Document Title

A7105 reference code for FIFO mode

Revision History

Rev. No.	History	Issue Date	Remark
0.0	Preliminary	Oct 15 , 2007	
0.1	Preliminary	Mar. 6, 2008	
0.2	Modify RF config setting	Apr. 9, 2008	
0.3	Add hopping function, modify calibration procedure	Jun, 11, 2008	

Table of contents

1. 簡介.....3
2. 系統概述.....3
3. 硬體.....4
3.1 系統方塊圖.....4
3.2 RF線路圖(Master & Slave).....5
4. 韌體程式設計:.....6
4.1 應用範例概述.....6
4.2 範例程式工作基本方塊.....7
5. 程式說明.....8

RF Chip-A7105
Reference code for FIFO mode

1. 簡介

這文件係對 RF chip -A7105 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

本範例程式使用簡單的跳頻(frequency hopping)機制，時序如下圖：

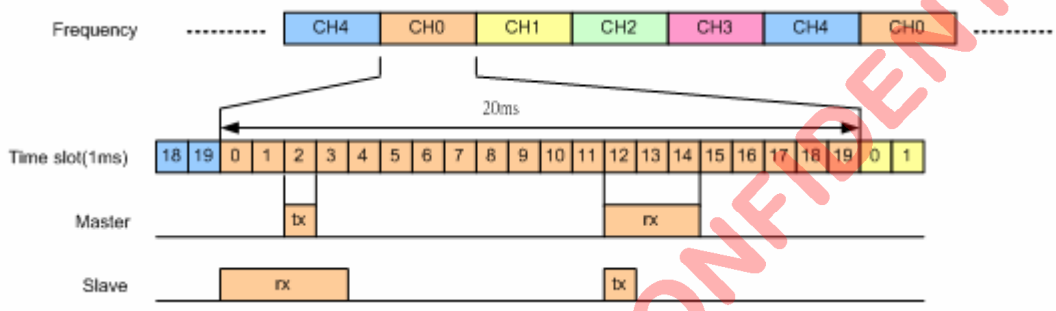

Fig1. 跳頻機制時序圖

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，等待 time slot=2 時，進入 TX 狀態，傳送 64 bytes 資料。之後等待 time slot=12 時，再進入 RX 狀態，等待接收。如收到資料，會自動改變下一次的工作頻率序列，重新另一次的時序週期動作。若未收到資料，Master 端會自動改變下一次的工作頻率序列，重新另一次的時序週期動作。

Slave 端：power on、initial 系統及 RF chip 後，等待 time slot=0 時，進入 RX 狀態等待接收。若無收到 Master 端所發送的資料，則會自動改變下一次的工作頻率序列，等待下一次 time slot=0 的時序週期動作。若仍未收到資料有 5 次時序週期，則停止跳頻機制，並回到初始工作頻率，進入 RX 狀態等待接收。若有收到 Master 端所發送的資料，則重新啓動跳頻機制，依時序完成 TX 及 RX 工作。

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。

3. 硬體

3.1 系統方塊圖

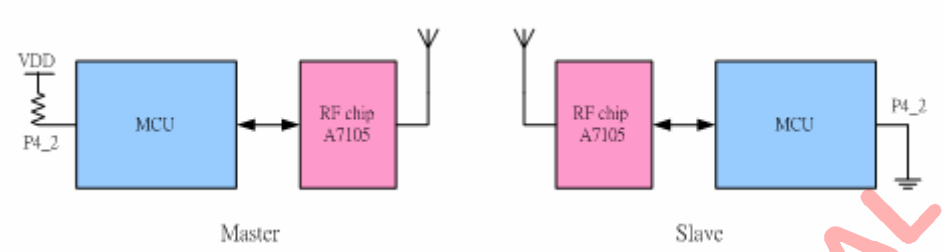

Fig2. 系統方塊圖

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。
GPIO1 - 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 A7105 RF chip 的 I/O 配置如下圖：

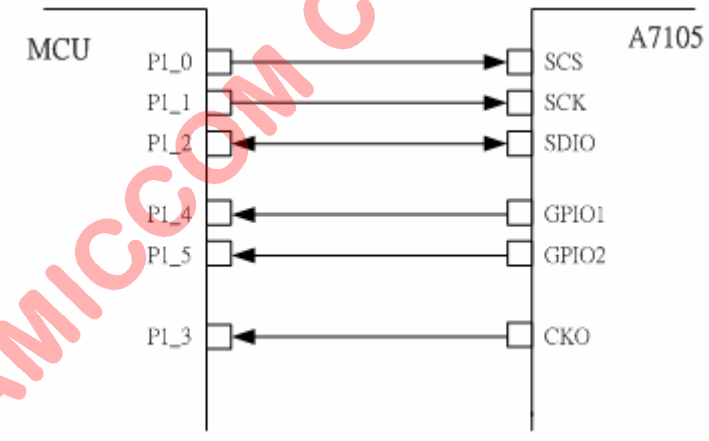

Fig3. I/O 配置圖

3.2 RF 線路圖(Master & Slave)

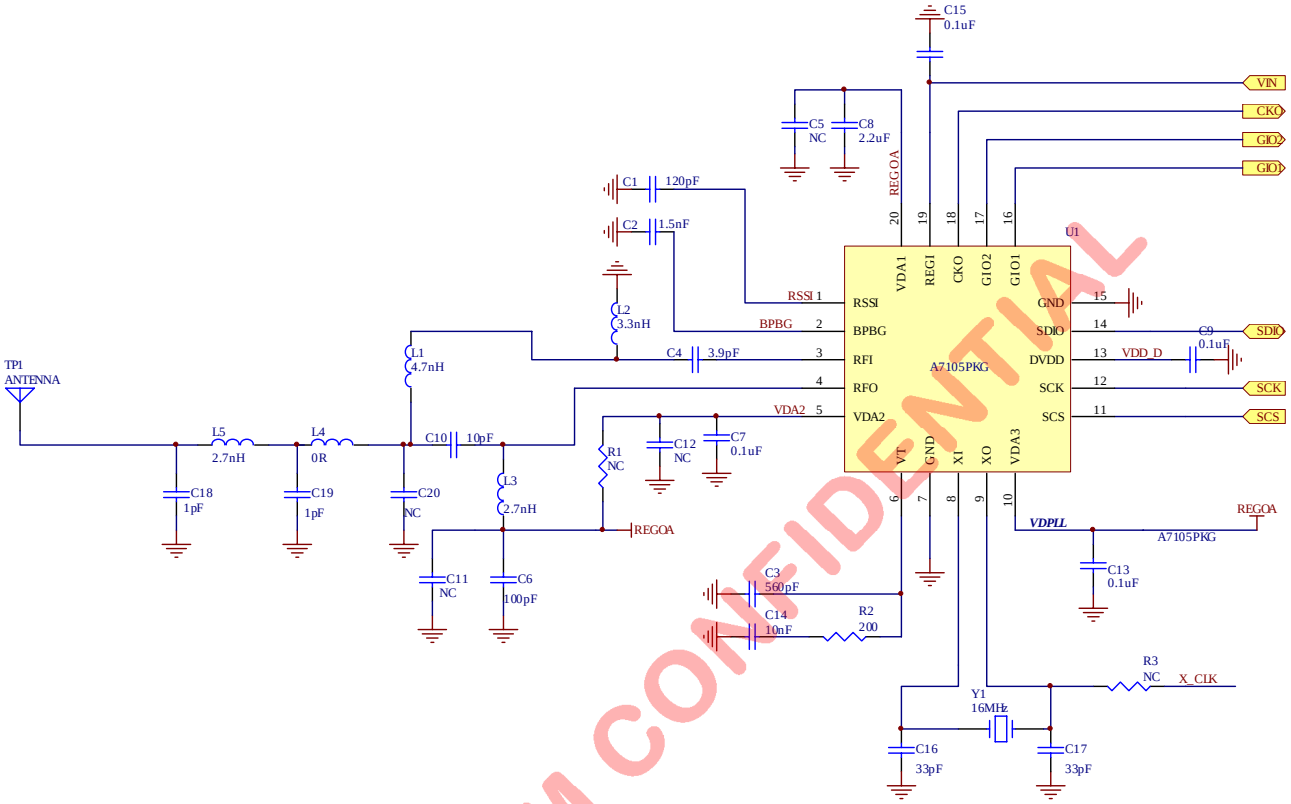

Fig4.RF 線路圖

4. 韌體程式設計:

4.1 **應用範例概述**

首先初始化 Timer0、Uart0 及 A7105RF chip，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) 等待 timer=2。
2) 進入 TX state，傳送封包。完成傳送後，RF chip 會自動結束 TX state，回復到 Standby state。
3) 等待 timer=12 時，進入 RX 狀態。
4) 進入 RX 狀態。
5) 如 Time>14 時，仍未收到資料，則結束 RX 狀態。變數 seq 值加 1。回到 Step 1 動作，開始下一周期時序(另一工作頻率)的傳送。
6) 如收到封包後，RF chip 會自動結束 RX state，回復到 Standby state。
7) 從 RX FIFO 讀出，並比較 PN9 code 共 64bytes，並計算 error bit 數目。
8) 變數 seq 值加 1，重新回到 Step 1 動作，重新開始下一周期時序工作。
9) 每 500ms，將所計算的 error bit 傳送至 PC。

Slave 端：

1) 等待 timer=0。
2) 進入 RX 狀態，等待封包收到。
3) 如 timer>3 時，仍未收到資料，則結束 RX 狀態。變數 seq 值加 1，變數 Err_HopCnt 值加 1。
4) 如 Err_HopCnt 值沒有大於 5 次，則重新回到 Step 1 動作，重新開始下一周期時序工作。
5) 如 Err_HopCnt 值大於 5 次，則清除變數 seq 值爲 0，及 Err_HopCnt 值爲 0。使用 seq=0 的工作頻率進入 RX state，等待封包收到。
6) 如收到封包後，RF chip 會自動結束 RX state，回復到 Standby state。
7) 從 RX FIFO 讀出，並比較 PN9 code 共 64bytes，計算 error bit 數目。
8) 等待 timer=12 時，進入 TX 狀態，傳送封包。完成傳送後，RF chip 會自動結束 TX state，回復到 Standby state。
9) 重新回到 Step 1 動作，重新開始下一周期時序工作。
10) 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

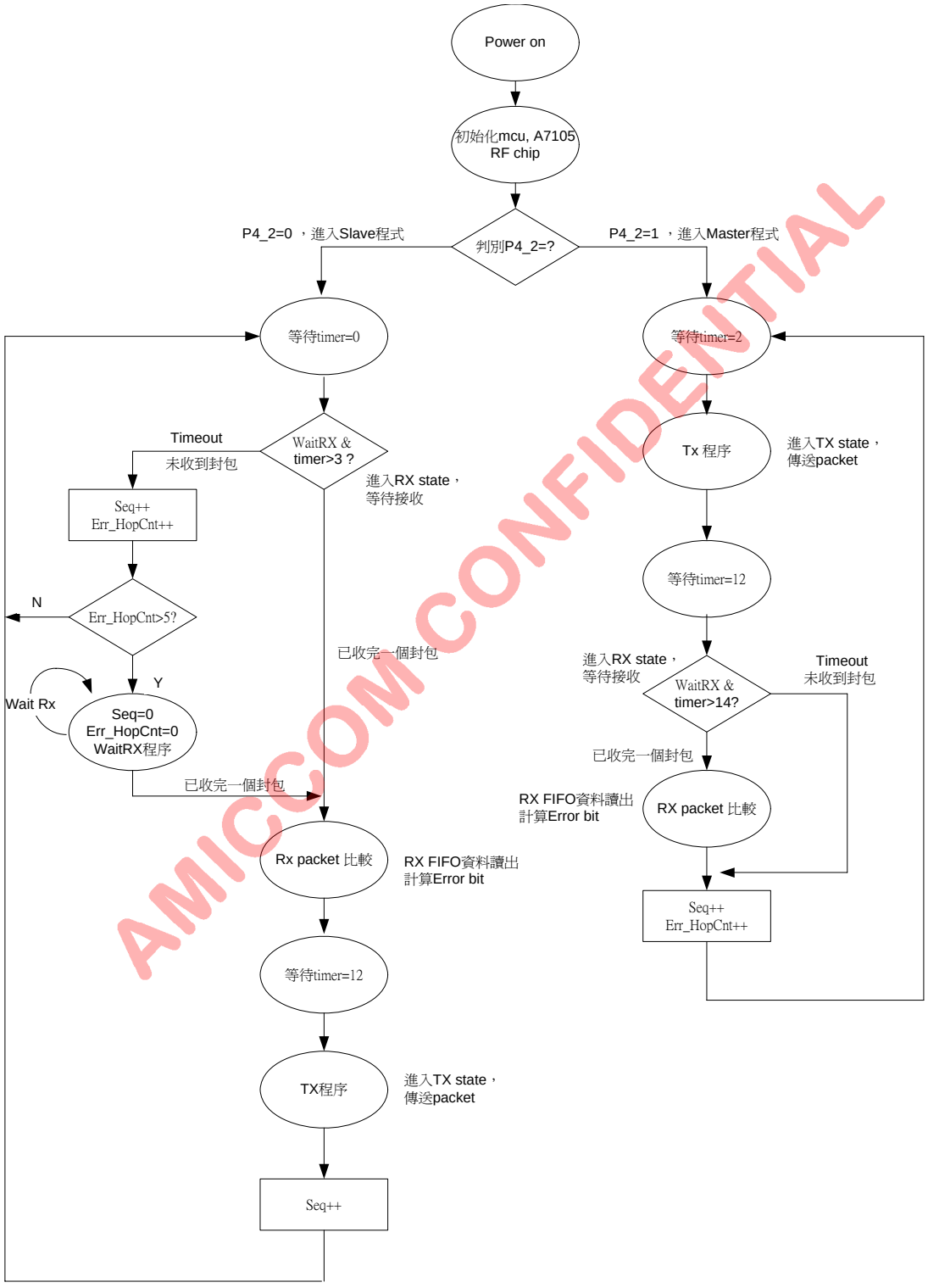

Fig5. 範例程式工作基本方塊

5. 程式說明

```
 1  /*********************************************************************
 2  ** Device:     A7105
 3  ** File:       main.c
 4  ** Author:     JPH
 5  ** Target:     Winbond W77LE58
 6  ** Tools:      ICE
 7  ** Created:    2008-06-11
 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 GPIO1         P1_4          //GPIO1
27  #define GPIO2         P1_5          //GPIO2
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  Uint16      idata   RxCnt;
 41  Uint32      idata   Err_ByteCnt;
 42  Uint32      idata   Err_BitCnt;
 43  Uint16      idata   TimerCnt0;
 44  Uint8       data    *Uartptr;
 45  Uint8       data    UartSendCnt;
 46  Uint8       data    CmdBuf[12];
 47  Uint8       xdata   tmpbuf[64];
 48  Uint8       idata   Err_Frame;
 49  Uint8       data    Seq;
 50  Uint8       data    Err_HopCnt;
 51
 52  const Uint8 code BitCount_Tab[16] = {0,1,1,2,1,2,2,3,1,2,2,3,2,3,3,4};
 53  const Uint8 code ID_Tab[4]={0x54,0x75,0xC5,0x2A}; //ID code
 54  const Uint8 code PN9_Tab[]=
 55  {   0xFF,0x83,0xDF,0x17,0x32,0x09,0x4E,0xD1,
 56      0xE7,0xCD,0x8A,0x91,0xC6,0xD5,0xC4,0xC4,
 57      0x40,0x21,0x18,0x4E,0x55,0x86,0xF4,0xDC,
 58      0x8A,0x15,0xA7,0xEC,0x92,0xDF,0x93,0x53,
 59      0x30,0x18,0xCA,0x34,0xBF,0xA2,0xC7,0x59,
 60      0x67,0x8F,0xBA,0x0D,0x6D,0xD8,0x2D,0x7D,
 61      0x54,0x0A,0x57,0x97,0x70,0x39,0xD2,0x7A,
 62      0xEA,0x24,0x33,0x85,0xED,0x9A,0x1D,0xE0
 63  };      // 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
39~50	程式中使用的變數宣告
52	BitCount_Tab 宣告
53	ID code 宣告
54~62	PN9 data 宣告

```
65   const Uint16 code A7105Config[]=
 66  {
 67          0x00, //RESET register,         only reset, not use on config
 68          0x42, //MODE register,
 69          0x00, //CALIBRATION register,only read, not use on config
 70          0x3F, //FIFO1 register,
 71          0x00, //FIFO2 register,
 72          0x00, //FIFO register,          for fifo read/write
 73          0x00, //IDDATA register,        for idcode
 74          0x00, //RCOSC1 register,
 75          0x00, //RCOSC2 register,
 76          0x00, //RCOSC3 register,
 77          0x00, //CKO register,
 78          0x01, //GPIO1 register
 79          0x21, //GPIO2 register,
 80          0x05, //CLOCK register,
 81          0x00, //DATARATE register,
 82          0x50, //PLL1 register,
 83          0x9E, //PLL2 register,          RFbase 2400MHz
 84          0x4B, //PLL3 register,
 85          0x00, //PLL4 register,
 86          0x02, //PLL5 register,
 87          0x16, //TX1 register,
 88          0x2B, //TX2 register,
 89          0x12, //DELAY1 register,
 90          0x00, //DELAY2 register,
 91          0x62, //RX register,
 92          0x80, //RXGAIN1 register,
 93          0x80, //RXGAIN2 register,
 94          0x00, //RXGAIN3 register,
 95          0x0A, //RXGAIN4 register,
 96          0x32, //RSSI register,
 97          0xC3, //ADC register,
 98          0x07, //CODE1 register,
 99          0x16, //CODE2 register,
100          0x00, //CODE3 register,
101          0x00, //IFCAL1 register,
102          0x00, //IFCAL2 register,    only read
103          0x00, //VCOCCAL register,
104          0x00, //VCOCAL1 register,
105          0x3B, //VCOCAL2 register,
106          0x00, //BATTERY register,
107          0x17, //TXTEST register,
108          0x47, //RXDEM1 register,
109          0x80, //RXDEM2 register,
110          0x03, //CPC register,
111          0x01, //CRYSTAL register,
112          0x45, //PLLTEST register,
113          0x18, //VCOTEST1 register,
114          0x00, //VCOTEST2 register,
115          0x01, //IFAT register,
116          0x0F, //RSCALE register,
117          0x00  //FILTERTEST
118   };
```

功能說明： RF chip 初始設定

行數	說明
67~117	RF chip 的初始設定

```
120   const Uint8 HopTab[]=
 121  {
 122      20, //2410
 123      40, //2420
 124      80, //2440
 125      120,//2460
 126      160 //2480
 127  };
```

功能說明：Hopping table 宣告

行數	說明
122~126	自行定義 5 個的 channel.

```
129   /**********************************************************************
 130   **  function Declaration
 131   **********************************************************************/
 132   void InitTimer0(void);
 133   void initUart0(void);
 134   void Timer0ISR (void);
 135   void Uart0Isr(void);
 136   void A7105_Reset(void);
 137   void A7105_WriteReg(Uint8, Uint8);
 138   Uint8 A7105_ReadReg(Uint8);
 139   void ByteSend(Uint8 src);
 140   Uint8 ByteRead(void);
 141   void A7105_WriteID(void);
 142   void A7105_WriteFIFO(void);
 143   void initRF(void);
 144   void A7105_Config(void);
 145   void A7105_Cal(void);
 146   void RxPacket(void);
 147   void StrobeCmd(Uint8);
 148   void SetCH(Uint8);
 149   void WaitBit_0(Uint8, Uint8);
 150   void SelVCOBand(Uint8, Uint8);
```

功能說明：副程式檔頭宣告

行數	說明
132~150	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;
    EA=1;

    if ((P4 & 0x04)==0x04) //if P4.2=1, master
    {
        initRF();
        StrobeCmd(CMD_STBY);
        A7105_WriteFIFO(); //write data to tx fifo
        Seq=0;

        while(1)
        {
            //Tx time-slot
            while(timer != 2); //wait until timer=2
                    SetCH(HopTab[Seq]);
            StrobeCmd(CMD_TX); //entry tx & transmit
            while(GPIO1); //wait transmit completed

            //Rx time-slot
            while(timer !=12); //wait until timer=12
            SetCH(HopTab[Seq]-1);
            StrobeCmd(CMD_RX);
            while(GPIO1 && timer <= 14);

            if (timer >14)
            {
                //timeout
                StrobeCmd(CMD_PLL);
            }
            else
            {
                //data procedure
                RxPacket();
            }

            Seq++;
            if (Seq > 4)
                Seq = 0;
        }
    }
```

```
209     else //if P4.2=0, slave
210     {
211         initRF();
212         StrobeCmd(CMD_STBY);
213
214         RxCnt = 0;
215         Err_ByteCnt = 0;
216         Err_BitCnt = 0;
217
218         Seq=0;
219         Err_HopCnt=0;
220
221         while(1)
222         {
223             if (P1_7==0)
224             {
225                 RxCnt = 0;
226                 Err_BitCnt = 0;
227             }
228
229             //Rx time-slot
230             while(timer!=0);
231             SetCH(HopTab[Seq]-1);
232             StrobeCmd(CMD_RX);
233             while(GPIO1 && timer <= 3); //wait receive completed
234             if (timer > 3)
235             {
236                 StrobeCmd(CMD_PLL);
237
238                 Seq++;
239                 if (Seq > 4)
240                     Seq = 0;
241
242                 Err_HopCnt++;
243                 if (Err_HopCnt > 5)
244                 {
245                     Seq = 0;
246                     Err_HopCnt = 0;
247
248                     SetCH(HopTab[Seq]-1);
249                     StrobeCmd(CMD_RX);
250                     while(1)
251                     {
252                         if (GPIO1==0)
253                         {
254                             break;
255                         }
256                     }
257                 }
258                 else
259                 {
260                     continue;
261                 }
262             }
263
264             timer = 2; //reSync
265             TF0 = 0; // Clear Timer0 interrupt
266             TH0 = (65536-t0hrel)>>8; // Reload Timer0 high byte,low byte
267             TL0 = 65536-t0hrel;
```

```
269             RxPacket();
 270
 271             //Tx time-slot
 272             while(timer != 12);
 273             SetCH(HopTab[Seq]);
 274             StrobeCmd(CMD_TX);
 275             while(GPIO1);
 276
 277             Seq++;
 278             if (Seq > 4)
 279                 Seq = 0;
 280         }
 281     }
 282 }
```

功能說明：主程式 main loop。Port4_2=1，進入 master 迴圈，行數 162~188 爲 master 程式。Port4_2=0，進入 slave 迴圈，行數 239~261 爲 slave 程式。

行數	說明
158	啓用 MCUon chip data SRAM
161~165	初始化 MCU I/O Port
167	呼叫副程式 initTimer0，致能中斷
168	呼叫副程式 initUart0，初始 Uart0
169~170	啓動 Timer0 ，致能中斷開啓
172	判別 port4_2=1，進入 master 迴圈。Port4_2=0，進入 slave 迴圈。
173~208	Master 迴圈程式
174	呼叫副程式 initRF，初始化 A7105 chip
175	使用 Strobe command，進入 Standby mode 模式
176	呼叫副程式 A7105_WriteFIFO，將 data 寫入 TX FIFO
177	清除變數 seq=0
182	等待 timer=2
183	呼叫副程式 SetCH ，依 HopTab 查表設置工作頻率
184	使用 Strobe command，進入 TX 模式，發送資料
185	判別 I/O GPIO1 等待是否完成資料傳送
188	等待 timer=12
189	呼叫副程式 SetCH ，依 HopTab 查表設置工作頻率
190	使用 Strobe command，進入 RX 模式
191	等待是否收妥資料或是 timer>14
193~197	判別是否 timer>14。如是，則使用 Strobe command，進入 PLL state
199~202	已收妥資料，呼叫 RxPacket 副程式，從 RX FIFO 讀出資料、比對、計算 error bit 數
204~206	變數 seq 加 1，判別變數 seq 是否大於 4。如是，則清爲 0
210~281	Slave 迴圈程式
211	呼叫副程式 initRF，初始化 A7105 chip
212	使用 Strobe command，進入 Standby mode 模式
214~219	清除變數 RxCnt, Err_ByteCnt, Err_BitCnt, seq, Err_HopCnt 值
223~227	判別 MCU pin P1_7 是否爲 0。如是，則清除變數 seq, Err_HopCnt 值
230	等待 timer=0
231	呼叫副程式 SetCH ，依 HopTab 查表設置工作頻率
232	使用 Strobe command，進入 RX 模式
233	等待資料的收妥或是 timer>3
234	判別變數 timer 是否大於 3
236	使用 Strobe command，進入 PLL state
238~240	變數 seq 加 1，判別變數 seq 是否大於 4。如是，則清爲 0
242	變數 Err_HopCnt 加 1

243	判別 Err_HpoCnt 是否大於 5
245~246	清除變數 seq=0, Err_HopCnt=0
248	呼叫副程式 SetCH ，設置工作頻率
249	使用 Strobe command，進入 RX 模式
250~256	等待資料進入
260	變數 Err_HopCnt 值沒有大於 5 次時，則回到 223 行，重新另一次周期的工作
264	設置變數 timer=2，重新同步
265~267	清除 TF0 旗標，重新設置 TH0, TL0 值
269	已收妥資料，呼叫 RxPacket 副程式，從 RX FIFO 讀出資料、比對、計算 error bit 數
272	等待 timer=12
273	呼叫副程式 SetCH，設置工作頻率
274	使用 Strobe command，進入 TX 模式，發送資料
275	等待是否完成資料傳送
277~279	變數 seq 加 1，判別變數 seq 是否大於 4。如是，則清爲 0

```
284  /*********************************************************************
285   **  init Timer0
286   *********************************************************************/
287   void InitTimer0(void)
288   {
289       TR0 = 0;
290       TMOD =(TMOD & 0xF0)|0x01; //timer0 mode=1
291       TH0 = (65536-t0hrel)>>8; // Reload Timer0 high byte,low byte
292       TL0 = 65536-t0hrel;
293       TF0 = 0; // Clear any pending Timer0 interrupts
294       ET0 = 1; // Enable Timer0 interrupt
295   }
```

功能說明：初始化 Timer0 程序

行數	說明
289	關閉 Timer0 計時動作
290	設置 Timer0 在 mode 1 模式
291~292	設置 TH0,TL0 的初始值
293	清除 Timer0 中斷旗標
294	致能 Timer0 中斷

```
297  /*********************************************************************
298   **  Timer0ISR
299   *********************************************************************/
300   void Timer0ISR (void) interrupt 1
301   {
302      TF0 = 0; // Clear Timer0 interrupt
303      TH0 = (65536-t0hrel)>>8; // Reload Timer0 high byte,low byte
304      TL0 = 65536-t0hrel;
305
306      timer++;
307      if (timer>=20)
308      {
309         timer=0;
310         P3_5= ~P3_5;
311      }
312
313      TimerCnt0++;
314      if (TimerCnt0 == 500)
315      {
316         TimerCnt0 = 0;
317         CmdBuf[0] = 0xF1;
318
319         memcpy(&CmdBuf[1], &RxCnt, 2);
320         memcpy(&CmdBuf[3], &Err_ByteCnt, 4);
321         memcpy(&CmdBuf[7], &Err_BitCnt, 4);
322         memcpy(&CmdBuf[11], &Err_Frame, 1);
323
324         UartSendCnt = 12;
325         Uartptr =& CmdBuf[0];
326         SBUF = CmdBuf[0];
327      }
328   }
```

功能說明：初始化 Timer0 的中斷副程式

行數	說明
302~304	清除 Timer0 中斷旗標，設置 TH0,TL0 的啓始值
306	變數 timer 加 1
307~311	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.
313	變數 TimerCnt0 加 1
314	判別變數 TimerCnt0 是否等於 500(即 500ms)
316	清除變數 TimerCnt0
317	CmdBuf[0]設置 0xF1 爲傳送啓始位元識別碼
319	CmdBuf[1] 、CmdBuf[1]設置變數 RxCnt 的值
320	CmdBuf[3] 、CmdBuf[4]、CmdBuf[5] 、CmdBuf[6]設置變數 Err_ByteCnt 的值
321	CmdBuf[7] 、CmdBuf[8]、CmdBuf[9] 、CmdBuf[10]設置變數 Err_BitCn 的值
322	CmdBuf[11]設置變數 Err_Frame 的值
324	設置變數UartSendCnt=12
325	設置指標變數 Uartptr 指到變數 CmdBuf[0]的啓始位址
326	傳送 SBUF 至 PC

```
330  /*******************************************************************************
331   **  Init Uart0
332   *******************************************************************************/
333   void initUart0(void)
334   {
335       TH1 = 0xFD; //BaudRate 9600;
336       TL1 = 0xFD;
337       SCON = 0x40;
338       TMOD = (TMOD & 0x0F) | 0x20;
339       REN = 1;
340       TR1 = 1;
341       ES = 1;
342   }
```

功能說明：初始化 Uart0 的程序

行數	說明
335~337	初始 TL1,TH1,SCON1 值，設置爲 9600bps @xtal=11.0592MHz
338	設置 Timer1 爲 mode 2
339~341	設置 REN,TR1,ES 爲 1，啓用 Uart0 的功能

```
344  /*******************************************************************************
345   **  Uart0 ISR
346   *******************************************************************************/
347   void Uart0Isr(void) interrupt 4 using 3
348   {
349       if (TI==1)
350       {
351           TI=0;
352           UartSendCnt--;
353           if(UartSendCnt !=0)
354           {
355               Uartptr++;
356               SBUF = *Uartptr;
357           }
358       }
359   }
```

功能說明：初始化 uart0 的中斷副程式

行數	說明
349	判別 TI 旗標是否爲 Uart 已傳送完成 1byte
351	清除 TI 旗標
352	變數 UartSendCnt 減 1
353	判別變數 UartSendCnt 是否爲 0。如不爲 0，則繼續傳送下一個資料
355~356	指標變數 Uartptr 加 1，並將其位址的資料，使用 Uart0 送至 PC

```
361  /*******************************************************************************
362   **  Reset_RF
363   *******************************************************************************/
364   void A7105_Reset(void)
365   {
366       A7105_WriteReg(MODE_REG, 0x00); //reset RF chip
367   }
```

功能說明：A7105 RF chip Reset 程序

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

```
369  /*********************************************************************
370   **  WriteID
371   *********************************************************************/
372   void A7105_WriteID(void)
373   {
374       Uint8 i;
375       Uint8 d1,d2,d3,d4;
376       Uint8 addr;
377
378       addr = IDCODE_REG; //send address 0x06, bit cmd=0, r/w=0
379       SCS = 0;
380       ByteSend(addr);
381       for (i=0; i < 4; i++)
382             ByteSend(ID_Tab[i]);
383       SCS = 1;
384
385       //for check
386       addr = IDCODE_REG | 0x40; //send address 0x06, bit cmd=0, r/w=1
387       SCS=0;
388       ByteSend(addr);
389       d1=ByteRead();
390       d2=ByteRead();
391       d3=ByteRead();
392       d4=ByteRead();
393       SCS=1;
394   }
```

功能說明：寫入 ID 的程序。

行數	說明
378	計算變數 addr 的值
379	SCS=0，設置控制暫存器讀寫功能
380~382	寫入 ID 控制暫存器的位址，及 4 bytes 的 ID code
383	SCS=1，清除 SPI 讀寫功能
386	計算讀出 ID code 的變數 addr 值
387	SCS=0，設置控制暫存器讀寫功能
388~392	讀出 ID code
393	SCS=1，清除控制暫存器讀寫功能

```
396  /*********************************************************************
397   **  A7105_WriteReg
398   *********************************************************************/
399   void A7105_WriteReg(Uint8 addr, Uint8 dataByte)
400   {
401       Uint8 i;
402
403       SCS = 0;
404       addr |= 0x00; //bit cmd=0,r/w=0
405       for(i = 0; i < 8; i++)
406       {
407           if(addr & 0x80)
408               SDIO = 1;
409           else
410               SDIO = 0;
411
412           SCK = 1;
413           _nop_();
414           SCK = 0;
415           addr = addr << 1;
416       }
417       _nop_();
418
419       //send data byte
420       for(i = 0; i < 8; i++)
421       {
422           if(dataByte & 0x80)
423               SDIO = 1;
424           else
425               SDIO = 0;
426
427           SCK = 1;
428           _nop_();
429           SCK = 0;
430           dataByte = dataByte<< 1;
431       }
432       SCS = 1;
433   }
```

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

行數	說明
403	SCS=0，致能控制暫存器讀寫功能
404	將 address Or 寫入控制暫存器命令。
405~416	寫入 address 的程序
420~431	寫入 data byte 的程序
432	SCS=1，清除控制暫存器讀寫功能

```
435  /*********************************************************************
 436   **  A7105_ReadReg
 437   *********************************************************************/
 438   Uint8 A7105_ReadReg(Uint8 addr)
 439   {
 440      Uint8 i;
 441      Uint8 tmp;
 442
 443      SCS = 0;
 444      addr |= 0x40; //bit cmd=0,r/w=1
 445      for(i = 0; i < 8; i++)
 446      {
 447
 448         if(addr & 0x80)
 449            SDIO = 1;
 450         else
 451            SDIO = 0;
 452
 453               _nop_();
 454         SCK = 1;
 455         _nop_();
 456         SCK = 0;
 457
 458         addr = addr << 1;
 459      }
 460
 461      _nop_();
 462      SDIO = 1;
 463
 464      //read data
 465      for(i = 0; i < 8; i++)
 466      {
 467         if(SDIO)
 468            tmp = (tmp << 1) | 0x01;
 469         else
 470            tmp = tmp << 1;
 471
 472         SCK = 1;
 473         _nop_();
 474         SCK = 0;
 475      }
 476      SCS = 1;
 477      return tmp;
 478   }
```

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

行數	說明
443	SCS=0，致能控制暫存器讀寫功能
444	將 address Or 讀出控制暫存器命令。
445~459	寫入 address 的程序
462	設置 SDIO 爲輸出模式
465~475	讀出資料
476	SCS=0，清除控制暫存器讀寫功能
477	回傳 1 byte 的讀值

```
480  /*********************************************************************
481   **  ByteSend
482   *********************************************************************/
483   void ByteSend(Uint8 src)
484   {
485       Uint8 i;
486
487       for(i = 0; i < 8; i++)
488       {
489           if(src & 0x80)
490               SDIO = 1;
491           else
492               SDIO = 0;
493
494           _nop_();
495           SCK = 1;
496           _nop_();
497           SCK = 0;
498           src = src << 1;
499       }
500   }
```

功能說明：寫入 1 byte 的程序	
行數	說明
487~499	寫入 1 個 byte 的程序

```
502  /*********************************************************************
503   **  ByteRead
504   *********************************************************************/
505   Uint8 ByteRead(void)
506   {
507       Uint8 i,tmp;
508
509       SDIO = 1; //sdio pull high
510       for(i = 0; i < 8; i++)
511       {
512           if(SDIO)
513               tmp = (tmp << 1) | 0x01;
514           else
515               tmp = tmp << 1;
516
517           SCK = 1;
518           _nop_();
519           SCK = 0;
520       }
521       return tmp;
522   }
```

功能說明：讀出 1byte 的程序	
行數	說明
509~520	讀出 1 個 byte 的程序
521	返回 8 bit 的讀值

```
524  /*********************************************************************
525   **  Send4Bit
526   *********************************************************************/
527   void Send4Bit(Uint8 src)
528   {
529      Uint8 i;
530
531      for(i = 0; i < 4; i++)
532      {
533         if(src & 0x80)
534            SDIO = 1;
535         else
536            SDIO = 0;
537
538         _nop_();
539         SCK = 1;
540         _nop_();
541         SCK = 0;
542         src = src << 1;
543      }
544   }
```

功能說明：寫入 4 bit 的程序

行數	說明
531~543	寫入 1 個 byte 的程序

```
546  /*********************************************************************
547   ** SetCH
548   *********************************************************************/
549   void SetCH(Uint8 ch)
550   {
551      A7105_WriteReg(PLL1_REG, ch);
552   }
```

功能說明：設置頻道的程序

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

```
554  /*********************************************************************
555   ** initRF
556   *********************************************************************/
557   void initRF(void)
558   {
559       //init io pin
560       SCS = 1;
561       SCK = 0;
562       SDIO = 1;
563       CKO = 1;
564       GPIO1 = 1;
565       GPIO2 = 1;
566
567       A7105_Reset(); //reset A7105 RF chip
568       A7105_WriteID(); //write ID code
569       A7105_Config(); //config A7105 chip
570       A7105_Cal(); //calibration IF,vco, vcoc
571   }
```

功能說明：初始化 Master 端的 RF chip

行數	說明
560~565	設置 RF chip 介面 I/O 初始值
567	呼叫副程式 A7105_Reset，重置 RF chip
568	呼叫副程式 A7105_WriteID，寫入 ID code 4ytes
569	呼叫副程式 A7105_Config，初始 RF 端的控制暫存器
570	呼叫副程式 A7105_Cal，IF, VCO, VCO current 的校準程序

```
573  /*********************************************************************
574   ** A7105_WriteFIFO
575   *********************************************************************/
576   void A7105_WriteFIFO(void)
577   {
578       Uint8 i;
579       Uint8 cmd;
580
581       cmd = FIFO_REG; //send address 0x05, bit cmd=0, r/w=0
582       SCS=0;
583       ByteSend(cmd);
584       for(i=0; i <64; i++)
585           ByteSend(PN9_Tab[i]);
586       SCS=1;
587   }
```

功能說明：Tx FIFO 寫入資料的程序

行數	說明
581	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
582	SCS=0，致能控制暫存器讀寫功能
583	送出 TX FIFO 寫入命令
584~585	寫入 64 bytes 的資料
586	SCS=1，清除控制暫存器讀寫功能

```
589  /*********************************************************************
590   ** Strobe Command
591   *********************************************************************/
592   void StrobeCmd(Uint8 cmd)
593   {
594       SCS = 0;
595       Send4Bit(cmd);
596       SCS = 1;
597   }
```

功能說明：Strobe 命令寫入的程序。

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

```
599  /*********************************************************************
600   ** RxPacket
601   *********************************************************************/
602   void RxPacket(void)
603   {
604       Uint8 i;
605       Uint8 recv;
606       Uint8 tmp;
607       Uint8 cmd;
608
609       RxCnt++;
610       cmd = FIFO_REG | 0x40; //address 0x05, bit cmd=0, r/w=1
611
612       SCS=0;
613       ByteSend(cmd);
614       for(i=0; i <64; i++)
615       {
616           recv = ByteRead();
617           tmpbuf[i]=recv;
618           if((recv ^ PN9_Tab[i])!=0)
619           {
620               tmp = recv ^ PN9_Tab[i];
621               Err_BitCnt += (BitCount_Tab[tmp>>4] + BitCount_Tab[tmp & 0x0F]);
622           }
623       }
624       SCS=1;
625   }
```

功能說明：從 RX FIFO 讀出資料，比對資料的程序

行數	說明
609	變數 RxCnt 加 1
610	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
612	SCS=0，致能控制暫存器讀寫功能
613	呼叫副程式 ByteSend，送出控制命令
614~623	讀出 data ，比較 data 的正確性，計算出 error bit
624	SCS=1，清除控制暫存器讀寫功能

```
627  /*********************************************************************
628   ** Err_State
629   *********************************************************************/
630   void Err_State(void)
631   {
632       //ERR display
633       //Error Proc...
634       //...
635   }
```

功能說明：Error State 處理程序	
632~634	Error 處理程序，使用者自訂

```
637  /*********************************************************************
638   **  WaitBit_0
639   *********************************************************************/
640   void WaitBit_0(Uint8 reg, Uint8 nbit)
641   {
642       do {
643       } while(A7105_ReadReg(reg) & nbit);
644   }
```

功能說明：WaitBit 程序	
642~643	呼叫 A7105_ReadReg 副程式，讀取該暫存器中，判定某 bit 值是否清爲 0.

```
646  /*********************************************************************
647   ** SelVCOBand
648   *********************************************************************/
649   void SelVCOBand(Uint8 vb1, Uint8 vb2)
650   {
651       Uint8 diff1,diff2;
652
653       if (vb1>=4)
654           diff1 = vb1-4;
655       else
656           diff1 = 4-vb1;
657
658       if (vb2>=4)
659           diff2 = vb2-4;
660       else
661           diff2 = 4-vb2;
662
663       if (diff1 == diff2 || diff1 > diff2)
664           A7105_WriteReg(VCOCAL1_REG, (vb1 | 0x08));//manual setting vb1 value
665       else
666           A7105_WriteReg(VCOCAL1_REG, (vb2 | 0x08));//manual setting vb2 value
667   }
```

功能說明：Select VCOBand 處理程序	
653~656	決定變數 vb1 值與 band 4 的差值.
658~661	決定變數 vb2 值與 band 4 的差值.
663~666	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.

```
669  /*********************************************************************
 670   ** calibration
 671   *********************************************************************/
 672   void A7105_Cal(void)
 673   {
 674       Uint8 tmp;
 675       Uint8 fb,fbcf;
 676       Uint8 vb1,vbcf1,dvt1;
 677       Uint8 vb2,vbcf2,dvt2;
 678
 679       StrobeCmd(CMD_STBY); //calibration @STB state
 680
 681       //calibration IF procedure
 682       A7105_WriteReg(CALIBRATION_REG, 0x01);
 683       WaitBit_0(CALIBRATION_REG, 0x01);
 684
 685       //for check
 686       tmp = A7105_ReadReg(IFCAL1_REG);
 687       fb = tmp & 0x0F;
 688       fbcf = (tmp >>4) & 0x01;
 689
 690       if (fbcf ==1)
 691       {
 692           Err_State();
 693           while(1);
 694       }
 695
 696       //calibration vco procedure
 697       A7105_WriteReg(VCOCCAL_REG, 0x13); //manual VCOC=3
 698       A7105_WriteReg(VCOCAL2_REG, 0x3B); //VTL=3, VTH=7
 699
 700       SetCH(0); //setting 2400MHz
 701       A7105_WriteReg(CALIBRATION_REG, 0x02);
 702       WaitBit_0(CALIBRATION_REG, 0x02);
 703
 704       tmp = A7105_ReadReg(VCOCAL1_REG);
 705       vb1 = tmp & 0x07;
 706       vbcf1 = (tmp >> 3) & 0x01;
 707       dvt1 = (tmp >> 4) & 0x03;
 708
 709       SetCH(160); //setting 2480MHz
 710       A7105_WriteReg(CALIBRATION_REG, 0x02);
 711       WaitBit_0(CALIBRATION_REG, 0x02);
 712
 713       tmp = A7105_ReadReg(VCOCAL1_REG);
 714       vb2 = tmp & 0x07;
 715       vbcf2 = (tmp >> 3) & 0x01;
 716       dvt2 = (tmp >> 4) & 0x03;
 717
 718       SelVCOBand(vb1, vb2);
 719
 720       if (vbcf1==1 && vbcf2==1)
 721       {
 722           Err_State();
 723           while(1);
 724       }
 725   }
```

功能說明：IF, VCO, VCO current 校準程序

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

683	讀出 calibration control register，並判別 bit FBC 是否爲 0。如爲 0，則跳出等待迴圈
686~688	讀出 IF Calibration I 控制暫存器，並檢示其值
690~694	判別 fbcf 旗標是否爲 0。如不爲 0，則判定 RF chip 爲 fail。
697	設置 VCO current Band 爲 3。
698	設置 VTH=7, VTL=3。
700	設置頻率在 2400MHz。
701	設置 calibration control register 中 bit VBC=1。
702	讀出 calibration control register，並判別 bit VBC 是否爲 0。如爲 0，則跳出等待迴圈
704~707	讀出 IF Calibration I 控制暫存器，並檢示其值
709	設置頻率在 2480MHz。
710	設置 calibration control register 中 bit VBC=1。
711	讀出 calibration control register，並判別 bit VBC 是否爲 0。如爲 0，則跳出等待迴圈
713~716	讀出 IF Calibration I 控制暫存器，並檢示其值
718	呼叫副程式 SelVCOBand，設置最佳 VCO band 值
720~724	判別 vbcf1,vbcf2 是否爲 1。如爲 1，則判定 fail.

```
727  /*********************************************************************
728   ** A7105_Config
729   *********************************************************************/
730  void A7105_Config(void)
731  {
732      Uint8 i;
733
734      //0x00 mode register, for reset
735      //0x05 fifo data register
736      //0x06 id code register
737      //0x23 IF calibration II, only read
738      //0x32 filter test register
739
740      for (i=0x01; i<=0x04; i++)
741          A7105_WriteReg(i, A7105Config[i]);
742
743      for (i=0x07; i<=0x22; i++)
744          A7105_WriteReg(i, A7105Config[i]);
745
746      for (i=0x24; i<=0x31; i++)
747          A7105_WriteReg(i, A7105Config[i]);
748  }
```

功能說明：初始 RF chip Master 端的程序

行數	說明
740~741	呼叫副程式 A7105_WriteReg，寫入控制暫存器位址 0x01~0x04
743~744	呼叫副程式 A7105_WriteReg，寫入控制暫存器位址 0x07~0x22
746~747	呼叫副程式 A7105_WriteReg，寫入控制暫存器位址 0x24~0x31