
0x30	Reset RF Chip
0x32	Close RF OSC
0x33	Auto-lock frequency. The timing is shorter than 1ms
0x34	Remove Receiving Buffer
0x35	Remove Transmitting Buffer
0x36	Chip into Static Mode
0x39	Chip into low power consumption Mode
0x3A	Chip into Receiving Mode
0x3B	Chip into transmitting Mode
0x7E	Send Power Setting
0x7F/0xFF	Write-Send/ Read-Receive Data of Buffer Area

1. TRW- 24C base frequently enactment: 0DH, 0EH, 0FH

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

One: Operating frequency for 2401MHz,

then FREQ [23:0] = 2401000000*2^16/26000000 = 6051997= 0x5C589D

Then 0DH = 0x5C 0EH= 0x58 0FH = 0x9D

Operating frequency for 2411.5MHz,

then FREQ [23:0] = 2411500000*2^16/26000000 = 6078464= 0x5CC000

Then 0DH = 0x5C 0EH = 0xC0 0FH = 0x00

2. Channel with channel's alternation frequency's enactment: 13H [1:0] 14H

fCHANNEL = 26000000* (256+ 11H [7:0])*2^10 [1:0] /2^18

If will set up channel's alternation frequency for 250K, then 14 = 0x3B, 13 [1:0] = 3;

3. Channel alternation's choice: 0x0A

fcarrier = fBASE + fCHANNEL* 0AH [7:0]

If base frequently for 2401MHz, fCHANNEL =250K, 0x0A= 00 word, then current operating frequency for: 2401MHz

If base frequently for 2401, fCHANNE = 250K, 0x0A= 04 word, then current operating frequency for: 2402MHz

4. **Precedes code enactment: 13H [6:4]**

13H [6:4] = 0 Then is 2 bytes pioneer yard
13H [6:4] = 1 Then is 3 bytes pioneer yard
13H [6:4] = 2 Then is 4 bytes pioneer yard
13H [6:4] = 3 Then is 6 bytes pioneer yard
13H [6:4] = 4 Then is 8 bytes pioneer yard
13H [6:4] = 5 Then is 12 bytes pioneer yard
13H [6:4] = 6 Then is 16 bytes pioneer yard
13H [6:4] = 7 Then is 24 bytes pioneer yard

5. **Receives bandwidth enactment: 10H [7:4],parallels each worth give parallel frequency under it**

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's enactment: f_{IF} = 25390.625*0BH [3:0]**

For example: 2.4K working speed, 0BH [3:0] =8, then f_{IF} = 203.125KHz

7. **Working speed's enactment: 10H [3:0] ,11H [7:0] ,beg the most approach worth:**

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

Example: Is like working speed for 2.4K, then 11H = 0x83, 10H [3:0] = 0xx6

Example: Is like working speed for 1.2K, then 11H = 0x83, 10H [3:0] = 0xx5

8. **Modulates selecting mode: 12H [7:4]**

12H [7:4] = 0, illustrates choose 2- FSK mode.

12H [7:4] = 1, illustrates choose MSK mode.

9. **Modulate set up 15H**

a. At MSK time, it worths forever all is one.

b. At 2- FSK time, it modulates calculate formula following, beg lately worth

fdev = (8+ 15H [2:0] *2^15H [6:4] *198

10. **Envelop wrap control 08H**

08H = 0x04, illustrates Envelop length is fixed, it's length worth saving in 06H.

Example: When in the 06H value is 0x70, explained a packet in the launch and receive byte is 112, needs to partition the transmission in the launch and the receiving process to be able to receive with the receive data.

Launch form: 0x7F+N bytes (N = 06H)

08H = 0x05, Explained the packet length is invariable, its length value for the launch is the first byte data, the 06H value is 0xFF. Launch form : 0x7F+ Envelops wrap length +N bytes data (N =Envelops wrap length)

Note:

a. As at launch condition time, if envelops wrap length ≤64, then can data read in TRW-24C by once, if envelops wrap length>64, then needs segment deliver data to TRW-24C, because TRW- 24C only have a 64 bytes buffer. The concrete partition method is as follows:

10.1.1 The first is to read in 64 bytes to TRW-24C.

10.1.2 Waits DR pin in high.

10.1.3 Reads 0xFA the value, if reads the value is 0x10, explained in the buffer also has 16 bytes not to deliver, if reads the value is 0x20, explained in the buffer also has 32 bytes not to deliver.

10.1.4 Bases read 0xFA the value, the user may decide when transmits the material to give TRW-24C once more, but cannot read it to be equal to 0, otherwise can have the mistake

10.1.5 If transmits the data not to deliver once more, the data duplicates 10.1.13 and 10.1.14.

b. Works as when the receive condition, waited for its DR line turns high, if the packet length is smaller than 61, after then waited for the DR line changes lowly reads two bytes the data to come out, reads two data, preceding is the RSSI value, latter is the CRC value, when latter CRC value bit7 is 1, explains the packet correctly, otherwise wrong, the RSSI value is has the mark the value, if reads bigger value, the explanation receive signal is better. Please note: 0xFF is -1 < 0. If the packet length is bigger than 60, then concrete reading method as follows:

10.2.1 The first waits for DR line turns to 1.

10.2.2 Reads in 0xFB the value if is equal to 51, then can read out 50 bytes data from TRW-24C (the note 51,/50 to be possible to define by user, only was smaller than the 0xFB in value.

10.2.3 Duplicates 10.2.2, if in when reads 0xFB, its DR line turns lowly, explained this data not enough to 50 bytes, this time does not have to withdraw situation of reads 0xFB, reads out the surplus data (the note: The read-out data is more than 2 of reality data.)

10.2.4 After the DR line turns lowly, how to know how many data must to read: In under the fixed packet condition, decided by the computation of 0x06, if supposes is N, then reads out the material is N+2 bytes.

In under the invariable packet condition, decided by the first byte value. If receives is N, then reads out the material is N+3 bytes.

11. Power enactment:

As set up worth for 0xFF/0xFE/0xBB/0xA9/0x7F/0x6E/0x97/0xC6/0x8D/0x55/0x93/0x46 parallel dB worth for+19/ + 18/+ 16/+ 14/+ 12/+10/+ 8/+ 6/+ 4/+ 2/+ 0/- 2dBm

Sets up power address for 0x7E+ 8 bytes data,8 bytes set up become same, chooses above power worth.

Reads set up power worth for 0xFE+ 8 bytes data, 8 bytes whether with set up become same.

Procedure explanation

1. Heavy fix TRW-24C Subprogram: Reset- TRW- 24C.
2. Configure TRW-24C Subprogram: Config- TRW- 24C.
3. Chooses work mode:
   - Shoots mode: Subprogram: TRW-24C-TxMode.
   - Receives mode: Subprogram: TRW-24C-RxMode.
   - Leisures mode: Subprogram: TRW-24C-TdleMode.
4. Data transference:
   - Sends data Subprogram: TRW-24C-Send- Data.
   - Receive data Subprogram: TRW-24C-Receive- Data.
5. Other mode:
   - Sets up power Subprogram: TRW-24C-Set-Power.
   - Low achievements consume mode Subprogram: TRW-24C- LowPowerMode.

Sample Program

```
// Function：How to use TRW-24C Module
// Resource：WENSHING ELECTRONICS CO., LTD.
// Website：http://www.wenshing.com.tw
// Update date：2007/01/16
// Version：Ver1.00

#include <C8051F320.H>
/* ************************************************************************ */
/* ************************************************************************ */
/* ************************************************************************ */
/* ************************************************************************ */
/* ************************************************************************ */
const unsigned char code TRW-24C-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-24C-Byte */
/* Functional Description：Read in Byte to TRW-24C */
/* Input：x */
/* Return：No */
/* ********************************************************************** */
void W-TRW-24C-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-24C-Byte */
/* Functional Description：Reads out Byte from TRW-24C */
/* Input：No */
/* Return：x */
/* ********************************************************************** */
char R-TRW-24C-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-24C-Byte */
/* Functional Description：Disposes a byte form TRW-24C module*/.
/* Transfer Function：W-TRW-24C-Byte */
/* R-TRW-24C-Byte */
/* Function instruction ：When the seventh of address is 1, reads its value */
/* Function Description ：Disposes one of bytes from TRW-24C *
/* Input：address */
/* Return：c_data */
/* ********************************************************************** */
char Config-TRW-24C-Byte(char address,char c_data)
{
CS = 0;
while(MISO); // To judge MISO if the low level, otherwise wait for turning to the
low level to secede. 判斷MISO是否爲低電平，否則等待變成低電平後退出
W-TRW-24C-Byte(address);
if(address&0x80)
c_data = R-TRW-24C-Byte();
else
W-TRW-24C-Byte(c_data);
CS = 1;
return(c_data);
}
/* ********************************************************************** */
/* Function Name：Reset-TRW-24C */
/* Functional Description ：ReplacementTRW-24C */
/* Transfer Function：W-TRW-24C-Byte */
/* Input：No */
/* Return：NO */
/* ********************************************************************** */
void Reset-TRW-24C(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);  // To judge MISO if the low level, otherwise wait for turning to
low level to secede.
W-TRW-24C-Byte(0x30);
CS = 1;
}
/* ********************************************************************** */
/* Function Name  : TRW-24C-Set-Power  */
/* Function Description  : Set up the TRW-24C module power */
/* Transfer Function  : W-TRW-24C-Byte  */
/*Function Instruction  : Writes 8 Bytes data at one time. */
/* Input  : x */
/* Return  : No */
/* ********************************************************************** */
void TRW-24C-Set-Power(char x)
{
unsigned char i;
CS = 0;
while(MISO);
W_TRW_24C_Byte(0x7E);
for(i=0;i<8;i++)
      W_TRW_24C_Byte(x);
CS = 1;
}
/* ********************************************************************** */
/* Function Name：Config-TRW-24C  */
/* Function Description  : Set up TRW-24C */
/* Transfer Function：Config-TRW-24C-Byte  */
/* : W-TRW-24C-Byte
/* Input：No */
/* Return：NO */
```

```
/* ********************************************************************** */
void Config-TRW-24C(void)
{
unsigned char i,x;
do
{
for(i=0;i<68;)
      Config-TRW-24C-Byte(TRW-24C-Table[i++],TRW-24C-Table[i++]);
TRW-24C-Set-Power(0xFF);  // Set up the highest- powered launch

CS = 0;
while(MISO);
W-TRW-24C-Byte(0x33);
CS = 1;

CS = 0;
while(MISO);
W-TRW-24C-Byte(0x37);
CS = 1;

for(i=0;i<200;i++);
      i = Config-TRW-24C-Byte(0x80,x);
}while(i!=0x1B);
}
/* ********************************************************************** */
/* Function Name：TRW-24C-TxMode */
/* Function Description：Set up TRW-24C Work in launch pattern */
/* Function Instruction：This function may cause TRW-24C directly to cut the launch pattern
from the receive pattern or the idle mode
/* Transfer Function：W-TRW-24C-Byte */
/*Input：No */
/* Return：No */
/* ********************************************************************** */
void TRW-24C-TxMode(void)
{
CS = 0;
while(MISO);
W-TRW-24C-Byte(0x36);
CS = 1;
```

```
CS = 0;
while(MISO);
W-TRW-24C-Byte(0x3B);
CS = 1;

CS = 0;
while(MISO);
W-TRW-24C-Byte(0x35);
CS = 1;
}
/* ********************************************************************** */
/* Function Name：TRW-24C-RxMode */
/* Function Description：Set up TRW-24C Work in receive pattern */
/* Function Instruction：This function may cause TRW-24C module directly to cut the launch
pattern from the…… pattern or the launch pattern
此函數可使TRW-24C模組直接從發射模式或發射模式切回發射模式 */
/* Transfer Function：W-TRW-24C-Byte */
/* Input：No */
/* Return ：No */
/* ********************************************************************** */
void TRW-24C-RxMode(void)
{
CS = 0;
while(MISO);
W-TRW-24C-Byte(0x36);
CS = 1;

CS = 0;
while(MISO);
W-TRW-24C-Byte(0x3A);
CS = 1;

CS = 0;
while(MISO);
W-TRW-24C-Byte(0x34);
CS = 1;
}
/* ********************************************************************** */
```

```
/* Function Name：TRW-24C-IdleMode */
/*Function Description：Set up the TRW-24C module works in the idle mode /
/* Function Instruction ：This function may cause TRW-24C module directly to cut the idle
   mode from the launch pattern or the receive pattern, the applicative
situation belongs to not receive, but requests fast to return to the launch or the receive
   condition
/* Transfer Function：W-TRW-24C-Byte */
/* Input：No */
/* Return：NO */
/* ********************************************************************** */
void TRW-24C-IdleMode(void)
{
CS = 0;
while(MISO);
W-TRW-24C-Byte(0x36);
CS = 1;
}
/* ********************************************************************** */
void Send_FIFO_Pointer(void)
{
unsigned char i;
while(Config-TRW-24C-Byte(0xFA,0x00)>0x10)
for(i=0;i<200;i++);
}
/* ********************************************************************** */
/* Function ：TRW-24C-Send-Data */
/* Function Description：TransmiteTRW-24C data */
/*Function Description：Fixed send data 88H, in the sending process, as far as
   possible avoids disposable many Bytes the material is 0x00,0xFF,
   otherwise receives has the mistake or can’t receive.
/* Transfer Function：W-TRW-24C-Byte */
/* Input ： Refers to how many bytes does one packet need, the example for take the
invariable packet, Very packet is under the 110 byte.
/*Return ：NO */
/* ********************************************************************** */
void TRW-24C-Send-Data(unsigned char x)
{
unsigned char i;
TRW-24C-TxMode();
```

```
CS = 0;
while(MISO);
W-TRW-24C-Byte(0x7F);
W-TRW-24C-Byte(x);
if(x<65)
{
for(i=0;i<x;i++)
W-TRW-24C-Byte(0x88);  // Transmits 88H data
CS = 1;
}
else
{
for(i=0;i<60;i++)
W-TRW-24C-Byte(0x88);  // Transmits 88H data
CS = 1;
while(!DR);
Send_FIFO_Pointer();
CS = 0;
while(MISO);
W-TRW-24C-Byte(0x7F);
for(i=0;i<(x-60);i++)
W-TRW-24C-Byte(0x88);  // Transmits 88H data
CS = 1;
}
while(!DR);  // The waiting launch completes, after the launch completes,
automatically enters the IDLE pattern, after if transmits wants to
enter the receive pattern, writes TRW-24C-RxMode  () the function
}
/* ********************************************************************** */
/* Function Name：Receive_FIFO_Pointer */
/* Function Description  ：Reads the TRW-24C receiving data index, when receives 51 data,
exit it. */
/* Transfer Function：W-TRW-24C-Byte */
/* Input：No  */
/* Return：0 ,To explain receive conclusion before not to 50 Bytes data */
/* Return：1 , To explain has received 50 byte data. */
/* ********************************************************************** */
unsigned char Receive_FIFO_Pointer(void)
{
```

```
unsigned char i = 0;
while(Config-TRW-24C-Byte(0xFB,0x00)<51)
{
TX_LED  = ~TX_LED;
for(i=0;i<200;i++)
if(!DR)
i = 210;
if(i==211)
break; //return to while sentence
}
if(i==211)
return(0);
else
return(1);
}
/* ********************************************************************** */
/* Function Name：TRW-24C-Receive-Data */
/*Function Description ： TRW-24C can receive data */
/* Function Instruction ：the example for the invariable package 以可變封包爲例， */
/*Transfer Function：W-TRW-24C-Byte */
/* Input：No */
/* Return：0 , To explain not receive data */
/* Return：1 ,To explain has received error data */
/* Return：2 ,To explain has received correct data. */
/* ********************************************************************** */
char TRW-24C-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-24C-Byte(0xFF);
i = 0;
if(x==0)
```

```
{
RF_Pointer = 0;
y = R_TRW-24C_Byte();
y ++ ; // When receiving data, must be more than two in actual launch data. The preceding is the RSSI value
i++; // Latter one BIT7 is the CRC value, if CRC is correct, then BIT7 = 1,
        otherwise is 0.
RF_Buffer[RF_Pointer++] = y-1;
}
for(;i<50;i++)
RF_Buffer[RF_Pointer++] = R_TRW_24C_Byte();
CS = 1;
z = y-50;
}
else
{
CS = 0;
while(MISO);
W_TRW_24C_Byte(0xFF);
if(x==0)
{
RF_Pointer = 0;
y = R_TRW_24C_Byte();
y ++;
RF_Buffer[RF_Pointer++] = y-1;
for(i=0;i<y;i++)
RF_Buffer[RF_Pointer++] = R_TRW_24C_Byte();
}
else
{
for(i=0;i<z;i++)
RF_Buffer[RF_Pointer++] = R_TRW_24C_Byte();
}
CRC_Value = R_TRW_24C_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-24C-LowPowerMode */
/* Function Description ：TRW-24C work in the low-powered losing condition */
/* Transfer Function：W-TRW-24C-Byte */
/* Input：No */
/* Return：NO */
/* ********************************************************************** */
void TRW-24C_LowPowerMode(void)
{
CS = 0;
while(MISO);
W_TRW-24C-Byte(0x36);
CS = 1;
Config-TRW-24C-Byte(0x00,0x6F);
CS = 0;
while(MISO);
W_TRW-24C-Byte(0x39);
CS = 1;
}
/* ********************************************************************** */
void Init_MCU_Status (void)
{
P1MDOUT = 0x4B;
P2MDOUT = 0x01;
P1 = 0xF4;
XBR1 = 0x40;
OSCICN |= 0x03; // Operating frequency is 24M
}
/* ********************************************************************** */
```

```
/* Function Name：Main  Function */
/* MCU Body：C8051F32x  */
/* Input：No  */
/* Return：No  */
/* ************************************************************************  */
void  main(void)
{
unsigned  int x;
unsigned  char i;
PCA0MD = 0x00;
for(x=0;x<30000;x++);
Init_MCU_Status();
Reset-TRW-24C();
Config-TRW-24C();
for(x=0;x<1000;x++);
while(1)
{
TRW-24C-Send-Data(90);
TRW-24C-RxMode();
for(x=0;x<60000;x++)
TRW-24C-Receive-Data();
}
}
const unsigned char code TRW_24C_Table[68]  =
{
0x0D,0x5C,  // 2.4K 2-FSK 2410M
0x0E,0x4e,
0x0F,0xc5,
0x0B,0x08,
0x0C,0x00,
0x10,0x86,
0x11,0x83,
0x12,0x03,
0x13,0x43,
0x14,0x3B,
0x0A,0x00,
0x15,0x44,
0x22,0x10,
0x21,0x56,
```

```
0x18,0x08,
0x19,0x16,
0x1A,0x6C,
0x1B,0x03,
0x1C,0x40,
0x1D,0x91,
0x23,0xA9,
0x24,0x0A,
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,0x5C, ; 250K MSK 2410
0x0E,0xB1,
0x0F,0x3B,
0x0B,0x09,
0x0C,0x00,
0x10,0x2D,
0x11,0x3B,
0x12,0x73,
0x13,0x32,
0x14,0xF8,
0x0A,0x00,
0x15,0x01,
0x22,0x10,
0x21,0x56,
0x18,0x08,
0x19,0x1D,
0x1A,0x1C,
0x1B,0xC7,
0x1C,0x00,
```

```
0x1D,0xB2,
0x23,0xEA,
0x24,0x0A,
0x25,0x00,
0x26,0x11,
0x29,0x59,
0x2C,0x88,
0x2D,0x31,
0x2E,0x0B,
0x08,0x05,
0x07,0x04,
0x02,0x06,
0x00,0x1B,
0x09,0x00,
0x06,0xFF
*/};
```