Document Title

A7125 reference code for FIFO mode

Revision History

Rev. No.	History	Issue Date	Remark
0.1	Preliminary	Nov.18, 2008	You
1.0	Modify initial value(Rcosc3,Tx_test,Crystal)	Dec.22, 2008	You

Table of contents

RF Chip-A7125
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

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

Master side ：After power on and initial RF chip procedure, Master will deliver 64 bytes data from TX FIFO, then jump into RX state to wait ACK data from Slave. If Master receives the ACK data, it will back to TX state to deliver next 64 byte data. If Master does NOT receive the ACK data, Master will also back to TX state for next 64 byte data delivery after staying in RX state for 50 ms.

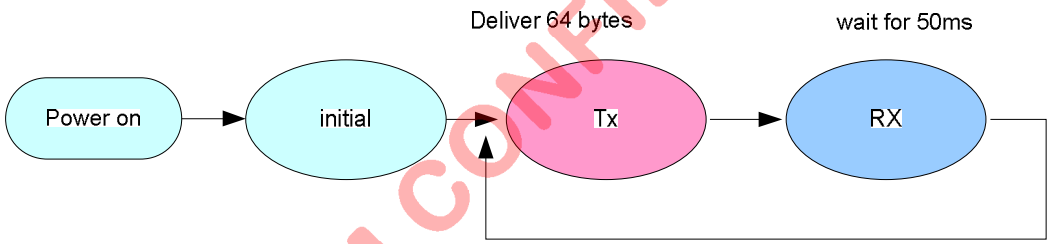

Fig1. The state diagram of master side

Slave side： After power on and initial RF procedure, Slave enters into RX state for receiving data from Master. Slave is set to stay in RX state until it receives the data. If Slave receives the data from Master, it will transit to TX state to deliver 64 bytes ACK data and then back to RX state for receiving next 64 byte data from Master.

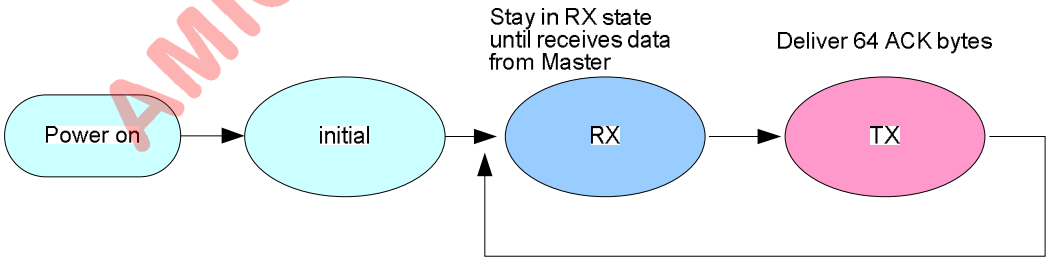

Fig2. The state diagram of Slave side

From Fig3, in Master side, Master enters into RX state to wait 64 byte ACK data once it delivers 64 byte data. If Master does NOT receive 64 byte ACK data within 50ms, it will back to TX state to deliver next 64 byte data. Once Master receives 64 byte packet, this packet will be authenticated and calculated bit error rate. After 10 ms, Master is set to back TX state for next 64 byte delivery.

From Fig3, in Slave side, Slave stays in RX state until it receives 64 byte data from Master. Once Slave receives 64 byte packet, this packet will be authenticated and calculated bit error rate. Then, Slave is set to enter TX state to deliver 64 byte ACK data to Master.
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).

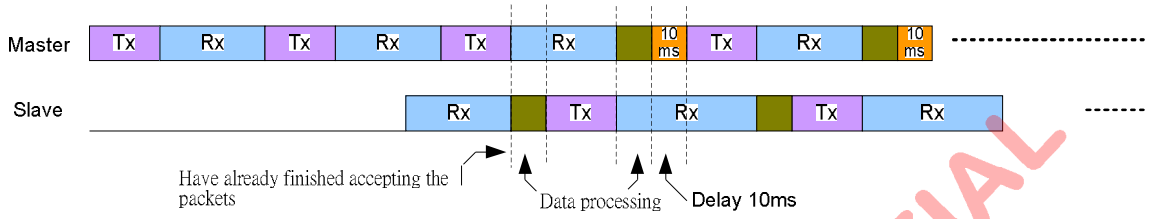

Fig3. Chronological chart between Master and Slave

3. Hardware

3.1 System block diagram

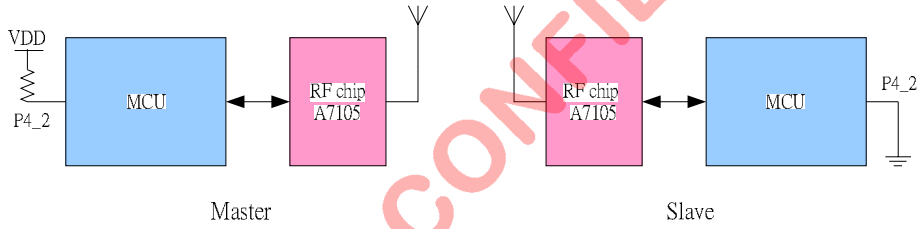

Fig4. Systematic block diagram

MCU I/O Pin Definition:
The example is explanation how to use the I/0:
SCS, SCK, SDIO -3 wire this bunch tabulate interface control 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:

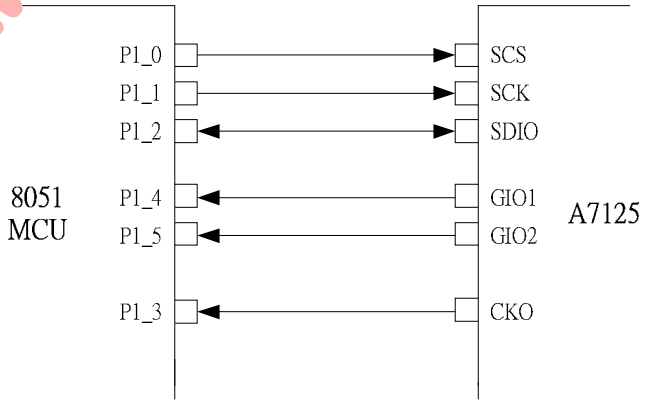

Fig5. Connections between 8051 MCU and A7125

4. Firmware Program

4.1Introduction

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) Writes 64 bytes PN9 code into TX FIFO.
2) MCU asks A7125 to enter TX State to deliver 64 byte PN9 code. After done, A7125 is auto back to Standby state.
3) MCU asks A7125 to enter RX state to wait 64 byte ACK data.
4) Enable Timer 0 and clear TimeoutFlag flag
5) If TimeoutFlag = 1 (timeout = 50ms), back to step(1).
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) MCU calls delay loop for 10 ms, then back to step (1).
9) For each 500 ms, MCU reports BER to personal computer.

Slave code：(Use frequency=2448.001MHz)
1) MCU asks A7125 to enter RX state until it receives 64 byte data from Master.
2) Once A7125 receives the packet, A7125 will be auto back to Standby state.
3) MCU compares received 64 bytes data with PN9 code and calculates BER (Bit Error Rate).
4) MCU writes 64 bytes PN9 code into TX FIFO.
5) MCU asks A7125 to enter TX State to deliver 64 byte PN9 code. After done, A7125 is auto back to Standby state.
6) Back to step (1).
7) For each 500 ms, MCU reports BER to personal computer.

4.2 State transition

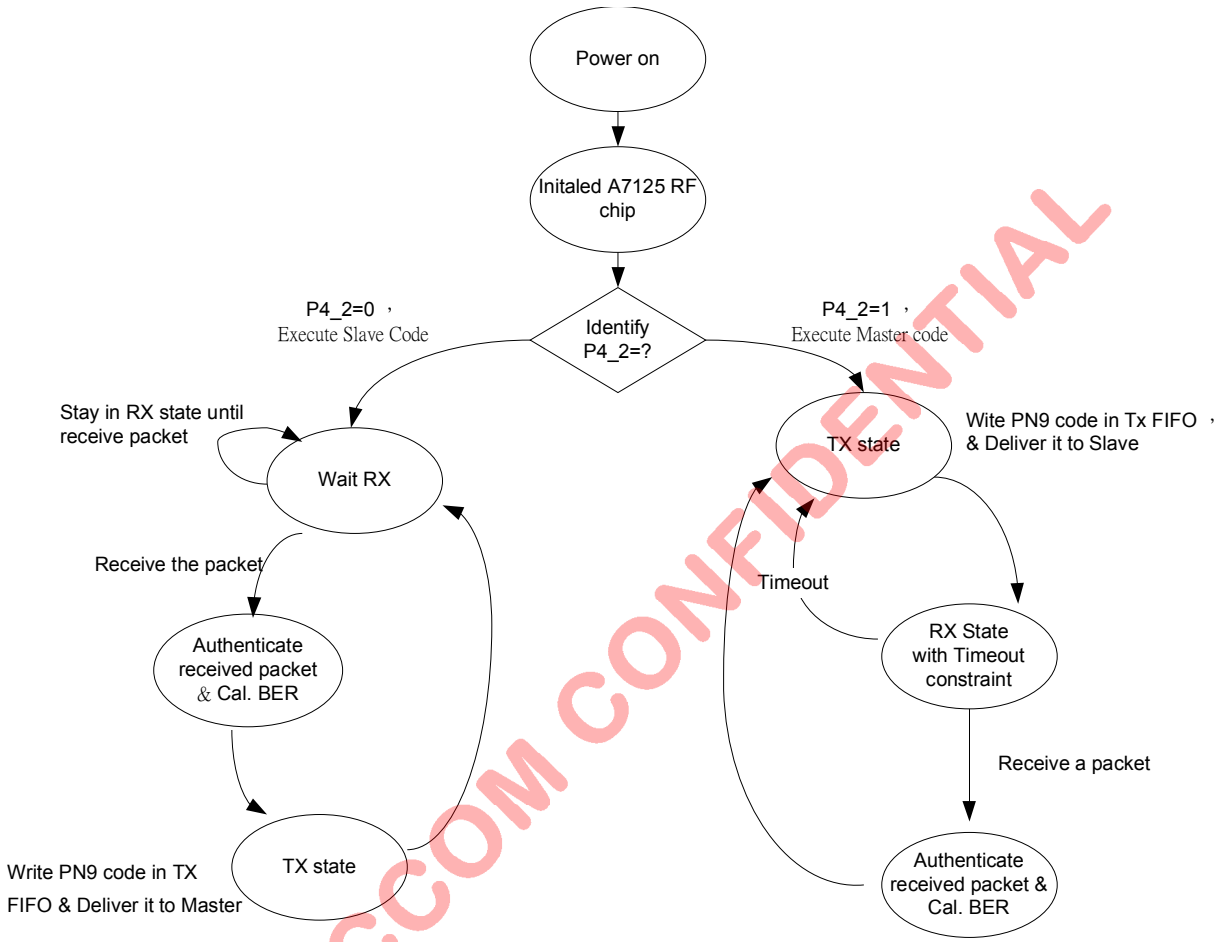

Fig6 State transition of Master code and Slave code

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:    2008-08-18
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"
```

The function explains：Include file declaration，Define the constant parameter

Row Number	Explanation
14~17	Include the link files

```
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       //GPIO1
27 #define GIO2       P1_5       //GPIO2
28
29 /*********************************************************************
30 **  Constant Declaration
31 *********************************************************************/
32 #define TIMEOUT      50
33 #define t0hrel     1000
```

The function explains： Define the I/O Port of A7125 RF chip by MCU，Define the constant parameter

Row Number	Explanation
22~27	MCU I/O definition
32~33	Define the constant parameter

```
35  /*****************************************************************
36  **  Global Variable Declaration
37  *****************************************************************/
38  Uint8        data    timer;
39  Uint8        data    TimeoutFlag;
40  Uint16       idata   RxCnt;
41  Uint32       idata   Err_ByteCnt;
42  Uint32       idata   Err_BitCnt;
43  Uint16       idata   TimerCnt0;
44  Uint8        data    *Uartptr;
45  Uint8        data    UartSendCnt;
46  Uint8        data    CmdBuf[12];
47  Uint8        xdata   tmpbuf[64];
48  Uint8        idata   Err_Frame;
49
50  const Uint8 code BitCount_Tab[16] = {0,1,1,2,1,2,2,3,1,2,2,3,2,3,3,4};
51  const Uint8 code ID_Tab[4]={0x34,0x75,0xC5,0x2A}; //ID code
52  const Uint8 code PN9_Tab[]=
53  {   0xFF,0x83,0xDF,0x17,0x32,0x09,0x4E,0xD1,
54      0xE7,0xCD,0x8A,0x91,0xC6,0xD5,0xC4,0xC4,
55      0x40,0x21,0x18,0x4E,0x55,0x86,0xF4,0xDC,
56      0x8A,0x15,0xA7,0xEC,0x92,0xDF,0x93,0x53,
57      0x30,0x18,0xCA,0x34,0xBF,0xA2,0xC7,0x59,
58      0x67,0x8F,0xBA,0x0D,0x6D,0xD8,0x2D,0x7D,
59      0x54,0x0A,0x57,0x97,0x70,0x39,0xD2,0x7A,
60      0xEA,0x24,0x33,0x85,0xED,0x9A,0x1D,0xE0
61  };      // This table are 64bytes PN9 pseudo random code.
```

The function explains：Declaration of the parameter

Row Number	Explanation
38~48	Declare parameters that are used in the procedure
50	Declare BitCount_Tab
51	Declare ID_Tab for ID code
52~61	Declare PN9_Tab of 64 bytes PN9 code

```
const Uint16 code A7125Config[]=
{
    0x00, //MODE register,          only reset, not use on config
    0x42, //MODE CTRL register,
    0x00, //CALIBRATION register,
    0x3F, //FIFO1 register,
    0x00, //FIFO2 register,
    0x00, //FIFO register,          for fifo read/write
    0x00, //IDDATA register,        for idcode
    0x00, //RCOSC1 register,
    0x00, //RCOSC2 register,
    0x00, //RCOSC3 register,
    0x00, //CKO register,
    0x01, //GPIO1 register
    0x00, //GPIO2 register,
    0x1F, //DATARATE register,
    0x50, //PLL1 register,
    0x0E, //PLL2 register,          RFbase 2400.001MHz
    0x96, //PLL3 register,
    0x00, //PLL4 register,
    0x04, //PLL5 register,
    0x3C, //chanel group I,
    0x78, //chanel group II,
    0xD7, //TX1 register,
    0x40, //TX2 register,
    0x12, //DELAY1 register,
    0x41, //DELAY2 register,
    0x62, //RX register,
    0xA0, //RXGAIN1 register,
    0x00, //RXGAIN2 register,
    0x00, //RXGAIN3 register,
    0xD2, //RXGAIN4 register,
    0x00, //RSSI register,
    0xE2, //ADC register,
    0x07, //CODE1 register,
    0x56, //CODE2 register,
    0x2A, //CODE3 register,
    0x06, //IFCAL1 register,
    0x00, //IFCAL2 register,   only read
    0x05, //VCOCCAL register,
    0x44, //VCOCAL1 register,
    0x80, //VCOCAL2 register,
    0x30, //VCO DEV Cal. I register,
    0x20, //VCO DEV Cal. II register,
    0x80, //VCO DEV Cal. III register,
    0x00, //VCO Mod. delay register
    0x6A, //BATTERY register,
    0x0F, //TXTEST register,
    0x47, //RXDEM1 register,
    0x80, //RXDEM2 register,
    0xF1, //CPC1 register,
    0x11, //CPC2 register,
    0x04, //CRYSTAL register,
    0x45, //PLLTEST register,
    0x18, //VCOTEST register,
    0x10, //RF Analog Test
    0x03, //IFAT register,
    0x37, //Channel select register,
    0xFF  //VRB register
};
```

The function explains： Configure of A7125's control registers for Master

Row	Explanation

Number	
65~121	Configure of A7125's control registers for Master

```
124  /*********************************************************************
125   **  function Declaration
126   *********************************************************************/
127   void InitTimer0(void);
128   void initUart0(void);
129   void Timer0ISR (void);
130   void Uart0Isr(void);
131   void A7125_Reset(void);
132   void A7125_WriteReg(Uint8, Uint8);
133   Uint8 A7125_ReadReg(Uint8);
134   void ByteSend(Uint8 src);
135   Uint8 ByteRead(void);
136   void A7125_WriteID(void);
137   void A7125_WriteFIFO(void);
138   void initRF(void);
139   void A7125_Config(void);
140   void A7125_Cal(void);
141   void RxPacket(void);
142   void StrobeCmd(Uint8);
143   void SetCH(Uint8);
144   void CHGroupCal(Uint8);
```

The function explains：Subroutine declaration

Row Number	Explanation
127~144	Subroutine declarations

```
/*****************************************************************
 * main loop
 *****************************************************************/
void main(void)
{
    //initsw
    PMR |= 0x01; //set DME0

    //initHW
    P0 = 0xFF;
    P1 = 0xFF;
    P2 = 0xFF;
    P3 = 0xFF;
    P4 = 0x0F;

    InitTimer0();
    initUart0();
    TR0=1;
    EA=1;

    if ((P4 & 0x04)==0x04) //if P4.2=1, master
    {
        initRF();
        StrobeCmd(CMD_STBY);

        while(1)
        {
            A7125_WriteFIFO(); //write data to tx fifo
            SetCH(100); //freq 2450.001MHz
            StrobeCmd(CMD_TX); //entry tx
            while(GIO1); //wait transmit completed

            SetCH(96); //freq 2448.001MHz
            StrobeCmd(CMD_RX); //entry rx

            timer=0;
            TimeoutFlag=0;
            while(GIO1==1 && TimeoutFlag==0); //wait receive completed
            if (TimeoutFlag)
            {
                StrobeCmd(CMD_STBY); //exit rx mode
                continue;
            }

            RxPacket();
            Delay10ms(1);
        }
    }
    else //if P4.2=0, slave
    {
        initRF();
        StrobeCmd(CMD_STBY);

        RxCnt = 0;
        Err_ByteCnt = 0;
        Err_BitCnt = 0;
        TR0 = 1;
```

```
203
 204        while(1)
 205        {
 206            SetCH(96); //freq 2448.001MHz
 207            StrobeCmd(CMD_RX); //entry rx
 208            while(GIO1); //wait receive completed
 209            RxPacket();
 210
 211            A7125_WriteFIFO(); //write data to tx fifo
 212            SetCH(100); //freq 2450.001MHz
 213            StrobeCmd(CMD_TX); //entry tx
 214            while(GIO1); //wait transmit completed
 215        }
 216    }
 217  }
```

The function explains： Main loop。MCU Identifies Port4_2 = 1，row 167~193 are Master code. Row 195~216 are Slave code.

Row Number	Expiring
152	To start the MCU on chip data SRAM
155~159	Initial MCU I/O Port
161	Call subroutine of initTimer0 and enable interrupt
162	Call subroutine of initUart0 and initial Uart0
163~164	Enable Timer0 and Timer0 interrupt
166	Identify if port4_2=1, jump to Master code, else jump to Slave code.
167~193	Master code
168	Call subroutine of initRF to initial A7125
169	Use Strobe command to enter Standby mode
173	Call subroutine of A7125_WriteFIFO to write data into TX FIFO
174	Call subroutine of SetCH to setup the TX radio freq. at 2450.001MHz
175	User Strobe command to enter TX mode and A7125 deliver TX FIFO automatically.
176	Monitor status of A7125's GIO1 pin for A7125's WTR output.
178	Call subroutine of SetCH to setup RX LO freq. at 2448.001MHz
179	Use Strobe command to enter RX mode
181~182	Clean timer0 variable and Timeout Flag
183	Wait for received data or Timeout
184~188	Check Timeout Flag，if Timeout，jump to row 173
190	Call subroutine of RxPacket to read data from RX FIFO , do authentication and calculate BER
191	Call subroutine of Delay10ms for 10 ms delay.
195~216	Slave code
196	Call subroutine of initRF to initial A7125
197	Use Strobe command to enter Standby mode
199~201	Clean the variable of RxCnt, Err_ByteCnt, Err_BitCnt
202	Enable timer0
206	Call subroutine of SetCH to setup TX radio freq. at 2448.001MHz
207	Use Strobe command to enter RX mode
208	Stay in RX state until receive 64 bytes data
209	Call subroutine of RxPacket to read data from RX FIFO , do authentication and calculate BER
211	Call subroutine of A7125_WriteFIFe to write data into TX FIFO
212	Call subroutine of SetCH to setup TX radio freq. at 2450.001MHz
213	Use Strobe command to enter Tx mode and A7125 deliver TX FIFO automatically.
214	Monitor status of A7125's GIO1 pin for for A7125's WTR output.

```
219 /*********************************************************************
220  **  init Timer0
221  *********************************************************************/
222 void InitTimer0(void)
223 {
224     TR0 = 0;
225     TMOD =(TMOD & 0xF0)|0x01; //timer0 mode=1
226     TH0 = (65536-t0hrel)>>8; // Reload Timer0 high byte,low byte
227     TL0 = 65536-t0hrel;
228     TF0 = 0; // Clear any pending Timer0 interrupts
229     ET0 = 1; // Enable Timer0 interrupt
230 }
```

The function explains：Subroutine of InitTimer0

Row Number	Explanation
224	Disable Timer0
225	Setup Timer0 in mode1 mode
226~227	Setup the initial value of TH0,TL0
228	Clean Timer0 interrupt flag
229	Enable Timer0 interrup

```
232 /*********************************************************************
233  **  Timer0ISR
234  *********************************************************************/
235 void Timer0ISR (void) interrupt 1
236 {
237     TF0 = 0; // Clear Timer0 interrupt
238     TH0 = (65536-t0hrel)>>8; // Reload Timer0 high byte,low byte
239     TL0 = 65536-t0hrel;
240
241     timer++;
242     if (timer == TIMEOUT)
243     {
244         TimeoutFlag=1;
245     }
246
247     TimerCnt0++;
248     if (TimerCnt0 == 500)
249     {
250         TimerCnt0 = 0;
251         CmdBuf[0] = 0xF1;
252
253         memcpy(&CmdBuf[1], &RxCnt, 2);
254         memcpy(&CmdBuf[3], &Err_ByteCnt, 4);
255         memcpy(&CmdBuf[7], &Err_BitCnt, 4);
256         memcpy(&CmdBuf[11], &Err_Frame, 1);
257
258         UartSendCnt = 12;
259         Uartptr =& CmdBuf[0];
260         SBUF = CmdBuf[0];
261     }
262 }
```

The function explains：Subroutine of Timer0 interrupt

Row Number	Explanation
237~239	Clear Timer0 interrupt flag and setup initial value of TH0 and TL0.
241	Increase timer (timer is a variable).

242~245	Check timer == TIMEOUT, if yes, set TimeoutFlag =1.
247	ncrease timer (timer is a variable).
248	Check TimerCnt0 == 500, if yes, 500ms delay is done.
250	Clear TimerCnt0
251	**CmdBuf[0] is set to 0xF1 for Windows VB.**
253	CmdBuf[1] is used to set up parameter RxCnt
254	CmdBuf[3] 、CmdBuf[4]、CmdBuf[5] 、CmdBuf[6] are used to set up parameter Err_ByteCnt
255	CmdBuf[7] 、CmdBuf[8]、CmdBuf[9] 、CmdBuf[10] are used to set up parameter Err_BitCn
256	CmdBuf[11] is used to set up parameter Err_Frame.
258	Set UartSendCnt=12
259	Set pointer Uartptr to indicate location of CmdBuf [0].
260	Deliver SBUF to PC for BER information.

```
264 /*************************************************************************
265  **  Init Uart0
266  *************************************************************************/
267  void initUart0(void)
268  {
269      TH1 = 0xFD; //BaudRate 9600;
270      TL1 = 0xFD;
271      SCON = 0x40;
272      TMOD = (TMOD & 0x0F) | 0x20;
273      REN = 1;
274      TR1 = 1;
275      ES = 1;
276  }
```

The function explains：Initialize the procedure of Uart0

Row Number	Explanation
269~271	Set initial value of TL1,TH1,SCON1 and Set 9600 baud rate @xtal=11.0592MHz
272	Set Timer1 in mode 2
273~275	Set REN=1, TR1=1, ES=1 to enable UART function.

```
278 /*************************************************************************
279  **  Uart0 ISR
280  *************************************************************************/
281  void Uart0Isr(void) interrupt 4 using 3
282  {
283      if (TI==1)
284      {
285          TI=0;
286          UartSendCnt--;
287          if(UartSendCnt !=0)
288          {
289              Uartptr++;
290              SBUF = *Uartptr;
291          }
292      }
293  }
```

The function explains：Initialize the interrupt subprogram of uart0.

Row Number	Explanation
283	Check TI flag, if TI=1, one byte of UART transmission is done.
285	Clear TI flag

286	Decrease UartSendCnt (UartSendCnt is a variable)
287	Check UartSendCnt == 0, if not, continue UART transmission until done.
289~290	Increase pointer Uartptr to assign address of next data via UART to PC

```
295  /*************************************************************************
296   **  Reset_RF
297   *************************************************************************/
298   void A7125_Reset(void)
299   {
300       A7125_WriteReg(MODE_REG, 0x00); //reset RF chip
301   }
```

The function explains：A7125 RF chip Reset step

Row Number	Explanation
300	Write 0x00 in MODE register to reset A7125

```
303  /*************************************************************************
304   **  WriteID
305   *************************************************************************/
306   void A7125_WriteID(void)
307   {
308       Uint8 i;
309       Uint8 d1,d2,d3,d4;
310       Uint8 addr;
311
312       addr = IDCODE_REG; //send address 0x06, bit cmd=0, r/w=0
313       SCS = 0;
314       ByteSend(addr);
315       for (i=0; i < 4; i++)
316           ByteSend(ID_Tab[i]);
317       SCS = 1;
318
319       //for check
320       addr = IDCODE_REG | 0x40; //send address 0x06, bit cmd=0, r/w=1
321       SCS=0;
322       ByteSend(addr);
323       d1=ByteRead();
324       d2=ByteRead();
325       d3=ByteRead();
326       d4=ByteRead();
327       SCS=1;
328   }
```

The function explains：subprogram of accessing ID code registers

Row Number	Explanation
312	Set parameter addr = 0x06 for ID code write operation.
313	Set SCS=0 to enable SPI interface.
314~316	Write ID_Tab Table into A7125 ID Code registers.
317	Set SCS=1 to disable SPI interface.
320	Set parameter addr = 0x06 for ID code read operation.
321	Set SCS=0 to enable SPI interface.
322~326	Read 4 byte ID code for ID check.
327	Set SCS=1 to disable SPI interface.

```
330  /*********************************************************************
331   **  A7125_WriteReg
332   *********************************************************************/
333   void A7125_WriteReg(Uint8 addr, Uint8 dataByte)
334   {
335      Uint8 i;
336
337      SCS = 0;
338      addr |= 0x00; //bit cmd=0,r/w=0
339      for(i = 0; i < 8; i++)
340      {
341         if(addr & 0x80)
342            SDIO = 1;
343         else
344            SDIO = 0;
345
346         SCK = 1;
347         _nop_();
348         SCK = 0;
349         addr = addr << 1;
350      }
351      _nop_();
352
353      //send data byte
354      for(i = 0; i < 8; i++)
355      {
356         if(dataByte & 0x80)
357            SDIO = 1;
358         else
359            SDIO = 0;
360
361         SCK = 1;
362         _nop_();
363         SCK = 0;
364         dataByte = dataByte<< 1;
365      }
366      SCS = 1;
367   }
```

The function explains：subroutine of A7125 control registers' write operation

Row Number	Explanation
337	Set SCS=0 to enable SPI interface.
338	Enable write operation of control registers (addr7=0 for control registers, addr6=0 for write operation)
339~350	Assign control register's address by input variable addr.
354~365	Assign control register's value by input variable dataByte
366	Set SCS=1 to disable SPI interface.

```
369 /*********************************************************************
370  **  A7125_ReadReg
371  *********************************************************************/
372 Uint8 A7125_ReadReg(Uint8 addr)
373 {
374     Uint8 i;
375     Uint8 tmp;
376
377     SCS = 0;
378     addr |= 0x40; //bit cmd=0,r/w=1
379     for(i = 0; i < 8; i++)
380     {
381
382         if(addr & 0x80)
383             SDIO = 1;
384         else
385             SDIO = 0;
386
387                 _nop_();
388         SCK = 1;
389         _nop_();
390         SCK = 0;
391
392         addr = addr << 1;
393     }
394
395     _nop_();
396     SDIO = 1;
397
398     //read data
399     for(i = 0; i < 8; i++)
400     {
401         if(SDIO)
402             tmp = (tmp << 1) | 0x01;
403         else
404             tmp = tmp << 1;
405
406         SCK = 1;
407         _nop_();
408         SCK = 0;
409     }
410     SCS = 1;
411     return tmp;
412 }
```

The function explains：subprogram of A7125 control registers' read operation

Row Number	Explanation
377	Enable read operation of control registers (addr7=0 for control registers, addr6=1 for read operation)
378	Assign control register's address by input variable addr.
379~393	Assign control register's address by input variable addr.
396	Set SDIO=1 for SPI Data Output
399~409	The control register's value is output to SPI interface.
410	Set SCS=1 to disable SPI interface.
411	Return tmp (tmp represents 8-bits control register's value)

```
414  /*********************************************************************
415   **  ByteSend
416   *********************************************************************/
417  void ByteSend(Uint8 src)
418  {
419      Uint8 i;
420
421      for(i = 0; i < 8; i++)
422      {
423          if(src & 0x80)
424              SDIO = 1;
425          else
426              SDIO = 0;
427
428          _nop_();
429          SCK = 1;
430          _nop_();
431          SCK = 0;
432          src = src << 1;
433      }
434  }
```

The function explains：subroutine of ByteSend

Row Number	Explanation
421~433	Procedures how to write one byte data into A7125.

```
436  /*********************************************************************
437   **  ByteRead
438   *********************************************************************/
439  Uint8 ByteRead(void)
440  {
441      Uint8 i,tmp;
442
443      SDIO = 1; //sdio pull high
444      for(i = 0; i < 8; i++)
445      {
446          if(SDIO)
447              tmp = (tmp << 1) | 0x01;
448          else
449              tmp = tmp << 1;
450
451          SCK = 1;
452          _nop_();
453          SCK = 0;
454      }
455      return tmp;
456  }
```

The function explains：subroutine of ByteRead

Row Number	Explanation
443~454	Procedures how to read one byte data from A7125.
455	Return tmp (tmp represents control register's value)

```
458  /*********************************************************************
459   **  Send4Bit
460   *********************************************************************/
461  void Send4Bit(Uint8 src)
462  {
463      Uint8 i;
464
465      for(i = 0; i < 4; i++)
466      {
467          if(src & 0x80)
468              SDIO = 1;
469          else
470              SDIO = 0;
471
472          _nop_();
473          SCK = 1;
474          _nop_();
475          SCK = 0;
476          src = src << 1;
477      }
478  }
```

The function explains：subroutine of Send4Bit

Row Number	Explanation
465~477	Procedures how to write 4 bits data into A7125, especially for Strobe command.

```
480  /*********************************************************************
481   ** SetCH
482   *********************************************************************/
483  void SetCH(Uint8 ch)
484  {
485      //RF freq = RFbase + (CH_Step * ch)
486      A7125_WriteReg(PLL1_REG, ch);
487  }
```

The function explains：subroutine of SetCH, procedures of setting channel

Row Number	Explanation
486	Call subroutine of A7125_WriteReg and to set up PLL1 control register.

```
489 /*********************************************************************
490  ** initRF
491  *********************************************************************/
492 void initRF(void)
493 {
494     //init io pin
495     SCS = 1;
496     SCK = 0;
497     SDIO = 1;
498     CKO = 1;
499     GIO1 = 1;
500     GIO2 = 1;
501
502     A7125_Reset(); //reset A7125 RF chip
503     A7125_WriteID(); //write ID code
504     A7125_Config(); //config A7125 chip
505     A7125_Cal(); //calibration IF,vco, vcoc
506 }
```

The function explains：Initialize RF chip of Master side

Row Number	Explanation
495~500	Set up I/O initial value between MCU and A7125
502	Call subroutine of A7125_Reset to reset A7125
503	Call subroutine of A7125_WriteID to write 4 byte ID code.
504	Call subroutine of A7125_Config to initial A7125's control registers.
505	Call subroutine of A7125_Cal to do calibration procedures for IF, VCO, and VCO current.

```
508 /*********************************************************************
509  ** A7125_WriteFIFO
510  *********************************************************************/
511 void A7125_WriteFIFO(void)
512 {
513     Uint8 i;
514     Uint8 cmd;
515
516     cmd = FIFO_REG; //send address 0x05, bit cmd=0, r/w=0
517     SCS=0;
518     ByteSend(cmd);
519     for(i=0; i <64; i++)
520         ByteSend(PN9_Tab[i]);
521     SCS=1;
522 }
```

The function explains：Subroutine of write Tx FIFO

Row Number	Explanation
516	Assign address = 0x05 for TX FIFO of write operation.
517	Set SCS=0 to enable SPI interface.
518	Call subroutine of ByteSend for one byte write operation.
519~520	Write PN9_Tab into TX FIFO, total 64 bytes.
521	Set SCS=1 to disable SPI interface.

```
524 /*********************************************************************
525  ** Strobe Command
526  *********************************************************************/
527  void StrobeCmd(Uint8 cmd)
528  {
529      SCS = 0;
530      Send4Bit(cmd);
531      SCS = 1;
532  }
```

The function explains：Subroutine of sending 4-bits Strobe command.

Row Number	Explanation
529	Set SCS=0 to enable SPI interface
530	Call subroutine of Send4Bit for 4-bits Strobe command.
531	Set SCS=1 to disable SPI interface.

```
534 /*********************************************************************
535  ** RxPacket
536  *********************************************************************/
537  void RxPacket(void)
538  {
539      Uint8 i;
540      Uint8 recv;
541      Uint8 tmp;
542      Uint8 cmd;
543
544      RxCnt++;
545      cmd = FIFO_REG | 0x40; //address 0x05, bit cmd=0, r/w=1
546
547      SCS=0;
548      ByteSend(cmd);
549      for(i=0; i <64; i++)
550      {
551          recv = ByteRead();
552          tmpbuf[i]=recv;
553          if((recv ^ PN9_Tab[i])!=0)
554          {
555              tmp = recv ^ PN9_Tab[i];
556              Err_BitCnt += (BitCount_Tab[tmp>>4] + BitCount_Tab[tmp & 0x0F]);
557          }
558      }
559      SCS=1;
560  }
```

The function explains：Read received data from RX FIFO, and do authentication

Row Number	Explanation
544	Increase parameter RxCnt
545	Assign address = 0x05 for RX FIFO of read operation.
547	Set SCS=0 to enable SPI interface.
548	Call subroutine of ByteSend for one byte write operation.
549~558	Read 64 byte RX FIFO, compare received data with PN9_Tab and calculate BER
559	Set SCS=1 to disable SPI interface.

```
562 /*********************************************************************
563  ** Err_State
564  *********************************************************************/
565 void Err_State(void)
566 {
567     //ERR display
568     //Error Proc...
569     //...
570     while(1);
571 }
```

The function explains：Procedure of BER

Row Number	Explanation
567~570	Procedure of BER is defined by user's applications.

```
573 /*********************************************************************
574  ** CHGroupCal
575  *********************************************************************/
576 void CHGroupCal(Uint8 ch)
577 {
578     Uint8 tmp;
579     Uint8 vb,vbcf;
580     Uint8 vcb,vccf;
581
582     A7125_WriteReg(PLL1_REG, ch);
583     A7125_WriteReg(CALIBRATION_REG, 0x1C);
584     do
585     {
586         tmp = A7125_ReadReg(CALIBRATION_REG);
587         tmp &= 0x1C;
588     }
589     while (tmp);
590
591     //for check
592     tmp = A7125_ReadReg(VCOCAL1_REG);
593     vb = tmp & 0x07;
594     vbcf = (tmp >>3) & 0x01;
595
596     tmp = A7125_ReadReg(VCOCCAL_REG);
597     vcb = tmp & 0x0F;
598     vccf= (tmp >> 4) & 0x01;
599
600     if (vbcf || vccf)
601       Err_State();//error
602 }
```

The function explains：Subroutine of CHGroupCal for easy channel group calibration.

Row Number	Explanation
582	Call Subroutine of A7125_WriteReg to write PLL1 register.
583	Set VCC=1, VBC=1, VDC=1 to enable VCO current, VCO band and VCO deviation.
584~589	Check VCC==0, VBC==0, VDC bit==0, if yes, exit the loop.
592~598	Assign variable vb, vbcf from read VCOCAL1_REG and assign variable vcb, vccf from read VCOCAL1_REG.
600~601	If (vbcf =1) or (vbcf2=1), call subroutine of Error_state to do error handling.

```
604 /*********************************************************************
605  ** calibration
606  *********************************************************************/
607  void A7125_Cal(void)
608  {
609      Uint8 tmp;
610      Uint8 fb,fbcf,fcd;
611      Uint8 dvt;
612
613      //calibration RSSI,IF procedure
614      StrobeCmd(CMD_PLL);
615      A7125_WriteReg(CALIBRATION_REG, 0x03);
616      do
617      {
618          tmp = A7125_ReadReg(CALIBRATION_REG);
619          tmp &= 0x03;
620      }
621      while (tmp);
622
623      //calibration VCO dev,VBC,VCC procedure
624
625      CHGroupCal(30);     //calibrate channel group Bank I
626      CHGroupCal(90);     //calibrate channel group Bank II
627      CHGroupCal(150);    //calibrate channel group Bank III
628      StrobeCmd(CMD_STBY); //return to STBY state
629
630      //for check
631      tmp = A7125_ReadReg(IFCAL1_REG);
632      fb = tmp & 0x0F;
633      fbcf = (tmp >>4) & 0x01;
634
635      tmp = A7125_ReadReg(IFCAL2_REG);
636      fcd = tmp & 0x1F;
637
638      tmp = A7125_ReadReg(VCOCAL2_REG);
639      dvt = tmp;
640
641      if (fbcf)
642          Err_State(); //error
643  }
```

The function explains：Subroutine of A7125 Auto Calibration of RSSI, IF, VCO, VCO current, VCO deviation

Row Number	Explanation
614	Use Strobe command to set A7125 in PLL state
615	Set RSSC=1 to enable RSSI auto calibration Set FBC=1 to enable IF auto calibration.
616~621	Checks RSSC==0 and FBC==0, if yes, exit the loop.
625~627	Call subroutine of CHGroupCal to enable channel group calibration.
628	Use Strobe command to set A7125 in Standby state.
631~639	Assign variable fb, fbcf from read IFCAL1_REG and Assign variable fdc from read IFCAL2_REG and Assign variable dvt from read VCOCAL2_REG
641~642	If (fbcf =1) , call subroutine of Error_state to do error handling.

```
645  /*********************************************************************
646   ** A7125_Config
647   *********************************************************************/
648   void A7125_Config(void)
649   {
650       Uint8 i;
651
652       //0x00 mode register, for reset
653       //0x05 fifo data register
654       //0x06 id code register
655       //0x24 IF calibration II, only read
656
657       for (i=0x01; i<=0x04; i++)
658           A7125_WriteReg(i, A7125Config[i]);
659
660       for (i=0x07; i<=0x23; i++)
661           A7125_WriteReg(i, A7125Config[i]);
662
663       for (i=0x25; i<=0x38; i++)
664           A7125_WriteReg(i, A7125Config[i]);
665   }
```

The function explains：Initial RF chip procedure in Master side

Row Number	Explanation
657~658	Call subroutine of A7125_WriteReg to initial control register from address 0x01 to 0x04
660~661	Call subroutine of A7125_WriteReg to initial control register from address 0x07 to 0x23
663~664	Call subroutine of A7125_WriteReg to initial control register from address 0x25 to 0x38