

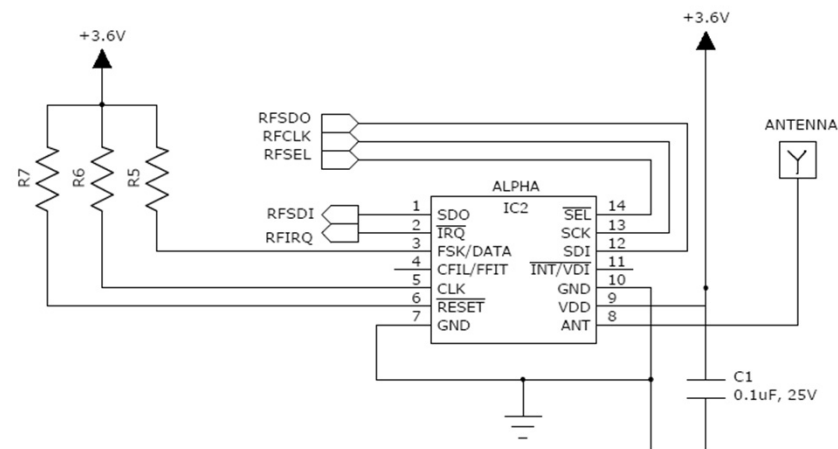
Mestrado Integrado em Engenharia de Comunicações

Transceiver RF
RFM12B

Microprocessadores I
2º Ano – A15

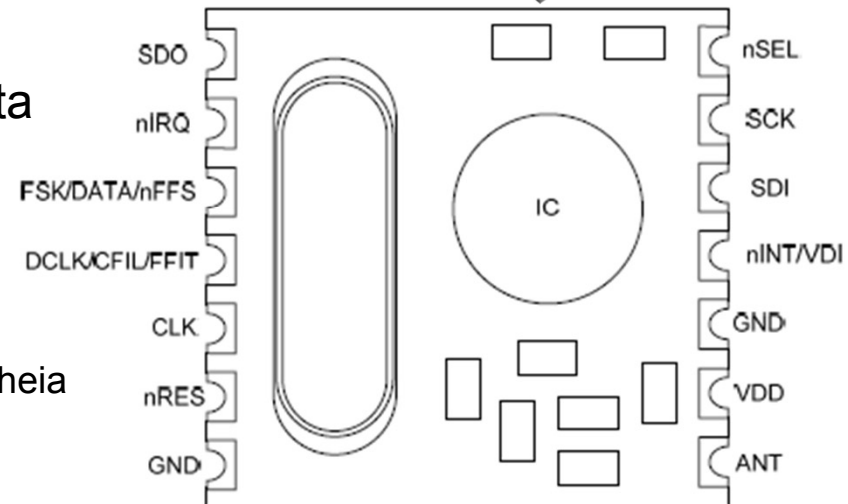
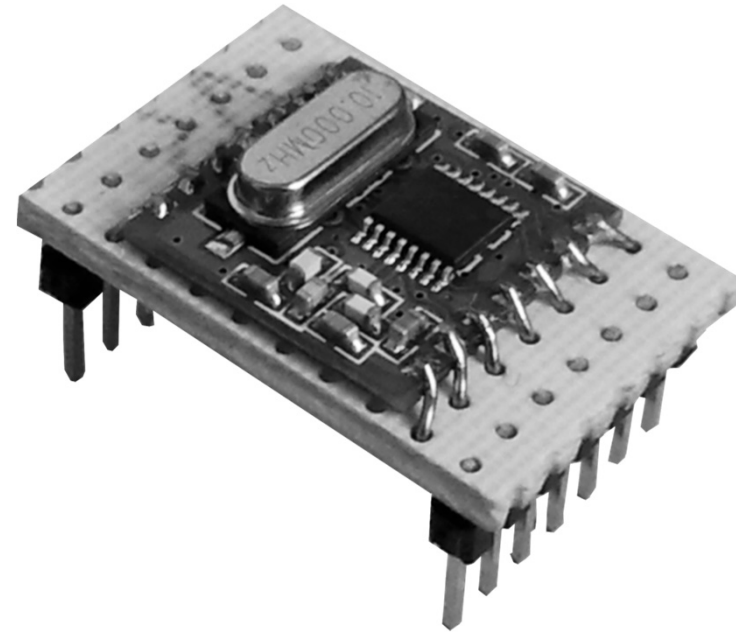
Alpha-TRX-12B

- *Transceiver RF*
 - Baixo custo
 - Baixo consumo
 - Frequência programável
 - Interface SPI
 - FIFO de dados RX
 - Indicação de força do sinal
 - Interrupção
 - Gerador de sinal de relógio
 - Disponível para Arduino
 - Interface “trabalhoso”
 - Documentação escassa
 - Fabricante original Hope RF (na minha opinião!)



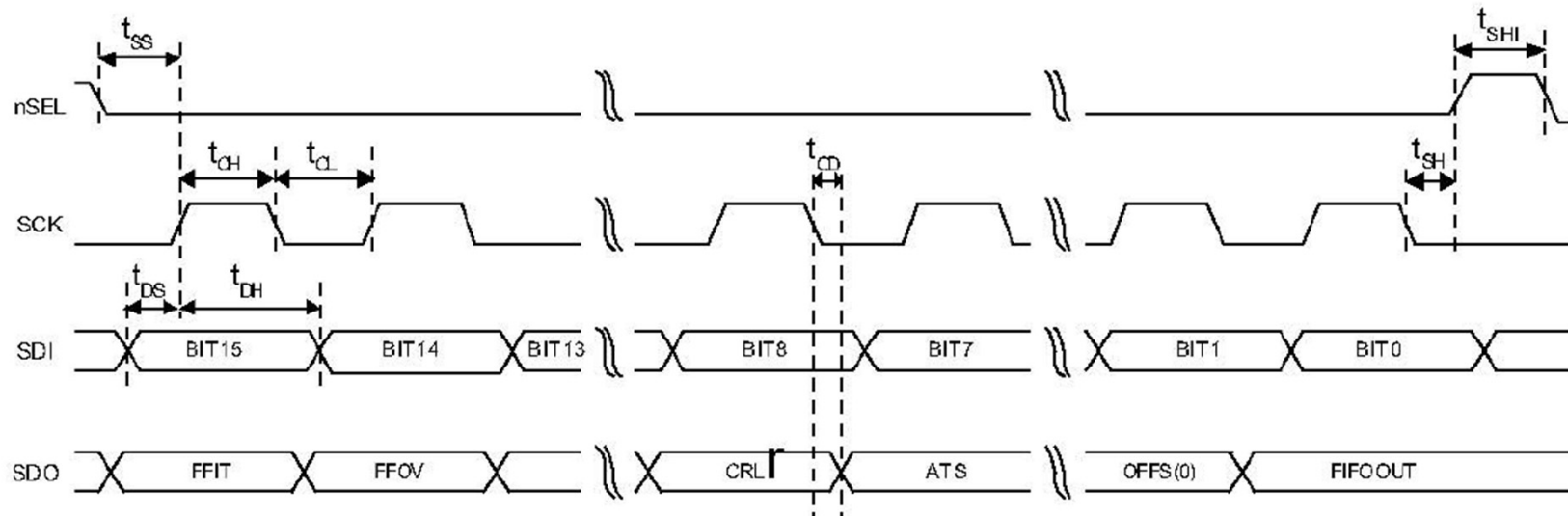
Alpha-TRX-12

- Ligações:
 - O módulo suporta 5V, pelo que pode ser alimentado pelo Kit8051USB
 - Os pinos nRES e FSK devem ser ligados a 5 volt através de uma resistência de 10Kohm
 - O *reset* ao módulo é feito colocando o pino nRES a “0”
 - A antena pode ser um fio normal com o comprimento de $\lambda/4 \approx 17,3\text{cm}$
 - A ligação ao microcontrolador é feita pelos pinos:
 - SDI – Serial Data Input do RFM12B
 - SCK – Serial Clock (gerado pelo 8051)
 - nSEL – Chip Select do RFM12B
 - nIRQ – Sinal que indica FIFO receção cheia
 - SDO – Serial Data Output do RFM12B



SPI - Alpha-TRX-12

- O interface é SPI:



- Para comunicar com o *transceiver* RF a linha nSEL tem de ser colocada a zero;
- A comunicação é bidirecional (16-bit)
- Para enviar um "1": SCK=0; SDI=1; delay (t_{DS}); SCK=1; delay (t_{CH})
- Para enviar um "0": SCK=0; SDI=0; delay (t_{DS}); SCK=1; delay (t_{CH})
- Para ler um bit: SCK=0; delay (t_{CL}); SCK=1; carry=SDO; delay (t_{DH})

SPI “nightmare”

- No código faz sentido definir com nomes adequados os pinos de E/S do 8051 que estão ligados aos pinos do *transceiver* RF
- Não esquecer que os pinos de E/S devem ser inicializados apenas uma vez no início do programa

```
SDI    EQU    P2.0    // saída - Serial Data Input do RF12B
SDO    EQU    P2.1    // entrada - Serial Data Output do RF12B
SCK    EQU    P2.2    // saída - Sinal relógio para o RF12B
NSEL   EQU    P2.3    // saída - Sinal de seleção para o RF12B
NIRQ   EQU    P2.4    // entrada - Sinal que avisa recepção
```

RF12_WRITE0:		RF12_WRITE1:	
CLR	SCK	CLR	SCK
CLR	SDI	SETB	SDI
NOP		NOP	
NOP		NOP	
NOP		NOP	
SETB	SCK	SETB	SCK
RET		RET	

- *Como fazer para enviar um comando de 16-bit ao transceiver RF?*

SPI “nightmare”

- No código utilizei os registos A e B para armazenar o MSB e o LSB da palavra de 16-bit a enviar via SPI
- O registo R3 é o contador de bits
- O bit mais significativo do MSB (Acumulador) é enviado primeiro
- O ciclo é repetido 2 vezes, valor de R2, sendo que na segunda iteração é transmitido o LSB
- Como esta rotina vai ser invocada muitas vezes no código e de modo a facilitar o esforço de programação optei por definir uma macro:

```
WRITE_RF MACRO FIRST,SECOND
    MOV A,#FIRST
    MOV B,#SECOND
    CALL RF12_WRITECMD
ENDM

WRITE_RF 82H,0D9H
WRITE_RF 0CAH,83H
```

Equivalente a:

```
MOV A,#82H
MOV B,#0D9H
CALL RF12_WRITECMD
```

e menos propenso a erros!

```
; A é MSB e B é LSB
RF12_WRITECMD:
    MOV     R2,#2
    MOV     R3,#8
    CLR     SCK
    CLR     NSEL
RFWC_LOOP:
    JB      ACC.7,RFWC_W1
    CALL    RF12_WRITE0
    JMP     RFWC_TEST
RFWC_W1:
    CALL    RF12_WRITE1
RFWC_TEST:
    RL      A
    DJNZ    R3,RFWC_LOOP
    MOV     R3,#8
    MOV     A,B
    DJNZ    R2,RFWC_LOOP
    CLR     SCK
    SETB    NSEL
    RET
```

RFM12B – Enviar dados

- Para enviar um byte de um *transceiver* RF para outro usa-se um comando de 16-bit
- O byte mais significativo é o 0xB8 e o menos significativo é o byte a enviar
- Apenas é necessário garantir que o *transceiver* RF está pronto a receber o byte

- *Ativar chip select*
- *Enviar pulsos de relógio com SDI a 0 (query ao status) até o transceiver RF colocar SDO a alto – significa que está livre*
- *Desativar chip select*
- *Invocar a rotina RF12_WRITECMD após colocar em A o valor 0xB8 e em B o byte a enviar.*

```
; Em B Byte a escrever
RF12_WRITEFSK:
    CLR     NSEL
    CLR     SCK
    NOP
    NOP
    CLR     SDI
    SETB    SCK
    NOP
    NOP
    MOV     C,SDO          ; ler SDO
    SETB    SDI
    SETB    NSEL
    JNC     RF12_WRITEFSK
    MOV     A,#0B8H
    CALL    RF12_WRITECMD
    RET
```

RFM12B – Ler dados

- Para ler um byte enviado de outro transceiver RF usa-se um comando de 16-bit
- O comando é B000h e o transceiver envia o byte em simultâneo com a transmissão do LSB
- Antes, é necessário garantir que o *transceiver* RF tem o byte na FIFO (nIRQ)
- Dependendo do modo de configuração pode ser necessário ler o status (comando 0000h)

- *Ativar chip select*
- *Enviar comando de leitura de estado*
- *Enviar MSB de comando de leitura de FIFO*
- *Durante o envio do LSB do comando de leitura (SDI a 0 antes de SCK a 1) é lido o SDO após a linha SCK ser colocada a 1*
- *Byte lido da FIFO de receção fica no Acumulador*

RF12_RDFIFO:		; LER BYTE FIFO	
CLR	SCK	RDFLOOP2:	
CLR	NSEL	CLR	SCK
CLR	SDI	CLR	SDI
WRITE_RF	00H,00H	NOP	
CLR	NSEL	NOP	
MOV	A,#0B0H	NOP	
MOV	R3,#8	SETB	SCK
RDFLOOP:		NOP	
CLR	SCK	NOP	
RLC	A	NOP	
MOV	SDI,C	MOV	C,SDO
NOP		RLC	A
NOP		DJNZ	R3,RDFLOOP2
NOP		CLR	SCK
SETB	SCK	SETB	NSEL
NOP		RET	
NOP			
NOP			
DJNZ	R3,RDFLOOP		
MOV	R3,#8		
CLR	SCK		

RFM12B – Inicializar

- Faltam apenas os comandos necessários para configurar o módulo
- Encontrei um assistente de configuração que simplifica muito a tarefa:
 - <http://tools.jeelabs.org/rfm12b>

RFM12B Command Calculator

80D8 : Configuration Settings
Band ☒ 433 ☐ 868 ☐ 915 MHz ☒ TX Register
Xtal cap pF ☒ RX FIFO Buffer

8201 : Power Management
☐ Enable Receiver ☐ Enable Crystal Osc
☐ Enable Base Band Block ☐ Enable Low-bat Detector
☐ Enable Transmitter ☐ Enable Wake-Up Timer
☐ Enable Synthesizer ☒ Disable Clock Output Pin

A640 : Frequency Setting
For 433 MHz: $F_c = 430 + F \times 0.0025$ MHz
F = : Center Frequency = 434.0000 MHz

C647 : Data Rate
☐ Enable Prescale (1/8)
Enter a value for R : Data Rate = 4.789 kbps

94A0 : Receiver Control
LNA Gain dBm Pin
RX Bandwidth kHz VDI
DRSSI dBm

C2AC : Data Filter & Clock Recovery
Filter Type Recovery Mode
Quality Threshold Recovery Speed

CA83 : FIFO and Reset Mode
FIFO INT Level Sync on bytes
FIFO Fill Start Reset Sensitivity
☒ FIFO Fill Enabled

CED4 : Synchronization Pattern
Synchronization Byte (HEX) : 0010110111010100

C483 : Automatic Frequency Control
AFC Mode
Offset Register Limit
☒ Enable AFC ☐ Enable High Accuracy (slower)
☐ Strobe ☒ Enable Frequency Offset Register

9820 : TX Control
Frequency Shift Power Out dB
Deviation kHz

CC57 : PLL Settings
Clock rise ☒ Disable Dither in PLL
PLL Band kbps ☐ Enable Phase Detector Delay

E000 : Wake-Up Timer
 $T = M \times 2^R$: R = M = : T = 0 ms

C800 : Low Duty-Cycle
 $DC = (D \times 2 + 1) / M \times 100\%$ ☐ Enable
D = : Duty Cycle = Infinity %

C000 : Low Battery Detect and μ C Clock
Low-Battery Threshold V
Clock Pin Frequency MHz

Other Commands
B000 : RX Read - read 8 bits from the receiver FIFO
B8xx : TX Write - write 8 bits to the transmitter register

INIT_RF12:
SETB SDO ; definir como entrada
SETB NIRQ ; definir como entrada
; valor das saídas
SETB NSEL
SETB SDI
CLR SCK

WRITE_RF 80H,0D8H
WRITE_RF 82H,01H
WRITE_RF 0A6H,40H
WRITE_RF 0C6H,047H ; 71 SEM PRE-ESCALAR
WRITE_RF 94H,0A0H ; dBm
WRITE_RF 0C2H,0ACH
WRITE_RF 0CAH,83H
WRITE_RF 98H,20H
WRITE_RF 0CCH,77H
WRITE_RF 0E0H,00H
WRITE_RF 0C8H,00H
WRITE_RF 0C0H,00H
WRITE_RF 0CEH,0D4H
WRITE_RF 0C4H,83H
RET

RFM12B – Transmissão

- Antes de transmitir ligar o módulo de transmissão
- Enviar 2 bytes preâmbulo 0xAA
- Enviar 2 bytes de *sync* (endereço) 0x2D 0xD4 (pode ser alterado por comando 0xCEXXh)
- Enviar bytes
- Enviar 2 bytes de preâmbulo 0xAA
- Desligar módulo de transmissão

```
MOV    A,#82H
MOV    B,#29H
CALL   RF12_WRITECMD
NOP
NOP
NOP
NOP
NOP
NOP
MOV    A,#82H
MOV    B,#39H
CALL   RF12_WRITECMD
NOP
NOP
NOP
```


```
MOV    B,#0AAH
CALL   RF12_WRITEFSK
MOV    B,#0AAH
CALL   RF12_WRITEFSK
MOV    B,#2DH
CALL   RF12_WRITEFSK
MOV    B,#0D4H
CALL   RF12_WRITEFSK
```

```
MOV    B,R3
CALL   RF12_WRITEFSK ; ENVIAR CHKSUM
MOV    B,#0AAH
CALL   RF12_WRITEFSK
MOV    B,#0AAH
CALL   RF12_WRITEFSK
MOV    A,#82H
MOV    B,#01H
CALL   RF12_WRITECMD ; CLOSE MODULO
```

*Bytes a enviar .
Procedimento igual ao
de enviar o checksum*

RFM12B – Receção

- Antes de receber ligar o módulo de receção
- Configurar tamanho da FIFO para acertar com tamanho da trama (valor de R7)
- Esperar que NIRQ seja nível baixo (FIFO de receção com dados)
- Ler 1 byte da FIFO de receção (repetir N vezes)
- Ler o valor do *checksum*
- Desligar módulo de receção

```
MOV      R7,#0
MAINLOOP:
WRITE_RF 82H,0D9H
WRITE_RF 0CAH,83H
MAINLOOP2:
JB       NIRQ,$
CALL     RF12_RDFIFO
        
CJNE     R7,#RF12TRAMA,MAINLOOP2
MOV      R7,#0
CALL     RF12_RDFIFO    ; CHKSUM ... no final
WRITE_RF 82H,01H
```

Byte recebido está no acumulador.



ESRG

Embedded Systems Research Group

Jorge Cabral

*Membro do ESRG (Centro ALGORITMI)
Universidade do Minho*

Azurém, Guimarães

21 Janeiro, 2012