Document Title

433MHz Band Reference Code for FIFO mode (100Kbps, 100KIF)

Revision History

Rev. No.	History	Issue Date	Remark
0.0	Preliminary	July 19, 2013	

Table of contents

AMICCOM RF Chip - A7139
Reference code for FIFO mode

1. 簡介

這文件針對 AMIC-COM RF chip -A7139 FIFO mode 做一簡單的應用範例程式 ，供使用者能夠快速應用這 RF chip。

2. 系統概述

本範例程式主要分二個部份，一個爲 master 端，另一個爲 slave 端。

Master 端：power on、initial 系統及 RF chip 後，進入 TX 狀態，傳送 64 bytes 資料，再進入 RX 接收狀態，等待 100ms。若 slave 端有發射資料，則 Master 端會接收到資料。否則，100ms 之後，Master 端又會回到 TX 傳送階段。

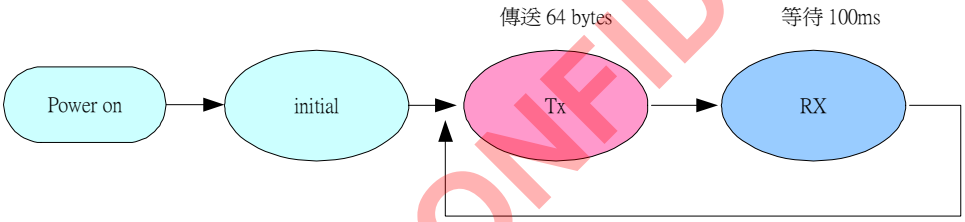

Fig1. Master 端方塊圖

Slave 端：power on、initial 系統及 RF chip 後，進入 RX 狀態等待接收。若無收到 Master 端所發送的資料，則仍再 RX 狀態，等待接收。若有收到 Master 端所發送的資料，則進入 TX 狀態，傳送 64bytes 資料。再次回到 RX 狀態等待下一次接收。

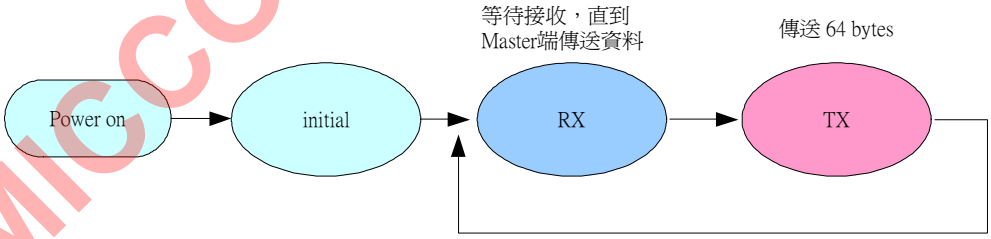

Fig2. Slave 端方塊圖

Master 端在 Power on 後，進入迴圈送出封包及等待 Slave 端所傳送合法的封包。Master 端如未收到封包，在 100ms 後，回到發送程序送出封包。一旦接收到封包，讀出資料、比對，計算 error bit，延遲 100ms 後，回到發送程序送出封包。Slave 端在 Power on 後，進入接收狀態，等待從 master 端所發送合法的封包。Slave 端如未收到封包，則仍繼續等待接收。一旦接收到封包，讀出資料、比對，計算 error bit 後，再發送封包給 Master 端。使用者可依據簡易的計算 error bit 及傳送封包數，得出 BER(bit error rate)，作爲傳輸品質的數據。

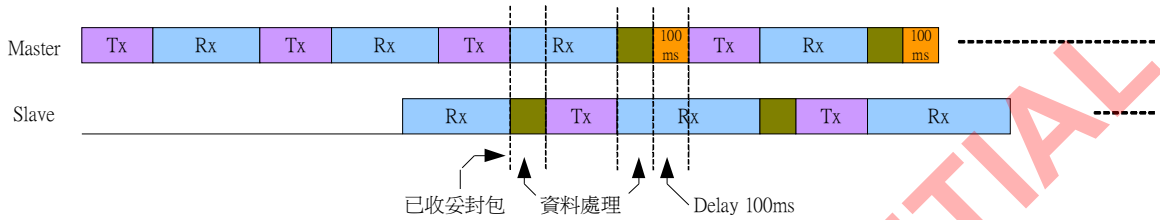

Fig3. TX/RX 時序圖

3. 硬體

3.1 系統方塊圖

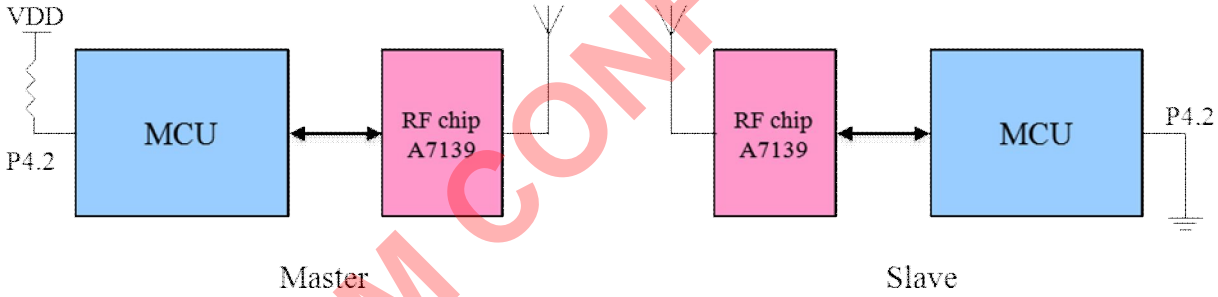

Fig4. 系統方塊圖

MCU I/O pin 設定：

- MCU 使用 I/O pin P4.2 的設定，判別 Master 端或 Slave 端。
- SCS, SCK, SDIO - 這 3 wire SPI 介面控制 A7139 內部 register。
- GIO2 - FIFO 動作完成的控制信號，MCU 可檢測該 pin 是否傳送或接收 packet 完成。

MCU 控制 A7139 RF chip 的 I/O 配置如下圖：

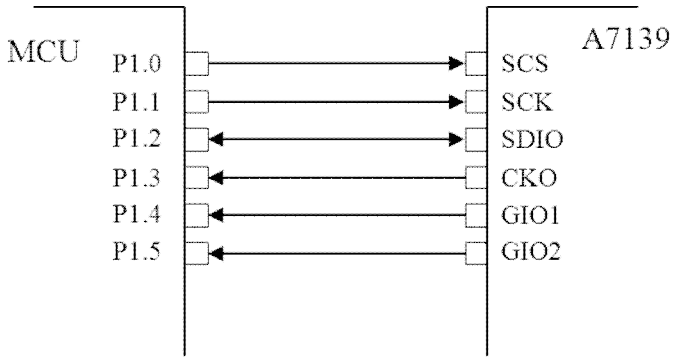

Fig5. I/O 配置圖

4. 韌體程式設計

4.1 應用範例概述

首先初始化 Timer0、Uart0 及 A7139 RF chip，之後判別 P4.2=1 進入 master 端的主程式；
P4.2 =0 進入 slave 端的主程式。

Master 端：
1) TX FIFO 寫入 PN9 code 共 64 bytes。
2) 進入 TX state，傳送封包後，自動結束 TX state，回到 Standby state。
3) 進入 RX state。
4) 啓動 Timer0 計時、清除 Timeout Flag 旗標。
5) 如發生 Timeout=100ms 後，離開 RX state。回到(1)。
6) 如收到封包後，自動結束 RX state，回到 Standby state。
7) 從 RX FIFO 讀出接收的資料，並比較 PN9 code 共 64bytes，計算 error bit 。
8) 延遲 100ms，重新回到(1)。
9) 每 500ms，將所計算的 error bit 傳送至 PC。

Slave 端：
1) 進入 RX state，等待封包收到。
2) 如收到封包後，自動結束 RX state，回到 Standby state。
3) 從 RX FIFO 讀出接收的資料，並比較 PN9 code 共 64bytes，計算 error bit 。
4) TX FIFO 寫入 PN9 code 共 64 bytes。
5) 進入 TX state，傳送封包後，自動結束 TX state，回到 Standby state。
6) 重新回到(1)。
7) 每 500ms，將所計算的 error bit 傳送至 PC。

4.2 範例程式工作基本方塊

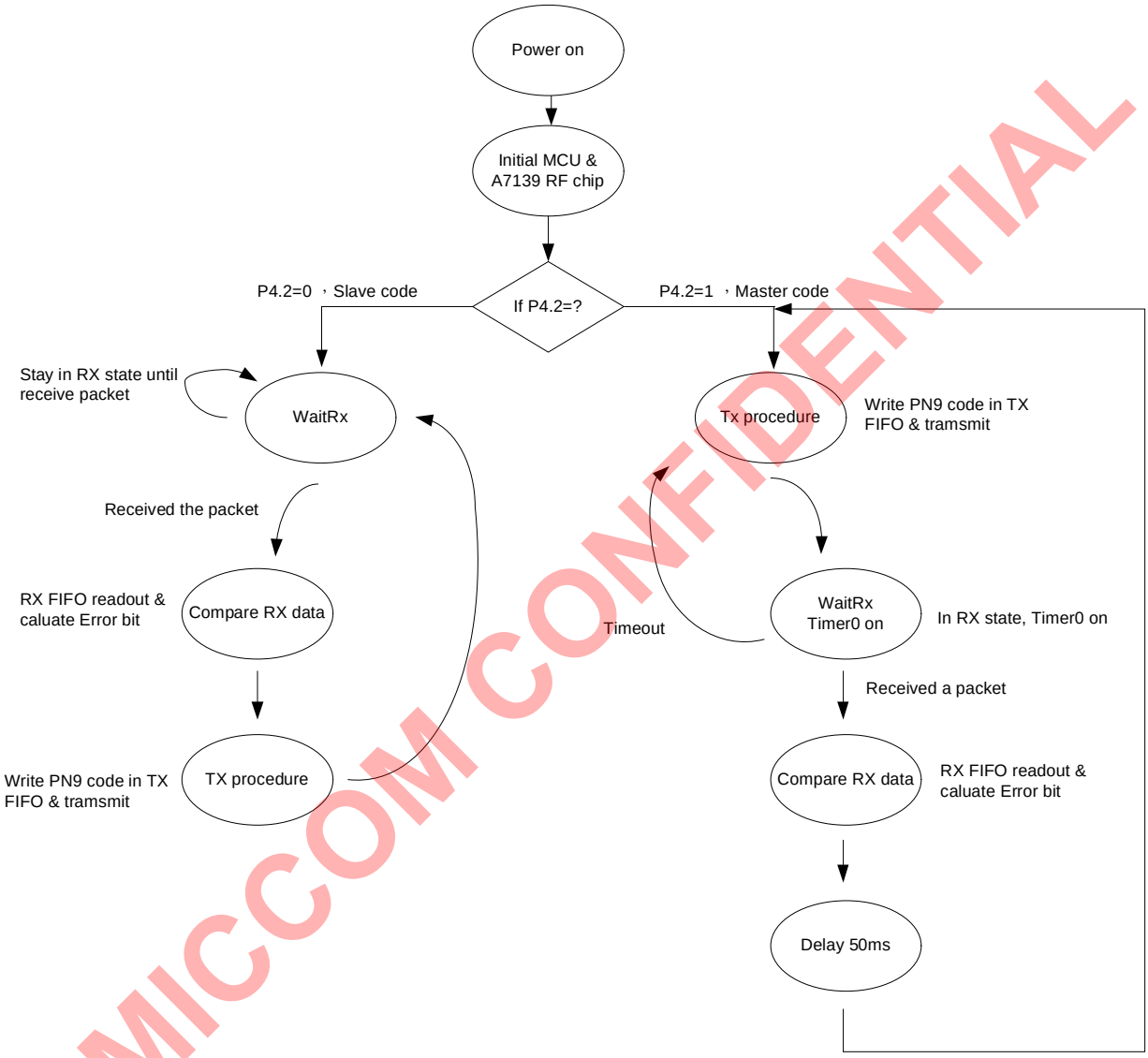

Fig6 範例程式工作基本方塊

5. 程式說明

```
 1 /*********************************************************************
 2 **  Device:       A7139
 3 **  File:         main.c
 4 **  Target:       Winbond W77LE58
 5 **  Tools:        ICE
 6 **  Updated:      2013-07-19
 7 **  Description:
 8 **  This file is a sample code for your reference.
 9 **
10 **  Copyright (C) 2011 AMICCOM Corp.
11 **
12 *********************************************************************/
13 #include "define.h"
14 #include "w77le58.h"
15 #include "A7139reg.h"
16 #include "Uti.h"
```

功能說明：Include 檔宣告

行數	說明
13~16	匯入程式庫設定檔

```
18 /*********************************************************************
19 **  I/O Declaration
20 *********************************************************************/
21 #define SCS             P1_0    //SPI SCS
22 #define SCK             P1_1    //SPI SCK
23 #define SDIO            P1_2    //SPI SDIO
24 #define CKO             P1_3    //CKO
25 #define GIO1            P1_4    //GIO1
26 #define GIO2            P1_5    //GIO2
27
28 /*********************************************************************
29 **  Constant Declaration
30 *********************************************************************/
31 #define TIMEOUT         100     //100ms
32 #define t0hrel          1000    //1ms
```

功能說明：MCU 對 A7139 RF chip I/O 接腳定義

行數	說明
21~26	MCU I/O 配置
31~32	常數變數宣告

```
34 /***********************************************************************
35 **  Global Variable Declaration
36 ***********************************************************************/
37 Uint8   data    timer;
38 Uint8   data    TimeoutFlag;
39 Uint16  idata   RxCnt;
40 Uint32  idata   Err_ByteCnt;
41 Uint32  idata   Err_BitCnt;
42 Uint16  idata   TimerCnt0;
43 Uint8   data    *Uartptr;
44 Uint8   data    UartSendCnt;
45 Uint8   data    CmdBuf[11];
46 Uint8   idata   tmpbuf[64];
47
48 const Uint8 code BitCount_Tab[16]={0,1,1,2,1,2,2,3,1,2,2,3,2,3,3,4};
49 const Uint8 code ID_Tab[8]={0x34,0x75,0xC5,0x8C,0xC7,0x33,0x45,0xE7};   //ID code
50 const Uint8 code PN9_Tab[]=
51 {   0xFF,0x83,0xDF,0x17,0x32,0x09,0x4E,0xD1,
52     0xE7,0xCD,0x8A,0x91,0xC6,0xD5,0xC4,0xC4,
53     0x40,0x21,0x18,0x4E,0x55,0x86,0xF4,0xDC,
54     0x8A,0x15,0xA7,0xEC,0x92,0xDF,0x93,0x53,
55     0x30,0x18,0xCA,0x34,0xBF,0xA2,0xC7,0x59,
56     0x67,0x8F,0xBA,0x0D,0x6D,0xD8,0x2D,0x7D,
57     0x54,0x0A,0x57,0x97,0x70,0x39,0xD2,0x7A,
58     0xEA,0x24,0x33,0x85,0xED,0x9A,0x1D,0xE0
59 };// This table are 64bytes PN9 pseudo random code.
```

功能說明：使用的整體變數宣告，常數變數的宣告

行數	說明
37~46	程式中使用的變數宣告
48	BitCount_Tab 宣告
49	ID code 宣告
50~59	PN9 data 宣告

```
 61 const Uint16 code A7139Config[]=            //433MHz, 100kbps (IFBW = 100KHz, Fdev = 37.5KHz)
 62 {
 63     0x0021,     //SYSTEM CLOCK register,
 64     0x0A21,     //PLL1 register,
 65     0xDA05,     //PLL2 register,     433.301MHz
 66     0x0000,     //PLL3 register,
 67     0x0A20,     //PLL4 register,
 68     0x0024,     //PLL5 register,
 69     0x0000,     //PLL6 register,
 70     0x0011,     //CRYSTAL register,
 71     0x0000,     //PAGEA,
 72     0x0000,     //PAGEB,
 73     0x18D4,     //RX1 register,      IFBW=100KHz
 74     0x7009,     //RX2 register,      by preamble
 75     0x4000,     //ADC register,
 76     0x0800,     //PIN CONTROL register,          Use Strobe CMD
 77     0x4C45,     //CALIBRATION register,
 78     0x20C0      //MODE CONTROL register,        Use FIFO mode
 79 };
 80
 81 const Uint16 code A7139Config_PageA[]=   //433MHz, 100kbps (IFBW = 100KHz, Fdev = 37.5KHz)
 82 {
 83     0xF706,     //TX1 register,      Fdev = 37.5kHz
 84     0x0000,     /WOR1 register,
 85     0xF800,     /WOR2 register,
 86     0x1107,     //RFI register,      Enable Tx Ramp up/down
 87     0x0170,     //PM register,
 88     0x0201,     //RTH register,
 89     0x400F,     //AGC1 register,
 90     0x2AC0,     //AGC2 register,
 91     0x0045,     //GIO register,      GIO2=WTR, GIO1=FSYNC
 92     0xD181,     //CKO register
 93     0x0004,     //VCB register,
 94     0x0A21,     //CHG1 register,     430MHz
 95     0x0022,     //CHG2 register,     435MHz
 96     0x003F,     //FIFO register,     FEP=63+1=64bytes
 97     0x1507,     //CODE register,     Preamble=4bytes, ID=4bytes
 98     0x0000      //WCAL register,
 99 };
100
101 const Uint16 code A7139Config_PageB[]=   //433MHz, 100kbps (IFBW = 100KHz, Fdev = 37.5KHz)
102 {
103     0x0337,     //TX2 register,
104     0x8400,     //IF1 register,      Enable Auto-IF, IF=200KHz
105     0x0000,     //IF2 register,
106     0x0000,     //ACK register,
107     0x0000      //ART register,
108 };
```

功能說明：A7139 configure table	
行數	說明
61~108	A7139 的初始設定

```
110 /***********************************************************************
111  **  function Declaration
112  ***********************************************************************/
113 void Timer0ISR(void);
114 void UART0Isr(void);
115 void InitTimer0(void);
116 void InitUART0(void);
117 void InitRF(void);
118 void A7139_Config(void);
119 void A7139_WriteID(void);
120 void A7139_Cal(void);
121 void StrobeCMD(Uint8);
122 void ByteSend(Uint8);
123 Uint8 ByteRead(void);
124 void A7139_WriteReg(Uint8, Uint16);
125 Uint16 A7139_ReadReg(Uint8);
126 void A7139_WritePageA(Uint8, Uint16);
127 Uint16 A7139_ReadPageA(Uint8);
128 void A7139_WritePageB(Uint8, Uint16);
129 Uint16 A7139_ReadPageB(Uint8);
130 void A7139_WriteFIFO(void);
131 void RxPacket(void);
132 void Err_State(void);
```

功能說明：副程式檔頭宣告	
行數	說明
113~132	副程式宣告

```
/*********************************************************************
 * main loop
 *********************************************************************/
void main(void)
{
    //initsw
    PMR |= 0x01;        //set DME0

    //initHW
    P0 = 0xFF;
    P1 = 0xFF;
    P2 = 0xFF;
    P3 = 0xFF;
    P4 = 0x0F;

    InitTimer0();
    InitUART0();
    TR0=1;    //Timer0 on
    EA=1;     //enable interrupt

    if((P4 & 0x04)==0x04)         //if P4.2=1, master
    {
        InitRF(); //init RF

        while(1)
        {
            A7139_WriteFIFO();    //write data to TX FIFO
            StrobeCMD(CMD_TX);
            Delay10us(1);
            while(GIO2);              //wait transmit completed
            StrobeCMD(CMD_RX);
            Delay10us(1);

            timer=0;
            TimeoutFlag=0;
            while((GIO2==1)&&(TimeoutFlag==0));   //wait receive completed
            if(TimeoutFlag)
            {
                StrobeCMD(CMD_STBY);
            }
            else
            {
                RxPacket();
                Delay10ms(10);
            }
        }
    }
    else        //if P4.2=0, slave
    {
        InitRF(); //init RF

        RxCnt = 0;
        Err_ByteCnt = 0;
        Err_BitCnt = 0;

```

```
189         while(1)
190         {
191             StrobeCMD(CMD_RX);
192             Delay10us(1);
193             while(GIO2);                //wait receive completed
194             RxPacket();
195
196             A7139_WriteFIFO();    //write data to TX FIFO
197             StrobeCMD(CMD_TX);
198             Delay10us(1);
199             while(GIO2);                //wait transmit completed
200
201           Delay10ms(9);
202         }
203     }
204 }
```

功能說明：主程式

行數	說明
140	啓用 MCU on chip data SRAM
143~147	初始化 MCU I/O Port
149	呼叫副程式 initTimer0，致能中斷
150	呼叫副程式 initUart0，Uart0 初始化
151	啓動 timer0
152	致能中斷開啓
154	判別 P4.2，如果 P4.2=1 則執行 Master 端程式, 否則, 執行 Slave 端程式。
156~179	Master 端程式
156	呼叫副程式 initRF，初始化 A7139 chip
160	呼叫副程式 A7139_WriteFIFO，將 64 bytes data 寫入 TX FIFO
161	RF chip 進入 TX 模式，發送資料
163	等待資料傳送完畢。
164	RF chip 進入 RX 模式
167~168	清除變數 timer0 及 Timeout Flag
169	等待是否接收到正確的資料或是 Timeout
170~173	判別是否爲 Timeout，如爲 Timeout，則呼叫副程式 StrobeCMD, 進入 standby 模式
176	呼叫副程式 RxPacket，從 RX FIFO 讀出資料、比對、計算 error bit 數
177	呼叫副程式 Delay10ms，程式延遲 100ms 動作
183~202	Slave 端程式
183	呼叫副程式 initRF，初始化 A7139 chip
185~187	清除變數 Seq,RxCnt, Err_ByteCnt, Err_BitCnt
191	RF chip 進入 RX 模式
193	等待接收到正確的資料
194	呼叫副程式 RxPacket，從 RX FIFO 讀出資料、比對、計算 error bit 數
196	呼叫副程式 A7139_WriteFIFO，將 64 bytes data 寫入 TX FIFO
197	呼叫副程式 StrobeCMD，進入 TX 模式，發送資料
199	等待資料傳送完畢。
201	呼叫副程式 Delay10ms，程式延遲 90ms 動作

```
206 /*********************************************************************
207 **  Timer0ISR
208 *********************************************************************/
209 void Timer0ISR (void) interrupt 1
210 {
211     TH0 = (65536-t0hrel)>>8;// Reload Timer0 high byte,low byte
212     TL0 = 65536-t0hrel;
213
214     timer++;
215     if (timer >= TIMEOUT)
216     {
217         TimeoutFlag=1;
218     }
219
220     TimerCnt0++;
221     if (TimerCnt0 == 500)
222     {
223         TimerCnt0=0;
224         CmdBuf[0]=0xF1;
225
226         memcpy(&CmdBuf[1], &RxCnt, 2);
227         memcpy(&CmdBuf[3], &Err_ByteCnt, 4);
228         memcpy(&CmdBuf[7], &Err_BitCnt, 4);
229
230         UartSendCnt=11;
231         Uartptr=&CmdBuf[0];
232         SBUF=CmdBuf[0];
233     }
234 }
```

功能說明：初始化 Timer0 的中斷副程式

行數	說明
211~212	設置 TH0, TL0 的啓始值
214	變數 timer 加 1
215~218	判別變數 timer 是否等於 TIMEOUT 值，如 Timeout，則設置旗標 TimeoutFlag=1
220	變數 TimerCnt0 加 1
221	判別變數 TimerCnt0 是否等於 500(即 500ms)
223	清除變數 TimerCnt0
224	CmdBuf[0]設置 0xF1 爲傳送啓始位元識別碼
226	CmdBuf[1] 、CmdBuf[1]設置變數 RxCnt 的值
227	CmdBuf[3] 、CmdBuf[4]、CmdBuf[5] 、CmdBuf[6]設置變數 Err_ByteCnt 的值
228	CmdBuf[7] 、CmdBuf[8]、CmdBuf[9] 、CmdBuf[10]設置變數 Err_BitCn 的值
230	設置變數UartSendCnt=11
231	設置指標變數 Uartptr 指到變數 CmdBuf[0]的啓始位址
232	將 BER 結果傳送 SBUF 至 PC

```
236 /*******************************************************************************
237 **  Uart0ISR
238 *******************************************************************************/
239 void Uart0Isr(void) interrupt 4 using 3
240 {
241     if (TI==1)
242     {
243         TI=0;
244         UartSendCnt--;
245         if(UartSendCnt !=0)
246         {
247             Uartptr++;
248             SBUF = *Uartptr;
249         }
250     }
251 }
```

功能說明：初始化 uart0 的中斷副程式	
行數	說明
241	判別 TI1 旗標是否爲 Uart 已傳送完成 1byte
243	清除 TI1 旗標
244	變數 UartSendCnt 減 1
245	判別變數 UartSendCnt 是否爲 0。如不爲 0，則繼續傳送下一個資料
247~248	指標變數 Uartptr 加 1，並將其位址的資料，使用 Uart0 送至 PC

```
253  /*******************************************************************************
254 **  init Timer0
255 *******************************************************************************/
256 void InitTimer0(void)
257 {
258     TR0 = 0;
259     TMOD =(TMOD & 0xF0)|0x01;       //timer0 mode=1
260     TH0 = (65536-t0hrel)>>8;        //setup Timer0 high byte,low byte
261     TL0 = 65536-t0hrel;
262     TF0 = 0;                        //Clear any pending Timer0 interrupts
263     ET0 = 1;                        // Enable Timer0 interrupt
264 }
```

功能說明：初始化 Timer0 程序	
行數	說明
258	關閉 Timer0 計時動作
259	設置 Timer0 在 mode 1 模式
260~261	設置 TH0,TL0 的初始值
262	清除 Timer0 中斷旗標
263	致能 Timer0 中斷

```
266 /*********************************************************************
267 **  Init Uart0
268 *********************************************************************/
269 void initUart0(void)
270 {
271     TH1 = 0xFD;         //BaudRate 9600;
272     TL1 = 0xFD;
273     SCON = 0x40;
274     TMOD = (TMOD & 0x0F) | 0x20;
275     REN = 1;
276     TR1 = 1;
277     ES = 1;
278 }
```

功能說明：初始化 Uart0 的程序

行數	說明
271~273	初始 TL1,TH1,SCON1 值，設置爲 9600bps @xtal=11.0592MHz
274	設置 Timer1 爲 mode 2
275~277	設置 REN1,TR1,ES1 爲 1，啓用 Uart0 的功能

```
280 /*********************************************************************
281 ** Strobe Command
282 *********************************************************************/
283 void StrobeCMD (Uint8 cmd)
284 {
285     Uint8 i;
286
287     SCS = 0;
288     for(i = 0; i < 8; i++)
289     {
290         if(cmd & 0x80)
291             SDIO = 1;
292         else
293             SDIO = 0;
294
295         _nop_();
296         SCK = 1;
297         _nop_();
298         SCK = 0;
299         cmd <<= 1;
300     }
301     SCS = 1;
302 }
```

功能說明： 執行 Strobe 指令副程式。

行數	說明
287~301	寫入一個 byte 的 strobe 指令。

```
304 /*********************************************************************
305 **  ByteSend
306 *********************************************************************/
307 void ByteSend(Uint8 src)
308 {
309     Uint8 i;
310
311     for(i = 0; i < 8; i++)
312     {
313         if(src & 0x80)
314             SDIO = 1;
315         else
316             SDIO = 0;
317
318         _nop_();
319         SCK = 1;
320         _nop_();
321         SCK = 0;
322         src <<= 1;
323     }
324 }
```

功能說明：寫入 1 byte 的程序。	
行數	說明
311~323	寫入 1 個 byte 的程序

```
326 /*********************************************************************
327 **  ByteRead
328 *********************************************************************/
329 Uint8 ByteRead(void)
330 {
331     Uint8 i, tmp;
332
333     //read data code
334     SDIO = 1;       //SDIO pull high
335     for(i = 0; i < 8; i++)
336     {
337         if(SDIO)
338             tmp = (tmp << 1) | 0x01;
339         else
340             tmp = tmp << 1;
341
342         SCK = 1;
343         _nop_();
344         SCK = 0;
345     }
346     return tmp;
347 }
```

功能說明：讀出 1byte 的程序。	
行數	說明
335~345	讀出 1 個 byte 的程序
346	傳回 tmp 的數值

```
349 /*********************************************************************
350 **  A7139_WriteReg
351 *********************************************************************/
352 void A7139_WriteReg(Uint8 address, Uint16 dataWord)
353 {
354     Uint8 i;
355
356     SCS = 0;
357     address |= CMD_Reg_W;
358     for(i = 0; i < 8; i++)
359     {
360         if(address & 0x80)
361             SDIO = 1;//bit=1
362         else
363             SDIO = 0;//bit=0
364
365         SCK = 1;
366         _nop_();
367         SCK = 0;
368         address <<= 1;
369     }
370     _nop_();
371
372     //send data word
373     for(i = 0; i < 16; i++)
374     {
375         if(dataWord & 0x8000)
376             SDIO = 1;
377         else
378             SDIO = 0;
379
380         SCK = 1;
381         _nop_();
382         SCK = 0;
383         dataWord <<= 1;
384     }
385     SCS = 1;
386 }
```

功能說明：對 A7139 內部控制暫存器(Control Register)寫入動作。

行數	說明
356	SCS=0，致能 SPI 讀寫功能
357	將 address 寫入控制暫存器命令。
358~369	寫入 address 的程序
373~384	寫入 data word 的程序
385	SCS=1，清除 SPI 讀寫功能

```
388 /*********************************************************************
389 **  A7139_ReadReg
390 *********************************************************************/
391 Uint16 A7139_ReadReg(Uint8 address)
392 {
393     Uint8 i;
394     Uint16 tmp;
395
396     SCS = 0;
397     address |= CMD_Reg_R;
398     for(i = 0; i < 8; i++)
399     {
400         if(address & 0x80)
401             SDIO = 1;
402         else
403             SDIO = 0;
404
405         _nop_();
406         SCK = 1;
407         _nop_();
408         SCK = 0;
409
410         address <<= 1;
411     }
412     _nop_();
413
414     //read data code
415     SDIO = 1;       //SDIO pull high
416     for(i = 0; i < 16; i++)
417     {
418         if(SDIO)
419             tmp = (tmp << 1) | 0x01;
420         else
421             tmp = tmp << 1;
422
423         SCK = 1;
424         _nop_();
425         SCK = 0;
426     }
427     SCS = 1;
428     return tmp;
429 }
```

功能說明：對 A7139 內部控制暫存器(Control Register)讀出動作。

行數	說明
396	SCS=0，致能 SPI 讀寫功能
397	將 address 寫入控制暫存器命令。
398~411	寫入 address 的程序
416~426	讀出 data word 的程序
427	SCS=1，清除 SPI 讀寫功能
428	傳回 tmp 的數值

```
431 /*********************************************************************
432 **  A7139_WritePageA
433 *********************************************************************/
434 void A7139_WritePageA(Uint8 address, Uint16 dataWord)
435 {
436     Uint16 tmp;
437
438     tmp = address;
439     tmp = ((tmp << 12) | A7139Config[CRYSTAL_REG]);
440     A7139_WriteReg(CRYSTAL_REG, tmp);
441     A7139_WriteReg(PAGEA_REG, dataWord);
442 }
```

功能說明：對 A7139 的 page A 暫存器做寫入動作.

Line	Description
438~440	在 CRYSTAL 暫存器 bit[15:12]中，設定欲寫入的 Page A 位址。
441	將資料寫入欲寫入的 page A 暫存器中。 (根據 CRYSTAL bit[15:12]設定的位址)

```
444 /*********************************************************************
445 **  A7139_ReadPageA
446 *********************************************************************/
447 Uint16 A7139_ReadPageA(Uint8 address)
448 {
449     Uint16 tmp;
450
451     tmp = address;
452     tmp = ((tmp << 12) | A7139Config[CRYSTAL_REG]);
453     A7139_WriteReg(CRYSTAL_REG, tmp);
454     tmp = A7139_ReadReg(PAGEA_REG);
455     return tmp;
456 }
```

功能說明：對 A7139 的 page A 暫存器做讀出動作

Line	Description
451~453	在 CRYSTAL 暫存器 bit[15:12]中，設定欲讀出的 Page A 位址。
454	將資料從欲讀出的 page A 暫存器中讀出。 (根據 CRYSTAL bit[15:12]設定的位址)
455	傳回 tmp 數值

```
458 /*********************************************************************
459 **  A7139_WritePageB
460 *********************************************************************/
461 void A7139_WritePageB(Uint8 address, Uint16 dataWord)
462 {
463     Uint16 tmp;
464
465     tmp = address;
466     tmp = ((tmp << 7) | A7139Config[CRYSTAL_REG]);
467     A7139_WriteReg(CRYSTAL_REG, tmp);
468     A7139_WriteReg(PAGEB_REG, dataWord);
469 }
```

功能說明：對 A7139 的 page B 暫存器做寫入動作.

Line	Description
465~467	在 CRYSTAL 暫存器 bit[9:7]中，設定欲寫入的 Page B 位址。
468	將資料寫入欲寫入的 page B 暫存器中。 (根據 CRYSTAL bit[9:7]設定的位址)

```
471 /*********************************************************************
472 **  A7139_ReadPageB
473 *********************************************************************/
474 Uint16 A7139_ReadPageB(Uint8 address)
475 {
476     Uint16 tmp;
477
478     tmp = address;
479     tmp = ((tmp << 7) | A7139Config[CRYSTAL_REG]);
480     A7139_WriteReg(CRYSTAL_REG, tmp);
481     tmp = A7139_ReadReg(PAGEB_REG);
482     return tmp;
483 }
```

功能說明：對 A7139 的 page B 暫存器做讀出動作	
Line	Description
478~480	在 CRYSTAL 暫存器 bit[9:7]中，設定欲讀出的 Page B 位址。
481	將資料從欲讀出的 page B 暫存器中讀出。 (根據 CRYSTAL bit[9:7]設定的位址)
482	傳回 tmp 數值

```
485 /*********************************************************************
486 ** initRF
487 *********************************************************************/
488 void InitRF(void)
489 {
490     //initial pin
491     SCS = 1;
492     SCK = 0;
493     SDIO= 1;
494     CKO = 1;
495     GIO1= 1;
496     GIO2= 1;
497
498     StrobeCMD(CMD_RF_RST);          //reset A7139 chip
499     A7139_Config();                 //config A7139 chip
500     Delay100us(8);                  //delay 800us for crystal stabilized
501     A7139_WriteID();                //write ID code
502     A7139_Cal();                    //IF and VCO calibration
503 }
```

功能說明：初始化 RF chip。	
行數	說明
491~496	設置 RF chip 介面 I/O 初始值
498	A7139 重置。
499	呼叫副程式 A7139_Config，初始 RF chip 控制暫存器
500	延遲時間，等待 crystal 振盪穩定
501	呼叫副程式 A7139_WriteID，寫入 ID code
502	呼叫副程式 A7139_Cal，做 VCO, IF, 和 RSSI 的校準程序

```
505 /*********************************************************************
506 ** A7139_Config
507 *********************************************************************/
508 void A7139_Config(void)
509 {
510     Uint8 i;
511     Uint16 tmp;
512
513     for(i=0; i<8; i++)
514         A7139_WriteReg(i, A7139Config[i]);
515
516     for(i=10; i<16; i++)
517         A7139_WriteReg(i, A7139Config[i]);
518
519     for(i=0; i<16; i++)
520         A7139_WritePageA(i, A7139Config_PageA[i]);
521
522     for(i=0; i<5; i++)
523         A7139_WritePageB(i, A7139Config_PageB[i]);
524
525     //for check
526     tmp = A7139_ReadReg(SYSTEMCLOCK_REG);
527     if(tmp != A7139Config[SYSTEMCLOCK_REG])
528     {
529         Err_State();
530     }
531 }
```

功能說明：初始 RF config 的程序

行數	說明
513~517	將初始值寫入一般暫存器。
519~520	將初始值寫入 page A 暫存器。
522~523	將初始值寫入 page B 暫存器。
526~530	檢查暫存器資料寫入是否正確

```
533 /*********************************************************************
534 **  WriteID
535 *********************************************************************/
536 void A7139_WriteID(void)
537 {
538     Uint8 i;
539     Uint8 d1, d2, d3, d4;
540
541     SCS=0;
542     ByteSend(CMD_ID_W);
543     for(i=0; i<4; i++)
544                 ByteSend(ID_Tab[i]);
545     SCS=1;
546
547     SCS=0;
548     ByteSend(CMD_ID_R);
549     d1=ByteRead();
550     d2=ByteRead();
551     d3=ByteRead();
552     d4=ByteRead();
553     SCS=1;
554
555     if((d1!=ID_Tab[0]) || (d2!=ID_Tab[1]) || (d3!=ID_Tab[2]) || (d4!=ID_Tab[3]))
556     {
557         Err_State();
558     }
559 }
```

功能說明：寫入 ID 程序

行數	說明
541	SCS=0，致能 SPI 讀寫功能
542	送出 Write ID 命令
543~544	將 ID_Tab Table 寫入 A7139 ID Code 暫存器.
545	SCS=1，清除 SPI 讀寫功能
547	SCS=0，致能 SPI 讀寫功能
548	送出 Read ID 命令
549~552	依序讀出 ID code 4bytes
553	SCS=1，清除 SPI 讀寫功能
555~558	檢查 ID Code 寫入是否正確

```
561 /*********************************************************************
562 ** A7139_Cal
563 *********************************************************************/
564 void A7139_Cal(void)
565 {
566     Uint8 fb, fcd, fbcf;            //IF Filter
567     Uint8 vb,vbcf;                  //VCO Current
568     Uint8 vcb, vccf;                //VCO Band
569     Uint16 tmp;
570
571     //IF calibration procedure @STB state
572     A7139_WriteReg(MODE_REG, A7139Config[MODE_REG] | 0x0802);    //IF Filter & VCO Current Calibration
573     do{
574         tmp = A7139_ReadReg(MODE_REG);
575     }while(tmp & 0x0802);
576
577     //for check(IF Filter)
578     tmp = A7139_ReadReg(CALIBRATION_REG);
579     fb = tmp & 0x0F;
580     fcd = (tmp>>11) & 0x1F;
581     fbcf = (tmp>>4) & 0x01;
582     if(fbcf)
583     {
584         Err_State();
585     }
586
587     //for check(VCO Current)
588     tmp = A7139_ReadPageA(VCB_PAGEA);
589     vcb = tmp & 0x0F;
590     vccf = (tmp>>4) & 0x01;
591     if(vccf)
592     {
593         Err_State();
594     }
595
596
597     //RSSI Calibration procedure @STB state
598     A7139_WriteReg(ADC_REG, 0x4C00);                    //set ADC average=64
599     A7139_WriteReg(MODE_REG, A7139Config[MODE_REG] | 0x1000);    //RSSI Calibration
600     do{
601         tmp = A7139_ReadReg(MODE_REG);
602     }while(tmp & 0x1000);
603     A7139_WriteReg(ADC_REG, A7139Config[ADC_REG]);
604
605
```

```
606     //VCO calibration procedure @STB state
607     A7139_WriteReg(PLL1_REG, A7139Config [PLL1_REG]);
608     A7139_WriteReg(PLL2_REG, A7139Config [PLL2_REG]);
609     A7139_WriteReg(MODE_REG, A7139Config[MODE_REG] | 0x0004);      //VCO Band Calibration
610     do{
611        tmp = A7139_ReadReg(MODE_REG);
612     }while(tmp & 0x0004);
613
614     //for check(VCO Band)
615     tmp = A7139_ReadReg(CALIBRATION_REG);
616     vb = (tmp >>5) & 0x07;
617     vbcf = (tmp >>8) & 0x01;
618     if(vbcf)
619     {
620        Err_State();
621     }
622 }
```

功能說明：VCO,IF 以及 RSSI 校準程序

行數	說明
572	校準程序建議在 Standby 或 PLL state 下執行 設置 mode control register 中 bit FBC=1 和 VCC=1。
573~575	讀出 mode control register 並判別 bit FBC 和 bit VCC 是否爲 0。如爲 0，則跳出等待迴圈。
578~585	讀出 calibration register，並檢示其值 判別 bit fbcf 是否爲 1。如爲 1，則進入 Err_State 處理程序。
588~594	讀出 vco current register，並檢示其值 判別 bit vccf 是否爲 1。如爲 1，則進入 Err_State 處理程序。
598	設置 ADC average , RSSC_D , RS_DLY, 和 RC_DLY
599	設置 mode control register 中 bit RSSC=1
600~602	讀出 mode control register 並判別 bit RSSC 是否爲 0。如爲 0，則跳出等待迴圈。
603	設置原本的 ADC average , RSSC_D, RS_DLY, 和 RC_DLY
607~621	對工作 band 做 VCO 校準程序。
607~608	設定工作頻率
609	設置 mode control register 中 bit VBC=1。
610~612	讀出 mode control register 並判別 bit VBC 是否爲 0。如爲 0，則跳出等待迴圈
615~621	讀出 calibration control register，並檢示其值 判別 bit vbcf 是否爲 1。如爲 1，則進入 Err_State 處理程序。

```
624 /*********************************************************************
625 ** A7139_WriteFIFO
626 *********************************************************************/
627 void A7139_WriteFIFO(void)
628 {
629     Uint8 i;
630
631     StrobeCMD(CMD_TFR);              //TX FIFO address pointer reset
632
633     SCS=0;
634     ByteSend(CMD_FIFO_W);            //TX FIFO write command
635     for(i=0; i <64; i++)
636         ByteSend(PN9_Tab[i]);
637     SCS=1;
638 }
```

功能說明：TX FIFO 寫入資料的程序	
行數	說明
631	重置 TX FIFO 的 write pointer.
633	SCS=0，致能 SPI 讀寫功能
634	送出 TX FIFO 寫入命令
635~636	寫入 64 bytes 的資料
637	SCS=1，清除 SPI 讀寫功能

```
640 /*********************************************************************
641 ** RxPacket
642 *********************************************************************/
643 void RxPacket(void)
644 {
645     Uint8 i;
646     Uint8 recv;
647     Uint8 tmp;
648
649     RxCnt++;
650
651     StrobeCMD(CMD_RFR);    //RX FIFO address pointer reset
652
653     SCS=0;
654     ByteSend(CMD_FIFO_R); //RX FIFO read command
655     for(i=0; i <64; i++)
656     {
657         tmpbuf[i] = ByteRead();
658     }
659     SCS=1;
660
661     for(i=0; i<64; i++)
662     {
663         recv = tmpbuf[i];
664         tmp = recv ^ PN9_Tab[i];
665         if(tmp!=0)
666         {
667             Err_ByteCnt++;
668             Err_BitCnt += (BitCount_Tab[tmp>>4] + BitCount_Tab[tmp & 0x0F]);
669         }
670     }
671 }
```

功能說明：從 RX FIFO 讀出 data 及資料的比對程序

行數	說明
649	變數 RxCnt 加 1
651	重置 RX FIFO 的 read pointer
653	SCS=0，致能 SPI 讀寫功能
654	送出 RX FIFO 讀出命令
655~658	從 RX FIFO 讀出 64bytes data
659	SCS=1，清除 SPI 讀寫功能
661~670	比較 data 的正確性，計算出 error bit

```
673 /*********************************************************************
674 ** Err_State
675 *********************************************************************/
676 void Err_State(void)
677 {
678     //ERR display
679     //Error Proc...
680     //...
681     while(1);
682 }
```

功能說明：Error state 處理程序

行數	說明
678~681	使用者自行定義 error state 的處理程序