Source Codes for Demo kit DAD03

V2.00

This document contains simplified source codes for demo kit DAD03 which can be used for testing RF front-end modules without MCU. It demonstrates how to configure registers by using of excel calculation from Silicon labs and fulfill the basic transmit/receive communication. The codes only can be used for design reference. DORJI isn't responsible for any mistake contained in this document and any risk from using of the codes will be undertaken by users.

```
// Module types:          DRF4431F13, DRF4432F20,and DRF4431F27
// Parameter setting:     FSK，433.5MHz，1.2KBPS，+/-10PPM, Deviation: 30KHz，BW：61.2KHz
//                        AFC, CRC Enable，PH + FIFO, Header: "swwx", Synch Word: 0x2d, 0xd4
// Data for Tx/Rx test:   0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,0x49,0x6d
//                        the last byte 0x6d is the checksum of the first nine bytes
// MCU :                  Microchip 8 bit PIC16F689 with internal OSC

#include<pic.h>
#include<math.h>


const unsigned char tx_test_data[10] = {0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,0x49,0x6d};
// Fixed testing data sent per second. The last byte is the checksum.


#define SI4432_PWRSTATE_READY              01      // Si4432 ready mode define
#define SI4432_PWRSTATE_TX                 0x09    // Si4432 Tx mode define
#define SI4432_PWRSTATE_RX                 05      // Si4432 Rx mode define
#define SI4432_PACKET_SENT_INTERRUPT       04      // Si4432 packet sent interrupt define
#define SI4432_Rx_packet_received_interrupt 0x02   // Si4432 packet received interrupt define

#define   TX1_RX0   spi_rw(0x0e|0x80, 0x01)        // Antenna switch to tx mode
#define   TX0_RX1   spi_rw(0x0e|0x80, 0x02)        // Antenna switch to Rx mode
#define   TX0_RX0   spi_rw(0x0e|0x80, 0x00)        // Antenna is not in Tx/Rx mode

#define   nIRQ      RC0     // MCU input port
#define   SDO       RC1     // MCU input port
#define   nSEL      RC2     // MCU output port
#define   SDI       RC3     // MCU output port
#define   SCK       RC4     // MCU output port
#define   SDN       RC5     // MCU output port

#define nIRQ_set    TRISC0  // Set the direction of input/output
#define SDO_set     TRISC1  // Set the direction of input/output
#define nSEL_set    TRISC2  // Set the direction of input/output
#define SDI_set     TRISC3  // Set the direction of input/output
#define SCK_set     TRISC4  // Set the direction of input/output
#define SDN_set     TRISC5  // Set the direction of input/output

#define INPUT       1
#define OUTPUT      0
```

```
unsigned char count_50hz;
unsigned char ItStatus1, ItStatus2;
unsigned char rf_timeout;
unsigned char rx_buf[15];

typedef struct
{

    unsigned char reach_1s              : 1;
    unsigned char rf_reach_timeout      : 1;
    unsigned char is_tx                 : 1;



}   FlagType;

FlagType        Flag;


void rx_data(void);
void tx_data(void);

unsigned char spi_byte(unsigned char data);
unsigned char spi_rw(unsigned char addr, unsigned char data);

void SI4432_init(void);
void delay_1ms(unsigned char time);
void port_init(void);
void timer_init(void);

void main()
{

    unsigned char   i, j, chksum;
    OSCCON = 0X70;      // Using internal OSC，  8M crystal
    WDTCON = 0X00;      // Watch dog disable
    port_init();        // Port initial
    SDN = 1;
    delay_1ms(10);      // RF module reset
    SDN = 0;
    delay_1ms(200);     // Delay 200 ms
    SI4432_init();       // RF module parameters initializing
    TX0_RX0;            // Set the antenna switch,not in Tx/Rx
     timer_init();      // timer initializing ,20ms/interrupt
```

```
count_50hz = 0;
Flag.reach_1s = 0;
INTCON = 0xc0;      // enable interrupt

while(1)
{
      if(Flag.reach_1s)
      {
            Flag.reach_1s = 0;
            tx_data();           // Transmit data per 1s and then receive the Acknowledge
      }
      if(!Flag.is_tx)
      {
            if(!nIRQ)
            {
                  ItStatus1 = spi_rw(0x03,0x00);      // clear interrupt factor
                                                      //read the Interrupt Status1 register
                  ItStatus2 = spi_rw(0x04,0x00);      // clear interrupt factor
                  SCK = 0;
                  nSEL = 0;
                  spi_byte(0x7f);                     // read data from the Si4432 FiFo
                  for(i = 0;i<10;i++)
                  {
                        rx_buf[i] = spi_byte(0x00);
                  }
                  nSEL = 1;
                  spi_rw(0x07|0x80, SI4432_PWRSTATE_READY);
                  // Exit Rx mode after all the data read from the FiFo

                  chksum = 0;
                  for(i=0;i<9;i++)          // calculate the checksum for the received data
                  chksum += rx_buf[i];

                        if(( chksum == rx_buf[9] )&&( rx_buf[0] == 0x41 ))

                        {
                              ;     // data verified OK
                  }
                  else
                  {
                        rx_data();          // Data Verified error, then restart to Rx
                  }
```

```
                }
            }
        }
}


void delay_1ms(unsigned char time)
{
    unsigned char i,k;
    for(k = 0; k< time; k++)
    {
        for(i = 0; i<130; i++)
        {
            NOP();
        }
    }
}


void timer_init(void)
{
    T1CON = 0x31;
    TMR1IE = 1;
    TMR1L = 0x78;   // timer initializing
    TMR1H = 0xec;

}

void interrupt ISR_timer(void)
{

    unsigned char i;
    if(TMR1IF)
    {
        TMR1L = 0x78;          // Reset the reload counter for the timer
        TMR1H = 0xec;
        rf_timeout++;
        if(rf_timeout == 25)
        {
            Flag.rf_reach_timeout = 1;   // 0.5s timeout for Tx
        }
        count_50hz++;
        if(count_50hz==50)    //
```

```
                {
                    count_50hz=0;
                    Flag.reach_1s = 1;      // 1s timeout

                }

            TMR1IF=0;
        }

}

void port_init(void)
{
        ANSEL = 0;
        ANSELH = 0;    // AD not used
        WPUA = 0;       // No pull-up
        IOCA = 0;       // No I/O interrupt


        nIRQ_set     = INPUT;
        SDO_set      = INPUT;
        nSEL_set      = OUTPUT;
        SDI_set       = OUTPUT;
        SCK_set       = OUTPUT;
        SDN_set       = OUTPUT;
        TRISB5        = OUTPUT;
}

void SI4432_init(void)
{
        ItStatus1 = spi_rw(0x03,0x00);     // clear interrupt factor of Si4432
        ItStatus2 = spi_rw(0x04,0x00);
        spi_rw(0x06|0x80, 0x00);            // disable the interrupt of Si4432
        spi_rw(0x07|0x80, SI4432_PWRSTATE_READY);     // to Ready mode
        spi_rw(0x09|0x80, 0x7f);            // load Capacitance = 12PF
        spi_rw(0x0a|0x80, 0x05);            // output clock
        spi_rw(0x0b|0x80, 0xea);            // GPIO 0 = digital output
        spi_rw(0x0c|0x80, 0xea);            //GPIO 1 = digital output
        spi_rw(0x0d|0x80, 0xf4);            // /GPIO 2 = Rx data

        // The settings below are obtained from the Excel calculation table from silicon labs
        spi_rw(0x70|0x80, 0x2c);
        spi_rw(0x1d|0x80, 0x40);    // enable AFC
```

```
// 1.2K bps setting
spi_rw(0x1c|0x80, 0x16);
spi_rw(0x20|0x80, 0x83);
spi_rw(0x21|0x80, 0xc0); //
spi_rw(0x22|0x80, 0x13);//
spi_rw(0x23|0x80, 0xa9); //
spi_rw(0x24|0x80, 0x00); //
spi_rw(0x25|0x80, 0x04); //
spi_rw(0x2a|0x80, 0x14);
spi_rw(0x6e|0x80, 0x09);
spi_rw(0x6f|0x80, 0xd5);

//1.2K bps setting end

spi_rw(0x30|0x80, 0x8c);       // PH + FiFo, MSB , CRC enabled
spi_rw(0x32|0x80, 0xff);       // Header= Byte0, 1, 2, 3
spi_rw(0x33|0x80, 0x42);       // Sync = byte 3,2
spi_rw(0x34|0x80, 16);         // Tx Preamble = 16 nibble
spi_rw(0x35|0x80, 0x20);       // Detected preamble = 4 nibble
spi_rw(0x36|0x80, 0x2d);       // Sync word = 0x2dd4
spi_rw(0x37|0x80, 0xd4);
spi_rw(0x38|0x80, 0x00);
spi_rw(0x39|0x80, 0x00);
spi_rw(0x3a|0x80, 's');        // Tx header = swwx"
spi_rw(0x3b|0x80, 'w');
spi_rw(0x3c|0x80, 'w');
spi_rw(0x3d|0x80, 'x');
spi_rw(0x3e|0x80, 10);         // payload = 10 byte
spi_rw(0x3f|0x80, 's');        // header checked = ” swwx"
spi_rw(0x40|0x80, 'w');
spi_rw(0x41|0x80, 'w');
spi_rw(0x42|0x80, 'x');
spi_rw(0x43|0x80, 0xff);       // all bit need be checked
spi_rw(0x44|0x80, 0xff);       //
spi_rw(0x45|0x80, 0xff);       //
spi_rw(0x46|0x80, 0xff);       //
spi_rw(0x6d|0x80, 0x07);       // max power output

spi_rw(0x79|0x80, 0x0);        // no hopping
spi_rw(0x7a|0x80, 0x0);        // no hopping
spi_rw(0x71|0x80, 0x22);       // RF mode = FSK , FiFo
spi_rw(0x72|0x80, 0x30);       // Frequency deviation =    30KHz
```

```
        spi_rw(0x73|0x80, 0x0);          // No freq offset
        spi_rw(0x74|0x80, 0x0);          // No freq offset
        spi_rw(0x75|0x80, 0x53);         // freq = 433.5MHz
        spi_rw(0x76|0x80, 0x57);         //
        spi_rw(0x77|0x80, 0x80);
}

void rx_data(void)
{
        unsigned char i, chksum;
        Flag.is_tx = 0;
        spi_rw(0x07|0x80, SI4432_PWRSTATE_READY);   // enter Ready mode
        delay_1ms(5);                                // stablize the OSC; not needed if OSC is on
        TX0_RX1;                                     // antenna switch = Rx mode
        spi_rw(0x08|0x80, 0x03);                     //clear Tx/Rx fifo
        spi_rw(0x08|0x80, 0x00);                     //clear Tx/Rx fifo
        spi_rw(0x07|0x80,SI4432_PWRSTATE_RX );       // enter Rx mode

        spi_rw(0x05|0x80, SI4432_Rx_packet_received_interrupt);   // interrupt for packet received
        ItStatus1 = spi_rw(0x03,0x00);                            //clear all interrupt factor
        ItStatus2 = spi_rw(0x04,0x00);                            //clear all interrupt factor
              }

void tx_data(void)
{
        unsigned char i;
        Flag.is_tx = 1;
        spi_rw(0x07|0x80, SI4432_PWRSTATE_READY);   // enter Ready mode
        TX1_RX0;                                     //Antenna switch = Tx mode
        delay_1ms(5);                // stablize the OSC; not needed if OSC is on
        spi_rw(0x08|0x80, 0x03);     // clear Tx/Rx fifo
        spi_rw(0x08|0x80, 0x00);     // clear Tx/Rx fifo
        spi_rw(0x34|0x80, 16);       // preamble tx = 16 nibble
        spi_rw(0x3e|0x80, 10);       // payload = 10 bytes
        for (i = 0; i<10; i++)
        {
              spi_rw(0x7f|0x80, tx_test_data[i]);                  //load payload into the fifo
        }
        spi_rw(0x05|0x80, SI4432_PACKET_SENT_INTERRUPT);    // interrupt after packet is sent
        ItStatus1 = spi_rw(0x03,0x00);                       // clear interrupt factor
        ItStatus2 = spi_rw(0x04,0x00);
        spi_rw(0x07|0x80, SI4432_PWRSTATE_TX);      // enter Tx mode
```

```
    rf_timeout = 0;
    Flag.rf_reach_timeout = 0;                  // set Tx timeout
    while(nIRQ)                                 // wait for interrupt
    {

        if(Flag.rf_reach_timeout)
        {

            SDN    = 1;                         //if no interrupt in 0.5s, then exit the loop
            delay_1ms(10);
            SDN = 0;
            delay_1ms(200);
            SI4432_init();
            break;
        }

    }
    rx_data();          //enter Rx mode after Tx
}


unsigned char spi_byte(unsigned char data)
{
    unsigned char i;

    for (i = 0; i < 8; i++)          // read data at rising edge of SCK and send data at the falling edge of SCK
    {                                //
        if (data & 0x80)
            SDI = 1;
        else
            SDI = 0;

        data <<= 1;
        SCK = 1;

        if (SDO)
            data |= 0x01;
        else
            data &= 0xfe;

        SCK = 0;
    }
```

```
	return (data);
}
//-------------------------------------------
unsigned char spi_rw(unsigned char addr, unsigned char data)
{
	unsigned char i;
	SCK = 0;
	nSEL = 0;
	for (i = 0; i < 8; i++)
	{
		if (addr & 0x80)
			SDI = 1;
		else
			SDI = 0;
		addr <<= 1;
		SCK = 1;
		asm("NOP");
		SCK = 0;
	}

	for (i = 0; i < 8; i++)
	{
		if (data & 0x80)
			SDI = 1;
		else
			SDI = 0;
		data <<= 1;
		SCK = 1;
		if (SDO)
			data |= 0x01;
		else
			data &= 0xfe;
		SCK = 0;
	}
	nSEL = 1;
	SCK = 1;
	return (data);
}
```

Dorji Applied Technologies
A division of Dorji Industrial Group Co., Ltd

Add.: Xinchenhuayuan 2, Dalangnanlu, Longhua, Baoan district, Shenzhen, China 518109
Tel: 0086-755-28156122
Fax.: 0086-755-28156133
Email: sales@dorji.com
Web: http://www.dorji.com

Dorji Industrial Group Co., Ltd reserves the right to make corrections, modifications, improvements and other changes to its products and services at any time and to discontinue any product or service without notice. Customers are expected to visit websites for getting newest product information before placing orders.

These products are not designed for use in life support appliances, devices or other products where malfunction of these products might result in personal injury. Customers using these products in such applications do so at their own risk and agree to fully indemnify Dorji Industrial Group for any damages resulting from improper use.