



Document Title

A7125 reference code for FIFO extension mode (2Mbps)

Revision History

<u>Rev. No.</u>	<u>History</u>	<u>Issue Date</u>	<u>Remark</u>
0.0	Preliminary	Jun 9 , 2009	
0.1	Modify initial configuration value QDS=1, XCP[1:0]=[00], IGFI[2:0]=[111], IGFI[2:0]=[111], PDL[1:0]=[00], WSEL[2:0]=[011]	Apr. 22, 2009	
0.2	Modify initial configuration value GIO2=0x01	Sep. 1, 2010	

AMICCOM CONFIDENTIAL

Important Notice:

AMICCOM reserves the right to make changes to its products or to discontinue any integrated circuit product or service without notice. AMICCOM integrated circuit products are not designed, intended, authorized, or warranted to be suitable for use in life-support applications, devices or systems or other critical applications. Use of AMICCOM products in such applications is understood to be fully at the risk of the customer.

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
4. 韌體程式設計.....	5
4.1 應用範例概述.....	5
4.2 範例程式工作基本方塊.....	6
5. 程式說明.....	7

AMICCOM CONFIDENTIAL

RF Chip-A7125 Reference code for FIFO extension mode (2Mbps)

1. 簡介

這文件係對 RF chip -A7125 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，一個為 master 端，另一個為 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。

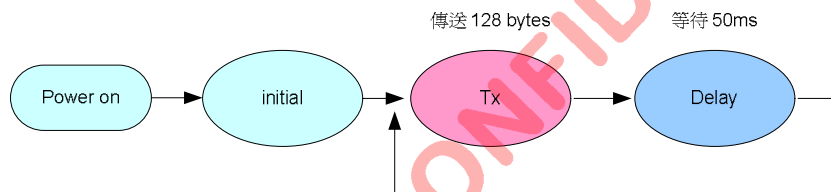


Fig1. Master 端方塊圖

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。

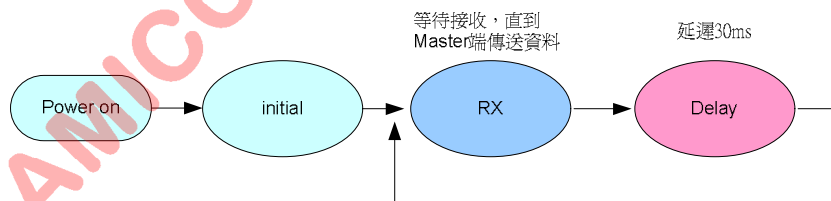


Fig2. Slave 端方塊圖

3. 硬體

3.1 系統方塊圖

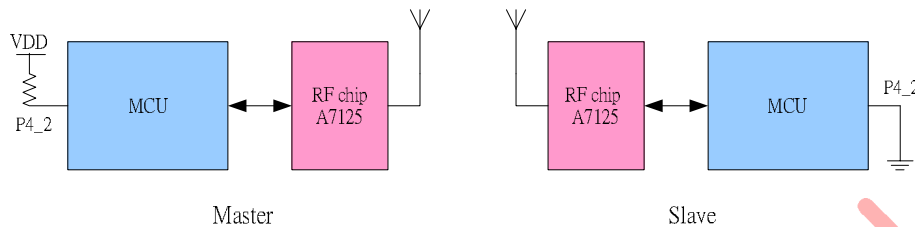


Fig3. 系統方塊圖

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 A7125 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 A7125 RF chip 的 I/O 配置如下圖：

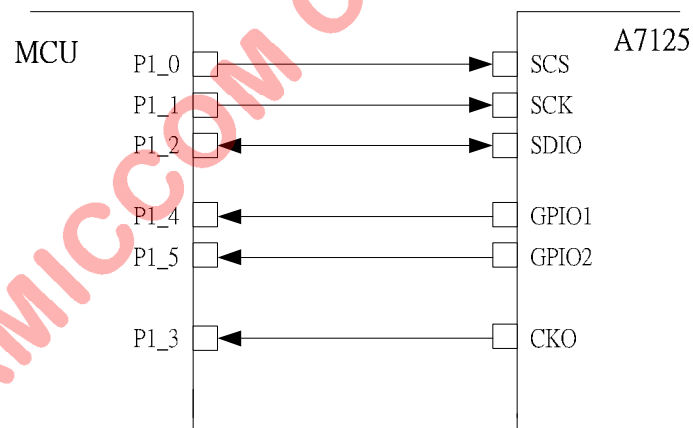


Fig4. I/O 配置圖

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 A7125RF 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 為 1 時，再對 TX FIFO 寫入 PN9 code 反向的資料，共 64 bytes。
- 5) 等待 GIO1 pin 為 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 A7125RF chip。
- 2) 進入 RX 狀態，等待封包收到。
- 3) 判斷 CKO pin 為 1 後，從 RX FIFO 讀出資料放置 tmpbuf 中
- 4) 等待 GIO1 pin 為 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。

AMICCOM CONFIDENTIAL

4.2 範例程式工作基本方塊

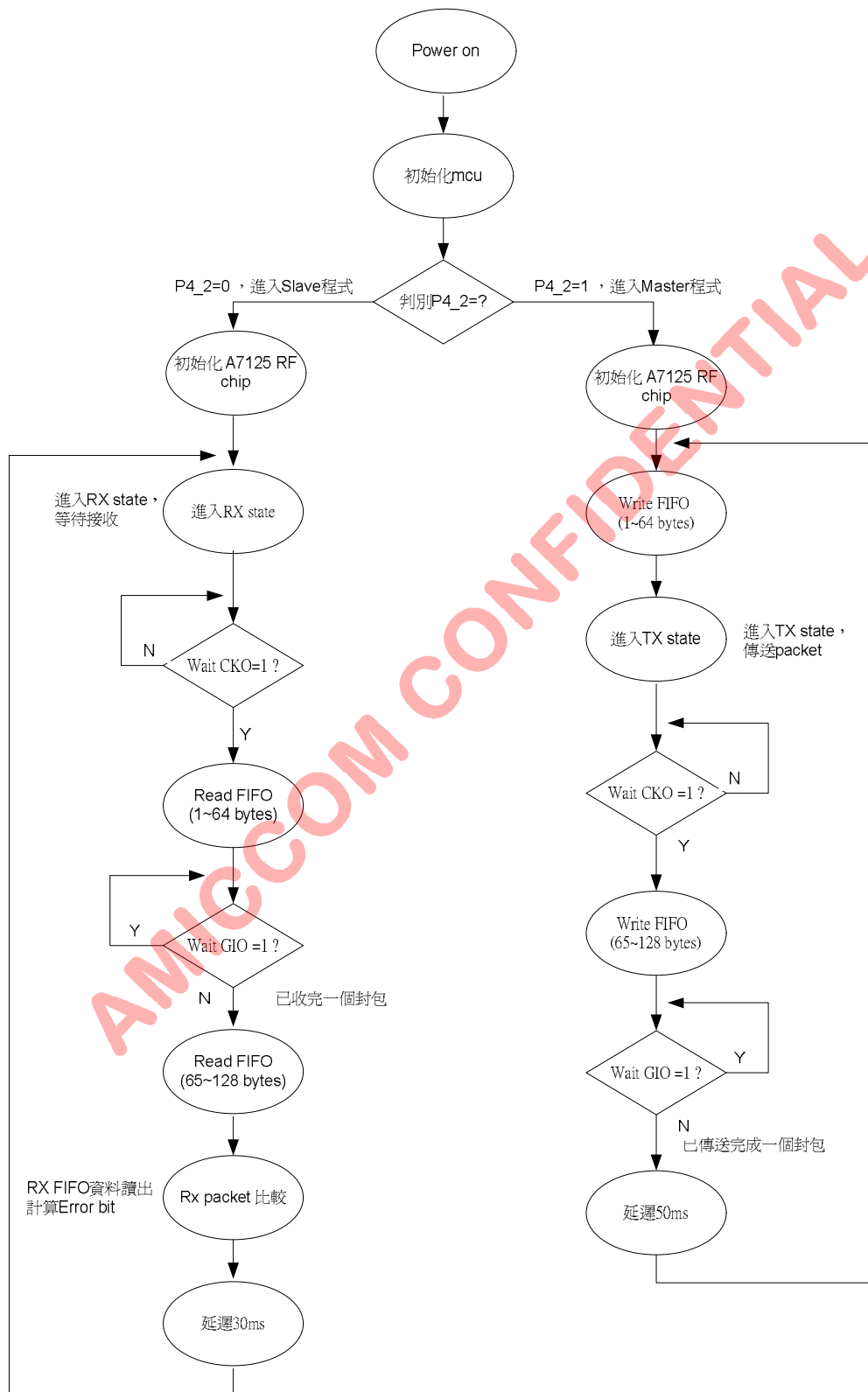


Fig5. 範例程式工作基本方塊

5. 程式說明

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。

<pre> 1 /***** 2 ** Device: A7125 3 ** File: main.c 4 ** Author: JPH 5 ** Target: Winbond W77LE58 6 ** Tools: ICE 7 ** Created: 2009-04-22 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 "a7125reg.h" 17 #include "Uti.h" </pre>	
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

<pre> 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 </pre>	
功能說明：MCU 對 A7125 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 A7125Config[]=
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     0x1F, //DATARATE register,
82     0x50, //PLL1 register,
83     0x0E, //PLL2 register,       RFbase 2400MHz
84     0x96, //PLL3 register,
85     0x00, //PLL4 register,
86     0x04, //PLL5 register,
87     0x3C, //chanel group I,
88     0x78, //chanel group II,
89     0xD7, //TX1 register,
90     0x40, //TX2 register,
91     0x10, //DELAY1 register,
92     0x61, //DELAY2 register,
93     0x62, //RX register,
94     0xA0, //RXGAIN1 register,
95     0x00, //RXGAIN2 register,
96     0x00, //RXGAIN3 register,
97     0xD2, //RXGAIN4 register,
98     0x00, //RSSI register,
99     0xE2, //ADC register,
100    0x07, //CODE1 register,
101    0x56, //CODE2 register,
102    0x2A, //CODE3 register,
103    0x06, //IFCAL1 register,
104    0x00, //IFCAL2 register,     only read
105    0x05, //VCOCCAL register,
106    0x44, //VCOCAL1 register,
107    0x80, //VCOCAL2 register,
108    0x30, //VCO DEV Cal. I register,
109    0x20, //VCO DEV Cal. II register,
110    0x80, //VCO DEV Cal. III register,
111    0x00, //VCO Mod. delay register
112    0x7A, //BATTERY register,
113    0x2F, //TXTEST register,
114    0x47, //RXDEM1 register,
115    0x80, //RXDEM2 register,
116    0xF1, //CPC1 register,
117    0x11, //CPC2 register,
118    0x04, //CRYSTAL register,
119    0x45, //PLLTTEST register,
120    0x18, //VCOTEST register,
121    0x10, //RF Analog Test,
122    0xFF, //IFAT register,
123    0x37, //Channel select register,
124    0xFF //VRB register
125 };

```

功能說明： RF chip 初始設定

行數	說明
----	----

68~124 RF chip 的初始設定

```

127 /*****
128 ** function Declaration
129 *****/
130 void InitTimer0(void);
131 void initUart0(void);
132 void Timer0ISR (void);
133 void Uart0Isr(void);
134 void A7125_Reset(void);
135 void A7125_WriteReg(Uint8, Uint8);
136 Uint8 A7125_ReadReg(Uint8);
137 void ByteSend(Uint8 src);
138 Uint8 ByteRead(void);
139 void A7125_WritelD(void);
140 void A7125_WriteFIFO(void);
141 void initRF(void);
142 void A7125_Config(void);
143 void CHGroupCal(Uint8);
144 void A7125_Cal(void);
145 void RxPacket(void);
146 void StrobeCmd(Uint8);
147 void SetCH(Uint8);
148 void A7125_WriteFIFO_1(void);
149 void ReadDataToBuf(Uint8 *);

```

功能說明：副程式檔頭宣告

行數	說明
----	----

130~149	副程式宣告
---------	-------

```

151 /*****
152 * main loop
153 *****/
154 void main(void)
155 {
156     //initsw
157     PMR |= 0x01; //set DME0
158
159     //initHW
160     P0 = 0xFF;
161     P1 = 0xFF;
162     P2 = 0xFF;
163     P3 = 0xFF;
164     P4 = 0x0F;
165
166     InitTimer0();
167     initUart0();
168     TR0=1;
169     EA=1;
170
171     if ((P4 & 0x04)==0x04) //if P4.2=1, master
172     {
173         initRF();
174         StrobeCmd(CMD_STBY);
175
176         while(1)
177         {
178             StrobeCmd(CMD_TFR); //reset tX FIFO pointer
179             A7125_WriteFIFO(); //write 1~64 bytes data to tx fifo
180             SetCH(100); //freq 2450.001MHz
181             StrobeCmd(CMD_TX); //entry tx
182
183             while(~CKO);
184             A7125_WriteFIFO_1(); //write 65~128 bytes data to tx fifo
185             while(GIO1); //wait transmit completed
186
187             Delay10ms(5);
188         }
189     }
190     else //if P4.2=0, slave
191     {
192         initRF();
193
194         RxCnt = 0;
195         Err_ByteCnt = 0;
196         Err_BitCnt = 0;
197         //TR0 = 1;
198         StrobeCmd(CMD_STBY);

```

```

200 while(1)
201 {
202     SetCH(96); //freq 2448.001MHz
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 172~189 為 master 程式。Port4_2=0，進入 slave 迴圈，行數 191~214 為 slave 程式。

行數	說明
157	啟用 MCU on chip data SRAM
160~164	初始化 MCU I/O Port
166	呼叫副程式 initTimer0，致能中斷
167	呼叫副程式 initUart0，初始 Uart0
168~169	啟動 Timer0，致能中斷開啓
171	判別 port4_2=1，進入 master 迴圈。Port4_2=0，進入 slave 迴圈。
172~189	Master 迴圈程式
173	呼叫副程式 initRF，初始化 A7125 chip
174	使用 Strobe command，進入 Standby mode 模式
178	使用 Strobe command，重置 TX FIFO pointer
179	呼叫副程式 A7125_WriteFIFO，將 data 寫入 TX FIFO
180	呼叫副程式 SetCH，設定工作頻率
181	使用 Strobe command，進入 TX 模式，發送資料
183	判別 I/O CKO，等待是否已傳送完 48 bytes
184	呼叫副程式 A7125_WriteFIFO_1，將 data 寫入 TX FIFO
185	判別 I/O GIO1，等待是否已傳送完成
187	延遲 50ms
191~214	Slave 迴圈程式
192	呼叫副程式 initRF，初始化 A7125 chip
194~196	清除變數 RxCnt, Err_ByteCnt, Err_BitCnt, 值
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 的初始值
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 為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，設置為 9600bps @xtal=11.0592MHz
270	設置 Timer1 為 mode 2
271~273	設置 REN,TR1,ES 為 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 旗標是否為 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 是否為 0。如不為 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 A7125_Reset(void)
297 {
298     A7125_WriteReg(MODE_REG, 0x00); //reset RF chip
299 }

```

功能說明：A7125 RF chip Reset 程序

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

```

301 /*****
302 ** WritelD
303 *****/
304 void A7125_WritelD(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 ** A7125_WriteReg
330 *****/
331 void A7125_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 A7125 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 ** A7125_ReadReg
369 *****/
370 Uint8 A7125_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         if(addr & 0x80)
380             SDIO = 1;
381         else
382             SDIO = 0;
383
384         _nop_();
385         SCK = 1;
386         _nop_();
387         SCK = 0;
388
389         addr = addr << 1;
390     }
391
392     _nop_();
393     SDIO = 1;
394
395     //read data
396     for(i = 0; i < 8; i++)
397     {
398         if(SDIO)
399             tmp = (tmp << 1) | 0x01;
400         else
401             tmp = tmp << 1;
402
403         SCK = 1;
404         _nop_();
405         SCK = 0;
406     }
407     SCS = 1;
408     return tmp;
409 }
410

```

功能說明：A7125 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 為輸出模式
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     for(i = 0; i < 4; i++)
463     {
464         if(src & 0x80)
465             SDIO = 1;
466         else
467             SDIO = 0;
468         _nop_();
469         SCK = 1;
470         _nop_();
471         SCK = 0;
472         src = src << 1;
473     }
474 }
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     A7125_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 A7125_WriteReg，對 PLL1 控制暫存器寫入工作頻道值。
-----	-------------------------------------------

```

487 /*****
488 ** initRF
489 *****/
490 void initRF(void)
491 {
492     //init io pin
493     SCS = 1;
494     SCK = 0;
495     SDIO = 1;
496     CKO = 1;
497     GIO1 = 1;
498     GIO2 = 1;
499
500     A7125_Reset(); //reset A7125 RF chip
501     A7125_WritelD(); //write ID code
502     A7125_Config(); //config A7125 chip
503     A7125_Cal(); //calibration RSSI, IF, vco, vcoc, vco deviation
504 }

```

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 A7125_Reset，重置 RF chip
501	呼叫副程式 A7125_WritelD，寫入 ID code 4ytes
502	呼叫副程式 A7125_Config，初始 RF 端的控制暫存器
503	呼叫副程式 A7125_Cal，做 RSSI, IF, VCO, VCO current, VCO deviation 的校準程序

```

506 /*****
507 ** A7125_WriteFIFO
508 *****/
509 void A7125_WriteFIFO(void)
510 {
511     Uint8 i;
512     Uint8 cmd;
513
514     cmd = FIFO_REG; //send address 0x05, bit cmd=0, r/w=0
515     SCS=0;
516     ByteSend(cmd);
517     for(i=0; i <64; i++)
518         ByteSend(PN9_Tab[i]);
519     SCS=1;
520 }

```

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 ** A7125_WriteFIFO_1
524 *****/
525 void A7125_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 }

```

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
557     RxCnt++;
558     for(i=0; i <64; i++)
559     {
560         recv = tmpbuff[i];
561         if((recv ^ PN9_Tab[i])!=0)
562         {
563             tmp = recv ^ PN9_Tab[i];
564             Err_BitCnt += (BitCount_Tab[tmp>>4] + BitCount_Tab[tmp & 0x0F]);
565         }
566     }
567
568     for(i=0; i <64; i++)
569     {
570         recv = tmpbuff[i+64];
571         if((recv ^ (PN9_Tab[i+1]))!=0)
572         {
573             tmp = recv ^ (~PN9_Tab[i]);
574             Err_BitCnt += (BitCount_Tab[tmp>>4] + BitCount_Tab[tmp & 0x0F]);
575         }
576     }
577 }

```

功能說明：從 tmpbuff 讀出資料，比對資料的程序

行數	說明
557	變數 RxCnt 加 1
558~566	將 tmpbuff[0]~tmpbuff[63]讀出，比較 data 的正確性，計算出 error bit。
568~576	將 tmpbuff[64]~tmpbuff[127]讀出，比較 data 的正確性，計算出 error bit。

```

579 /*****
580 ** Err_State
581 *****/
582 void Err_State(void)
583 {
584     //ERR display
585     //Error Proc...
586     //...
587 }

```

功能說明：Error State 處理程序

584~585	Error 處理程序，使用者自訂
---------	------------------

```

589 /*****
590 ** CHGroupCal
591 *****/
592 void CHGroupCal(Uint8 ch)
593 {
594     Uint8 tmp;
595     Uint8 vb,vbcf;
596     Uint8 vcb,vccf;
597
598     A7125_WriteReg(PLL1_REG, ch);
599     Delay10us(4);
600     A7125_WriteReg(CALIBRATION_REG, 0x1C);
601     do{
602         tmp = A7125_ReadReg(CALIBRATION_REG);
603         tmp &= 0x1C;
604     }
605     while (tmp);
606
607     //for check
608     tmp = A7125_ReadReg(VCOCAL1_REG);
609     vb = tmp & 0x07;
610     vbcf = (tmp >>3) & 0x01;
611
612     tmp = A7125_ReadReg(VCOCCAL_REG);
613     vcb = tmp & 0x0F;
614     vccf= (tmp >> 4) & 0x01;
615
616     if (vbcf || vccf)
617         Err_State();//error
618 }

```

功能說明：Channel Group 校準程序

598	呼叫 A7125_WriteReg 副程式，設置 PLL1 控制暫存器。
599	延遲 40us，使 PLL setting 穩定。
600	設置 calibration control register 中 bit VDC=1, VBC=1, VCC=1。
601~605	讀出 calibration control register，並判別 bit VDC, VBC, VCC 是否為 0。如為 0，則跳出等待迴圈
608~610	讀出 VCO Calibration I 控制暫存器，並檢示其值
612~614	讀出 VCO Current Calibration 控制暫存器，並檢示其值
616~617	判別 vbcf,vccf 是否為 1。如為 1，則判定 fail。進入 Err_State 處理程序

```

620 /*****
621 ** calibration
622 *****/
623 void A7125_Cal(void)
624 {
625     Uint8 tmp;
626     Uint8 fb,fbcf,fcd;
627     Uint8 dvt;
628
629     //calibration RSSI,IF procedure
630     StrobeCmd(CMD_PLL);
631     A7125_WriteReg(CALIBRATION_REG, 0x03);
632     do
633     {
634         tmp = A7125_ReadReg(CALIBRATION_REG);
635         tmp &= 0x03;
636     }
637     while (tmp);
638
639     //calibration VCO dev,VBC,VCC procedure
640
641     CHGroupCal(30); //calibrate channel group Bank I
642     CHGroupCal(90); //calibrate channel group Bank II
643     CHGroupCal(150); //calibrate channel group Bank III
644     StrobeCmd(CMD_STBY); //return to STBY state
645
646     //for check
647     tmp = A7125_ReadReg(IFCAL1_REG);
648     fb = tmp & 0x0F;
649     fbcf = (tmp >>4) & 0x01;
650
651     tmp = A7125_ReadReg(IFCAL2_REG);
652     fcd = tmp & 0x1F;
653
654     tmp = A7125_ReadReg(VCOCAL2_REG);
655     dvt = tmp;
656
657     if (fbcf)
658         Err_State(); //error
659 }

```

功能說明：RSSI, IF, VCO, VCO current 校準程序

行數	說明
630	使用 Strobe 命令，設置 RF chip 進入 PLL state
631	設置 calibration control register 中 bit RSSC=1, FBC=1。
632~637	讀出 calibration control register，並判別 bit RSSC, FBC 是否為 0。如為 0，則跳出等待迴圈
641	呼叫 CHGropuCal 副程式，對 channel 30 做校正程序。
642	呼叫 CHGropuCal 副程式，對 channel 90 做校正程序。
643	呼叫 CHGropuCal 副程式，對 channel 150 做校正程序。
644	使用 Strobe 命令，設置 RF chip 進入 Standby state
647~649	讀出 IF Calibration I 控制暫存器，並檢示其值
651~652	讀出 IF Calibration II 控制暫存器，並檢示其值
654~655	讀出 VCO Calibration II 控制暫存器，並檢示其值
657~658	判別 fbcf 旗標是否為 0。如不為 0，則判定 RF chip 為 fail，進入 Err_State 處理程序。


```

661 /*****
662 ** A7125_Config
663 *****/
664 void A7125_Config(void)
665 {
666     Uint8 i;
667
668     //0x00 mode register, for reset
669     //0x05 fifo data register
670     //0x06 id code register
671     //0x24 IF calibration II, only read
672
673     for (i=0x01; i<=0x04; i++)
674         A7125_WriteReg(i, A7125Config[i]);
675
676     for (i=0x07; i<=0x23; i++)
677         A7125_WriteReg(i, A7125Config[i]);
678
679     for (i=0x25; i<=0x38; i++)
680         A7125_WriteReg(i, A7125Config[i]);
681 }

```

功能說明：初始 RF chip Master 端的程序

行數	說明
673~674	呼叫副程式 A7125_WriteReg，寫入控制暫存器位址 0x01~0x04
676~677	呼叫副程式 A7125_WriteReg，寫入控制暫存器位址 0x07~0x23
679~680	呼叫副程式 A7125_WriteReg，寫入控制暫存器位址 0x25~0x38

```

683 /*****
684 ** ReadDataToBuf
685 *****/
686 void ReadDataToBuf(Uint8 *dest)
687 {
688     Uint8 i;
689     Uint8 recv;
690     Uint8 cmd;
691
692     cmd = FIFO_REG | 0x40; //address 0x05, bit cmd=0, r/w=1
693
694     SCS=0;
695     ByteSend(cmd);
696     for(i=0; i <64; i++)
697     {
698         recv = ByteRead();
699         *dest++ = recv;
700     }
701     SCS=1;
702 }

```

功能說明：讀資料到 tmpbuf 的程序

行數	說明
692	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
694	SCS=0，致能控制暫存器讀寫功能
695	呼叫副程式 ByteSend，送出控制命令
696~700	讀出共 64 bytes 的值，並存入變數 tmpbuf
701	SCS=1，清除控制暫存器讀寫功能