Document Title

A7105 reference code for FIFO extension mode (500Kbps)

Revision History

Rev. No.	History	Issue Date	Remark
0.0	Preliminary	Jan 9 , 2009	

Table of contents

RF Chip-A7105
Reference code for FIFO extension mode (500Kbps)

1. 簡介

這文件係對 RF chip -A7105 FIFO mode，使用 FIFO data 大於 64 bytes，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 128 bytes 資料。之後，延遲 50ms 後再進入 TX 狀態，重新另一次的傳送循環動作 。

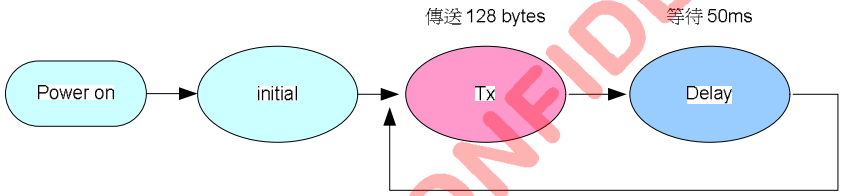

Fig1. Master 端方塊圖

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，則仍在 RX 狀態，等待接收。若有收到 Master 端所發送的資料，則讀出資料、比對，計算 error bit。之後，延遲 30ms 後再次回到 RX 狀態等待下一次接收。

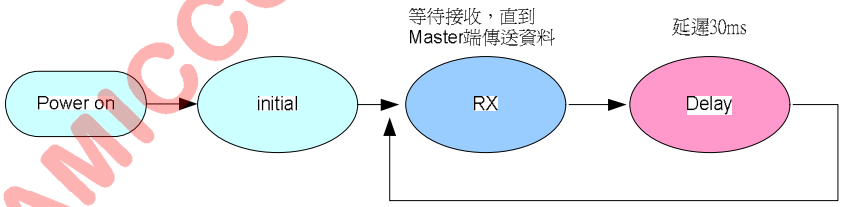

Fig2. Slave 端方塊圖

3. 硬體

3.1 系統方塊圖

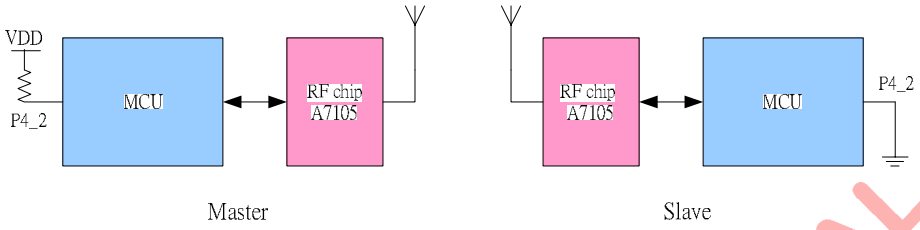

Fig2. 系統方塊圖

MCU 使用 I/O pin 4_2 的設定，判別 Master 端或 Slave 端。

使用 I/O pin 設定：
應用範例使用 I/O：
SCS, SCK, SDIO - 這 3 wire 串列介面控制 A7105 內部 register。
GIO1 - 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。
CKO – 監控 TX FIFO 或 RX FIFO 在 FIFO extension 下的 FIFO Pointer 臨界值的變化，藉以控制資料寫入或讀出 FIFO 的時機。

MCU 控制 A7105 RF chip 的 I/O 配置如下圖：

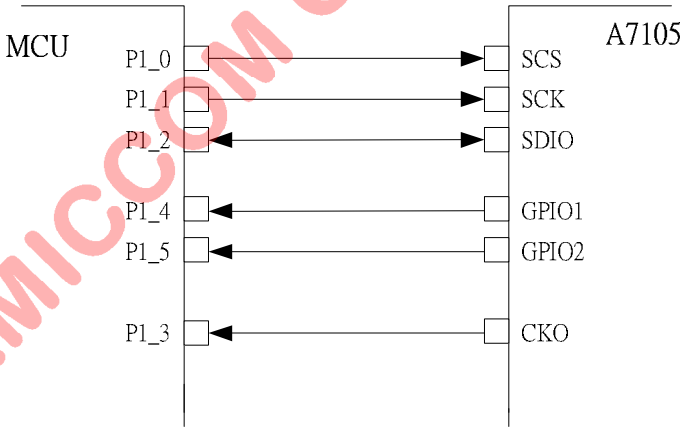

Fig3. I/O 配置圖

4. 韌體程式設計:

4.1 應用範例概述

首先初始化 Timer0、Uart0，之後判別 Port 4_2 =1 進入 master 端的主程式或 Port 4_2 =0 進入 slave 端的主程式。

Master 端：

1) 初始化 A7105RF chip。
2) TX FIFO 先寫入 PN9 code 共 64 bytes。
3) 進入 TX state，傳送封包。
4) 等待 CKO pin 爲 1 時，再對 TX FIFO 寫入 PN9 code 反向的資料， 共 64 bytes。
5) 等待 GIO1 pin 爲 0 後，RF chip 會自動結束 TX state，回復到 Standby state。
6) 延遲 50ms，重新回到 Step 2 動作，重新開始下一周時序工作。

Slave 端：

1) 初始化 A7105RF chip。
2) 進入 RX 狀態，等待封包收到。
3) 判斷 CKO pin 爲 1 後，從 RX FIFO 讀出資料放置 tmpbuf 中
4) 等待 GIO1 pin 爲 0 後，完成封包接收，RF chip 會自動結束 RX state，回復到 Standby state。
5) 從 RX FIFO 讀出資料放置 tmpbuf 中。
6) 從 tmpbuf 讀出、比對資料、計算 error bit 數目。
7) 延遲 30ms，重新回到 Step 2 動作，重新開始下一周期時序工作。
8) 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

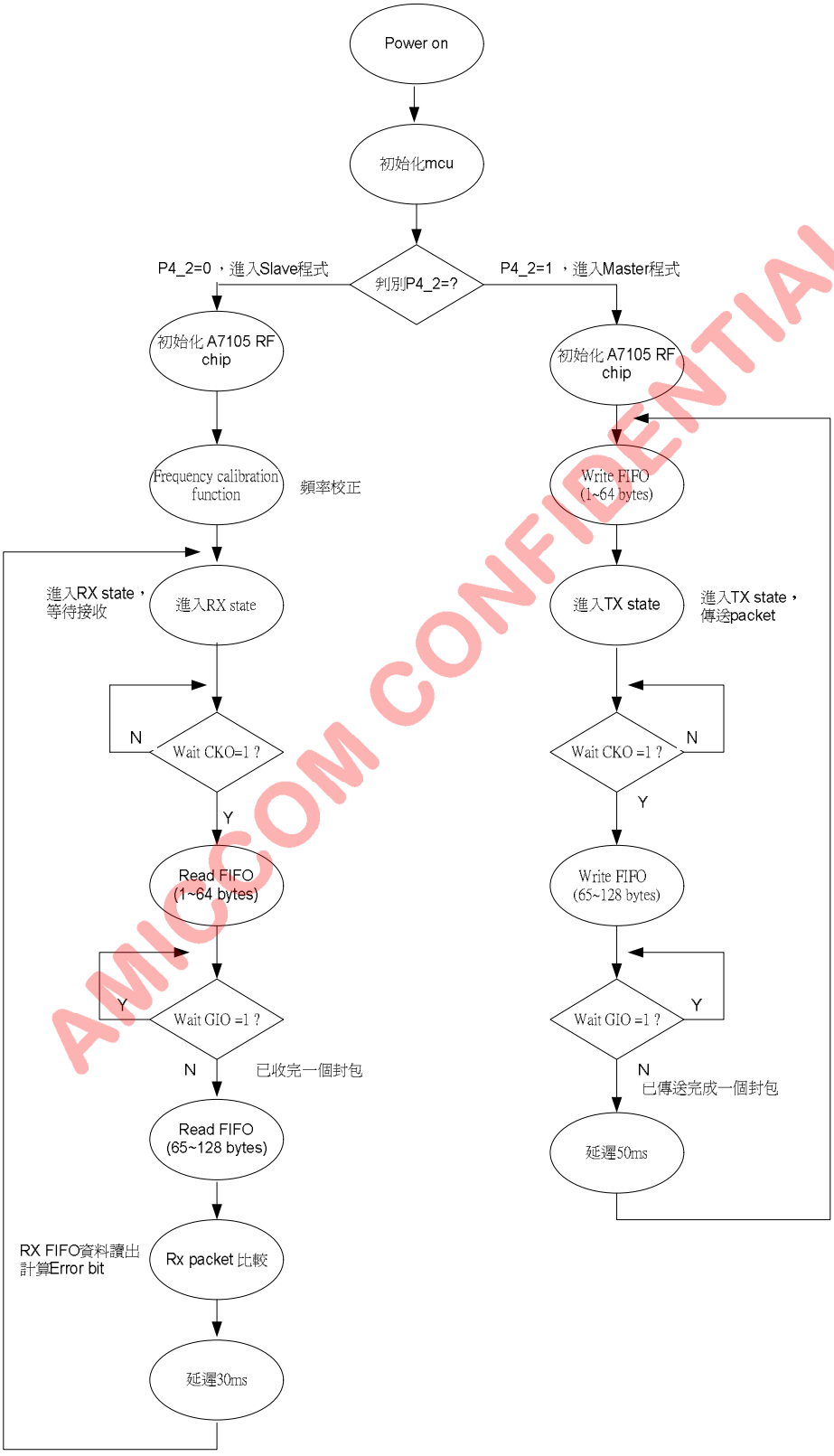

Fig4. 範例程式工作基本方塊

5. 程式說明

注意：
※本程式僅供 **FIFO extension** 模式下的動作參考程序，對於 **FIFO data** 寫入/讀出時間的控制，使用者需依 **MCU** 的工作速度，緊慎處理 **FIFO data** 寫入/讀出的時間，避免錯誤發生。

```
1  /*********************************************************************
2  ** Device:     A7105
3  ** File:       main.c
4  ** Author:     JPH
5  ** Target:     Winbond W77LE58
6  ** Tools:      ICE
7  ** Created:    2009-01-09
8  ** Description:
9  ** This file is a sample code for your reference.
10 **
11 **  Copyright (C) 2008 AMICCOM Corp.
12 **
13 *********************************************************************/
14 #include "define.h"
15 #include "w77le58.h"
16 #include "a7105reg.h"
17 #include "Uti.h"
```

功能說明：Include 檔宣告，定義常數變數

行數	說明
14~17	匯入程式庫設定檔

```
19 /*********************************************************************
20 **  I/O Declaration
21 *********************************************************************/
22 #define SCS          P1_0        //spi SCS
23 #define SCK          P1_1        //spi SCK
24 #define SDIO         P1_2        //spi SDIO
25 #define CKO          P1_3        //CKO
26 #define GIO1         P1_4        //GIO1
27 #define GIO2         P1_5        //GIO2
28 #define Button       P1_7        //test Button
29
30 /*********************************************************************
31 **  Constant Declaration
32 *********************************************************************/
33 #define TIMEOUT       50
34 #define t0hrel        1000
```

功能說明：MCU 對 A7105 RF chip I/O 接腳定義，常數定義

行數	說明
22~28	MCU I/O 配置
33~34	常數定義

```
36  /*********************************************************************
37  **  Global Variable Declaration
38  *********************************************************************/
39  Uint8       data    timer;
40  Uint8       data    TimeoutFlag;
41  Uint16      idata   RxCnt;
42  Uint32      idata   Err_ByteCnt;
43  Uint32      idata   Err_BitCnt;
44  Uint16      idata   TimerCnt0;
45  Uint8       data    *Uartptr;
46  Uint8       data    UartSendCnt;
47  Uint8       data    CmdBuf[12];
48  Uint8       xdata   tmpbuf[256];
49  Uint16      xdata   AFCBuf[12];
50  Uint8       data    sts_ModeReg;
51  Uint8       idata   Err_Frame;
52
53  const Uint8 code BitCount_Tab[16] = {0,1,1,2,1,2,2,3,1,2,2,3,2,3,3,4};
54  const Uint8 code ID_Tab[4]={0x54,0x75,0xC5,0x2A}; //ID code
55  const Uint8 code PN9_Tab[]=
56  {   0xFF,0x83,0xDF,0x17,0x32,0x09,0x4E,0xD1,
57      0xE7,0xCD,0x8A,0x91,0xC6,0xD5,0xC4,0xC4,
58      0x40,0x21,0x18,0x4E,0x55,0x86,0xF4,0xDC,
59      0x8A,0x15,0xA7,0xEC,0x92,0xDF,0x93,0x53,
60      0x30,0x18,0xCA,0x34,0xBF,0xA2,0xC7,0x59,
61      0x67,0x8F,0xBA,0x0D,0x6D,0xD8,0x2D,0x7D,
62      0x54,0x0A,0x57,0x97,0x70,0x39,0xD2,0x7A,
63      0xEA,0x24,0x33,0x85,0xED,0x9A,0x1D,0xE0
64  };      // 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
39~51	程式中使用的變數宣告
53	BitCount_Tab 宣告
54	ID code 宣告
55~64	PN9 data 宣告

```
66   const Uint16 code A7105Config[]=
67   {
68      0x00, //RESET register,         only reset, not use on config
69      0x42, //MODE register,
70      0x00, //CALIBRATION register,only read, not use on config
71      0x7F, //FIFO1 register,      128 bytes
72      0xC0, //FIFO2 register,      PFM[1:0]=[11]
73      0x00, //FIFO register,          for fifo read/write
74      0x00, //IDDATA register,        for idcode
75      0x00, //RCOSC1 register,
76      0x00, //RCOSC2 register,
77      0x00, //RCOSC3 register,
78      0x12, //CKO register,
79      0x01, //GIO1 register
80      0x00, //GIO2 register,
81      0x05, //CLOCK register,
82      0x00, //DATARATE register,  datarate=500Kbps
83      0x50, //PLL1 register,
84      0x9E, //PLL2 register,          RFbase 2400MHz
85      0x4B, //PLL3 register,
86      0x00, //PLL4 register,
87      0x02, //PLL5 register,
88      0x16, //TX1 register,
89      0x2B, //TX2 register,
90      0x12, //DELAY1 register,
91      0x00, //DELAY2 register,
92      0x62, //RX register,
93      0x80, //RXGAIN1 register,
94      0x80, //RXGAIN2 register,
95      0x00, //RXGAIN3 register,
96      0x0A, //RXGAIN4 register,
97      0x32, //RSSI register,
98      0xC3, //ADC register,
99      0x07, //CODE1 register,
100     0x16, //CODE2 register,
101     0x00, //CODE3 register,
102     0x00, //IFCAL1 register,
103     0x00, //IFCAL2 register,    only read
104     0x00, //VCOCCAL register,
105     0x00, //VCOCAL1 register,
106     0x3B, //VCOCAL2 register,
107     0x00, //BATTERY register,
108     0x17, //TXTEST register,
109     0x47, //RXDEM1 register,
110     0x80, //RXDEM2 register,
111     0x03, //CPC register,
112     0x01, //CRYSTAL register,
113     0x45, //PLLTEST register,
114     0x18, //VCOTEST1 register,
115     0x00, //VCOTEST2 register,
116     0x01, //IFAT register,
117     0x0F, //RSCALE register,
118     0x00  //FILTERTEST
119  };
```

功能說明： RF chip 初始設定

行數	說明
68~118	RF chip 的初始設定
100	Date rate 在 500kbps，建議設置 PMD[1:0]=[10].
109	Data rate 在 500kbps，建議設置 DCM[1:0]=[10].

```
121  /*********************************************************************
122   **  function Declaration
123   *********************************************************************/
124   void InitTimer0(void);
125   void initUart0(void);
126   void Timer0ISR (void);
127   void Uart0Isr(void);
128   void A7105_Reset(void);
129   void A7105_WriteReg(Uint8, Uint8);
130   Uint8 A7105_ReadReg(Uint8);
131   void ByteSend(Uint8 src);
132   Uint8 ByteRead(void);
133   void A7105_WriteID(void);
134   void A7105_WriteFIFO(void);
135   void initRF(void);
136   void A7105_Config(void);
137   void A7105_Cal(void);
138   void RxPacket(void);
139   void StrobeCmd(Uint8);
140   void SetCH(Uint8);
141   void Swap(Uint8, Uint8);
142   void FrequencyCal(void);
143   void A7105_WriteFIFO_1(void);
144   void ReadDataToBuf(Uint8 *);
```

功能說明：副程式檔頭宣告

行數	說明
124~144	副程式宣告

```
146  /*********************************************************************
147   * main loop
148   *********************************************************************/
149  void main(void)
150  {
151      //initsw
152      PMR |= 0x01; //set DME0
153
154      //initHW
155      P0 = 0xFF;
156      P1 = 0xFF;
157      P2 = 0xFF;
158      P3 = 0xFF;
159      P4 = 0x0F;
160
161      InitTimer0();
162      initUart0();
163      TR0=1;
164      EA=1;
165
166      if ((P4 & 0x04)==0x04) //if P4.2=1, master
167      {
168          initRF();
169          StrobeCmd(CMD_STBY);
170
171          while(1)
172          {
173              StrobeCmd(CMD_TFR); //reset tX FIFO pointer
174              A7105_WriteFIFO(); //write 1~64 bytes data to tx fifo
175              SetCH(100); //freq 2450.001MHz
176              StrobeCmd(CMD_TX); //entry tx
177
178              while(~CKO);
179              A7105_WriteFIFO_1(); //write 65~128 bytes data to tx fifo
180              while(GIO1); //wait transmit completed
181
182              Delay10ms(5);
183          }
184      }
185      else //if P4.2=0, slave
186      {
187          initRF();
188
189          RxCnt = 0;
190          Err_ByteCnt = 0;
191          Err_BitCnt = 0;
192          TR0 = 1;
193
194          //Frequency calibration with sync data
195          //at default link channel
196          SetCH(99);
197          FrequencyCal();
198          StrobeCmd(CMD_STBY);
199
```

```
200         while(1)
201         {
202             SetCH(99); //freq 2449.501MHz
203             StrobeCmd(CMD_RX); //entry rx
204
205             while(~CKO);
206             StrobeCmd(CMD_RFR); //reset rx FIFO pointer
207             ReadDataToBuf(&tmpbuf[0]); //read data to buf
208             while(GIO1); //wait receive completed
209             ReadDataToBuf(&tmpbuf[64]); //read data to buf
210             RxPacket(); //data compare
211
212             Delay10ms(3);
213         }
214     }
215 }
```

功能說明：主程式 main loop。Port4_2=1，進入 master 迴圈，行數 167~184 爲 master 程式。Port4_2=0，進入 slave 迴圈，行數 186~214 爲 slave 程式。

行數	說明
152	啓用 MCUon chip data SRAM
155~159	初始化 MCU I/O Port
161	呼叫副程式 initTimer0，致能中斷
162	呼叫副程式 initUart0，初始 Uart0
163~164	啓動 Timer0 ，致能中斷開啓
166	判別 port4_2=1，進入 master 迴圈。Port4_2=0，進入 slave 迴圈。
167~184	Master 迴圈程式
168	呼叫副程式 initRF，初始化 A7105 chip
169	使用 Strobe command，進入 Standby mode 模式
173	使用 Strobe command，重置 TX FIFO pointer
174	呼叫副程式 A7105_WriteFIFO，將 data 寫入 TX FIFO
175	呼叫副程式 SetCH，設定工作頻率
176	使用 Strobe command，進入 TX 模式，發送資料
178	判別 I/O CKO，等待是否已傳送完 48 bytes
179	呼叫副程式 A7105_WriteFIFO_1，將 data 寫入 TX FIFO
180	判別 I/O GIO1，等待是已傳送完成
182	延遲 50ms
186~214	Slave 迴圈程式
187	呼叫副程式 initRF，初始化 A7105 chip
189~192	清除變數 RxCnt, Err_ByteCnt, Err_BitCnt, seq, Err_HopCnt 值
196	呼叫副程式 SetCH，設定工作頻率
197	呼叫副程式 FrequencyCal，校正頻率偏移
198	使用 Strobe command，進入 Standby 模式
202	呼叫副程式 SetCH ，設定頻率
203	使用 Strobe command，進入 RX 模式
205	判別 I/O CKO，等待 RF chip 是否已接收完 48 bytes
206	使用 Strobe command，重置 RX FIFO pointer
207	使用 ReadDataToBuf 副程式，將 RX FIFO 的資料讀出，放至 tmpbuf
208	判別 I/O GIO1，等待是否已接收完成
209	使用 ReadDataToBuf 副程式，將 RX FIFO 的資料讀出，放至 tmpbuf
210	呼叫 RxPacket 副程式，從 RX FIFO 讀出資料、比對、計算 error bit 數
212	延遲 30ms

```
217  /*********************************************************************
218   **  init Timer0
219   *********************************************************************/
220  void InitTimer0(void)
221  {
222      TR0 = 0;
223      TMOD =(TMOD & 0xF0)|0x01; //timer0 mode=1
224      TH0 = (65536-t0hrel)>>8; // Reload Timer0 high byte,low byte
225      TL0 = 65536-t0hrel;
226      TF0 = 0; // Clear any pending Timer0 interrupts
227      ET0 = 1; // Enable Timer0 interrupt
228  }
```

功能說明：初始化 Timer0 程序

行數	說明
222	關閉 Timer0 計時動作
223	設置 Timer0 在 mode 1 模式
224~225	設置 TH0,TL0 的初始值
226	清除 Timer0 中斷旗標
227	致能 Timer0 中斷

```
230  /*********************************************************************
231   **  Timer0ISR
232   *********************************************************************/
233  void Timer0ISR (void) interrupt 1
234  {
235      TF0 = 0; // Clear Timer0 interrupt
236      TH0 = (65536-t0hrel)>>8; // Reload Timer0 high byte,low byte
237      TL0 = 65536-t0hrel;
238
239      timer++;
240      if (timer == TIMEOUT)
241      {
242          TimeoutFlag=1;
243      }
244
245      TimerCnt0++;
246      if (TimerCnt0 == 500)
247      {
248          TimerCnt0 = 0;
249          CmdBuf[0] = 0xF1;
250
251          memcpy(&CmdBuf[1], &RxCnt, 2);
252          memcpy(&CmdBuf[3], &Err_ByteCnt, 4);
253          memcpy(&CmdBuf[7], &Err_BitCnt, 4);
254          memcpy(&CmdBuf[11], &Err_Frame, 1);
255
256          UartSendCnt = 12;
257          Uartptr =& CmdBuf[0];
258          SBUF = CmdBuf[0];
259      }
260  }
```

功能說明：初始化 Timer0 的中斷副程式

行數	說明
235~237	清除 Timer0 中斷旗標，設置 TH0,TL0 的啓始值
239	變數 timer 加 1
240~243	判別變數 timer 是否等於 20ms。如是，清除變數 timer=0，pin P3_5 信號反向.

245	變數 TimerCnt0 加 1
246	判別變數 TimerCnt0 是否等於 500(即 500ms)
248	清除變數 TimerCnt0
249	CmdBuf[0]設置 0xF1 爲傳送啓始位元識別碼
251	CmdBuf[1] 、CmdBuf[1]設置變數 RxCnt 的值
252	CmdBuf[3] 、CmdBuf[4]、CmdBuf[5] 、CmdBuf[6]設置變數 Err_ByteCnt 的值
253	CmdBuf[7] 、CmdBuf[8]、CmdBuf[9] 、CmdBuf[10]設置變數 Err_BitCn 的值
254	CmdBuf[11]設置變數 Err_Frame 的值
256	設置變數UartSendCnt=12
257	設置指標變數 Uartptr 指到變數 CmdBuf[0]的啓始位址
258	傳送 SBUF 至 PC

```
262  /*******************************************************************************
263   **  Init Uart0
264   *******************************************************************************/
265  void initUart0(void)
266  {
267      TH1 = 0xFD; //BaudRate 9600;
268      TL1 = 0xFD;
269      SCON = 0x40;
270      TMOD = (TMOD & 0x0F) | 0x20;
271      REN = 1;
272      TR1 = 1;
273      ES = 1;
274  }
```

功能說明：初始化 Uart0 的程序

行數	說明
267~269	初始 TL1,TH1,SCON1 值，設置爲 9600bps @xtal=11.0592MHz
270	設置 Timer1 爲 mode 2
271~273	設置 REN,TR1,ES 爲 1，啓用 Uart0 的功能

```
276  /*******************************************************************************
277   **  Uart0 ISR
278   *******************************************************************************/
279  void Uart0Isr(void) interrupt 4 using 3
280  {
281      if (TI==1)
282      {
283          TI=0;
284          UartSendCnt--;
285          if(UartSendCnt !=0)
286          {
287              Uartptr++;
288              SBUF = *Uartptr;
289          }
290      }
291  }
```

功能說明：初始化 uart0 的中斷副程式

行數	說明
281	判別 TI 旗標是否爲 Uart 已傳送完成 1byte
283	清除 TI 旗標
284	變數 UartSendCnt 減 1

285	判別變數 UartSendCnt 是否爲 0。如不爲 0，則繼續傳送下一個資料
287~288	指標變數 Uartptr 加 1，並將其位址的資料，使用 Uart0 送至 PC

```
293  /*********************************************************************
 294  **  Reset_RF
 295  *********************************************************************/
 296  void A7105_Reset(void)
 297  {
 298      A7105_WriteReg(MODE_REG, 0x00); //reset RF chip
 299  }
```

功能說明：A7105 RF chip Reset 程序

行數	說明
298	對 register 位址 0 ，寫入 0x00，重置 RF chip。

```
301  /*********************************************************************
 302  **  WriteID
 303  *********************************************************************/
 304  void A7105_WriteID(void)
 305  {
 306      Uint8 i;
 307      Uint8 d1,d2,d3,d4;
 308      Uint8 addr;
 309
 310      addr = IDCODE_REG; //send address 0x06, bit cmd=0, r/w=0
 311      SCS = 0;
 312      ByteSend(addr);
 313      for (i=0; i < 4; i++)
 314          ByteSend(ID_Tab[i]);
 315      SCS = 1;
 316
 317      //for check
 318      addr = IDCODE_REG | 0x40; //send address 0x06, bit cmd=0, r/w=1
 319      SCS=0;
 320      ByteSend(addr);
 321      d1=ByteRead();
 322      d2=ByteRead();
 323      d3=ByteRead();
 324      d4=ByteRead();
 325      SCS=1;
 326  }
```

功能說明：寫入 ID 的程序。

行數	說明
310	計算變數 addr 的值
311	SCS=0，設置控制暫存器讀寫功能
312~314	寫入 ID 控制暫存器的位址，及 4 bytes 的 ID code
315	SCS=1，清除 SPI 讀寫功能
318	計算讀出 ID code 的變數 addr 值
319	SCS=0，設置控制暫存器讀寫功能
320~324	讀出 ID code
325	SCS=1，清除控制暫存器讀寫功能

```
328  /*********************************************************************
329   **  A7105_WriteReg
330   *********************************************************************/
331  void A7105_WriteReg(Uint8 addr, Uint8 dataByte)
332  {
333      Uint8 i;
334
335      SCS = 0;
336      addr |= 0x00; //bit cmd=0,r/w=0
337      for(i = 0; i < 8; i++)
338      {
339          if(addr & 0x80)
340              SDIO = 1;
341          else
342              SDIO = 0;
343
344          SCK = 1;
345          _nop_();
346          SCK = 0;
347          addr = addr << 1;
348      }
349      _nop_();
350
351      //send data byte
352      for(i = 0; i < 8; i++)
353      {
354          if(dataByte & 0x80)
355              SDIO = 1;
356          else
357              SDIO = 0;
358
359          SCK = 1;
360          _nop_();
361          SCK = 0;
362          dataByte = dataByte<< 1;
363      }
364      SCS = 1;
365  }
```

功能說明：對 A7105 控制暫存器(Control Register)寫入動作

行數	說明
335	SCS=0，致能控制暫存器讀寫功能
336	將 address Or 寫入控制暫存器命令。
337~348	寫入 address 的程序
352~363	寫入 data byte 的程序
364	SCS=1，清除控制暫存器讀寫功能

```
367  /*********************************************************************
368   **  A7105_ReadReg
369   *********************************************************************/
370  Uint8 A7105_ReadReg(Uint8 addr)
371  {
372      Uint8 i;
373      Uint8 tmp;
374
375      SCS = 0;
376      addr |= 0x40; //bit cmd=0,r/w=1
377      for(i = 0; i < 8; i++)
378      {
379
380          if(addr & 0x80)
381              SDIO = 1;
382          else
383              SDIO = 0;
384
385          _nop_();
386          SCK = 1;
387          _nop_();
388          SCK = 0;
389
390          addr = addr << 1;
391      }
392
393      _nop_();
394      SDIO = 1;
395
396      //read data
397      for(i = 0; i < 8; i++)
398      {
399          if(SDIO)
400              tmp = (tmp << 1) | 0x01;
401          else
402              tmp = tmp << 1;
403
404          SCK = 1;
405          _nop_();
406          SCK = 0;
407      }
408      SCS = 1;
409      return tmp;
410  }
```

功能說明：A7105 控制暫存器(Control Register)讀出動作

行數	說明
375	SCS=0，致能控制暫存器讀寫功能
376	將 address Or 讀出控制暫存器命令。
377~391	寫入 address 的程序
394	設置 SDIO 爲輸出模式
397~407	讀出資料
408	SCS=0，清除控制暫存器讀寫功能
409	回傳 1 byte 的讀值

```
412  /*********************************************************************
413   **  ByteSend
414   *********************************************************************/
415  void ByteSend(Uint8 src)
416  {
417      Uint8 i;
418
419      for(i = 0; i < 8; i++)
420      {
421          if(src & 0x80)
422              SDIO = 1;
423          else
424              SDIO = 0;
425
426          _nop_();
427          SCK = 1;
428          _nop_();
429          SCK = 0;
430          src = src << 1;
431      }
432  }
```

功能說明：寫入 1 byte 的程序	
行數	說明
419~431	寫入 1 個 byte 的程序

```
434  /*********************************************************************
435   **  ByteRead
436   *********************************************************************/
437  Uint8 ByteRead(void)
438  {
439      Uint8 i,tmp;
440
441      SDIO = 1; //sdio pull high
442      for(i = 0; i < 8; i++)
443      {
444          if(SDIO)
445              tmp = (tmp << 1) | 0x01;
446          else
447              tmp = tmp << 1;
448
449          SCK = 1;
450          _nop_();
451          SCK = 0;
452      }
453      return tmp;
454  }
```

功能說明：讀出 1byte 的程序	
行數	說明
441~452	讀出 1 個 byte 的程序
453	返回 8 bit 的讀值

```
456  /*******************************************************************************
 457   **  Send4Bit
 458   *******************************************************************************/
 459  void Send4Bit(Uint8 src)
 460  {
 461      Uint8 i;
 462
 463      for(i = 0; i < 4; i++)
 464      {
 465          if(src & 0x80)
 466              SDIO = 1;
 467          else
 468              SDIO = 0;
 469
 470          _nop_();
 471          SCK = 1;
 472          _nop_();
 473          SCK = 0;
 474          src = src << 1;
 475      }
 476  }
```

功能說明：寫入 4 bit 的程序

行數	說明
463~475	寫入 1 個 byte 的程序

```
478  /*******************************************************************************
 479   ** SetCH
 480   *******************************************************************************/
 481  void SetCH(Uint8 ch)
 482  {
 483      //RF freq = RFbase + (CH_Step * ch)
 484      A7105_WriteReg(PLL1_REG, ch);
 485  }
```

功能說明：設置頻道的程序

行數	說明
484	呼叫副程式 A7105_WriteReg，對 PLL1 控制暫存器寫入工作頻道值。

```
/*********************************************************************
 ** initRF
 *********************************************************************/
void initRF(void)
{
    //init io pin
    SCS = 1;
    SCK = 0;
    SDIO = 1;
    CKO = 1;
    GIO1 = 1;
    GIO2 = 1;

    A7105_Reset(); //reset A7105 RF chip
    A7105_WriteID(); //write ID code
    A7105_Config(); //config A7105 chip
    A7105_Cal(); //calibration IF,vco, vcoc
}
```

功能說明：初始化 Master 端的 RF chip

行數	說明
493~498	設置 RF chip 介面 I/O 初始值
500	呼叫副程式 A7105_Reset，重置 RF chip
501	呼叫副程式 A7105_WriteID，寫入 ID code 4ytes
502	呼叫副程式 A7105_Config，初始 RF 端的控制暫存器
503	呼叫副程式 A7105_Cal，IF, VCO, VCO current 的校準程序

```
/*********************************************************************
 ** A7105_WriteFIFO
 *********************************************************************/
void A7105_WriteFIFO(void)
{
    Uint8 i;
    Uint8 cmd;

    cmd = FIFO_REG; //send address 0x05, bit cmd=0, r/w=0
    SCS=0;
    ByteSend(cmd);
    for(i=0; i <64; i++)
        ByteSend(PN9_Tab[i]);
    SCS=1;
}
```

功能說明：Tx FIFO 寫入資料的程序

行數	說明
514	將 FIFO 控制暫存器位址與 cmd bit, r/w bit 作運算，寫入 TX FIFO 控制暫存器命令
515	SCS=0，致能控制暫存器讀寫功能
516	送出 TX FIFO 寫入命令
517~518	寫入 64 bytes 的資料
519	SCS=1，清除控制暫存器讀寫功能

```
522  /*********************************************************************
523   ** A7105_WriteFIFO_1
524   *********************************************************************/
525  void A7105_WriteFIFO_1(void)
526  {
527      Uint8 i;
528      Uint8 cmd;
529
530      cmd = FIFO_REG; //send address 0x05, bit cmd=0, r/w=0
531      SCS=0;
532      ByteSend(cmd);
533      for(i=0; i <64; i++)
534          ByteSend(~PN9_Tab[i]);
535      SCS=1;
536  }
```

功能說明：Tx FIFO 寫入資料的程序

行數	說明
530	將 FIFO 控制暫存器位址與 cmd bit, r/w bit 作運算，寫入 TX FIFO 控制暫存器命令
531	SCS=0，致能控制暫存器讀寫功能
532	送出 TX FIFO 寫入命令
533~534	寫入 64 bytes 的資料
535	SCS=1，清除控制暫存器讀寫功能

```
538  /*********************************************************************
539   ** Strobe Command
540   *********************************************************************/
541  void StrobeCmd(Uint8 cmd)
542  {
543      SCS = 0;
544      Send4Bit(cmd);
545      SCS = 1;
546  }
547
```

功能說明：Strobe 命令寫入的程序。

行數	說明
543	SCS=0，致能控制暫存器讀寫功能
544	呼叫副程式 Send4Bit，將控制指令寫入
545	SCS=1，清除控制暫存器讀寫功能

```
548  /*********************************************************************
549   ** RxPacket
550   *********************************************************************/
551  void RxPacket(void)
552  {
553      Uint8 i;
554      Uint8 recv;
555      Uint8 tmp;
556      RxCnt++;
557      for(i=0; i <64; i++)
558      {
559          recv = tmpbuf[i];
560          if((recv ^ PN9_Tab[i])!=0)
561          {
562              tmp = recv ^ PN9_Tab[i];
563              Err_BitCnt += (BitCount_Tab[tmp>>4] + BitCount_Tab[tmp & 0x0F]);
564          }
565      }
566
567      for(i=0; i <64; i++)
568      {
569          recv = tmpbuf[i+64];
570          if((recv ^ (PN9_Tab[i]+1))!=0)
571          {
572              tmp = recv ^ (~PN9_Tab[i]);
573              Err_BitCnt += (BitCount_Tab[tmp>>4] + BitCount_Tab[tmp & 0x0F]);
574          }
575      }
576  }
```

功能說明：從 RX FIFO 讀出資料，比對資料的程序

行數	說明
556	變數 RxCnt 加 1
557~565	將 tmpbuf[0]~tmpbuf[63]讀出，比較 data 的正確性，計算出 error bit。
567~575	將 tmpbuf[64]~tmpbuf[127]讀出，比較 data 的正確性，計算出 error bit。

```
578  /*********************************************************************
579   ** Err_State
580   *********************************************************************/
581  void Err_State(void)
582  {
583      //ERR display
584      //Error Proc...
585      //...
586  }
```

功能說明：Error State 處理程序

583~584	Error 處理程序，使用者自訂

```
588  /*********************************************************************
589   **  WaitBit_0
590   *********************************************************************/
591   void WaitBit_0(Uint8 reg, Uint8 nbit)
592   {
593       do {
594       } while(A7105_ReadReg(reg) & nbit);
595   }
```

功能說明：WaitBit 程序	
593~594	呼叫 A7105_ReadReg 副程式，讀取該暫存器中，判定某 bit 值是否清爲 0.

```
597  /*********************************************************************
598   ** SelVCOBand
599   *********************************************************************/
600   void SelVCOBand(Uint8 vb1, Uint8 vb2)
601   {
602       Uint8 diff1,diff2;
603
604       if (vb1>=4)
605           diff1 = vb1-4;
606       else
607           diff1 = 4-vb1;
608
609       if (vb2>=4)
610           diff2 = vb2-4;
611       else
612           diff2 = 4-vb2;
613
614       if (diff1 == diff2 || diff1 > diff2)
615           A7105_WriteReg(VCOCAL1_REG, (vb1 | 0x08));//manual setting vb1 value
616       else
617           A7105_WriteReg(VCOCAL1_REG, (vb2 | 0x08));//manual setting vb2 value
618   }
```

功能說明：Select VCOBand 處理程序	
604~607	決定變數 vb1 值與 band 4 的差值.
609~612	決定變數 vb2 值與 band 4 的差值.
614~617	判別 diff1 與 diff2 值是否相等或是找出與 band 4 相差較大的 band 值，並將該 band 值以手動設定 vco band 值.

```
620  /*********************************************************************
621   ** calibration
622   *********************************************************************/
623   void A7105_Cal(void)
624   {
625       Uint8 tmp;
626       Uint8 fb,fbcf;
627       Uint8 vb1,vbcf1,dvt1;
628       Uint8 vb2,vbcf2,dvt2;
629
630       StrobeCmd(CMD_STBY); //calibration @STB state
631
632       //calibration IF procedure
633       A7105_WriteReg(CALIBRATION_REG, 0x01);
634       WaitBit_0(CALIBRATION_REG, 0x01);
635
636       //for check
637       tmp = A7105_ReadReg(IFCAL1_REG);
638       fb = tmp & 0x0F;
639       fbcf = (tmp >>4) & 0x01;
640
641       if (fbcf ==1)
642       {
643           Err_State();
644           while(1);
645       }
646
647           //calibration vco procedure
648           A7105_WriteReg(VCOCCAL_REG, 0x13); //manual VCOC=3
649           A7105_WriteReg(VCOCAL2_REG, 0x3B); //VTL=3, VTH=7
650
651           SetCH(0); //setting 2400MHz
652           A7105_WriteReg(CALIBRATION_REG, 0x02);
653           WaitBit_0(CALIBRATION_REG, 0x02);
654
655           tmp = A7105_ReadReg(VCOCAL1_REG);
656           vb1 = tmp & 0x07;
657          vbcf1 = (tmp >> 3) & 0x01;
658          dvt1 = (tmp >> 4) & 0x03;
659
660          SetCH(160); //setting 2480MHz
661          A7105_WriteReg(CALIBRATION_REG, 0x02);
662          WaitBit_0(CALIBRATION_REG, 0x02);
663
664          tmp = A7105_ReadReg(VCOCAL1_REG);
665          vb2 = tmp & 0x07;
666          vbcf2 = (tmp >> 3) & 0x01;
667          dvt2 = (tmp >> 4) & 0x03;
668
669          SelVCOBand(vb1, vb2);
670
671          if (vbcf1==1 && vbcf2==1)
672          {
673              Err_State();
674              while(1);
675          }
676   }
```

功能說明：IF, VCO, VCO current 校準程序

行數	說明
630	使用 Strobe 命令，設置 RF chip 進入 Standby state
633	設置 calibration control register 中 bit FBC=1。

634	讀出 calibration control register，並判別 bit FBC 是否爲 0。如爲 0，則跳出等待迴圈
637~639	讀出 IF Calibration I 控制暫存器，並檢示其值
641~645	判別 fbcf 旗標是否爲 0。如不爲 0，則判定 RF chip 爲 fail。
648	設置 VCO current Band 爲 3。
649	設置 VTH=7, VTL=3。
651	設置頻率在 2400MHz。
652	設置 calibration control register 中 bit VBC=1。
653	讀出 calibration control register，並判別 bit VBC 是否爲 0。如爲 0，則跳出等待迴圈
655~658	讀出 IF Calibration I 控制暫存器，並檢示其值
660	設置頻率在 2480MHz。
661	設置 calibration control register 中 bit VBC=1。
662	讀出 calibration control register，並判別 bit VBC 是否爲 0。如爲 0，則跳出等待迴圈
664~667	讀出 IF Calibration I 控制暫存器，並檢示其值
669	呼叫副程式 SelVCOBand，設置最佳 VCO band 值
671~675	判別 vbcf1,vbcf2 是否爲 1。如爲 1，則判定 fail.

```
678  /*********************************************************************
679   ** A7105_Config
680   *********************************************************************/
681  void A7105_Config(void)
682  {
683      Uint8 i;
684
685      //0x00 mode register, for reset
686      //0x05 fifo data register
687      //0x06 id code register
688      //0x23 IF calibration II, only read
689      //0x32 filter test register
690
691      for (i=0x01; i<=0x04; i++)
692          A7105_WriteReg(i, A7105Config[i]);
693
694      for (i=0x07; i<=0x22; i++)
695          A7105_WriteReg(i, A7105Config[i]);
696
697      for (i=0x24; i<=0x31; i++)
698          A7105_WriteReg(i, A7105Config[i]);
699  }
```

功能說明：初始 RF chip Master 端的程序

行數	說明
691~692	呼叫副程式 A7105_WriteReg，寫入控制暫存器位址 0x01~0x04
694~695	呼叫副程式 A7105_WriteReg，寫入控制暫存器位址 0x07~0x22
697~698	呼叫副程式 A7105_WriteReg，寫入控制暫存器位址 0x24~0x31

```
701  /*********************************************************************
702   ** Swap
703   *********************************************************************/
704  void Swap(Uint8 idx1, Uint8 idx2)
705  {
706      Uint16 tmp;
707
708      tmp= AFCBuf[idx1];
709      AFCBuf[idx1] = AFCBuf[idx2];
710      AFCBuf[idx2]= tmp;
711  }
```

功能說明：二個暫存器值互換的程序

行數	說明
708~710	將二個暫存器值互換

```
713  /*********************************************************************
714   ** FrequencyCal
715   *********************************************************************/
716  void FrequencyCal(void)
717  {
718      Uint8 tmp_h, tmp_l;
719      Uint32 tmp;
720      Uint16 tmp1,tmp2;
721      Uint8 i,j;
722      Uint8 cnt;
723      Uint8 flag_1n,flag_2n;
724      Uint8 pll3,pll4,pll5;
725      Uint16 offsetValue,pll;
726      Uint8 sign;
727
728      //Step1. enable AFC bit
729      A7105_WriteReg(RX_REG, 0x72);
730
731      //Step2. receive data with sync
732      cnt=0;
733      while(1)
734      {
735          StrobeCmd(CMD_RX);
736          while(GIO1);
737
738          cnt++;
739          if (cnt>3)//ignore AFCcnt 1~3
740          {
741              tmp_h = A7105_ReadReg(PLL4_REG) & 0x7F;
742              tmp_l = A7105_ReadReg(PLL5_REG);
743              tmp1 = (Uint16)tmp_h * 256 + (Uint16)tmp_l;
744              AFCBuf[cnt-4]=tmp1;
745          }
746
747          if (cnt>=15)
748              break;
749
750          Delay1ms(1);
751      }
```

```
753     //Step3. bubble sort
754     for(i = 0; i < 12-1; i++)
755     {
756         for(j = i+1; j < 12; j++)
757         {
758             if(AFCBuf[i] & 0x4000)
759             {
760                 flag_1n=1;
761                 tmp1= ~AFCBuf[i] & 0x7FFF;
762             }
763             else
764             {
765                 flag_1n=0;
766                 tmp1= AFCBuf[i];
767             }
768
769             if(AFCBuf[j] & 0x4000)
770             {
771                 flag_2n=1;
772                 tmp2= ~AFCBuf[j] & 0x7FFF;
773             }
774             else
775             {
776                 flag_2n=0;
777                 tmp2= AFCBuf[j];
778             }
779
780             if (flag_1n==0 && flag_2n==0)
781             {
782                 if (tmp1 > tmp2)
783                     Swap(i, j);
784             }
785
786             if(flag_1n==1 && flag_2n==1)
787             {
788                 if (tmp1 < tmp2)
789                     Swap(i, j);
790             }
791
792             if (flag_1n==0 && flag_2n==1)
793                 Swap(i, j);
794         }
795     }
796
797     //Step4. ignore AFCBuf[0~1], AFCBuf[10~11], average AFCBuf[2~9]
798     //Plus, sign operation
799     tmp=0;
800     for(i=2; i<=9; i++)
801     {
802         if (AFCBuf[i] & 0x4000)
803             tmp = tmp + (AFCBuf[i] & 0x3FFF);
804         else
805             tmp = tmp + AFCBuf[i] + 0x4000;
806     }
```

```
808     tmp = tmp /8;
 809     if (tmp & 0x4000)
 810         offsetValue = tmp & 0x3FFF;
 811     else
 812         offsetValue = tmp | 0x4000;
 813
 814     //Step5. disable AFC bit
 815     A7105_WriteReg(RX_REG, 0x62);
 816
 817     //Step6. update PLL register
 818     pll3 = A7105Config[PLL3_REG];
 819     pll4 = A7105Config[PLL4_REG];
 820     pll5 = A7105Config[PLL5_REG];
 821
 822     pll = (Uint16)pll4*8 + (Uint16)pll5;
 823
 824     if (offsetValue & 0x4000)
 825     {
 826         sign=1;
 827         offsetValue = (~offsetValue + 1) & 0x7FFF;
 828     }
 829     else
 830     {
 831         sign=0;
 832     }
 833
 834     if (sign)
 835     {
 836         pll = pll - offsetValue;
 837         if (CY)
 838             A7105_WriteReg(PLL3_REG, pll3-1);
 839
 840         A7105_WriteReg(PLL4_REG, pll>>8);
 841         A7105_WriteReg(PLL5_REG, pll);
 842     }
 843     else
 844     {
 845         pll = pll + offsetValue;
 846         if (CY)
 847             A7105_WriteReg(PLL3_REG, pll3+1);
 848
 849         A7105_WriteReg(PLL4_REG, pll>>8);
 850         A7105_WriteReg(PLL5_REG, pll);
 851     }
 852 }
```

功能說明：頻率偏移修正程序
** a) 抓取 15 次 Master 端 TX 頻率偏差值。
** b) 刪除前 3 次偏差值。
** c) 將偏移值做大小排序,刪除最大、最小偏差值各二筆。
** d) 計算 8 次平均偏差值。
** e)修正設定 Slave 端 PLL control register 的值。

行數	說明
729	開啓 bit AFC=1
732	清除變數 cnt =0
733~745	接收 15 次，在有 RX sync 動作時，記錄其 compensate 的值
735	使用 Strobe command，進入 Rx mode 模式
736	等待 GIO1 爲 0
738	變數 cnt 加 1
739	判斷變數 cnt 是否大於 3。如不是，則忽略不運算。

741	讀取控制暫存器 PLL4 的值，將值存入變數 temp_h
742	讀取控制暫存器 PLL5 的值，將值存入變數 temp_l
743	計算 tmp1 值
744	將所計算的值，存入 AFCBuf[xx]中。
747	判斷變數 cnt 是否爲 15。如果是，則離開這迴圈。
750	呼叫副程式 Delay1ms，延遲 1ms。
754~795	對 AFCBuf[]做大小排序。
799	清除變數 tmp=0
800~806	對 AFCBuf[2]~AFCBuf[9]做有號數的相加
808~812	變數 tmp 除 8，做平均值，計算出頻率偏移值。
815	關閉 bit AFC=0
818	讀取初始 PLL3 的值，存入變數 pll3
819	讀取初始 PLL4 的值，存入變數 pll4
820	讀取初始 PLL5 的值，存入變數 pll5
822	計算變數 pll 值，即 RF chip 中的 FP 值。
824~832	判斷變數 offsetValue 的值是正向偏移或是負向偏移。
835~842	如 offsetValue 爲負向偏移，則做控制暫存器 PLL3, PLL4,PLL5 值的修正
844~851	如 offsetValue 爲正向偏移，則做控制暫存器 PLL3, PLL4,PLL5 值的修正

```
854  /*********************************************************************
855   ** ReadDataToBuf
856   *********************************************************************/
857   void ReadDataToBuf(Uint8 *dest)
858   {
859       Uint8 i;
860       Uint8 recv;
861       Uint8 cmd;
862
863
864       cmd = FIFO_REG | 0x40; //address 0x05, bit cmd=0, r/w=1
865
866       SCS=0;
867       ByteSend(cmd);
868       for(i=0; i <64; i++)
869       {
870           recv = ByteRead();
871           *dest++ = recv;
872       }
873       SCS=1;
874   }
```

功能說明：讀資料到 tmpbuf 的程序

行數	說明
864	將 FIFO 控制暫存器位址與 cmd bit, r/w bit 作運算，讀出 RX FIFO 控制暫存器命令
866	SCS=0，致能控制暫存器讀寫功能
867	呼叫副程式 ByteSend，送出控制命令
868~872	讀出共 64 bytes 的值，並存入變數 tmpbuf
873	SCS=1，清除控制暫存器讀寫功能