Verzenden en ontvangen met de RFM12

In dit document vindt u de code voor het verzenden en ontvangen met de RFM12. Veel mensen hebben om te verzenden of te ontvangen met de modules. Om ze op weg te helpen vindt u hier een simpele werkende code voor het verzenden en ontvangen met de RFM12 en de ATmega32. Met deze code kunt u een begin maken en vervolgens zelf aanpassen, zodat de RFM12 modules doen wat u voor ogen had.

Inhoud

Verzenden met de RFM12.....3

Ontvangen met de RFM12 6

Verschillen RFM12 en RFM12B 10

Verzenden met de RFM12

```
#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 8000000UL
#include <util/delay.h>

#define RF_PORT         PORTB
#define RF_DDR          DDRB
#define RF_PIN PINB

#define DDR_IN                  0
#define DDR_OUT                 1

#define PORT_SEL        PORTB
#define PIN_SEL                 PINB
#define DDR_SEL                 DDRB

#define PORT_SDI        PORTB
#define PIN_SDI                 PINB
#define DDR_SDI                 DDRB

#define PORT_SCK        PORTB
#define PIN_SCK                 PINB
#define DDR_SCK                 DDRB

#define PORT_SDO        PORTB
#define PIN_SDO                 PINB
#define DDR_SDO                 DDRB

#define PORT_INT        PORTD
#define PIN_INT                 PIND
#define DDR_INT                 DDRD

#define PORT_LED        PORTB
#define PIN_LED                 PINB
#define DDR_LED                 DDRB

#define SCK     7
#define SDO     6
#define SDI     5
#define SEL     4
#define INT             2
#define LED     1

#define LED_OUTPUT() DDR_LED  |= (1<<LED)
#define HI_LED()                PORT_LED |= (1<<LED)
#define LOW_LED()               PORT_LED &=~(1<<LED)

#define SEL_OUTPUT()  DDR_SEL  |= (1<<SEL)
#define HI_SEL()                PORT_SEL |= (1<<SEL)
```

```
#define LOW_SEL()                PORT_SEL &=~(1<<SEL)

#define SDI_OUTPUT()  DDR_SDI  |= (1<<SDI)
#define HI_SDI()                 PORT_SDI |= (1<<SDI)
#define LOW_SDI()                PORT_SDI &=~(1<<SDI)

#define SDO_INPUT()       DDR_SDO  &=~(1<<SDO)
#define SDO_HI()                 PIN_SDO&(1<<SDO)

#define INT_INPUT()              DDR_INT  &=~(1<<INT)
#define INT_HI()                 PIN_INT&(1<<INT)

#define SCK_OUTPUT() DDR_SCK  |=(1<<SCK)
#define HI_SCK()                 PORT_SCK |=(1<<SCK)
#define LOW_SCK()                PORT_SCK &=~(1<<SCK)


void portInit() {
        HI_SEL();
        HI_SDI();
        LOW_SCK();
        SEL_OUTPUT();
        SDI_OUTPUT();
        SCK_OUTPUT();
        LED_OUTPUT();
}


unsigned int writeCmd(unsigned int cmd) {
        unsigned char i;
        unsigned int recv;
        recv = 0;
        LOW_SCK();
        LOW_SEL();
        for(i=0; i<16; i++) {
                if(cmd&0x8000)
                        HI_SDI();
                else
                        LOW_SDI();
                HI_SCK();
                recv<<=1;
                if( PINB&(1<<SDO) ) {
                        recv|=0x0001;
                }
                LOW_SCK();
                cmd<<=1;
        }
        HI_SEL();
        return recv;
}
```

```
void rfInit() {
	writeCmd(0x80D7); //EL,EF,434band,12.0pF
	writeCmd(0x8239); //!er,!ebb,ET,ES,EX,!eb,!ew,DC
	writeCmd(0xA640); //frequency select
	writeCmd(0xC647); //4.8kbps
	writeCmd(0x94D0); //VDI,FAST,67kHz,0dBm,-103dBm
	writeCmd(0xC2AC); //AL,!ml,DIG,DQD4
	writeCmd(0xCA81); //FIFO8,SYNC,!ff,DR
	writeCmd(0xC483); //@PWR,NO RSTRIC,!st,!fi,OE,EN
	writeCmd(0x9820); //!mp,45kHz,MAX OUT
	writeCmd(0xE000); //NOT USED
	writeCmd(0xC800); //NOT USED
	writeCmd(0xC040); //1.66MHz,2.2V
}


void rfSend(unsigned char data){
	while(PIND&(1<<INT));  // wait until nIRQ is low
	writeCmd(0xB800 + data);  // write the next data to B800 register
}


int main() {
	volatile unsigned int i;
	_delay_ms(50);  // iniating
	portInit();
	rfInit();
	while(1) {
		HI_LED();
		writeCmd(0x0000);
		rfSend(0xAA); // PREAMBLE
		rfSend(0xAA);
		rfSend(0xAA);
		rfSend(0x2D); // SYNC
		rfSend(0xD4); // SYNC
		for(i=0; i<16; i++) {
		rfSend(0xA1+i);  // send the data
		}
		rfSend(0xAA); // DUMMY BYTES
		rfSend(0xAA);
		rfSend(0xAA);
		LOW_LED();
		_delay_ms(1000);
	}
}
```

Ontvangen met de RFM12

```
#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 8000000UL
#include <util/delay.h>

#define RF_PORT         PORTB
#define RF_DDR          DDRB
#define RF_PIN PINB

#define DDR_IN                  0
#define DDR_OUT                 1

#define PORT_SEL        PORTB
#define PIN_SEL                 PINB
#define DDR_SEL                 DDRB

#define PORT_SDI        PORTB
#define PIN_SDI                 PINB
#define DDR_SDI                 DDRB

#define PORT_SCK        PORTB
#define PIN_SCK                 PINB
#define DDR_SCK                 DDRB

#define PORT_SDO        PORTB
#define PIN_SDO                 PINB
#define DDR_SDO                 DDRB

#define PORT_INT        PORTD
#define PIN_INT                 PIND
#define DDR_INT                 DDRD

#define PORT_LED        PORTB
#define PIN_LED                 PINB
#define DDR_LED                 DDRB

#define SCK     7
#define SDO     6
#define SDI     5
#define SEL     4
#define INT             2
#define LED     1

#define LED_OUTPUT() DDR_LED  |= (1<<LED)
#define HI_LED()                PORT_LED |= (1<<LED)
#define LOW_LED()               PORT_LED &=~(1<<LED)

#define SEL_OUTPUT()  DDR_SEL  |= (1<<SEL)
```

```
#define HI_SEL()                PORT_SEL |= (1<<SEL)
#define LOW_SEL()               PORT_SEL &=~(1<<SEL)

#define SDI_OUTPUT()  DDR_SDI  |= (1<<SDI)
#define HI_SDI()                PORT_SDI |= (1<<SDI)
#define LOW_SDI()               PORT_SDI &=~(1<<SDI)

#define SDO_INPUT()       DDR_SDO  &=~(1<<SDO)
#define SDO_HI()                PIN_SDO&(1<<SDO)

#define INT_INPUT()             DDR_INT  &=~(1<<INT)
#define INT_HI()                PIN_INT&(1<<INT)

#define SCK_OUTPUT() DDR_SCK  |=(1<<SCK)
#define HI_SCK()                PORT_SCK |=(1<<SCK)
#define LOW_SCK()               PORT_SCK &=~(1<<SCK)

#define BAUD   38400   // F_CPU is 8000000 Hz
#define UBRR_VALUE ( ((F_CPU) + 8UL*(BAUD)) / (16UL*(BAUD)) - 1UL )

unsigned int status;

void uart_init(unsigned int ubrr)
{
 UBRRL = ubrr;
 UBRRH = (ubrr >> 8);
 UCSRC = _BV(URSEL)|_BV(UCSZ1)|_BV(UCSZ0);
 UCSRB = _BV(TXEN);
}


void portInit() {
        HI_SEL();
        HI_SDI();
        LOW_SCK();
        SEL_OUTPUT();
        SDI_OUTPUT();
        SCK_OUTPUT();
        LED_OUTPUT();
}


unsigned int writeCmd(unsigned int cmd) {
        unsigned char i;
        unsigned int recv;
        recv = 0;
        LOW_SCK();
        LOW_SEL();
        for(i=0; i<16; i++) {
                if(cmd&0x8000)
                        HI_SDI();
                else LOW_SDI();
```

```
                HI_SCK();
            recv<<=1;
            if( PINB&(1<<SDO) ) {
                recv|=0x0001;
            }
            LOW_SCK();
            cmd<<=1;
        }
        HI_SEL();
        return recv;
}


void rfInit() {
        writeCmd(0x80D7); //EL,EF,434band,12.0pF
        writeCmd(0x8299); //er,!ebb,ET,ES,EX,!eb,!ew,D
        writeCmd(0xA640); //frequency select
        writeCmd(0xC647); //4.8kbps
        writeCmd(0x94D0); //VDI,FAST,67kHz,0dBm,-103dBm
        writeCmd(0xC2AC); //AL,!ml,DIG,DQD4
        writeCmd(0xCA81); //FIFO8,SYNC,!ff,DR
        writeCmd(0xC483); //@PWR,NO RSTRIC,!st,!fi,OE,EN
        writeCmd(0x9820); //!mp,45kHz,MAX OUT
        writeCmd(0xE000); //NOT USED
        writeCmd(0xC800); //NOT USED
        writeCmd(0xC040); //1.66MHz,2.2V
}

void FIFOReset() {
        writeCmd(0xCA81);
        writeCmd(0xCA83);
}

unsigned char rfRecv() {
        unsigned int data=0;
        while(PIND&(1<<INT));  // wait until nIRQ is low
        status = writeCmd(0x0000);  // read the status register
        if ( (status&0x8000) ) {  // if the status has the right register value
                data = writeCmd(0xB000);  // copy data from FIFO register into data variable
        }

        return (data&0x00FF);
}


int main(void) {
        _delay_ms(50);
        unsigned char data, i;
        DDRA=0xFF;
        PORTA=0xFF;
        portInit();
        uart_init(UBRR_VALUE);
```

```
	rfInit();
	FIFOReset();
	while(1) {
		for (i=0; i<16; i++) {
			HI_LED();
			data = rfRecv();  // copy received data to variable
			while ( !(UCSRA & _BV(UDRE)) ) {};  // wait until UDR register is empty
			UDR=data;  // send data to hyperterminal

			PORTA=~status;  // PORTA is used to check the data
		}
		FIFOReset();  // reset the FIFO for next data package
		LOW_LED();
	}
	return 0;
}
```

Verschillen RFM12 en RFM12B

- The RFM12 works up to 5V, the RFM12B only up to 3.8V (but it won't be damaged by 5V).
- The RFM12 needs a pull-up resistor of 10..100 KΩ on the FSK/DATA/nFFS pin, the RFM12B doesn't.
- The RFM12 can only sync on the hex "2DD4" 2-byte pattern, the RFM12B can sync on any value for the second byte, not just D4.
- The B version can be set to a single sync byte.
- The "PLL setting command" (CCxx), which only exists on the RFM12B.
- Different values for RX sensitivity and TX output power.
- A slightly different formula to calculate the time of the Wake-Up Timer.

Bron