Document Title

A7125 reference code for FIFO extension mode (2Mbps)

Revision History

Rev. No.	History	Issue Date	Remark
0.1	Preliminary	May 20, 2009	
0.2	Modify initial configuration value GIO2=0x01	Sep. 1, 2010	

Table of contents

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

1. Introduction

This document describes development of simple example procedures by A7125 FIFO extension mode. It could support user how to implement.

2. System 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 128 bytes data from TX FIFO. After delay 50ms, Master will enter TX state to transmit 128 bytes again.

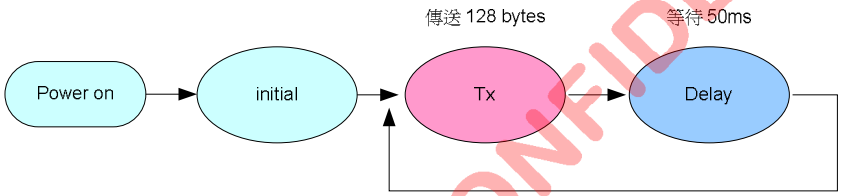

Fig1. The state diagram of master side

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 read out data, compared to right , calculates error bit. Delay 30ms, Slave enter into RX state for receiving data again.

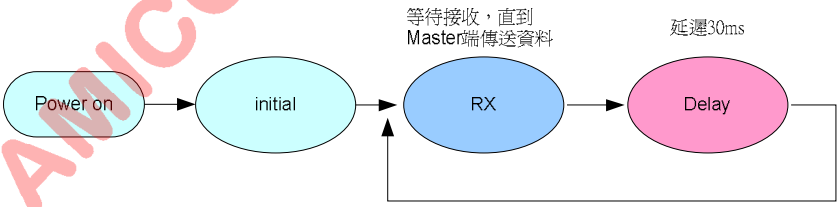

Fig2. The state diagram of Slave side

3. Hardware

3.1 System block diagram

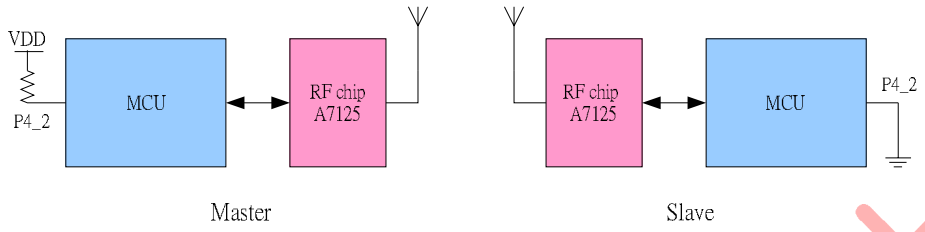

Fig3. System block diagram

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.
CKO - The control signal monitors TX FIFO or RX FIFO under FIFO extension FIFO the Pointer marginal value change, so as to the restricted data writes in or reads out FIFO the opportunity.

MCU controls A7125 RF chip disposes the following picture:

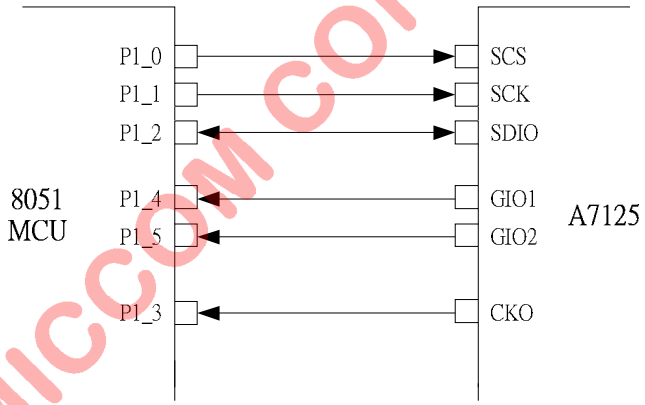

Fig4. Connections between 8051 MCU and A7125

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：

1) Initial A7125RF chip.
2) Write 64 bytes PN9 code to RF chip
3) Enter TX state, tramsmit packet.
4) Waiting for CKO pin is 1, write 64 bytes invert-PN9 code to TX FIFO again.
5) When GIO1 pin is 0, RF chip automatically exit TX state to standby state.
6) Delay 50ms, back to Step 2 and restart next time working.

Slave code：

1) Initial A7125RF chip.
2) Enter RX state until it receives 64 byte data from Master.
3) If CKO pin is 1, MCU read out data from RX FIFO.
4) After GIO1 pin is 0, A7125 will be auto back to Standby state.
5) MCU read out data from RX FIFO.
6) Compared to data, calculate error bit.
7) Delay 30ms, back to step2, restart to next time working period.
8) For each 500 ms, MCU reports BER to PC.

4.2 Example State Diagram

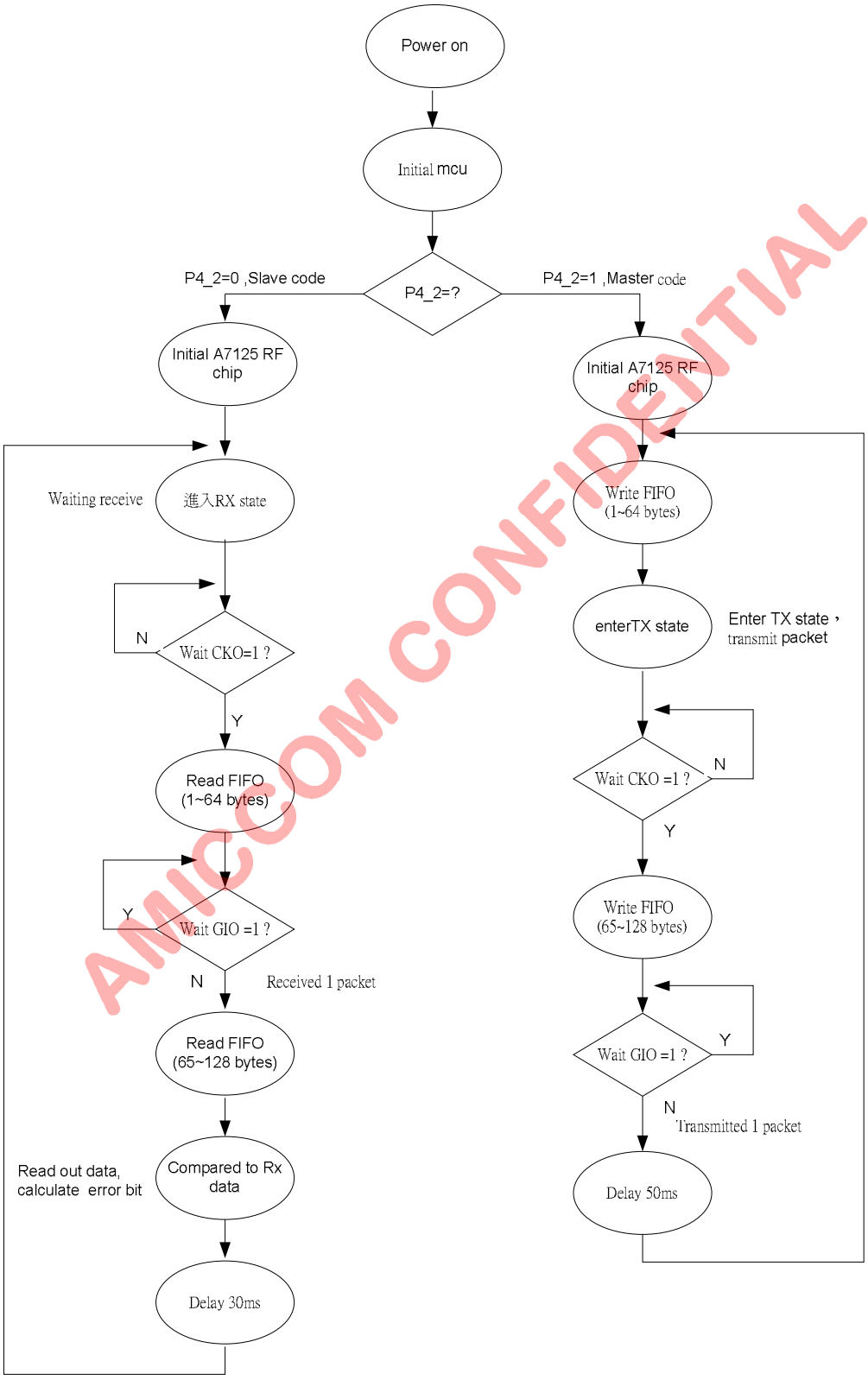

Fig5 Example state diagram

5. Explanation of reference code

Notes:

※This program only supplies reference procedure in FIFO extension mode , write in / read out for FIFO control register. User must according to the MCU working speed, processes FIFO data to write in / read-out time cautiously tightly, avoids occurring wrongly.

```
 1  /*********************************************************************
 2  **  Device:     A7125
 3  **  File:       main.c
 4  **  Author:     JPH
 5  **  Target:     Winbond W77LE58
 6  **  Tools:      ICE
 7  **  Created:    2009-04-22
 8  **  Description:
 9  **  This file is a sample code for your reference.
10  **
11  **  Copyright (C) 2008 AMICCOM Corp.
12  **
13  *********************************************************************/
14  #include "define.h"
15  #include "w77le58.h"
16  #include "a7125reg.h"
17  #include "Uti.h"
```

Function：Include file declaration，Define the constant parameter	
Line	Description
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        //GIO1
27  #define GIO2         P1_5        //GIO2
28  #define Button       P1_7        //test Button
29
30  /*********************************************************************
31  **  Constant Declaration
32  *********************************************************************/
33  #define TIMEOUT        50
34  #define t0hrel       1000
```

Function： Define the I/O Port of A7125 RF chip by MCU，Define the constant parameter	
Line	Description
22~28	MCU I/O definition
33~34	Define the constant parameter

```
36  /*****************************************************************
37  **  Global Variable Declaration
38  *****************************************************************/
39  Uint8      data    timer;
40  Uint8      data    TimeoutFlag;
41  Uint16     idata   RxCnt;
42  Uint32     idata   Err_ByteCnt;
43  Uint32     idata   Err_BitCnt;
44  Uint16     idata   TimerCnt0;
45  Uint8      data    *Uartptr;
46  Uint8      data    UartSendCnt;
47  Uint8      data    CmdBuf[12];
48  Uint8      xdata   tmpbuf[256];
49  Uint16     xdata   AFCBuf[12];
50  Uint8      data    sts_ModeReg;
51  Uint8      idata   Err_Frame;
52
53  const Uint8 code BitCount_Tab[16] = {0,1,1,2,1,2,2,3,1,2,2,3,2,3,3,4};
54  const Uint8 code ID_Tab[4]={0x54,0x75,0xC5,0x2A}; //ID code
55  const Uint8 code PN9_Tab[]=
56  {   0xFF,0x83,0xDF,0x17,0x32,0x09,0x4E,0xD1,
57      0xE7,0xCD,0x8A,0x91,0xC6,0xD5,0xC4,0xC4,
58      0x40,0x21,0x18,0x4E,0x55,0x86,0xF4,0xDC,
59      0x8A,0x15,0xA7,0xEC,0x92,0xDF,0x93,0x53,
60      0x30,0x18,0xCA,0x34,0xBF,0xA2,0xC7,0x59,
61      0x67,0x8F,0xBA,0x0D,0x6D,0xD8,0x2D,0x7D,
62      0x54,0x0A,0x57,0x97,0x70,0x39,0xD2,0x7A,
63      0xEA,0x24,0x33,0x85,0xED,0x9A,0x1D,0xE0
64  };  // This table are 64bytes PN9 pseudo random code.
```

Function：Declaration of the parameter

Line	Description
39~51	Declare parameters that are used in the procedure
53	Declare BitCount_Tab
54	Declare ID_Tab for ID code
55~64	Declare PN9_Tab of 64 bytes PN9 code

```
const Uint16 code A7125Config[]=
{
    0x00, //RESET register,        only reset, not use on config
    0x42, //MODE register,
    0x00, //CALIBRATION register,only read, not use on config
    0x7F, //FIFO1 register,     128 bytes
    0xC0, //FIFO2 register,     PFM[1:0]=[11]
    0x00, //FIFO register,         for fifo read/write
    0x00, //IDDATA register,       for idcode
    0x00, //RCOSC1 register,
    0x00, //RCOSC2 register,
    0x00, //RCOSC3 register,
    0x12, //CKO register,
    0x01, //GIO1 register
    0x00, //GIO2 register,
    0x1F, //DATARATE register,
    0x50, //PLL1 register,
    0x0E, //PLL2 register,         RFbase 2400MHz
    0x96, //PLL3 register,
    0x00, //PLL4 register,
    0x04, //PLL5 register,
    0x3C, //chanel group I,
    0x78, //chanel group II,
    0xD7, //TX1 register,
    0x40, //TX2 register,
    0x10, //DELAY1 register,
    0x61, //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
    0x7A, //BATTERY register,
    0x2F, //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,
    0xFF, //IFAT register,
    0x37, //Channel select register,
    0xFF  //VRB register
};
```

Function： Configure of A7125's control registers for Master

Line	Description

68~124	Configure of A7125's control registers for Master

```
127  /*********************************************************************
128   **  function Declaration
129   *********************************************************************/
130   void InitTimer0(void);
131   void initUart0(void);
132   void Timer0ISR (void);
133   void Uart0Isr(void);
134   void A7125_Reset(void);
135   void A7125_WriteReg(Uint8, Uint8);
136   Uint8 A7125_ReadReg(Uint8);
137   void ByteSend(Uint8 src);
138   Uint8 ByteRead(void);
139   void A7125_WriteID(void);
140   void A7125_WriteFIFO(void);
141   void initRF(void);
142   void A7125_Config(void);
143   void CHGroupCal(Uint8);
144   void A7125_Cal(void);
145   void RxPacket(void);
146   void StrobeCmd(Uint8);
147   void SetCH(Uint8);
148   void A7125_WriteFIFO_1(void);
149   void ReadDataToBuf(Uint8 *);
```

Function：Subroutine declaration

Line	Description
130~149	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)
        {
            StrobeCmd(CMD_TFR); //reset tX FIFO pointer
            A7125_WriteFIFO(); //write 1~64 bytes data to tx fifo
            SetCH(100); //freq 2450.001MHz
            StrobeCmd(CMD_TX); //entry tx

            while(~CKO);
            A7125_WriteFIFO_1(); //write 65~128 bytes data to tx fifo
            while(GIO1); //wait transmit completed

            Delay10ms(5);
        }
    }
    else //if P4.2=0, slave
    {
        initRF();

        RxCnt = 0;
        Err_ByteCnt = 0;
        Err_BitCnt = 0;
        //TR0 = 1;
        StrobeCmd(CMD_STBY);
```

```
200         while(1)
201         {
202             SetCH(96); //freq 2448.001MHz
203             StrobeCmd(CMD_RX); //entry rx
204
205             while(~CKO);
206             StrobeCmd(CMD_RFR); //reset rx FIFO pointer
207             ReadDataToBuf(&tmpbuf[0]); //read data to buf
208             while(GIO1); //wait receive completed
209             ReadDataToBuf(&tmpbuf[64]); //read data to buf
210             RxPacket(); //data compare
211
212             Delay10ms(3);
213         }
214     }
215 }
```

Function： Main loop。MCU Identifies Port4_2 = 1，line 172~189 are Master code. line 191~214 are Slave code.

Line	Description
157	Enanle the MCU on chip data SRAM
160~164	Initial MCU I/O Port
166	Call initTimer0 subroutine and enable interrupt
167	Call initUart0 subroutine and initial Uart0
168~169	Enable Timer0 and Timer0 interrupt
171	Identify if port4_2=1, jump to Master code, else jump to Slave code.
172~189	Master code
173	Call initRF subroutine to initial A7125
174	Use Strobe command to enter Standby mode
178	Use Strobe command to reset TX FIFO pointer
179	Call A7125_WriteFIFO subroutine to write data into TX FIFO
180	Call SetCH subroutine to setup the TX radio frequency.
181	User Strobe command to enter TX mode and A7125 deliver TX FIFO automatically.
183	Identify I/O CKO, waiting for whether to have transmitted 48 bytes
184	Call A7125_WriteFIFO_1 subroutine to write data into TX FIFO
185	Identify I/O GIO1, waiting for TX transmitted
187	Delay 50ms
191~214	Slave code
192	Call initRF subroutine to initial A7125
194~196	Clean the variable of RxCnt, Err_ByteCnt, Err_BitCnt
198	Use Strobe command to enter Standby mode
202	Call SetCH subroutine to setup the RX radio frequency.
203	Use Strobe command to enter RX mode
205	Identify I/O CKO, waiting for whether to have received 48 bytes
206	Use Strobe command to reset RX FIFO pointer.
207	Call ReadDataToBuf subroutine to read out data
208	Identify I/O GIO1, waiting for whether to have received.
209	Call ReadDataToBuf subroutine to read out data
210	Call RxPacket subroutine to read data from RX FIFO , do authentication and calculate BER
212	Delay 30ms

```
217 /*********************************************************************
218  **  init Timer0
219  *********************************************************************/
220 void InitTimer0(void)
221 {
222     TR0 = 0;
223     TMOD =(TMOD & 0xF0)|0x01; //timer0 mode=1
224     TH0 = (65536-t0hrel)>>8; // Reload Timer0 high byte, low byte
225     TL0 = 65536-t0hrel;
226     TF0 = 0; // Clear any pending Timer0 interrupts
227     ET0 = 1; // Enable Timer0 interrupt
228 }
```

Function： Initial Timer0	
Line	Description
222	Disable Timer0
223	Setup Timer0 in mode1 mode
224~225	Setup the initial value of TH0,TL0
226	Clean Timer0 interrupt flag
227	Enable Timer0 interrup

```
230 /*********************************************************************
231  **  Timer0ISR
232  *********************************************************************/
233 void Timer0ISR (void) interrupt 1
234 {
235     TF0 = 0; // Clear Timer0 interrupt
236     TH0 = (65536-t0hrel)>>8; // Reload Timer0 high byte,low byte
237     TL0 = 65536-t0hrel;
238
239     timer++;
240     if (timer == TIMEOUT)
241     {
242         TimeoutFlag=1;
243     }
244
245     TimerCnt0++;
246     if (TimerCnt0 == 500)
247     {
248         TimerCnt0 = 0;
249         CmdBuf[0] = 0xF1;
250
251         memcpy(&CmdBuf[1], &RxCnt, 2);
252         memcpy(&CmdBuf[3], &Err_ByteCnt, 4);
253         memcpy(&CmdBuf[7], &Err_BitCnt, 4);
254         memcpy(&CmdBuf[11], &Err_Frame, 1);
255
256         UartSendCnt = 12;
257         Uartptr =& CmdBuf[0];
258         SBUF = CmdBuf[0];
259     }
260 }
```

Function： Timer0 interrupt service routine	
Line	Description
235~237	Clear Timer0 interrupt flag and setup initial value of TH0 and TL0.
239	Increase timer (timer is a variable).
240~243	Check timer == TIMEOUT, if yes, set TimeoutFlag =1.

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

```
262 /*************************************************************************
263  **  Init Uart0
264  *************************************************************************/
265 void initUart0(void)
266 {
267     TH1 = 0xFD; //BaudRate 9600;
268     TL1 = 0xFD;
269     SCON = 0x40;
270     TMOD = (TMOD & 0x0F) | 0x20;
271     REN = 1;
272     TR1 = 1;
273     ES = 1;
274 }
```

Function：Initialize the procedure of Uart0

Line	Description
267~269	Set initial value of TL1,TH1,SCON1 and Set 9600 baud rate @xtal=11.0592MHz
270	Set Timer1 in mode 2
271~273	Set REN=1, TR1=1, ES=1 to enable UART function.

```
276 /*************************************************************************
277  **  Uart0 ISR
278  *************************************************************************/
279 void Uart0Isr(void) interrupt 4 using 3
280 {
281     if (TI==1)
282     {
283         TI=0;
284         UartSendCnt--;
285         if(UartSendCnt !=0)
286         {
287             Uartptr++;
288             SBUF = *Uartptr;
289         }
290     }
291 }
```

Function ：Initialize the interrupt subprogram of uart0.

Line	Description
281	Check TI flag, if TI=1, one byte of UART transmission is done.
283	Clear TI flag
284	Decrease UartSendCnt (UartSendCnt is a variable)
285	Check UartSendCnt == 0, if not, continue UART transmission until done.

287~288	Increase pointer Uartptr to assign address of next data via UART to PC

```
293  /*************************************************************************
294   **  Reset_RF
295   *************************************************************************/
296   void A7125_Reset(void)
297   {
298       A7125_WriteReg(MODE_REG, 0x00); //reset RF chip
299   }
```

Function：A7125 RF chip Reset step

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

```
301  /*************************************************************************
302   **  WriteID
303   *************************************************************************/
304   void A7125_WriteID(void)
305   {
306       Uint8 i;
307       Uint8 d1,d2,d3,d4;
308       Uint8 addr;
309
310       addr = IDCODE_REG; //send address 0x06, bit cmd=0, r/w=0
311       SCS = 0;
312       ByteSend(addr);
313       for (i=0; i < 4; i++)
314           ByteSend(ID_Tab[i]);
315       SCS = 1;
316
317       //for check
318       addr = IDCODE_REG | 0x40; //send address 0x06, bit cmd=0, r/w=1
319       SCS=0;
320       ByteSend(addr);
321       d1=ByteRead();
322       d2=ByteRead();
323       d3=ByteRead();
324       d4=ByteRead();
325       SCS=1;
326   }
```

Function：Write ID procedure.

Line	Description
310	Set parameter addr = 0x06 for ID code write operation.
311	Set SCS=0 to enable SPI interface.
312~314	Write ID_Tab Table into A7125 ID Code registers.
315	Set SCS=1 to disable SPI interface.
318	Set parameter addr = 0x06 for ID code read operation.
319	Set SCS=0 to enable SPI interface.
320~324	Read 4 byte ID code for ID check.
325	Set SCS=1 to disable SPI interface.

```
328  /*********************************************************************
329   **  A7125_WriteReg
330   *********************************************************************/
331  void A7125_WriteReg(Uint8 addr, Uint8 dataByte)
332  {
333      Uint8 i;
334
335      SCS = 0;
336      addr |= 0x00; //bit cmd=0,r/w=0
337      for(i = 0; i < 8; i++)
338      {
339          if(addr & 0x80)
340              SDIO = 1;
341          else
342              SDIO = 0;
343
344          SCK = 1;
345          _nop_();
346          SCK = 0;
347          addr = addr << 1;
348      }
349      _nop_();
350
351      //send data byte
352      for(i = 0; i < 8; i++)
353      {
354          if(dataByte & 0x80)
355              SDIO = 1;
356          else
357              SDIO = 0;
358
359          SCK = 1;
360          _nop_();
361          SCK = 0;
362          dataByte = dataByte<< 1;
363      }
364      SCS = 1;
365  }
```

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

Line	Description
335	Set SCS=0 to enable SPI interface.
336	Enable write operation of control registers (addr7=0 for control registers, addr6=0 for write operation)
337~348	Assign control register's address by input variable addr.
352~363	Assign control register's value by input variable dataByte
364	Set SCS=1 to disable SPI interface.

```
367 /*********************************************************************
368  **  A7125_ReadReg
369  *********************************************************************/
370 Uint8 A7125_ReadReg(Uint8 addr)
371 {
372     Uint8 i;
373     Uint8 tmp;
374
375     SCS = 0;
376     addr |= 0x40; //bit cmd=0,r/w=1
377     for(i = 0; i < 8; i++)
378     {
379
380         if(addr & 0x80)
381             SDIO = 1;
382         else
383             SDIO = 0;
384
385                 _nop_();
386         SCK = 1;
387         _nop_();
388         SCK = 0;
389
390         addr = addr << 1;
391     }
392
393     _nop_();
394     SDIO = 1;
395
396     //read data
397     for(i = 0; i < 8; i++)
398     {
399         if(SDIO)
400             tmp = (tmp << 1) | 0x01;
401         else
402             tmp = tmp << 1;
403
404         SCK = 1;
405         _nop_();
406         SCK = 0;
407     }
408     SCS = 1;
409     return tmp;
410 }
```

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

Line	Description
375	Enable read operation of control registers (addr7=0 for control registers, addr6=1 for read operation)
376	Assign control register's address by input variable addr.
377~391	Assign control register's address by input variable addr.
394	Set SDIO=1 for SPI Data Output
397~407	The control register's value is output to SPI interface.
408	Set SCS=1 to disable SPI interface.
409	Return 1 byte read value.

```
412  /*********************************************************************
413   **  ByteSend
414   *********************************************************************/
415   void ByteSend(Uint8 src)
416   {
417       Uint8 i;
418
419       for(i = 0; i < 8; i++)
420       {
421           if(src & 0x80)
422               SDIO = 1;
423           else
424               SDIO = 0;
425
426                   _nop_();
427           SCK = 1;
428           _nop_();
429           SCK = 0;
430           src = src << 1;
431       }
432   }
```

Function：subroutine of ByteSend	
Line	Description
419~431	Write 1 byte data to A7125

```
434  /*********************************************************************
435   **  ByteRead
436   *********************************************************************/
437   Uint8 ByteRead(void)
438   {
439       Uint8 i,tmp;
440
441       SDIO = 1; //sdio pull high
442       for(i = 0; i < 8; i++)
443       {
444           if(SDIO)
445               tmp = (tmp << 1) | 0x01;
446           else
447               tmp = tmp << 1;
448
449           SCK = 1;
450           _nop_();
451           SCK = 0;
452       }
453       return tmp;
454   }
```

Function：Subroutine of ByteRead	
Line	Description
441~452	Read 1 byte data from A7125
453	Return 1 byte value

```
456 /*********************************************************************
457  **  Send4Bit
458  *********************************************************************/
459 void Send4Bit(Uint8 src)
460 {
461     Uint8 i;
462
463     for(i = 0; i < 4; i++)
464     {
465         if(src & 0x80)
466             SDIO = 1;
467         else
468             SDIO = 0;
469
470         _nop_();
471         SCK = 1;
472         _nop_();
473         SCK = 0;
474         src = src << 1;
475     }
476 }
```

Function：Subroutine of Send4Bit

Line	Description
463~475	Write 4 bits data to A7125.

```
478 /*********************************************************************
479  ** SetCH
480  *********************************************************************/
481 void SetCH(Uint8 ch)
482 {
483     //RF freq = RFbase + (CH_Step * ch)
484     A7125_WriteReg(PLL1_REG, ch);
485 }
```

Function：Channel setting procedure

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

```
487  /*****************************************************************
488   ** initRF
489   *****************************************************************/
490   void initRF(void)
491   {
492       //init io pin
493       SCS = 1;
494       SCK = 0;
495       SDIO = 1;
496       CKO = 1;
497       GIO1 = 1;
498       GIO2 = 1;
499
500       A7125_Reset(); //reset A7125 RF chip
501       A7125_WriteID(); //write ID code
502       A7125_Config(); //config A7125 chip
503       A7125_Cal(); //calibration RSSI, IF,vco, vcoc, vco deviation
504   }
```

Function：Initial RF chip procedure

Line	Description
493~498	Set up I/O initial value between MCU and A7125
500	Call A7125_Reset subroutine to reset A7125
501	Call A7125_WriteID subroutine to write 4 byte ID code.
502	Call A7125_Config subroutine to initial A7125's control registers.
503	Call A7125_Cal subroutine to do calibration procedures for IF, VCO, and VCO current.

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

Function： write Tx FIFO procedure

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

```
522  /*********************************************************************
523   ** A7125_WriteFIFO_1
524   *********************************************************************/
525   void A7125_WriteFIFO_1(void)
526   {
527       Uint8 i;
528       Uint8 cmd;
529
530       cmd = FIFO_REG; //send address 0x05, bit cmd=0, r/w=0
531       SCS=0;
532       ByteSend(cmd);
533       for(i=0; i <64; i++)
534           ByteSend(~PN9_Tab[i]);
535       SCS=1;
536   }
```

Function： write Tx FIFO procedure

Line	Description
530	Assign address = 0x05 for TX FIFO of write operation.
531	Set SCS=0 to enable SPI interface.
532	Call ByteSend subroutine of for one byte write operation.
533~534	Write PN9_Tab into TX FIFO, total 64 bytes.
535	Set SCS=1 to disable SPI interface.

```
538  /*********************************************************************
539   ** Strobe Command
540   *********************************************************************/
541   void StrobeCmd(Uint8 cmd)
542   {
543       SCS = 0;
544       Send4Bit(cmd);
545       SCS = 1;
546   }
```

Function：Strobe command procedure.

Line	Description
543	Set SCS=0 to enable SPI interface
544	Call Send4Bit subroutine for 4-bits Strobe command.
545	Set SCS=1 to disable SPI interface.

```
548  /*********************************************************************
549   ** RxPacket
550   *********************************************************************/
551   void RxPacket(void)
552   {
553       Uint8 i;
554       Uint8 recv;
555       Uint8 tmp;
556
557       RxCnt++;
558       for(i=0; i <64; i++)
559       {
560           recv = tmpbuf[i];
561           if((recv ^ PN9_Tab[i])!=0)
562           {
563               tmp = recv ^ PN9_Tab[i];
564               Err_BitCnt += (BitCount_Tab[tmp>>4] + BitCount_Tab[tmp & 0x0F]);
565           }
566       }
567
568       for(i=0; i <64; i++)
569       {
570           recv = tmpbuf[i+64];
571           if((recv ^ (PN9_Tab[i]+1))!=0)
572           {
573               tmp = recv ^ (~PN9_Tab[i]);
574               Err_BitCnt += (BitCount_Tab[tmp>>4] + BitCount_Tab[tmp & 0x0F]);
575           }
576       }
577   }
```

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

Line	Description
557	Increase parameter RxCnt
558~566	Read out tmpbuf[0]~tmpbuf[63], compare data , calculate error bit.
568~576	Read out tmpbuf[64]~tmpbuf[127], compare data , calculate error bit.

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

Function：Error state processing procedure

Line	Description
584~585	Error processing procedure by user define

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

Function：Calibration procedure of Channel Group

Line	Description
598	Call Subroutine of A7125_WriteReg to write PLL1 register.
599	Delay 40us for PLL settling stable
600	Set VCC=1, VBC=1, VDC=1 to enable VCO current, VCO band and VCO deviation.
601~605	Check VCC=0, VBC=0, VDC bit=0, if yes, exit the loop.
608~610	Read out from VCO Calibration register, check flag, value
612~614	Read out from VCO Current calibration register, check flag, value
616~617	If (vbcf =1) or (vbcf2=1), call subroutine of Error_state to do error handling.

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

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

Line	Description
630	Use Strobe command to set A7125 in PLL state
631	Set RSSC=1 to enable RSSI auto calibration Set FBC=1 to enable IF auto calibration.
632~637	Checks RSSC==0 and FBC==0, if yes, exit the loop.
641~643	Call CHGroupCal subroutine I to enable channel group calibration.
644	Use Strobe command to set A7125 in Standby state.
647~655	Assign variable fb, fbcf from read IFCAL1_REG and Assign variable fdc from read IFCAL2_REG and Assign variable dvt from read VCOCAL2_REG
657~658	Check flag fbcf. If fbcf=1, jump to Error state processing process.

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

Function：Initial RF chip procedure

Line	Description
673~674	Call A7125_WriteReg subroutine to initial control register from address 0x01 to 0x04
676~677	Call A7125_WriteReg subroutine to initial control register from address 0x07 to 0x23
679~680	Call A7125_WriteReg subroutine to initial control register from address 0x25 to 0x38

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

Function：Read out data to tmpbuf procedure

Line	Description
692	Assign FIFO control register's address to read out data
694	Set SCS=0，enable SPI interface.
695	Call ByteSend subroutine
696~700	Read out 64 bytes data and setting to variable tmpbuf.
701	Set SCS=1, disable SPI interface.