Document Title

A7121 reference code for FIFO mode

Revision History

Rev. No.	History	Issue Date	Remark
1.0	Initial issue	Jun 1 , 2007	

Table of contents

AMIC-COM RF Chip-A7121
Reference code for FIFO mode

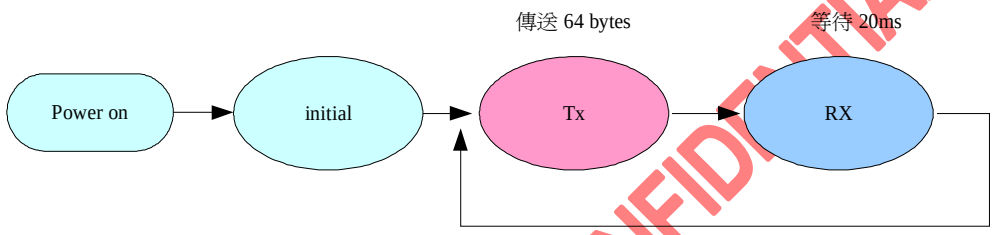
n 1. 簡介：

這文件係對 AMIC-COM RF chip -A7121 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。

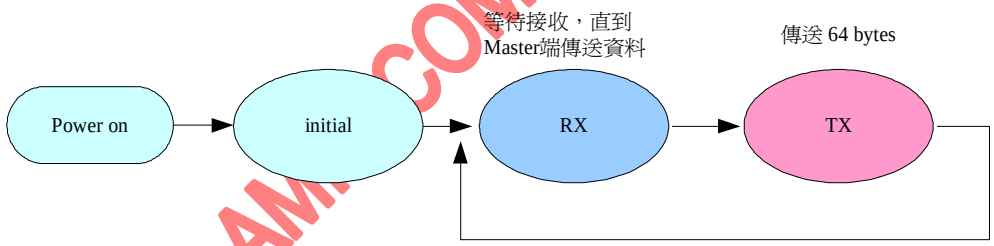
n 2. 系統概述：

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

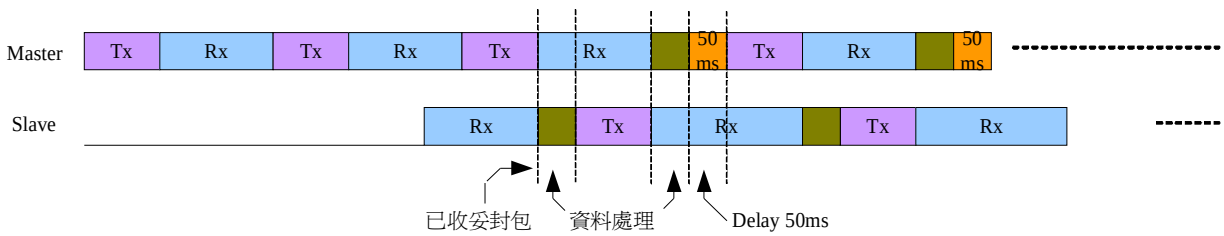
Master 端：power on、initial 系統及 RF chip 後，進入 TX 狀態，傳送 64 bytes 資料，再進入 RX 接收狀態，等待 20ms。若 slave 端有發射資料，則 Master 端會接收到資料。否則，20ms 之後，Master 端又會回到 TX 傳送階段。



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

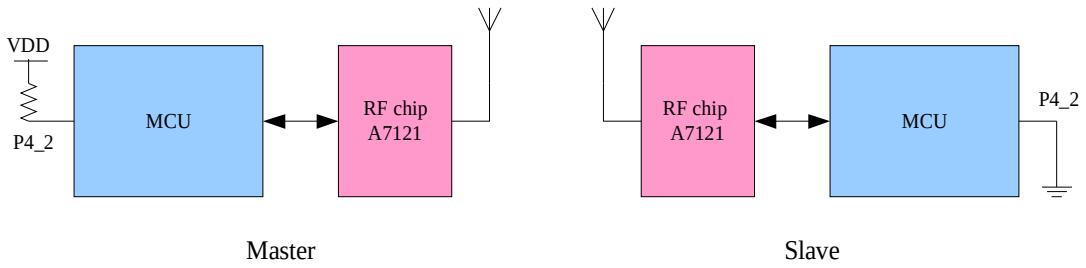


Master 端在 Power on 後，進入迴圈送出封包及等待 Slave 端所傳送合法的封包。Master 端如未收到封包，在 20ms 後，回到發送程序送出封包。一旦接收到封包，讀出資料、比對，計算 error bit，延遲 50ms 後，回到發送程序送出封包。Slave 端在 Power on 後，進入接收狀態，等待從 master 端所發送合法的封包。Slave 端如未收到封包，則仍繼續等待接收。一旦接收到封包，讀出資料、比對，計算 error bit 後，再發送封包給 Master 端。使用者可依據簡易的計算 error bit 及傳送封包數，得出 BER(bit error rate)，作爲傳輸品質的數據。



n 3. 硬體:

¨ **系統方塊圖:**



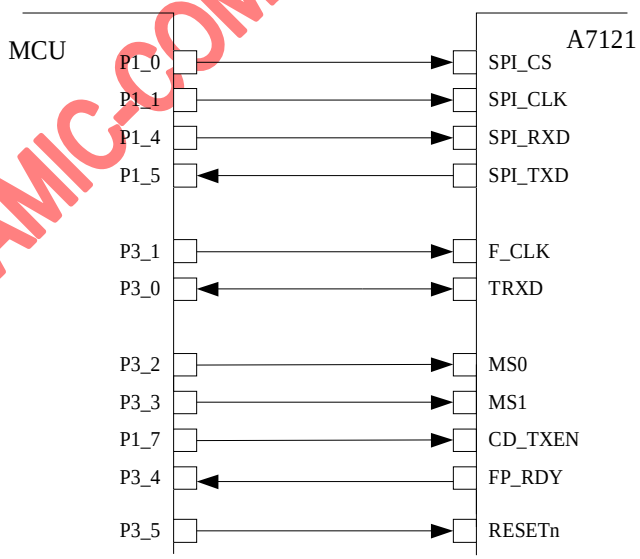
MCU 使用 I/O pin 4_2 的設定，判別 Master 端或 Slave 端。

¨ **使用 I/O pin 設定：**

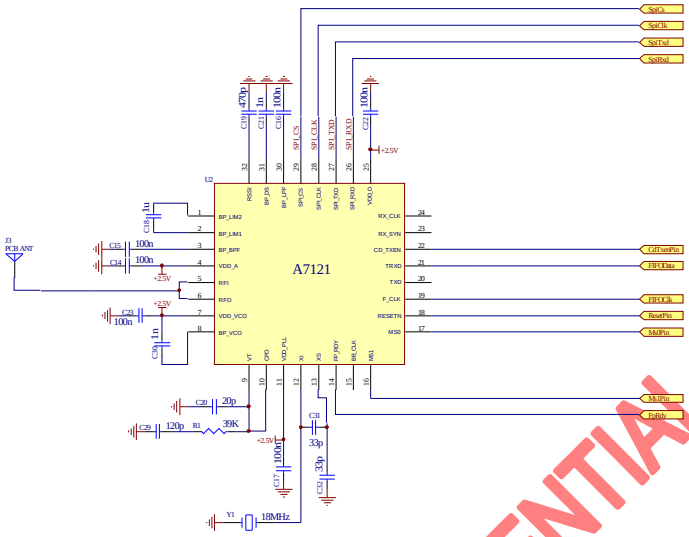
應用範例使用 I/O：

1. SPI_CS, SPI_CLK, SPI_RXD, SPI_TXD 這 4 wire SPI 介面控制 A7121 內部 register。Spi config 請參考附件一。
2. F_CLK, TRXD 用於控制 FIFO 讀寫介面。FIFO config 請參考附件二。
3. MS0, MS1, CD_TXEN 是 A7121 工作模式控制動作介面，能夠快速切換 mode 的選擇，減少使用 SPI 介面控制所耗費的時間。Mode of operation 及 Timing chart ，請參考附三。
4. FP_RDY 是 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。
5. RESETn 是 A7121RF chip 的硬體重置控制。

¨ **MCU 控制 A7121 RF chip 的 I/O 配置如下圖：**



- **RF 線路圖(Master & Slave)**



n 4. 韌體程式設計:

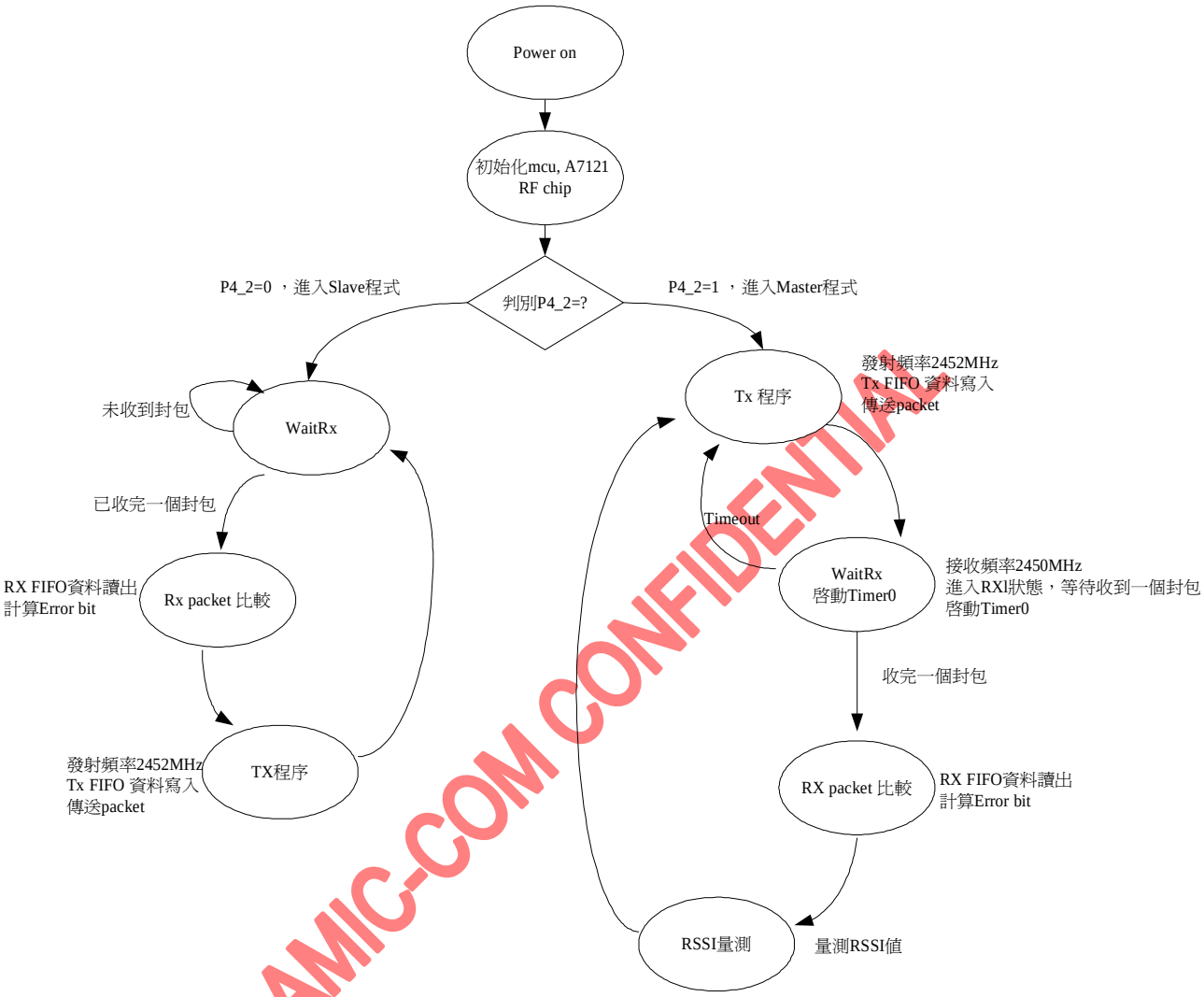
- **應用範例概述：**

首先初始化 Timer0 及 A7121RF 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) TX FIFO 寫入 PN9 code 共 64 bytes。
2) 設定頻率在 2452MHz，進入 TX 狀態，致能 CD_TXEN，發送封包，結束 TX 狀態。
4) 設定頻率在 2450MHz，進入 RX 狀態，啓動 Timer0 計時。
5) 如發生 Timeout=20ms 後，程式重新回到頻率 2452MHz，Tx 狀態，再一次傳送封包。
6) 如收到封包後，關閉 Timer0 計時，結束 RX 狀態。
7) 從 RX FIFO 讀出，並比較 PN9 code 共 64bytes，計算 error bit 數目。
8) 延遲 50ms，重新回到第一動作，重新再傳送下一次封包。

Slave 端：
1) 設定頻率在 2450MHz， 進入 RX 狀態，等待封包收到。
2) 如收到封包後，結束 RX 狀態。
3) 從 RX FIFO 讀出，並比較 PN9 code 共 64bytes，計算 error bit 數目。
4) TX FIFO 寫入 PN9 code 共 64 bytes。
5) 設定頻率在 2452MHz， 進入 TX 狀態，致能 CD_TXEN，發送封包，結束 TX 狀態。
6) 重新回到第一動作，等待下一次封包的進入。

- 範例程式工作基本方塊圖如下：



- 設置條件：
  freq = 2452 or 2450MHz
  Data Rate = 1Mbps
  Compare freq = 0.666666KHz
  IF(中頻) = 2MHz

5. 程式說明：

```
 1  /*****************************************************************************
 2  **  Device:     A7121
 3  **  File:       main.c
 4  **  Author:     JPH
 5  **  Target:     Winbond W77LE58
 6  **  Tools:      ICE
 7  **  Created:    2006-12-01
 8  **  Description:
 9  **  This file is a sample code for your reference.
10  **
11  **  Copyright (C) 2006 AMICCOM Corp.
12  **
13  **  Revision history:
14  **
15  **  Revision 0.0  2006/12/1
16  **  Initial version.
17  *****************************************************************************/
18  #include "define.h"
19  #include "w77le58.h"
20  #include "a7121reg.h"
21  #include "Uti.h"
22
23  #define TIMEOUT        20
24  #define t0hrel         1000
25  #define Freq_Base      0x0960//2400MHz
```

功能說明：Include 檔宣告，定義常數變數	
行數	說明
18~21	匯入程式庫設定檔
23~25	定義常數變數

```
27  /*****************************************************************************
28  **  I/O Declaration
29  *****************************************************************************/
30  #define SpiCs       P1_0
31  #define SpiClk      P1_1
32  #define SpiRxd      P1_4
33  #define SpiTxd      P1_5
34  #define CdTxenPin   P1_7
35
36  #define FIFOData    P3_0
37  #define FIFOClk     P3_1
38  #define Ms0Pin      P3_2
39  #define Ms1Pin      P3_3
40  #define FpRdy       P3_4
41  #define ResetPin    P3_5
```

功能說明：MCU 對 A7121 RF chip I/O 接腳定義	
行數	說明
30~41	MCU I/O 配置

```
43  /*********************************************************************
44   **  Global Variable Declaration
45   *********************************************************************/
46  Uint8       data    timer;
47  Uint8       data    TimeoutFlag;
48  Uint16      data    Err_BitCnt;
49  Uint16      idata   ModeCtlReg;
50  Uint16      idata   FifoCtlReg;
51  Uint8       xdata   FreqBank[84];
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 AccessCode_Tab[9] = {0x54,0x75,0xC5,0x8C,0xC7,0x33,0x45,0xE7,0x2A};
55  const Uint8 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
46~51	程式中使用的變數宣告
53	BitCount_Tab 宣告
54	Access code 宣告
55~64	PN9 data 宣告

```
66  /*********************************************************************
67   **  function Declaration
68   *********************************************************************/
69  void InitTimer0(void);
70  void Timer0ISR (void);
71  void Rst_Timer0(void);
72  void SpiWrite(Uint16 dataWord,Uint8 address);
73  Uint16 SpiRead(Uint8 address);
74  void FIFOWrite(Uint8 source);
75  Uint8 FIFORead(void);
76  void FIFOWr_En(void);
77  void FIFOWr_Di(void);
78  void FIFORd_En(void);
79  void FIFORd_Di(void);
80  void SYNth_En(void);
81  void SYNth_Di(void);
82  void SetFreq(Uint8 ch);
83  void SetTRC(Uint8 trc);
84  void ResetRF(void);
85  void initRF(void);
86  Uint8 Calibration_VCO(Uint8 ch);
87  void TxPacket(void);
88  void RxPacket(void);
89  void WaitRx(void);
90  Uint8 WaitAck(void);
91  void MeasureRSSI(void);
```

功能說明：副程式檔頭宣告

行數	說明
69~91	副程式宣告

```
93  /*********************************************************************
 94  * main loop
 95  *********************************************************************/
 96  void main(void)
 97  {
 98     //initsw
 99     PMR |= 0x01;//set DME0
100
101     //initHW
102     P0 = 0xFF;
103     P1 = 0xFF;
104     P2 = 0xFF;
105     P3 = 0xFF;
106     P4 = 0x0F;
107
108     InitTimer0();
109     EA=1;
110
111     initRF(); //init RF
112     SYNth_En(); //synthesizer on
113
114     if ((P4 & 0x04)==0x04) // 1=master
115     {
116        while(1)
117        {
118           //Tx state
119           TxPacket();
120
121           //Rx stste
122           if (WaitAck())
123              RxPacket();
124
125           //measure RSSI value
126           MeasureRSSI();
127
128           Delay10ms(5); //delay 50ms
129         }
130     }
131     else // 0=slave
132     {
133        while(1)
134        {
135           //Rx state
136           WaitRx();
137           RxPacket();
138
139           //Tx state
140           TxPacket();
141        }
142     }
143  }
```

功能說明：主程式 main loop。port4_2=1，進入 master 迴圈，行數 115~130 爲 master 程式。Port4_2=0，進入 slave 迴圈，行數 132~142 爲 slaver 程式。

行數	說明
99~106	初始化 mcu 及 I/O Port
108~109	呼叫副程式 initTimer0，致能中斷
111	呼叫副程式 initRF，初始化 A7121 chip
112	呼叫副程式 SYNth_En，設置 A7121 chip 進入 synthesizer mode
114	判別 port4_2=1，進入 master 迴圈。port4_2=0，進入 slave 迴圈。
115~130	Master 迴圈程式

119	呼叫副程式 TxPacket，進入 TX 模式，發送資料
122	呼叫副程式 WaitAck，進入 RX 模式，啓動 Timer0，等待資料的收妥
123	如資料已收妥後，呼叫副程式 RxPacket，從 RX FIFO 讀出資料、比對、計算 error bit 數
126	呼叫副程式 MeasureRSSI，量測 TX 端週邊附近的 RSSI 值，參考使用
128	呼叫副程式 Dealy10ms，程式延遲 50ms 動作
132~142	Slave 迴圈程式
136	呼叫副程式 WaitRx，進入 RX 模式，等待資料的收妥。如未收到資料，仍在 RX 模式中，等待接收
137	如資料已收妥後，呼叫副程式 RxPacket，從 RX FIFO 讀出資料、比對、計算 error bit 數
140	呼叫副程式 TxPacket，進入 TX 模式，發送資料

```
145  /*********************************************************************
146  ** initRF
147  *********************************************************************/
148  void initRF(void)
149  {
150      Uint8 i;
151      Uint8 a,b,c,RH_value,RL_value;
152      Uint16 tmp;
153
154      //init i/o pin
155      Ms0Pin = 1; //standby mode
156      Ms1Pin = 0;
157      SpiCs  = 1;
158      SpiClk = 0;
159      SpiTxd = 1;
160      SpiRxd = 1;
161      CdTxenPin = 0; //disable CDTXEN
162      FIFOClk = 0;
163
164      //reset RF chip
165      ResetRF();
166
167      //init register
168      SpiWrite(0x0960, SYNTH1_REG); //BNk=0,default freq=2400MHz
169      SpiWrite(0x0B1B, SYNTH2_REG); //VTH=6,CP=500uA,default R=27
170      SpiWrite(0x2231, SYSTEM_CLK_REG); //XBR=17,XDR=17,XIR=8
171
172      ModeCtlReg = 0x03D3; //RSTN=1,CE=1,SYN=1,TRC=tx
173      SpiWrite(ModeCtlReg, MODE_CONTROL_REG);
174
175      SpiWrite(0xFFD0, TX_CONTROL1_REG); //TXDI=normal,DEV=8(250kHz),GF=off,IA=31,QA=31
176      SpiWrite(0xE23F, TX_CONTROL2_REG); //PC=63,IO=8,QO=8,IQC=3
177      SpiWrite(0x06F6, RX_CONTROL1_REG); //RXDI=0,DPC=3,ETH=6,RCP=3,DS=rxsyn->off,SYNI=0
178      SpiWrite(0x0030, RX_CONTROL2_REG); //VGA=0db,DFG=6
179
180      FifoCtlReg = 0x0FC0; //fifo byte count = 64 bytes
181      SpiWrite(FifoCtlReg, FIFO_CONTROL_REG);
182
183      for (i=0;i<9;i++)
184          SpiWrite((AccessCode_Tab[i]<<8) | AccessCode_Tab[i],ACCESS_CODE_REG); //access code
185
186      SpiWrite(0x01F4, CALIBRATION_CONTROL1_REG); //MCAL=0,IFC=1,DFC=1,DEMC=1,RHC=1,RLC=1
187      SpiWrite(0x0001, CALIBRATION_CONTROL2_REG); //ETR=1,FPRS=0
188      SpiWrite(0x0D83, ADC_REG); //ADC=3,AD[2:0]=0,AD[5:3]=3,AD[8:6]=3
189
190      //rx cal
191      SpiWrite(0x01D3, MODE_CONTROL_REG); //EXIF=0
```

```
192     SpiWrite(0x0003, CALIBRATION_CONTROL2_REG); //ECAL=1
193        Delay10ms(2); //delay 20ms
194     SpiWrite(0x03D3, MODE_CONTROL_REG); //EXIF=1
195     do
196     {
197        tmp = SpiRead(CALIBRATION_CONTROL2_REG);
198        tmp &= 0x0002;
199     }
200     while (tmp); //check calibration ok?
201        SpiWrite(0x0037, RX_CONTROL2_REG); //VGA=20db,DFG=6
202
203     //read calibration value for check
204     a=(SpiRead(IF_FILTER_REG) & 0xFE00)>>9;
205     b=(SpiRead(DATA_FILTER_REG)& 0xFE00)>>9;
206     c=(SpiRead(DEMODULATOR_REG) & 0xFF00)>>8;
207     RH_value=(SpiRead(RH_REG) & 0xFF00)>>8;
208     RL_value=(SpiRead(RL_REG)& 0xFF00)>>8;
209
210     //vco cal
211     for(i=0;i<84;i++)
212        FreqBank[i]=Calibration_VCO(i);
213  }
```

功能說明：初始化 A7121 RF Chip，分爲 I/O 設定，初始化 control register，Rx calibration，vco calibration 程序

行數	說明
155~162	初始 A7121 RF chip I/O 設定
165	呼叫副程式 ResetRF ，使 A7121RF chip 重置動作
168~188	初始 A7121 內部 control register 的設定
191~201	Rx calibration 程序
191	設置 mode control register 中 bit EXIR=0
192	設置 calibration control register II 中 bit ECAL=1
193	延遲 > 20ms
194	設置 mode control register 中 bit EXIR=1
195~200	讀出 calibration control register 中 bit ECAL，判別是否完成 calibration 動作
201	設置 rx control register II 中 bit VGA[2:0]=111(VGA gain=20dB)
204~208	檢查 calibration 值 if filter register: 50~80 data filter register: 50 ~80 demodulator register:80~110 RH register: 50~70 RL register: 70~90
21~212	對工作頻率 2400~2483 MHz 做 vco calibration，並將 bank value 儲存至 FreqBank[ ]陣列變數中

```
215  /*********************************************************************
216  * Calibration_VCO
217  *********************************************************************/
218  Uint8 Calibration_VCO(Uint8 ch)
219  {
220     Uint8 var_bank;
221     Uint8 ackError;
222     Uint8 i;
223     Uint16 ack;
224     Uint16 var_data;
225
226     var_bank=3;//init var_bank value
```

功能說明：初始化 Timer0 的設定，計時 1ms

行數	說明
279~284	初始化 MCU Timer0 的 register 設定

```
227     SYNth_En();//set SYN=1
228     var_data = (Freq_Base + ch) & 0x1FFF;//mask Bank[2:0] bits
229
230     for (i=1; i<=8 ;i++)//retry 8 times
231     {
232         var_data &= 0x1FFF;
233         var_data |= (var_bank << 13);
234         SpiWrite(var_data, SYNTH1_REG);//write synthesizer I
235
236         Delay100us(3);//delay300us
237
238         ack=SpiRead(SYNTH2_REG);
239
240         //vt=01 calibration ok
241         if((ack & 0xc000)==0x4000)
242         {
243             ackError=0;
244             break;
245         }
246
247         //vt=00 freq too low
248         if((ack & 0xc000)==0x0000)
249         {
250             if(var_bank==7)
251             {
252                 ackError=1;
253                 break;
254             }
255             var_bank++;
256         }
257         //vt=11 freq too high
258         if((ack & 0xc000)==0xc000)
259         {
260            if(var_bank==0)
261             {
262                 ackError=1;
263                 break;
264             }
265             var_bank--;
266         }
267     }
268
269         SYNth_Di();//set SYN=0
270     if(ackError) return 0xFF;
271     else return var_bank;
272 }
```

功能說明：Calibration_VCO 程序，用於對每一個工作頻率做 calibration，並儲存於 MCU 中 register	
行數	說明
226	初始變數 var_bank=3
227	進入 synthesizer mode
228	變數 var_data= Freq_Base 值+CH 值，
230~267	For loop 迴圈，Vco calibration 做 8 次
232~233	變數 var_data= Freq_Base 值+CH 值 + 初始變數 var_bank
234	將變數 var_data 值，寫入 synthesizer control register I
236	延遲 300us

238	讀出 synthesizer control register II
241~266	取出 DVT[1:0]，並判別 DVT[1:0]=00：表示 BNK[2:0]值太小，須往上加 1。例 BNK[2:0]=011 -> 100。 DVT[1:0]=11：表示 BNK[2:0]值太大，須往下減 1。例 BNK[2:0]=011 -> 010。 DVT[1:0]=01：表示 BNK[2:0]值適中。
269	離開 synthesizer mode，進入 standby mode
270~271	返回正確的 var_bank 值或 error 值 0xff

```
/*********************************************************************
** init Timer0
*********************************************************************/
void InitTimer0(void)
{
    TR0 = 0;
    TMOD =0x01;//timer0 mode=1
    TH0 = (65536-t0hrel)>>8;// Reload Timer0 high byte,low byte
    TL0 = 65536-t0hrel;
    TF0 = 0;// Clear any pending Timer0 interrupts
    ET0 = 1;// Enable Timer0 interrupt
}
```

功能說明：初始化 Timer0 的設定，計時 1ms

行數	說明
279~284	初始化 MCU Timer0 的 register 設定

```
/*********************************************************************
** timer0ISR
*********************************************************************/
void Timer0ISR (void) interrupt 1
{
    TF0 = 0;// Clear Timer0 interrupt
    TH0 = (65536-t0hrel)>>8;// Reload Timer0 high byte
    TL0 = 65536-t0hrel;// Reload Timer0 low byte
    timer++;
    if (timer == TIMEOUT)
    {
        TimeoutFlag=1;
        TR0 = 0;// Stop timer
        timer=0;
    }
}
```

功能說明：Timer0ISR 中斷程序，是 Timer0 每 1ms 產生中斷的向量服務程式，主要在重新初始 timer0，計算是否 timeout，產生 TimerFlag 的旗標。

行數	說明
292~294	初始化 timer0 register 設定
295	每 1ms 產中斷，變數 timer 加 1
296~301	判別是否 timerout。如果 timeout，則設定 TimeoutFlag=1

```
/*********************************************************************
** Rst_Timer0
*********************************************************************/
void Rst_Timer0(void)
{
    TR0 = 0;
```

```
310     TH0 = (65536-t0hrel)>>8; // Reload Timer0 high byte
311     TL0 = 65536-t0hrel;
312     TF0 = 0;
313     timer = 0;
314     TimeoutFlag = 0;
315   }
```

功能說明：Rst_Timer0 程序是初始 Timer0 register 的設定。

行數	說明
309~314	初始化 timer0 register 設定

```
317   /*********************************************************************
318   ** SpiWrite
319   *********************************************************************/
320   void SpiWrite(Uint16 dataWord,Uint8 address)
321   {
322     Uint8 i;
323
324     SpiCs  = 0;// Enable A7121 SPI
325     SpiClk = 0;
326     SpiRxd = 1;
327
328     //send address code
329     address=(address<<2) | 0x80;//fill wirte bit
330     for(i = 0; i < 8; i++)
331     {
332       SpiClk = 1;
333
334       if(address & 0x80)
335         SpiRxd = 1;//bit=1
336       else
337         SpiRxd = 0;//bit=0
338
339       _nop_();
340       address = address << 1;
341       SpiClk = 0;
342     }
343
344     _nop_();
345
346     //send data code
347     for(i = 0; i < 16; i++)
348     {
349       SpiClk = 1;
350
351       if(dataWord & 0x8000)
352         SpiRxd = 1;
353       else
354         SpiRxd = 0;
355
356       dataWord = dataWord<< 1;
357       _nop_();
358       SpiClk = 0;
359     }
360
361     _nop_();
362     SpiCs = 1; //Disable A7121 SPI
363     _nop_();
364     SpiRxd = 1;
365   }
```

功能說明：SpiWrite 副程式是對 A7121 內部 control register 寫入動作的程式	
行數	說明
324~326	致能 SPI_CS pin
329~342	寫入 address 的程序
347~359	寫入 dataword 的程序
362	清除 SPI_CS pin

```
367  /*********************************************************************
368  ** SpiRead
369  *********************************************************************/
370  Uint16 SpiRead(Uint8 address)
371  {
372      Uint8 i;
373      Uint16 spiData;
374
375      spiData=0;
376      SpiClk=0;
377      SpiTxd = 1;
378
379      SpiCs  = 0; //Enable A7121 SPI
380      _nop_();
381
382      //send address code
383      address=(address<<2) | 0x00;//fill read bit
384      for(i = 0; i < 8; i++)
385      {
386          SpiClk = 1;
387
388          if(address & 0x80)
389              SpiRxd = 1;
390          else
391              SpiRxd = 0;
392
393          _nop_();
394          address = address << 1;
395          SpiClk = 0;
396      }
397
398      _nop_();
399
400      //read data code
401      for(i = 0; i < 16; i++)
402      {
403          SpiClk = 1;
404          _nop_();
405
406          if(SpiTxd)
407              spiData = (spiData << 1) | 0x01;  // SpiRxd: read "1"
408          else
409              spiData = spiData << 1;           // SpiRxd: read "0"
410
411          SpiClk = 0;
412      }
413
414     _nop_();
415      SpiCs = 1;
416      SpiRxd = 1;
417
```

```
418     return spiData;
419 }
```

功能說明：SpirRead 副程式是對 A7121 內部 control register 可讀出的 register 做讀出動作。使用者可讀出 DVT 值、RSSI 值等。

行數	說明
375~377	初始 SPI I/O 的設定
379	致能 SPI_CS pin
383~396	寫入 address 的程序
401~412	讀出 dataword 的程序
415	清除 SPI_CS pin

```
421 /*********************************************************************
422 *FIFOWrite
423 *********************************************************************/
424 void FIFOWrite(Uint8 source)
425 {
426     Uint8 i;
427
428     for(i = 0; i < 8; i++)
429     {
430         FIFOClk = 1;
431
432         if(source & 0x80)
433             FIFOData = 1;
434         else
435             FIFOData = 0;
436
437         _nop_();
438         source = source << 1;
439         FIFOClk = 0;
440     }
441
442     _nop_();
443     FIFOData=1;
444 }
```

功能說明：副程式 FIFOWrite，是對 A7121 TX FIFO 寫入動作。

行數	說明
428~443	對 A7121 TX FIFO 寫入程序。

```
446 /*********************************************************************
447 *FIFORead
448 *********************************************************************/
449 Uint8 FIFORead(void)
450 {
451     Uint8 i;
452     Uint8 tmp=0;
453
454     for(i = 0; i < 8; i++)
455     {
456         FIFOClk = 1;
457         _nop_();
458         if(FIFOData)
459             tmp = (tmp << 1) | 0x01;
460         else
```

```
461             tmp  = tmp << 1;
462
463           FIFOClk = 0;
464       }
465
466       return tmp;
467   }
```

功能說明：副程式 FIFORead，是對 A7121 RX FIFO 讀出動作。

行數	說明
454~464	A7121 RX FIFO 讀出程序
466	返回 16 bit 的讀值

```
469   /*********************************************************************
470   **  SYNth_En
471   *********************************************************************/
472   void SYNth_En(void)
473   {
474       ModeCtlReg |= 0x0004; //enable synth bit
475       SpiWrite(ModeCtlReg, MODE_CONTROL_REG);
476       Delay10us(15); //delay 150us for pll settling time
477   }
```

功能說明：副程式 Synth_En，是對 mode control register 中 bit SYN 設定爲 1。

行數	說明
474~475	設置 mode control register 中 bit SYN=1
476	延遲 Max 150us 的 PLL on settling time

```
479   /*********************************************************************
480   **  SYNth_Di
481   *********************************************************************/
482   void SYNth_Di(void)
483   {
484       ModeCtlReg &= (~0x0004); //disable synth bit
485       SpiWrite(ModeCtlReg, MODE_CONTROL_REG);
486   }
487
```

功能說明：副程式 Synth_Di，是對 mode control register 中 bit SYN 設定爲 0。

行數	說明
484~485	設置 mode control register 中 bit SYN=0

```
488   /*********************************************************************
489   ** ResetRF
490   *********************************************************************/
491   void ResetRF(void)
492   {
493         //hardware reset
494       ResetPin = 0;
495       _nop_();
496       ResetPin = 1;
497
498       // or register reset
499       //SpiWrite(0x0002,MODE_CONTROL_REG);//reset enable
500       //delay();
501       //SpiWrite(0x0003,MODE_CONTROL_REG);//reset disable
502   }
```

功能說明：副程式 ResetRF，是對 A7121 register 作初始設定。

行數	說明
494~496	使用 RESETn pin 對 A7121 做 Reset 動作

```
504  /*********************************************************************
505  **  FIFOWr_En
506  *********************************************************************/
507  void FIFOWr_En(void)
508  {
509      FifoCtlReg |= 0x0004;
510      SpiWrite(FifoCtlReg, FIFO_CONTROL_REG);
511  }
```

功能說明：副程式 FIFOWr_En，是對 A7121 TX FIFO 寫入置能設定。

行數	說明
509~510	設置 fifo control register 中 bit EFW=1

```
513  /*********************************************************************
514  **  FIFOWr_Di
515  *********************************************************************/
516  void FIFOWr_Di(void)
517  {
518      FifoCtlReg &= ~0x0004;
519      SpiWrite(FifoCtlReg, FIFO_CONTROL_REG);
520  }
```

功能說明：副程式 FIFOWr_Di，是對 A7121 TX FIFO 寫入清除設定。

行數	說明
518~519	設置 fifo control register 中 bit EFW=0

```
522  /*********************************************************************
523  **  FIFORd_En
524  *********************************************************************/
525  void FIFORd_En(void)
526  {
527      FifoCtlReg |= 0x0010;
528      SpiWrite(FifoCtlReg, FIFO_CONTROL_REG);
529  }
```

功能說明：副程式 FIFORd_En，是對 A7121 RX FIFO 讀出置能設定。

行數	說明
527~528	設置 fifo control register 中 bit EFR=1

```
531  /*********************************************************************
532  **  FIFORd_Di
533  *********************************************************************/
534  void FIFORd_Di(void)
535  {
536      FifoCtlReg &= ~0x0010;
537      SpiWrite(FifoCtlReg, FIFO_CONTROL_REG);
538  }
```

功能說明：副程式 FIFORd_Di，是對 A7121 RX FIFO 讀出清除設定。

行數	說明
536~537	設置 fifo control register 中 bit EFR=0

```
/*********************************************************************
** setFreq
*********************************************************************/
void SetFreq(Uint8 ch)
{
    Uint16 tmp;

    tmp = Freq_Base + ch;       //freq 2400MHz + offset ch
    tmp |= FreqBank[ch]<<13;    //find BNK[2:0] value
    SpiWrite(tmp, SYNTH1_REG);
}
```

功能說明：副程式 SetFreq，是設定工作頻率的程序。

行數	說明
547	Freq_Base 值+ch 值，如 SetFreq(50)即設定 2450 頻率
548	查表該頻率的正確 vco bank 值
549	寫入 synthesizer control register I


```
/*********************************************************************
** SetTRC
*********************************************************************/
void SetTRC(Uint8 trc)
{
        if (trc)
        {
        ModeCtlReg |= 0x0008;
        SpiWrite(ModeCtlReg, MODE_CONTROL_REG);//set TRC=1
    }
    else
        {
            ModeCtlReg &= (~0x0008);
            SpiWrite(ModeCtlReg, MODE_CONTROL_REG);//set TRC=0
        }
}
```

功能說明：副程式 SetTRC，是對 A7121 mode control register 中 TRC bit 的設定。

行數	說明
557	判別傳入參數 trc 值。Trc=1，設定 TX 模式。Trc=0，設定 RX 模式。
559~560	設置 mode control register 中 bit TRC=1，設定在 TX 模式。A7121 真正進入 TX 狀態，須先切換 MS1 由 0 到 1 的設定動作
564~565	設置 mode control register 中 bit TRC=0，設定在 RX 模式。A7121 真正進入 RX 狀態，須先切換 MS1 由 0 到 1 的設定動作

```
/*********************************************************************
** TxPacket
*********************************************************************/
void TxPacket(void)
{
    Uint8 i;

    //write data to fifo
        FIFOWr_En();
    for (i=0; i<64; i++)
        FIFOWrite(PN9_Tab[i]);
    FIFOWr_Di();

    SetFreq(52);                        //freq 2452MHz
    SetTRC(1);                  //set bit TRC=1
```

```
584     Ms1Pin = 1;                 //enter Tx mode
585     Delay10us(6);               //delay 60us for tx settling time
586
587     CdTxenPin = 1;              //start modulation
588       while(!FpRdy);  //wait until FP_RDY=1
589     delay();            //delay>4us
590       CdTxenPin = 0;
591     Ms1Pin = 0;         //quit Tx mode, enter synthesizer mode
592   }
```

功能說明：副程式 TxPacket，是傳送 data 的程序動作。

行數	說明
577	TX FIFO 寫入動作置能
578~579	對 TX FIFO 寫入 64 bytes 的 PN9 code 的資料
580	TX FIFO 寫入動作清除
582	設定工作頻率 2452MHz
583	設定在 TX 模式
584	設置 MS1 pin ，進入 TX 狀態
585	延遲 Max 60us 的 Tx settling time
587	設置 CD_TXEN pin，A7121 開始發送 data
588	等待 FP_RDY pin 是否爲 1，TX FIFO 資料傳送完成
589	延遲 > 4us 的緩衝時間
590	設置 CD_TXEN pin=0，停止傳送。
591	結束 TX 狀態，進入 synthesizer mode

```
594   /*********************************************************************
595   ** RxPacket
596   *********************************************************************/
597   void RxPacket(void)
598   {
599       Uint8 i;
600       Uint8 RecvData;
601       Uint8 tmp;
602
603       Err_BitCnt=0; //clear Err_BitCnt
604         FIFORd_En(); //rx fifo enable
605       for (i=0; i<64; i++)
606       {
607         RecvData = FIFORead();
608
609         if((RecvData ^ PN9_Tab[i])!=0)
610         {
611           tmp = RecvData ^ PN9_Tab[i];
612           Err_BitCnt += (BitCount_Tab[tmp>>4] + BitCount_Tab[tmp & 0x0F]);
613         }
614       }
615         FIFORd_Di(); //rx fifo disable
616   }
```

功能說明：副程式 RxPacket，是接收 data 的程序動作。

行數	說明
603	清除變數 Err_BitCnt=0
604	致能 RX FIFO 讀出動作
605~614	讀出 data ，並比較 data 的正確性，計算出 error bit
615	清除 RX FIFO 讀出動作

```
618  /*********************************************************************
619  ** WaitRx
620  *********************************************************************/
621  void WaitRx(void)
622  {
623          SetFreq(50);     //freq 2450MHz
624          SetTRC(0);       //set bit TRC=0
625          Ms1Pin = 1;      //enter Rx mdoe
626          while(FpRdy==0); //wait FP_RDY=1
627          Ms1Pin = 0;      //exit Rxmode, enter synthesizer mode
628  }
```

功能說明：副程式 WaitRx，是設定 A7121 進入 RX 狀態，等待 pin FP_RDY=1 完成接收 data 的程序動作。

行數	說明
623	設定工作頻率 2450MHz
624	設定在 RX 模式
625	致能 pin MS1，進入 RX 狀態
626	等待 pin FP_RDY=1，接收資料已收妥
627	離開 RX 狀態，進入 synthesizer mode

```
630  /*********************************************************************
631  ** WaitAck
632  *********************************************************************/
633  Uint8 WaitAck(void)
634  {
635          SetFreq(50);          //freq 2450MHz
636          SetTRC(0);            //set bit TRC=0
637          Ms1Pin = 1;                    //enter Rx mdoe
638
639          Rst_Timer0();         //reload timer0
640          TR0=1;                //start timer0
641          while(TimeoutFlag==0 && FpRdy==0); //wait timerout or data ready
642          if (TimeoutFlag==1)
643          {
644                  Ms1Pin=0;             //exit rx mode
645                  return 0;     //return 0
646          }
647
648      TR0=0;                    //stop timer0
649      Ms1Pin = 0;               //exit rx mode
650      return 1;                 //return 1
651  }
```

功能說明：副程式 WaitAck，是設定 A7121 進入 RX 狀態，等待 pin FP_RDY=1 完成接收 data 的程序，同時啓動 timer0。如超過時間 20ms ，傳回 0 值。如已有資料進入，傳回 1 值

行數	說明
635	設定工作頻率 2450MHz
636	設定在 RX 模式
637	進入 RX 狀態
639	Reset Timer0 的設置
640	啓動 Timer0
641	判別 pin FP_RDY 是否爲 1 ，或 Timeout 發生。
642	判別是否 timeout 已發生
643~646	Timeout 時，離開 RX 狀態，傳回 0 值，表示接收失敗
648~650	資料已收妥，停止 timer0 計時，離開 RX 狀態，傳回 1 值，表示接收成功

```
653  /*********************************************************************
654  ** MeasureRSSI
655  *********************************************************************/
656  void MeasureRSSI(void)
657  {
658          Uint16 tmp;
659          Uint8 rssiValue;
660
661          Ms1Pin = 1;                  //enter rx mode
662          Delay10us(6);   //delay 60us for pll settling time
663
664          SpiWrite(0x0005, CALIBRATION_CONTROL2_REG); //ERSS=1
665          do
666          {
667                  tmp = SpiRead(CALIBRATION_CONTROL2_REG);
668                  tmp &= 0x0004;
669          }
670          while (tmp!=0); //check bit ERSS ok?
671          Ms1Pin = 0;     //exit rx mode
672          rssiValue = (SpiRead(RSSI_REG) & 0xFF00)>>8;
673          _nop_();
674  }
```

功能說明：副程式 MeasureRSSI，是設定 A7121 做 RSSI 的量測程序。

行數	說明
661	致能 pin MS1，進入 RX 狀態
662	延遲 Max 60us 的 Rx settling time
664	設置 calibration control register 中 bit ERSS=1，啓動 RSSI 量測
665~670	讀出 calibration control register 中 bit ERSS 是否爲 0，RSSI 量測動作完成
671	離開 Rx 狀態
672	讀出 RSSI register，存入變數 rssiValue

附件一：

n 4 wire serial configuration：

A7121 control register 的控制係藉由簡單的4線串列相容的介面操作讀出或寫入資料（SPI_CS, SPI_CLK, SPI_RXD, SPI_TXD）。

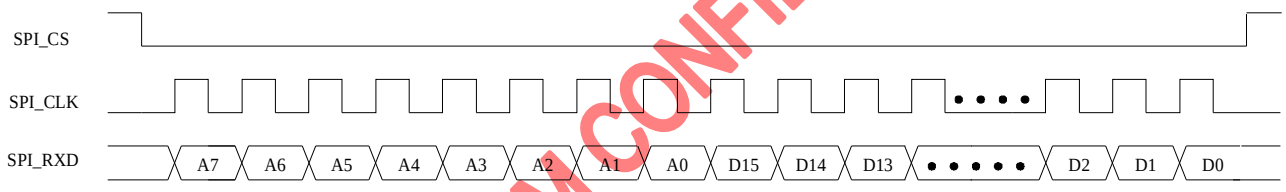
1. SPI format:

Address Byte(8 bits)								Data words(16 bits)													
R/W	Address					Reserved		Data													
7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

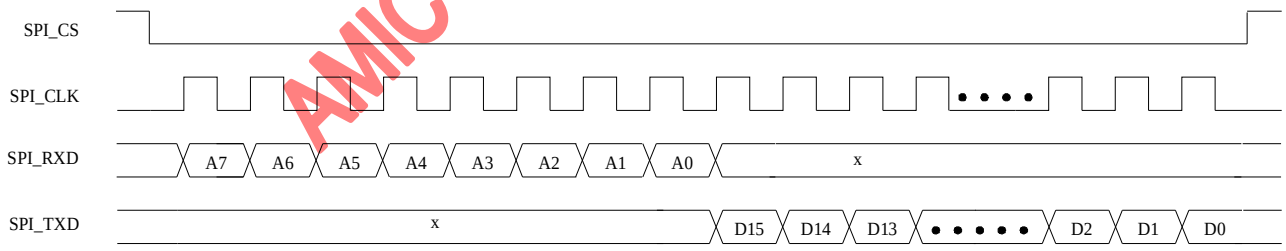
Address bytes:
Bit 7: R/W bit, 1-寫data至register, 0- 從register讀出data.
Bit [6:2]: Register Address.
Bit 1:0]: Reserved.

Data words:
Bit[15:0]: data, 請參考datasheet設置.

2. SPI timing chart:



SPI writr operation



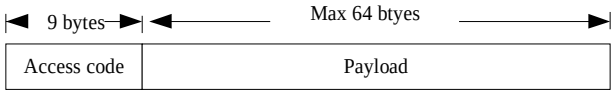
SPI read operatiom

附件二：

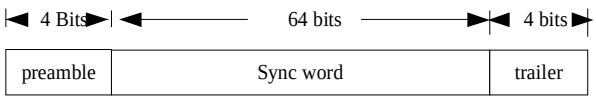
n FIFO read/write configuration：

A7121 RF chip 內建 TX 和 RX FIFO register，各自擁有 FIFO 長度 64 bytes。TX FIFO 僅能寫入 data，Rx FIFO 僅能讀出 data。寫入和讀出係由 pin F_CLK，TRXD（TXD）操作介面完成。

1. FIFO mode transmit packet format：



Packet format



Access code format

1) Access code: access code 長度共 9 bytes。
2) Payload: 長度由 FIFO control register 中 FBC[5:0]設定。
3) Sync word: A7121RF IC 在接收模式下，係以 sync word 8 bytes 爲比對條件。可由 Rx control register I 中 bit ETH[2:0]設定接收容許 sync word 錯誤的 bit 數，建議值是 6bits。如條件成立時，pin RX_SYN 會設爲 high。如需再次接收下一 packet，須將 pin MS1 pull low 後，再 pull high，完成一次切換動作， 再一次進入 Rx 模式，等待下一次有相同的 sync word。
4) 資料傳送時間計算：
   T = (access code 72 bits) + (Payload total bits) * (1 / Data rate)

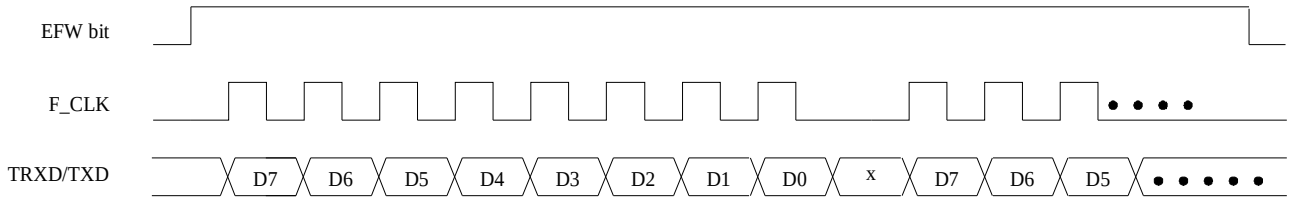
2. TX FIFO write:

a) FIFO control register 中 bit EFW 設置爲 1 後,才可執行 data 寫入動作。
b) FIFO control register 中 FBC[5:0]決定寫入 data bytes 最大長度。如寫入動作超過 FBC[5:0]的設定 Bytes 時，之後所寫的 data 將不會寫入 TX FIFO。
c) 重新寫入 data 至 TX FIFO，須設置 bit EFW 爲 0，作一重置動作，再設置 bit EFW 爲 1，再執行寫入動作。
d) FIFO 的 data 可從 TRXD 或 TXD pin 寫入，由 mode control register 中 bit TRD 決定。 Bit TRD=1，data 由 TRXD pin 寫入 TX FIFO。TRD=0，data 由 TXD pin 寫入 TX FIFO。

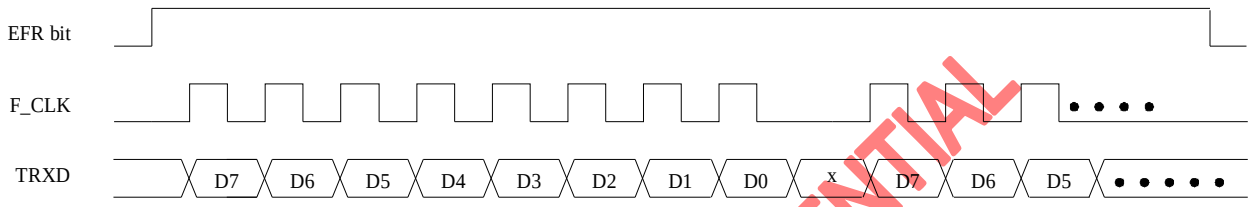
3. RX FIFO read:

a) FIFO control register 中 bit EFR 設置爲 1 後,才可執行 data 讀出動作。
b) FIFO control register 中 FBC[5:0]決定讀出 data bytes 最大長度。如讀出動作超過 FBC[5:0]的設定 Bytes 時， data 將不會有所輸出。
c) 重新從 RX FIFO 讀出 data ，須設置 bit EFR 爲 0，作一重置動作，再設置 bit EFR 爲 1，再執行讀出動作。
d) RX FIFO 的 data 僅能從 TRXD pin 讀出，不論 mode control register 中 bit TRD=1 或 0。

4. FIFO timing chart:



TX FIFO write timing chart



RX FIFO Read timing chart

附件三：

n Mode of operation:

1. A7121 mode 的決定由 pin MS0, pin MS1, SYN bit, TRC bit 設定.

MS0	MS1	SYN	TRC	Operation mode
0	x	x	x	Sleep mode
1	0	0	x	Standby mode
1	0	1	x	Synthesizer mode(PLL on)
1	1	1	1	TX mode
1	1	1	0	RX mode

ü 使用 I/O pin MS0,MS1 作 mode control，請參照上表設置。

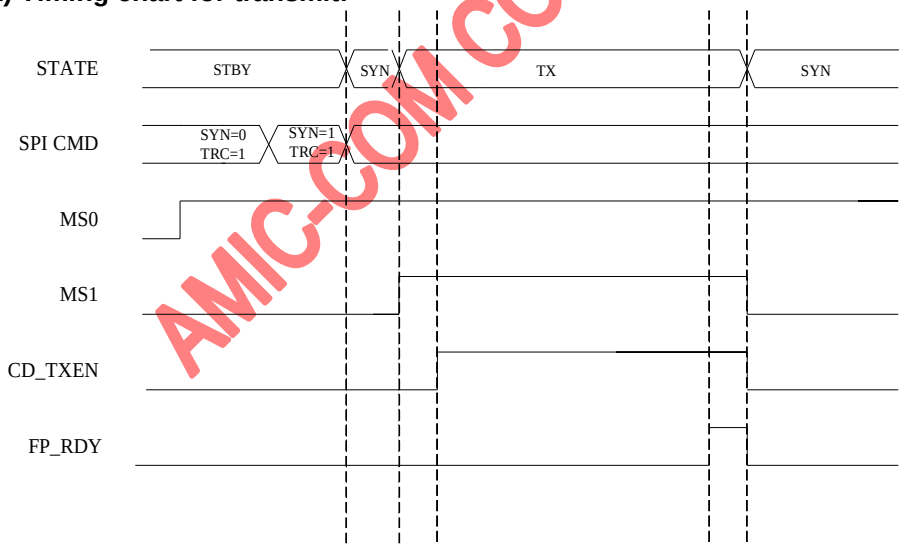
ü 如不使用 I/O pin 控制 MS0, MS1，可使用 SPI 介面設定 mode control register 中 CE bit 視爲 MS0，calibration control register II 中 ETR bit 視爲 MS1。Pin MS0, MS1 在外部電路須 Pull high.

2. Mode change timing information:

Standby -> Synthesizer	Max. 150us (PLL settling time)
Synthesizer -> TX or RX	Max. 60us (change TRX settling time)
TX -> RX	Max. 210us (change TRX settling time)
Rx -> TX	Max. 210us

3. Timing chart for FIFO mode:

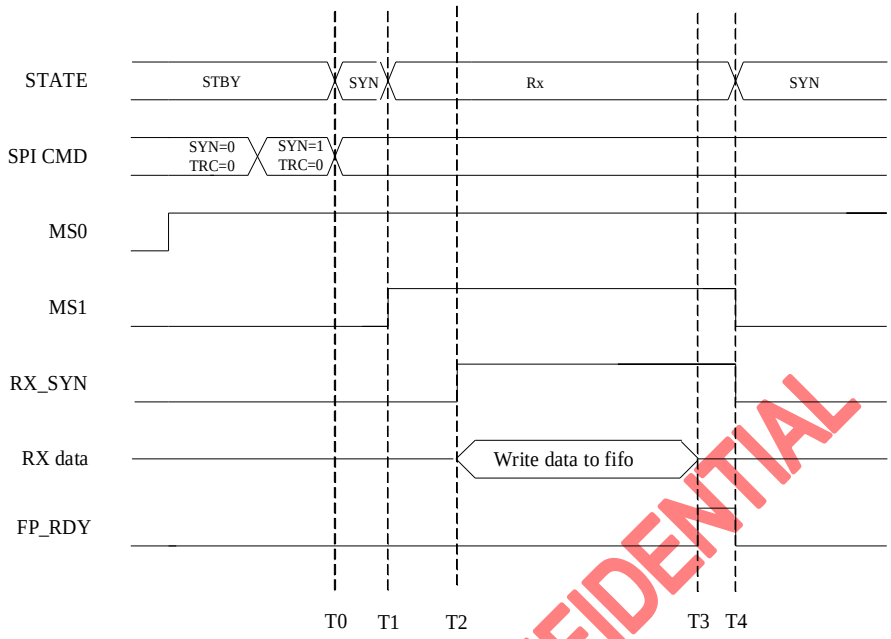
1) Timing chart for transmit:



T0: synthesizer mode enabled
T1: MS1 pin enabled
T2: CE_TXEN pin enabled
T3: Transmitted data ready
T4: end of Tx mode

T1-T0: PLL settling time
T2-T1: change to Tx settling time

2) Timing chart: for receiving:



T0: synthesizer mode enabled
T1: MS1 pin enabled
T2: correct sync word found
T3: received data ready
T4: end of Rx mode

T1-T0: PLL settling time
T2-T1: change to Rx settling time