



A7125 Hopping Reference code for FIFO mode

RC_A7125_01

Document Title

A7125 Hopping reference code for FIFO mode

Revision History

<u>Rev. No.</u>	<u>History</u>	<u>Issue Date</u>	<u>Remark</u>
0.0	Preliminary	Nov.18, 2008	
1.0	Modify initial value(Rcosc3,Tx_test,Crystal)	Dec.22, 2008	
1.2	Modify initial configuration value QDS=1, XCP[1:0]=[00], IGFAQ[2:0]=[111], IGFI[2:0]=[111], PDL[1:0]=[00], WSEL[2:0]=[011]	May 20, 2009	
1.3	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. Introduction.....	3
2. Systematic summary	3
3. Hardware	4
3.1 System block diagram.....	4
4. Firmware Program.....	5
4.1Introduction	5
4.2 Example State siagram.....	6
5. Explanation of reference code.....	7

AMICCOM CONFIDENTIAL

RF Chip-A7125 Hopping Reference code for FIFO mode

1. Introduction

This document describes development of simple example procedures by A7125 FIFO mode. It could support user how to implement two-way radio and how to initial A7125.

2. Systematic summary

This example procedure simple to use frequency hopping, the following picture:

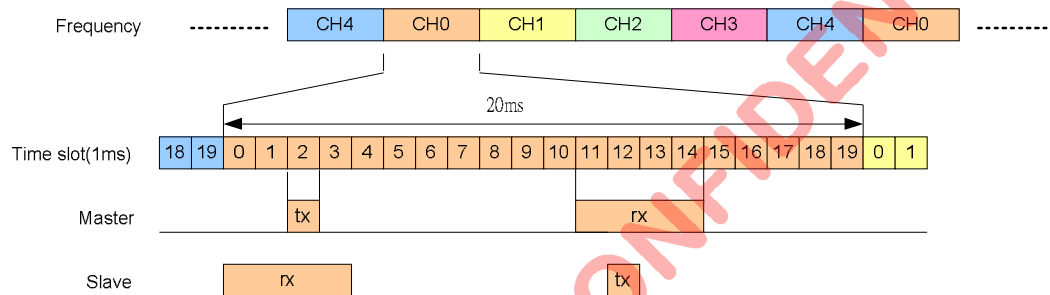


Fig1. Hopping timing chart

The procedure is divided into two parts, one is Master, and another one is Slave.

Master side : Waiting time slot=2 after power on and initial RF chip procedure, Master will deliver 64 bytes data from TX FIFO, then waiting time slot=11, jump into RX state to wait ACK data from Slave. If received, the automatic change next time operating frequency sequence, again another time succession periodic motion. If not received, master side change next time operating frequency sequence, again another time slot period.

Slave side : Waiting time slot=0 after power on and initial RF procedure, Slave enters into RX state for receiving data from Master. If not received from Master, Slave change next time operating frequency sequence to wait next time operating period. If had still not received data to have 5 cycles, then the stop frequency-hopping mechanism, and returns to the initial operating frequency, enters the RX state waiting receive. If has receives the data which the Master side transmits, then starts the frequency-hopping mechanism, completes TX and the RX work according to the time slot.

Based on the sample procedures between Master and Slave, user can learn how to implement two-way radio as well as how to calculate BER (bit error rate).

3. Hardware

3.1 System block diagram

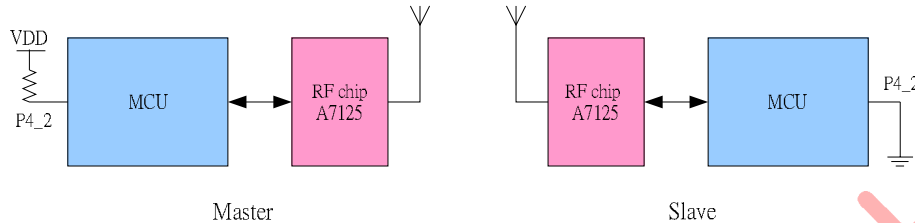


Fig2. System block diagram

MCU uses I/O pin 4 _ 2 to identify the Master and Slave.

MCU I/O Pin Definition:

The example is explanation how to use the I/O:

SCS, SCK, SDIO -3 wire serial interface to access A7125 register.

GIO1 - The control signal that FIFO movements finish, MCU can measure this pin and convey or receive packet to finish.

I/O that MCU controls A7125 RF chip disposes the following picture:

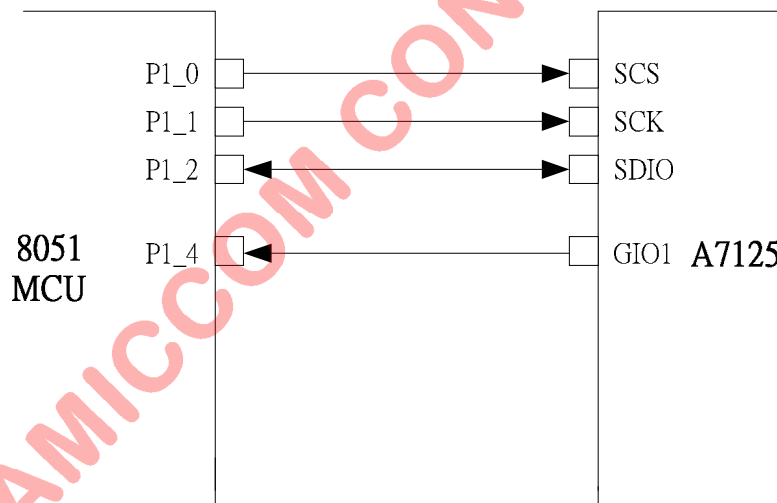


Fig3. Connections between 8051 MCU and A7125

4. Firmware Program

4.1 Introduction

After power on reset, MCU do initialization of its Timer0 and Uart0 as well as A7125. Then, MCU check its Port 4_2 to identify Master or Slave. If Port 4_2 = 1, MCU executes Master code in the main program; else, MCU executes Slave code in the main program.

Master code : (Use frequency=2450.001MHz)

- 1) Wait for timer = 2.
- 2) MCU asks A7125 to enter TX State to deliver 64 byte PN9 code. After done, A7125 is auto back to Standby state.
- 3) Wait for timer = 12.
- 4) Into RX state.
- 5) If Timer > 14, RF doesn't receive 64 bytes data with PN9 code, it will exit RX state and value of seq adds by one.
- 6) Once A7125 receives the packet, A7125 will be auto back to Standby state.
- 7) MCU compares received 64 bytes data with PN9 code and calculates BER (Bit Error Rate).
- 8) Value of seq adds by one, then back to step (1).
- 9) For each 500 ms, MCU reports BER to personal computer.

Slave code : (Use frequency=2448.001MHz)

- 1) Wait timer = 0;
- 2) MCU asks A7125 to enter RX state to receive 64 byte data from Master.
- 3) If timer > 3, RF doesn't receive 64 bytes data with PN9 code, it will exit RX state, value of seq and Err_HopCnt adds by one.
- 4) If Err_HopCnt <5, then back to step (1).
- 5) if Err_HopCnt >5, the parameter seq value, and Err _ HopCnt value is 0. Use seq =0 operating frequencies enter RX state, wait for the package to receive.
- 6) Once A7125 receives the packet, A7125 will be auto back to Standby state.
- 7) MCU compares received 64 bytes data with PN9 code and calculates BER (Bit Error Rate).
- 8) Wait for timer = 12, then MCU asks A7125 to enter TX State to deliver 64 byte PN9 code. After done, A7125 is auto back to Standby state.
- 9) Back to step (1).
- 10) For each 500 ms, MCU reports BER to personal computer.

4.2 Example State diagram

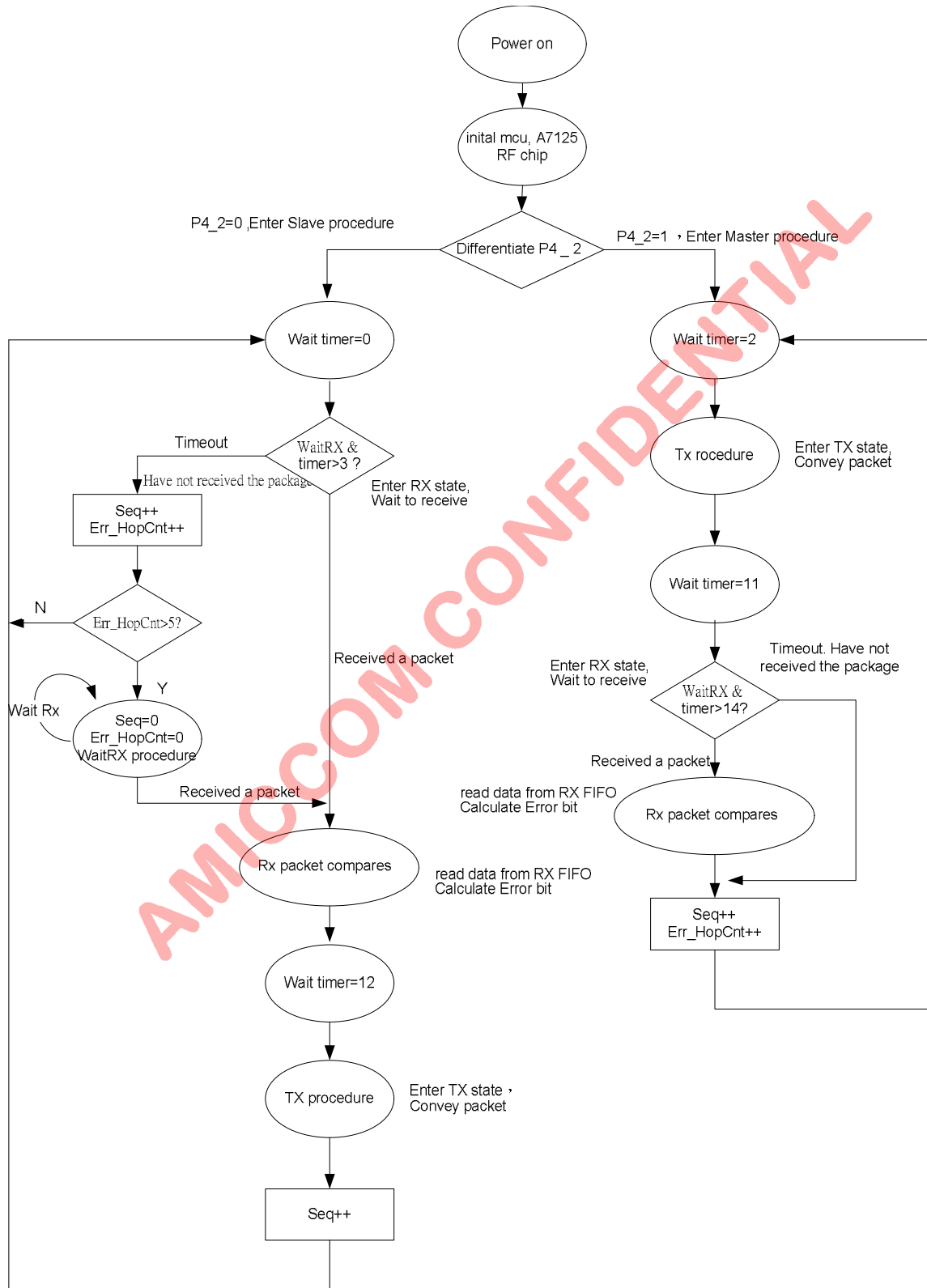


Fig4 Example State diagram

5. Explanation of reference code

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	**
15	**
16	**
17	*****/
18	#include "define.h"
19	#include "w77le58.h"
20	#include "a7125reg.h"
21	#include "Uti.h"
Function : Include file declaration , Define the constant parameter	
Line	Description
18~21	Include the link files

23	*****
24	** I/O Declaration
25	*****/
26	#define SCS P1_0 //spi SCS
27	#define SCK P1_1 //spi SCK
28	#define SDIO P1_2 //spi SDIO
29	#define CKO P1_3 //CKO
30	#define GIO1 P1_4 //GIO1
31	#define GIO2 P1_5 //GIO2
32	#define Button P1_7 //test Button
33	
34	*****
35	** Constant Declaration
36	*****/
37	#define TIMEOUT 50
38	#define t0hrel 1000
Function : Define the I/O Port of A7125 RF chip by MCU , Define the constant parameter	
Line	Description
26~32	MCU I/O definition
37~38	Define the constant parameter

40 /*****

41 ** Global Variable Declaration

42 *****/

43 Uint8 data timer;

44 Uint16 idata RxCnt;

45 Uint32 idata Err_ByteCnt;

46 Uint32 idata Err_BitCnt;

47 Uint16 idata TimerCnt0;

48 Uint8 data *Uartptr;

49 Uint8 data UartSendCnt;

50 Uint8 data CmdBuf[12];

51 Uint8 xdata tmpbuf[64];

52 Uint8 idata Err_Frame;

53 Uint8 data ReportTimeOut;

54 Uint8 data Seq;

55 Uint8 data Err_HopCnt;

56

57 const Uint8 code BitCount_Tab[16] = {0,1,1,2,1,2,2,3,1,2,2,3,2,3,3,4};

58 const Uint8 code ID_Tab[4]={0x54,0x75,0xC5,0x2A}; //ID code

59 const Uint8 code PN9_Tab[]=

60 { 0xFF,0x83,0xDF,0x17,0x32,0x09,0x4E,0xD1,

61 0xE7,0xCD,0x8A,0x91,0xC6,0xD5,0xC4,0xC4,

62 0x40,0x21,0x18,0x4E,0x55,0x86,0xF4,0xDC,

63 0x8A,0x15,0xA7,0xEC,0x92,0xDF,0x93,0x53,

64 0x30,0x18,0xCA,0x34,0xBF,0xA2,0xC7,0x59,

65 0x67,0x8F,0xBA,0x0D,0x6D,0xD8,0x2D,0x7D,

66 0x54,0x0A,0x57,0x97,0x70,0x39,0xD2,0x7A,

67 0xEA,0x24,0x33,0x85,0xED,0x9A,0x1D,0xE0

68 }; // This table are 64bytes PN9 pseudo random code.

Function : Declaration of the parameter

Line	Description
43~55	Declare parameters that are used in the procedure
57	Declare BitCount_Tab
58	Declare ID_Tab for ID code
59~68	Declare PN9_Tab of 64 bytes PN9 code


```

70 const Uint16 code A7125Config[]=
71 {
72     0x00, //MODE register,          only reset, not use on config
73     0x42, //MODE CTRL register,
74     0x00, //CALIBRATION register,
75     0x3F, //FIFO1 register,
76     0x00, //FIFO2 register,
77     0x00, //FIFO register,          for fifo read/write
78     0x00, //IDDATA register,        for idcode
79     0x00, //RCOSC1 register,
80     0x00, //RCOSC2 register,
81     0x00, //RCOSC3 register,
82     0x00, //CKO register,
83     0x01, //GPIO1 register
84     0x01, //GPIO2 register,
85     0x1F, //DATARATE register,
86     0x50, //PLL1 register,
87     0x0E, //PLL2 register,          RFbase 2400.001MHz
88     0x96, //PLL3 register,
89     0x00, //PLL4 register,
90     0x04, //PLL5 register,
91     0x3C, //chanel group I,
92     0x78, //chanel group II,
93     0xD7, //TX1 register,
94     0x40, //TX2 register,
95     0x10, //DELAY1 register,
96     0x61, //DELAY2 register,
97     0x62, //RX register,
98     0xA0, //RXGAIN1 register,
99     0x00, //RXGAIN2 register,
100    0x00, //RXGAIN3 register,
101    0xD2, //RXGAIN4 register,
102    0x00, //RSSI register,
103    0xE2, //ADC register,
104    0x07, //CODE1 register,
105    0x56, //CODE2 register,
106    0x2A, //CODE3 register,
107    0x06, //IFCAL1 register,
108    0x00, //IFCAL2 register,        only read
109    0x05, //VCOCCAL register,
110    0x44, //VCOCAL1 register,
111    0x80, //VCOCAL2 register,
112    0x30, //VCO DEV Cal. I register,
113    0x20, //VCO DEV Cal. II register,
114    0x80, //VCO DEV Cal. III register,
115    0x00, //VCO Mod. delay register
116    0x7A, //BATTERY register,
117    0x2F, //TXTEST register,
118    0x47, //RXDEM1 register,
119    0x80, //RXDEM2 register,
120    0xF1, //CPC1 register,
121    0x11, //CPC2 register,
122    0x04, //CRYSTAL register,
123    0x45, //PLLTTEST register,
124    0x18, //VCOTEST register,
125    0x10, //RF Analog Test
126    0xFF, //IFAT register,
127    0x37, //Channel select register,
128    0xFF //VRB register
129 };

```

Function : Configure of A7125's control registers for Master

Line	Description
------	-------------

72~128	Configure of A7125's control registers for Master
--------	---

```

131 const Uint8 HopTab[]=
132 {
133     20, //2410
134     40, //2420
135     80, //2440
136     120, //2460
137     160 //2480
138 };

```

Function : Declare Hopping table

Line	Description
133~137	Define 5 channels by oneself.

```

140 /*****
141 ** function Declaration
142 *****/
143 void InitTimer0(void);
144 void initUart0(void);
145 void Timer0ISR (void);
146 void Uart0Isr(void);
147 void A7125_Reset(void);
148 void A7125_WriteReg(Uint8, Uint8);
149 Uint8 A7125_ReadReg(Uint8);
150 void ByteSend(Uint8 src);
151 Uint8 ByteRead(void);
152 void A7125_WriteID(void);
153 void A7125_WriteFIFO(void);
154 void initRF(void);
155 void A7125_Config(void);
156 void A7125_Cal(void);
157 void RxPacket(void);
158 void StrobeCmd(Uint8);
159 void SetCH(Uint8);
160 void CHGroupCal(Uint8);
161 void ReportData(void);

```

Function : Subroutine declaration

Line	Description
143~161	Subroutine declarations

```

163 /*****
164  * main loop
165  *****/
166 void main(void)
167 {
168     //initSW
169     PMR |= 0x01; //set DME0
170
171     //initHW
172     P0 = 0xFF;
173     P1 = 0xFF;
174     P2 = 0xFF;
175     P3 = 0xFF;
176     P4 = 0x0F;
177
178     InitTimer0();
179     initUart0();
180     TR0=1; //timer0 on
181     EA=1; //enable interrupt
182
183     if ((P4 & 0x04)==0x04) //if P4.2=1, master
184     {
185         initRF(); //init RF
186
187         StrobeCmd(CMD_STBY);
188         A7125_WriteFIFO(); //write data to tx fifo
189         Seq=0;
190
191         while(1)
192         {
193             //Tx time-slot
194             while(timer != 2); //wait until timer=2
195             SetCH(HopTab[Seq]);
196             StrobeCmd(CMD_TX); //entry tx & transmit
197             while(GIO1); //wait transmit completed
198
199             //Rx time-slot
200             while(timer !=11); //wait until timer=11
201             SetCH(HopTab[Seq]-4);
202             StrobeCmd(CMD_RX);
203             while(GIO1 && timer <= 14);
204
205             if (timer >14)
206             {
207                 //timeout
208                 StrobeCmd(CMD_STBY);
209             }
210             else
211             {
212                 //data procedure
213                 RxPacket();
214             }
215
216             Seq++;
217             if (Seq > 4)
218                 Seq = 0;
219         }
220     }

```

```

221 else //if P4.2=0, slave
222 {
223     initRF(); //init RF
224
225     StrobeCmd(CMD_STBY);
226     A7125_WriteFIFO(); //write data to tx fifo
227
228     RxCnt = 0;
229     Err_ByteCnt = 0;
230     Err_BitCnt = 0;
231
232     Seq=0;
233     Err_HopCnt=0;
234
235     while(1)
236     {
237         if (P1_7==0)
238         {
239             RxCnt = 0;
240             Err_BitCnt = 0;
241         }
242
243         //Rx time-slot
244         while(timer!=0);
245         SetCH(HopTab[Seq]-4);
246         StrobeCmd(CMD_RX);
247         while(GIO1 && timer <= 3); //wait receive completed
248         if (timer > 3)
249         {
250             StrobeCmd(CMD_PLL);
251
252             Seq++;
253             if (Seq > 4)
254                 Seq = 0;
255
256             Err_HopCnt++;
257             if (Err_HopCnt > 5)
258             {
259                 Seq = 0;
260                 Err_HopCnt = 0;
261
262                 SetCH(HopTab[Seq]-4);
263                 StrobeCmd(CMD_RX);
264                 while(1)
265                 {
266                     if (GIO1==0)
267                     {
268                         break;
269                     }
270                 }
271             }
272             else
273             {
274                 continue;
275             }
276         }
277
278         timer = 2; //reSync
279         TF0 = 0; // Clear Timer0 interrupt
280         TH0 = (65536-t0hrel)>>8; // Reload Timer0 high byte,low byte
281         TL0 = 65536-t0hrel;

```

```

282
283     RxPacket();
284
285     //Tx time-slot
286     while(timer != 12);
287     SetCH(HopTab[Seq]);
288     StrobeCmd(CMD_TX);
289     while(GIO1);
290
291     Seq++;
292     if (Seq > 4)
293         Seq = 0;
294 }
295 }
296 }

```

Function : Main loop • MCU Identifies Port4_2 = 1 , row 183~220 are Master code. Row 221~296 are Slave code.

Line	Description
169	To start the MCU on chip data SRAM
172~176	Initial MCU I/O Port
178	Call initTimer0 subroutine and enable interrupt
179	Call initUart0 subroutine and initial Uart0
180~181	Enable Timer0 and Timer0 interrupt
183	Identify if port4_2=1, jump to Master code, else jump to Slave code.
184~220	Master code
185	Call initRF subroutine to initial A7125
187	Use Strobe command to enter Standby mode
188	Set the value of parameter seq=0
194	Wait timer=2
195	Call SetCH subroutine and Check the table to sets up the operating frequency in accordance with HopTab
196	User Strobe command to enter TX mode
197	Monitor status of A7125's GIO1 pin for A7125's WTR output.
200	Wait timer=11
201	Call SetCH subroutine and Check the table to sets up the operating frequency in accordance with HopTab
202	User Strobe command to enter RX mode
203	Wait for received data or timer>14
205~209	Check Timer >14 , if Yes jump to row User Strobe command to enter PLL state
211~214	Call RxPacket subroutine to read data from RX FIFO , do authentication and calculate BER
216~218	The parameter seq is added by 1, whether it is greater than 4 to differentiate the parameter seq. In this way, it is to be clear seq = 0.
222~295	Slave code
223	Call initRF subroutine to initial A7125
225	Use Strobe command to enter Standby mode
226	Call A7125_WriteFIFO subroutine to write data into TX FIFO
228~233	Clean the variable of RxCnt, Err_ByteCnt, Err_BitCnt,seq, Err_HopCnt.
237~241	Identify if MCU pin P1_7=0.Yes, Clean the variable of seq, Err_HopCnt
244	Wait timer=0
245	Call SetCH subroutine and Check the table to sets up the operating frequency in accordance with HopTab
246	User Strobe command to enter RX mode
247	Wait for received data or timer>3
248	Whether it is greater than 3 to differentiate the parameter timer
250	Use Strobe command to enter PLL mode

252~254	The parameter seq is added by 1, whether it is greater than 4 to differentiate the parameter seq. In this way, it is to be clear seq = 0.
256	Parameter Err _ HopCnt adds by 1
257	Differentiate Err _ Is HpoCnt greater than 5
259~260	Clean the variable of seq,Err_HopCnt.
262	Call SetCH subroutine and Check the table to sets up the operating frequency in accordance with HopTab
263	User Strobe command to enter RX mode
264~271	Wait for received data
274	Parameter Err _ HopCnt value without greater than 5 times than gets back to 237 line and Job of an another one cycle.
278	Set timer = 2
279~281	Clear TF0 Flage, reset the value of TH0, TL0.
283	Call RxPacket subroutine to read data from RX FIFO , do authentication and calculate BER
286	Wait timer=12
287	Call SetCH subroutine and Check the table to sets up the operating frequency in accordance with HopTab
288	Use Strobe command to enter Tx mode and A7125 deliver TX FIFO automatically.
289	Monitor status of A7125's GIO1 pin for A7125's WTR output.
291~293	The parameter seq is added by 1, whether it is greater than 4 to differentiate the parameter seq. In this way, it is to be clear seq = 0.

```

298 /*****
299 ** init Timer0
300 *****/
301 void InitTimer0(void)
302 {
303     TR0 = 0;
304     TMOD =(TMOD & 0xF0)|0x01; //timer0 mode=1
305     TH0 = (65536-t0hrel)>>8; // Reload Timer0 high byte,low byte
306     TL0 = 65536-t0hrel;
307     TF0 = 0; // Clear any pending Timer0 interrupts
308     ET0 = 1; // Enable Timer0 interrupt
309 }

```

Function : Initial Timer0

Line	Description
303	Disable Timer0
304	Setup Timer0 in mode1 mode
305~306	Setup the initial value of TH0,TL0
307	Clean Timer0 interrupt flag
308	Enable Timer0 interrupt

```

311 /*****
312 ** Timer0ISR
313 *****/
314 void Timer0ISR (void) interrupt 1
315 {
316     TF0 = 0; // Clear Timer0 interrupt
317     TH0 = (65536-t0hrel)>>8; // Reload Timer0 high byte,low byte
318     TL0 = 65536-t0hrel;
319
320     timer++;
321     if (timer>=20)
322     {
323         timer=0;
324         P3_5= ~P3_5;
325     }
326
327     TimerCnt0++;
328     if (TimerCnt0 == 500)
329     {
330         TimerCnt0=0;
331         CmdBuf[0]=0xF1;
332
333         memcpy(&CmdBuf[1], &RxCnt, 2);
334         memcpy(&CmdBuf[3], &Err_ByteCnt, 4);
335         memcpy(&CmdBuf[7], &Err_BitCnt, 4);
336         memcpy(&CmdBuf[11], &Err_Frame, 1);
337
338         UartSendCnt=12;
339         Uartptr=&CmdBuf[0];
340         SBUF=CmdBuf[0];
341     }
342 }

```

Function : Timer0 interrupt service routine

Line	Description
316~318	Clear Timer0 interrupt flag and setup initial value of TH0 and TL0.
320	Increase timer (timer is a variable).
321~325	Check timer == TIMEOUT, if yes, set TimeoutFlag =1.
327	Increase timer (timer is a variable).
328	Check TimerCnt0 == 500, if yes, 500ms delay is done.
330	Clear TimerCnt0
331	CmdBuf[0] is set to 0xF1 for Windows VB.
333	CmdBuf[1] is used to set up parameter RxCnt
334	CmdBuf[3] 、 CmdBuf[4] 、 CmdBuf[5] 、 CmdBuf[6] are used to set up parameter Err_ByteCnt
335	CmdBuf[7] 、 CmdBuf[8] 、 CmdBuf[9] 、 CmdBuf[10] are used to set up parameter Err_BitCnt
336	CmdBuf[11] is used to set up parameter Err_Frame.
338	Set UartSendCnt=12
339	Set pointer Uartptr to indicate location of CmdBuf [0].
340	Deliver SBUF to PC for BER information.

```

344 /*****
345 ** Init Uart0
346 *****/
347 void initUart0(void)
348 {
349     TH1 = 0xFD; //BaudRate 9600;
350     TL1 = 0xFD;
351     SCON = 0x40;
352     TMOD = (TMOD & 0x0F) | 0x20;
353     REN = 1;
354     TR1 = 1;
355     ES = 1;
356 }

```

Function : Initialize the procedure of Uart0

Line	Description
349~351	Set initial value of TL1,TH1,SCON1 and Set 9600 baud rate @xtal=11.0592MHz
352	Set Timer1 in mode 2
353~355	Set REN=1, TR1=1, ES=1 to enable UART function.

```

358 /*****
359 ** Uart0 ISR
360 *****/
361 void Uart0Isr(void) interrupt 4 using 3
362 {
363     if (TI==1)
364     {
365         TI=0;
366         UartSendCnt--;
367         if(UartSendCnt !=0)
368         {
369             Uartptr++;
370             SBUF = *Uartptr;
371         }
372     }
373 }

```

Function : Initialize the interrupt subprogram of uart0.

Line	Description
363	Check TI flag, if TI=1, one byte of UART transmission is done.
365	Clear TI flag
366	Decrease UartSendCnt (UartSendCnt is a variable)
367	Check UartSendCnt == 0, if not, continue UART transmission until done.
369~370	Increase pointer Uartptr to assign address of next data via UART to PC

```

375 /*****
376 ** Reset_RF
377 *****/
378 void A7125_Reset(void)
379 {
380     A7125_WriteReg(MODE_REG, 0x00); //reset RF chip
381 }

```

Function : A7125 RF chip Reset step

Line	Description
380	Write 0x00 in MODE register to reset A7125


```

383 /*****
384 ** WritelD
385 *****/
386 void A7125_WritelD(void)
387 {
388     Uint8 i;
389     Uint8 d1,d2,d3,d4;
390     Uint8 addr;
391
392     addr = IDCODE_REG; //send address 0x06, bit cmd=0, r/w=0
393     SCS = 0;
394     ByteSend(addr);
395     for (i=0; i < 4; i++)
396         ByteSend(ID_Tab[i]);
397     SCS = 1;
398
399     addr = IDCODE_REG | 0x40; //send address 0x06, bit cmd=0, r/w=1
400     SCS=0;
401     ByteSend(addr);
402     d1=ByteRead();
403     d2=ByteRead();
404     d3=ByteRead();
405     d4=ByteRead();
406     SCS=1;
407 }

```

Function : Write ID procedure.

Line	Description
392	Set parameter addr = 0x06 for ID code write operation.
393	Set SCS=0 to enable SPI interface.
394~396	Write ID_Tab Table into A7125 ID Code registers.
397	Set SCS=1 to disable SPI interface.
399	Set parameter addr = 0x06 for ID code read operation.
400	Set SCS=0 to enable SPI interface.
401~405	Read 4 byte ID code for ID check.
406	Set SCS=1 to disable SPI interface.

```

409 /*****
410 ** A7125_WriteReg
411 *****/
412 void A7125_WriteReg(Uint8 addr, Uint8 dataByte)
413 {
414     Uint8 i;
415
416     SCS = 0;
417
418     addr |= 0x00; //bit cmd=0,r/w=0
419     for(i = 0; i < 8; i++)
420     {
421         if(addr & 0x80)
422             SDIO = 1;
423         else
424             SDIO = 0;
425
426         SCK = 1;
427         _nop_();
428         SCK = 0;
429         addr = addr << 1;
430     }
431     _nop_();
432
433     //send data byte
434     for(i = 0; i < 8; i++)
435     {
436         if(dataByte & 0x80)
437             SDIO = 1;
438         else
439             SDIO = 0;
440
441         SCK = 1;
442         _nop_();
443         SCK = 0;
444         dataByte = dataByte << 1;
445     }
446
447     SCS = 1;
448 }

```

Function : Write a 8-bit value to specified address in A7125 control register.

Line	Description
416	Enable read operation of control registers (addr7=0 for control registers, addr6=1 for read operation)
418	Assign control register's address by input variable addr.
419~430	Assign control register's address by input variable addr.
434~445	The control register's value is output to SPI interface.
447	Set SCS=1 to disable SPI interface.

```

450 /*****
451 ** A7125_ReadReg
452 *****/
453 Uint8 A7125_ReadReg(Uint8 addr)
454 {
455     Uint8 i;
456     Uint8 tmp;
457
458     SCS = 0;
459     addr |= 0x40; //bit cmd=0,r/w=1
460     for(i = 0; i < 8; i++)
461     {
462         if(addr & 0x80)
463             SDIO = 1;
464         else
465             SDIO = 0;
466
467         _nop_();
468         SCK = 1;
469         _nop_();
470         SCK = 0;
471
472         addr = addr << 1;
473     }
474
475     _nop_();
476     SDIO = 1;
477
478     //read data
479     for(i = 0; i < 8; i++)
480     {
481         if(SDIO)
482             tmp = (tmp << 1) | 0x01;
483         else
484             tmp = tmp << 1;
485
486         SCK = 1;
487         _nop_();
488         SCK = 0;
489     }
490     SCS = 1;
491     return tmp;
492 }
493 
```

Function : Read a 8-bit value from specified address in A7125 control register.

Line	Description
458	Enable read operation of control registers (addr7=0 for control registers, addr6=1 for read operation)
459	Assign control register's address by input variable addr.
460~474	Assign control register's address by input variable addr.
477	Set SDIO=1 for SPI Data Output
480~490	The control register's value is output to SPI interface.
491	Set SCS=1 to disable SPI interface.
492	Return tmp (tmp represents 8-bits control register's value)

```

495 /*****
496 ** ByteSend
497 *****/
498 void ByteSend(UInt8 src)
499 {
500     UInt8 i;
501
502     for(i = 0; i < 8; i++)
503     {
504         if(src & 0x80)
505             SDIO = 1;
506         else
507             SDIO = 0;
508
509         _nop_();
510         SCK = 1;
511         _nop_();
512         SCK = 0;
513         src = src << 1;
514     }
515 }

```

Function : subroutine of ByteSend

Line	Description
------	-------------

502~514	Procedures how to write one byte data into A7125.
---------	---

```

517 /*****
518 ** ByteRead
519 *****/
520 UInt8 ByteRead(void)
521 {
522     UInt8 i,tmp;
523
524     SDIO = 1; //sdio pull high
525     for(i = 0; i < 8; i++)
526     {
527         if(SDIO)
528             tmp = (tmp << 1) | 0x01;
529         else
530             tmp = tmp << 1;
531
532         SCK = 1;
533         _nop_();
534         SCK = 0;
535     }
536     return tmp;
537 }

```

Function : Subroutine of ByteRead

Line	Description
------	-------------

524~535	Read 1 byte data from A7125
---------	-----------------------------

536	Return 1 byte value
-----	---------------------

```

539 /*****
540 ** Send4Bit
541 *****/
542 void Send4Bit(Uint8 src)
543 {
544     Uint8 i;
545     for(i = 0; i < 4; i++)
546     {
547         if(src & 0x80)
548             SDIO = 1;
549         else
550             SDIO = 0;
551         _nop_();
552         SCK = 1;
553         _nop_();
554         SCK = 0;
555         src = src << 1;
556     }
557 }
558
559

```

Function : Subroutine of Send4Bit

Line	Description
544~558	Procedures how to write 4 bits data into A7125, especially for Strobe command.

```

561 /*****
562 ** SetCH
563 *****/
564 void SetCH(Uint8 ch)
565 {
566     A7125_WriteReg(PLL1_REG, ch); //RF freq = RFbase + (CH_Step * ch)
567 }

```

Function : Channel setting procedure

Line	Description
566	Call subroutine of A7125_WriteReg and to set up PLL1 control register.

```

569 /*****
570 ** initRF
571 *****/
572 void initRF(void)
573 {
574     //init io pin
575     SCS = 1;
576     SCK = 0;
577     SDIO = 1;
578     CKO = 1;
579     GIO1 = 1;
580     GIO2 = 1;
581
582     A7125_Reset(); //reset A7125 RF chip
583     A7125_WritelD(); //write ID code
584     A7125_Config(); //config A7125 chip
585     A7125_Cal(); //calibration IF,VCO,VCO
586 }

```

Function : Initial RF chip procedure

Line	Description
575~580	Set up I/O initial value between MCU and A7125
582	Call A7125_Reset subroutine to reset A7125
583	Call A7125_WritelD subroutine to write 4 byte ID code.
584	Call A7125_Config subroutine to initial A7125's control registers.
585	Set up I/O initial value between MCU and A7125

```

588 /*****
589 ** A7125_WriteFIFO
590 *****/
591 void A7125_WriteFIFO(void)
592 {
593     Uint8 i;
594     Uint8 cmd;
595
596     cmd = FIFO_REG; //send address 0x05, bit cmd=0, r/w=0
597     SCS=0;
598     ByteSend(cmd);
599     for(i=0; i <64; i++)
600         ByteSend(PN9_Tab[i]);
601     SCS=1;
602 }

```

Function : write Tx FIFO procedure

Line	Description
596	Assign address = 0x05 for TX FIFO of write operation.
597	Set SCS=0 to enable SPI interface.
598	Call ByteSend subroutine for one byte write operation.
599~600	Write PN9_Tab into TX FIFO, total 64 bytes.
601	Set SCS=1 to disable SPI interface.

```

604 /*****
605 ** Strobe Command
606 *****/
607 void StrobeCmd(Uint8 cmd)
608 {
609     SCS = 0;
610     Send4Bit(cmd);
611     SCS = 1;
612 }

```

Function : Strobe command procedure.

Line	Description
609	Set SCS=0 to enable SPI interface
610	Call Send4Bit for 4-bits subroutine Strobe command.
611	Set SCS=1 to disable SPI interface.

```

614 /*****
615 ** RxPacket
616 *****/
617 void RxPacket(void)
618 {
619     Uint8 i;
620     Uint8 recv;
621     Uint8 tmp;
622     Uint8 cmd;
623
624     RxCnt++;
625     cmd = FIFO_REG | 0x40; //address 0x05, bit cmd=0, r/w=1
626
627     SCS=0;
628     ByteSend(cmd);
629     for(i=0; i <64; i++)
630     {
631         recv = ByteRead();
632         tmpbuf[i]=recv;
633         if((recv ^ PN9_Tab[i])!=0)
634         {
635             tmp = recv ^ PN9_Tab[i];
636             Err_BitCnt += (BitCount_Tab[tmp>>4] + BitCount_Tab[tmp & 0x0F]);
637         }
638     }
639     SCS=1;
640 }

```

Function : Read received data from RX FIFO, and do authentication

Line	Description
624	Increase parameter RxCnt
625	Assign address = 0x05 for RX FIFO of read operation.
627	Set SCS=0 to enable SPI interface.
628	Call ByteSend subroutine for one byte write operation.
629~638	Read 64 byte RX FIFO, compare received data with PN9_Tab and calculate BER
639	Set SCS=1 to disable SPI interface.

```

642 /*****
643 ** Err_State
644 *****/
645 void Err_State(void)
646 {
647     //ERR display
648     //Error Proc...
649     //...
650     while(1);
651 }

```

Function : Error state processing procedure

Line	Description
647~650	Procedure of BER is defined by user's applications.

```

653 /*****
654 ** CHGroupCal
655 *****/
656 void CHGroupCal(Uint8 ch)
657 {
658     Uint8 tmp;
659     Uint8 vb,vbcf;
660     Uint8 vcb,vccf;
661
662     A7125_WriteReg(PLL1_REG, ch);
663     Delay10us(4);
664     A7125_WriteReg(CALIBRATION_REG, 0x1C);
665     do {
666         tmp = A7125_ReadReg(CALIBRATION_REG);
667         tmp &= 0x1C;
668     }
669     while (tmp);
670
671     //for check
672     tmp = A7125_ReadReg(VCOCAL1_REG);
673     vb = tmp & 0x07;
674     vbcf = (tmp >> 3) & 0x01;
675
676     tmp = A7125_ReadReg(VCOCCAL_REG);
677     vcb = tmp & 0x0F;
678     vccf = (tmp >> 4) & 0x01;
679
680     if (vbcf || vccf)
681         Err_State();//error
682 }

```

Function : Calibration procedure of Channel Group

Line	Description
662	Call A7125_WriteReg subroutine to write PLL1 register.
663	Delay 40us for PLL settling stable
664	Set VCC=1, VBC=1, VDC=1 to enable VCO current, VCO band and VCO deviation.
665~669	Check VCC==0, VBC==0, VDC bit==0, if yes, exit the loop.
672~678	Assign variable vb, vbcf from read VCOCAL1_REG and assign variable vcb, vccf from read VCOCCAL1_REG.
680~681	If (vbcf =1) or (vbcf2=1), call Error_state subroutine to do error handling.


```

684 /*****
685 ** calibration
686 *****/
687 void A7125_Cal(void)
688 {
689     Uint8 tmp;
690     Uint8 fb,fbcf,fcd;
691     Uint8 dvt;
692
693     //calibration RSSI,IF procedure
694     StrobeCmd(CMD_PLL);
695     A7125_WriteReg(CALIBRATION_REG, 0x03);
696     do
697     {
698         tmp = A7125_ReadReg(CALIBRATION_REG);
699         tmp &= 0x03;
700     }
701     while (tmp);
702
703     //calibration VCO dev,VBC,VCC procedure
704     CHGroupCal(30); //calibrate channel group Bank I
705     CHGroupCal(90); //calibrate channel group Bank II
706     CHGroupCal(150); //calibrate channel group Bank III
707     StrobeCmd(CMD_STBY); //return to STBY state
708
709     //for check
710     tmp = A7125_ReadReg(IFCAL1_REG);
711     fb = tmp & 0x0F;
712     fbcf = (tmp >>4) & 0x01;
713
714     tmp = A7125_ReadReg(IFCAL2_REG);
715     fcd = tmp & 0x1F;
716
717     tmp = A7125_ReadReg(VCOCAL2_REG);
718     dvt = tmp;
719
720     if (fbcf)
721         Err_State(); //error
722 }

```

Function : RSSI , IF, VCO, VCO current, VCO deviation calibration procedure

Line	Description
694	Use Strobe command to set A7125 in PLL state
695	Set RSSC=1 to enable RSSI auto calibration Set FBC=1 to enable IF auto calibration.
696~701	Check RSSC==0 and FBC==0, if yes, exit the loop.
704~706	Call CHGroupCal subroutine to enable channel group calibration.
707	Use Strobe command to set A7125 in Standby state.
710~718	Assign variable fb, fbcf from read IFCAL1_REG and Assign variable fdc from read IFCAL2_REG and Assign variable dvt from read VCOCAL2_REG
720~721	Use Strobe command to set A7125 in PLL state

```

724 /*****
725 ** A7125_Config
726 *****/
727 void A7125_Config(void)
728 {
729     Uint8 i;
730
731     //0x00 mode register, for reset
732     //0x05 fifo data register
733     //0x06 id code register
734     //0x24 IF calibration II, only read
735
736     for (i=0x01; i<=0x04; i++)
737         A7125_WriteReg(i, A7125Config[i]);
738
739     for (i=0x07; i<=0x23; i++)
740         A7125_WriteReg(i, A7125Config[i]);
741
742     for (i=0x25; i<=0x38; i++)
743         A7125_WriteReg(i, A7125Config[i]);
744 }

```

Function : Initial RF chip procedure

Line	Description
736~737	Call A7125_WriteReg subroutine to initial control register from address 0x01 to 0x04
739~740	Call A7125_WriteReg subroutine to initial control register from address 0x07 to 0x23
742~743	Call A7125_WriteReg subroutine to initial control register from address 0x25 to 0x38