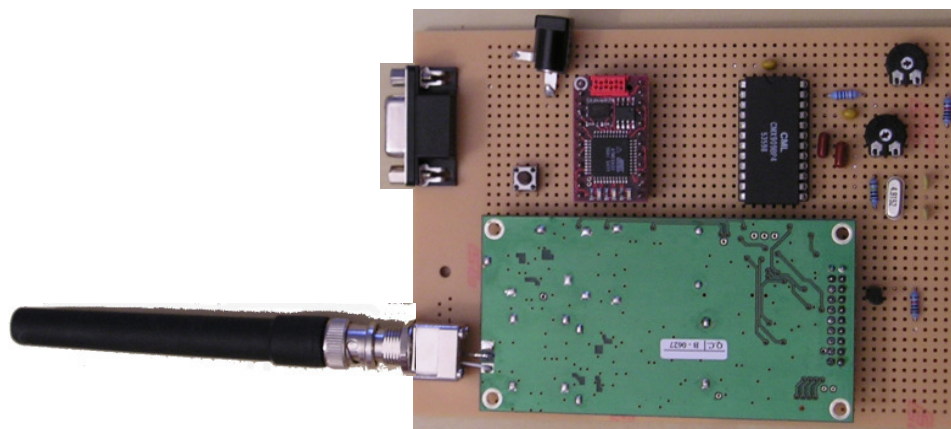
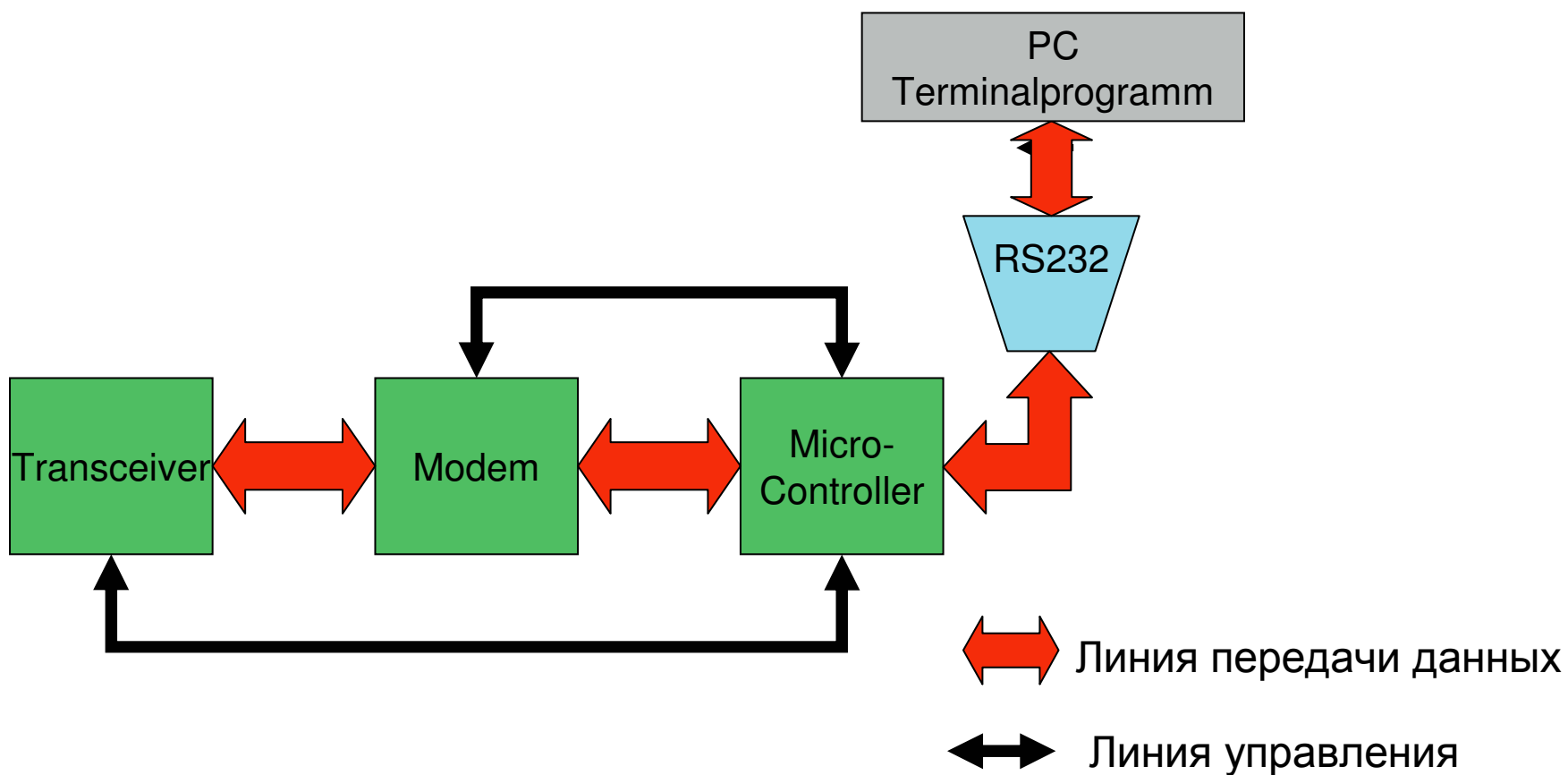


Методические указания по теме "Создание модулей связи космического назначения"



Упрощенная системная схема



Основные задачи компонентов

Микропроцессор выполняет следующие функции:

- Управляет модемом и передатчиком
- Осуществляет передачу данных между модемом и последовательным портом ПК
- Управляет прерываниями последовательного порта, модема и таймера

Модем обеспечивает:

- Модуляцию цифрового сигнала данных поступающих от микроконтроллера
- Демодуляцию аналогового сигнала от передатчика

Передатчик:

- Обеспечивает наложение сигнала от модема на высокую частоту и последующую передачу
- Обеспечивает прием сигнала и демодуляцию на базовую частоту с последующей передачей сигнала на модем

В данном руководстве будут рассмотрены две схемы с применением модемов различной модуляции: CMX869 (GMSK) и FX469J(FFSK)

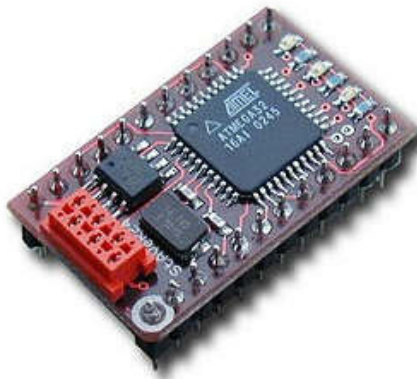


Figure 1: Staver24

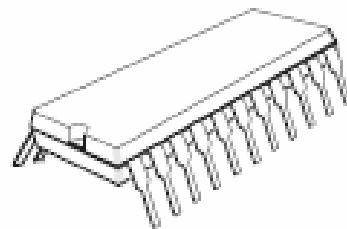


Figure 2: Externes FFSK(GMSK) Modem

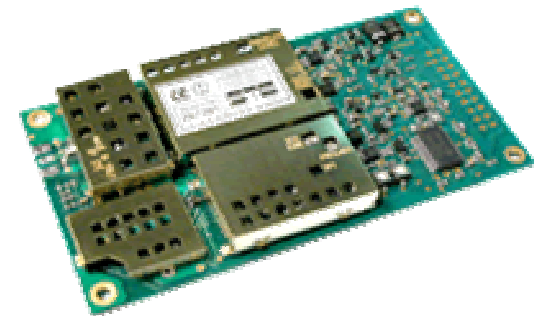
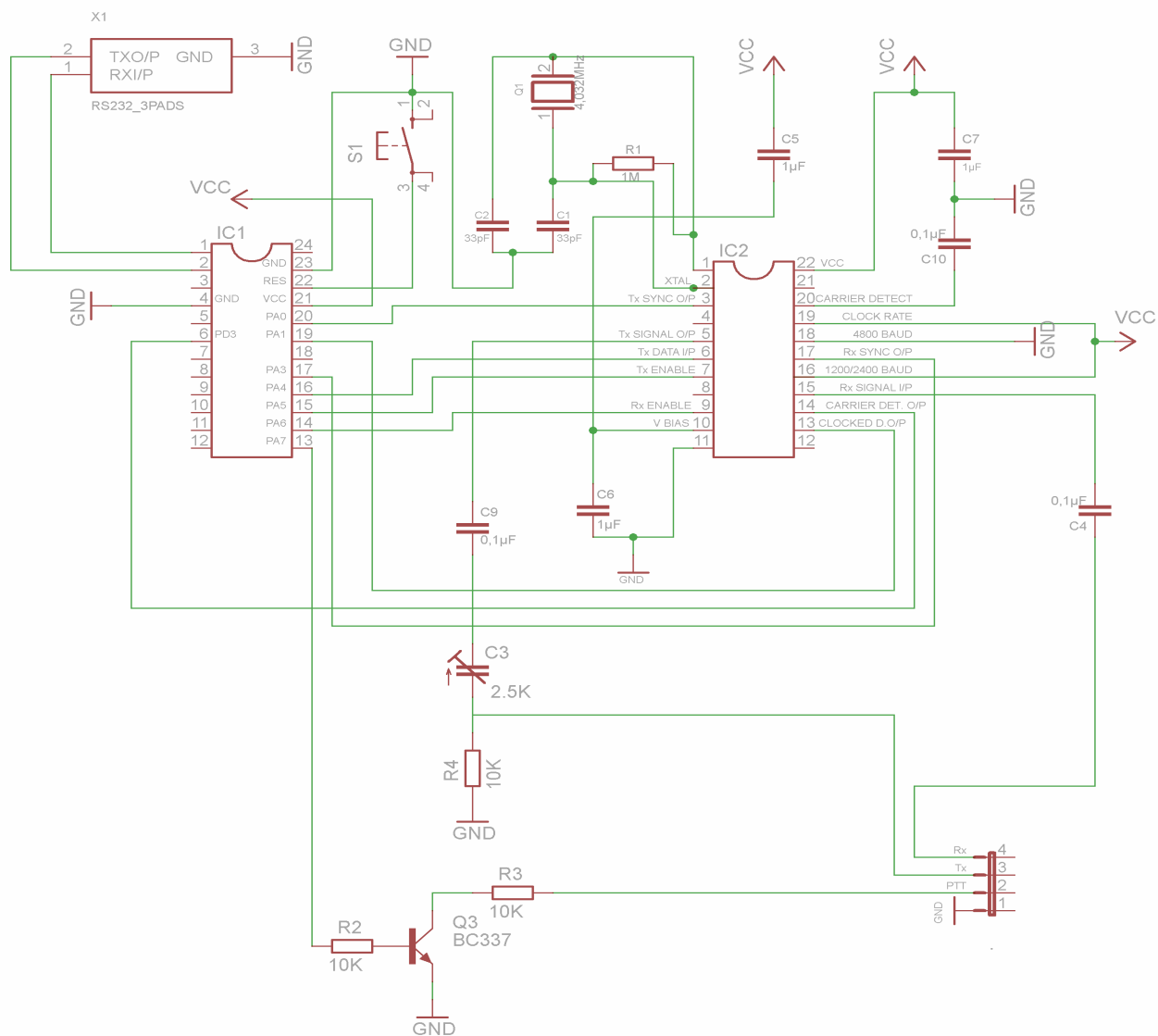
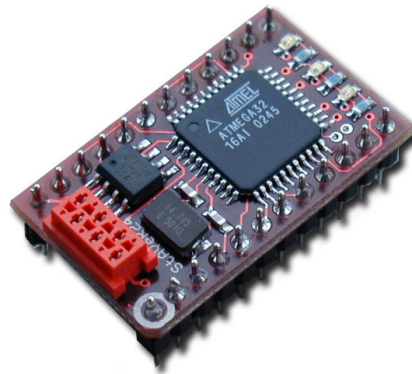


Figure 3: BK77

Модуль с применением модема FX469J (FFSK)



Mikrokontroller Staver



Распределение точек вход/выход микропроцессора STAVeR

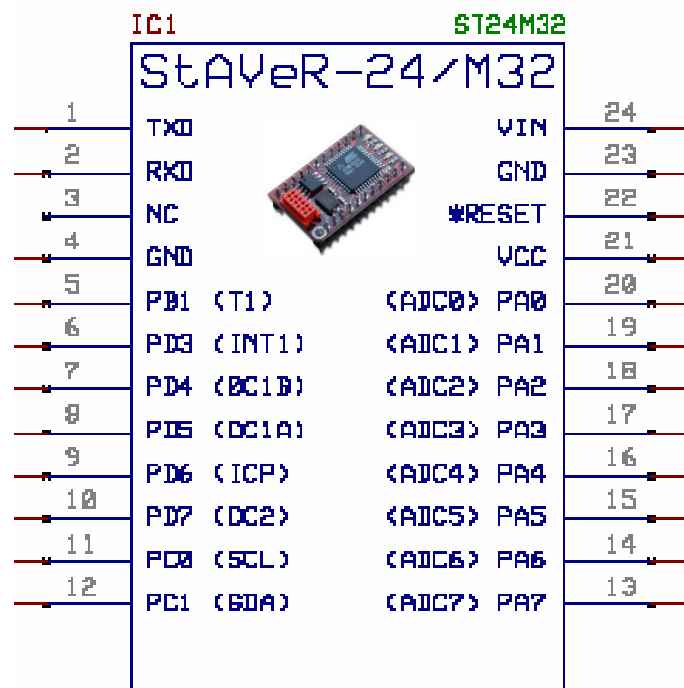
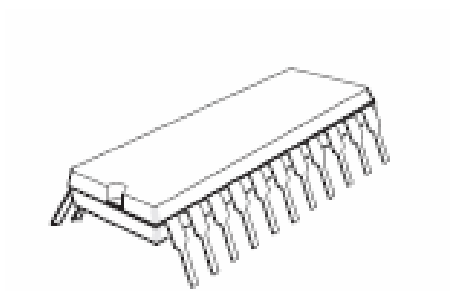


Схема подключения микропроцессора к модему FX469J и передатчику

Pin	Описание	Подсоединение к компоненту	Номер точки компонента	Описание
1	TXD	PC/ RS-232	2	
2	RXD	PC / RS-232	3	
4	GND		GND	
6	Int1	Modem	14	Carrier Detect
7	PA0	BK77	4	RSI
13	PA7	BK77	Через Transistor 8	Tx Enable
14	PC1	Modem	9	Rx Enable
15	PC0	Modem	7	Tx Enable
16	PD5	Modem	6	Tx Data
17	PA2	Modem	17	Rx Sync
19	PA1	Modem	13	ClockedData
20	PD4	Modem	3	Tx sync
21	VCC		4,5 V	
22	Reset		Через переключатель на GND	
23	GND		GND	

Modem FX469J



Распределение точек вход/выход FX469J

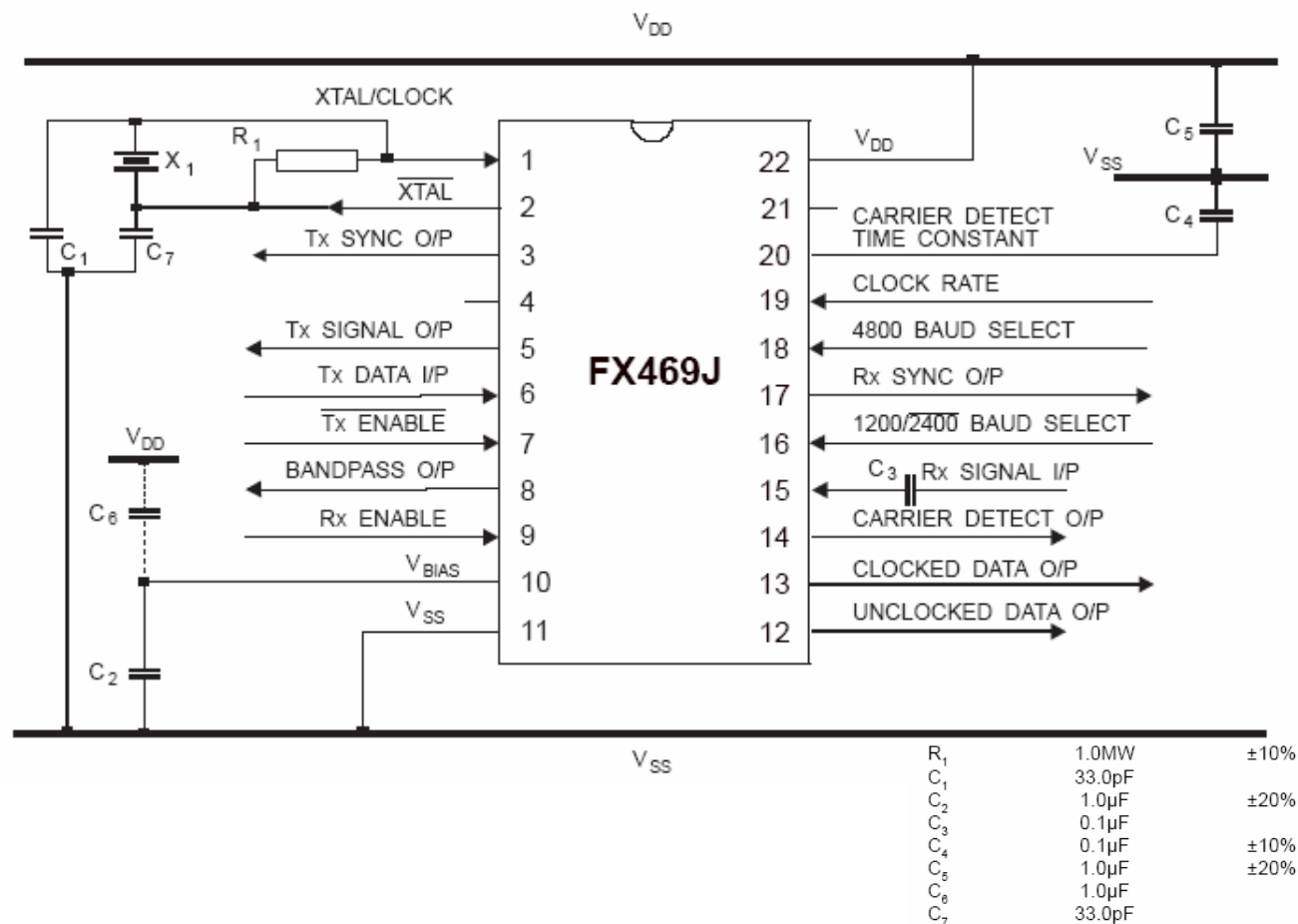


Схема подсоединения модема к микропроцессору и передатчику

Pin	Описание	Подсоединение к компоненту	Номер точки компонента	Описание
1	Xtal/Clock	Quarz		
2	Xtal	Quarz		
3	Tx sync	STAVER	7	PD4
5	Tx Signal O/P	BK77	9	ATX
6	Tx Data	STAVER	8	PD5
7	Tx Enable	STAVER	11	PC0
9	Rx Enable	STAVER	12	PC1
10	V_{bias}		через конденсатор C2 на GND	
11	V_{SS}		GND	
13	Clocked Data	STAVER	19	PA1
14	Carrier Detect	STAVER	6	Int1
15	Rx Signal I/P	BK77	через конденсатор C3 на 5	ARX
16	1200/2400		GND	
17	Rx Sync	STAVER	18	PA2
18	4800		4,5 V	
19	Clock Rate		4,5 V	
20			через конденсатор C4 на VSS	
22	VDD		4,5 V	

HF Voice/Data Transceiver BK77



Распределение точек вход/выход передатчика

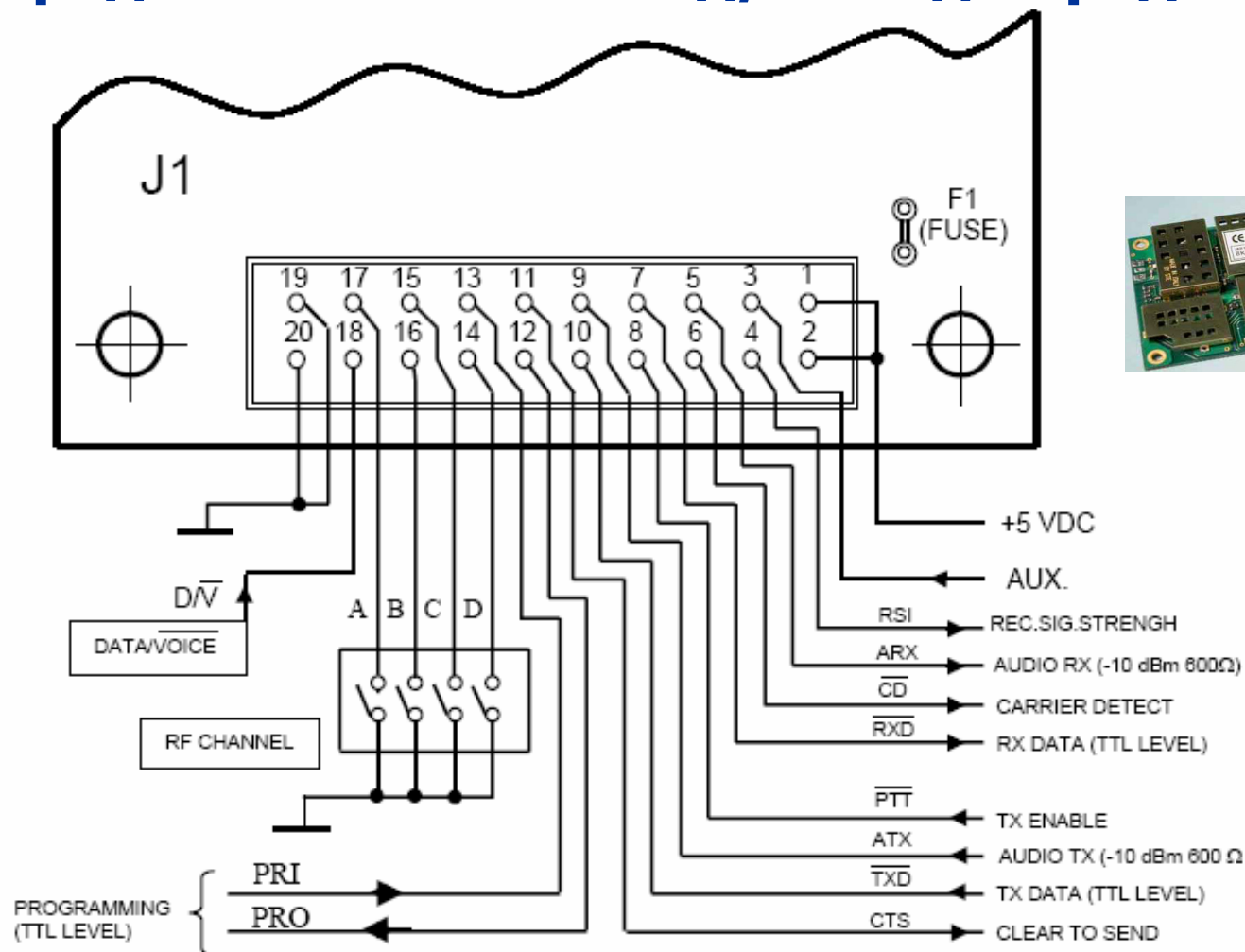






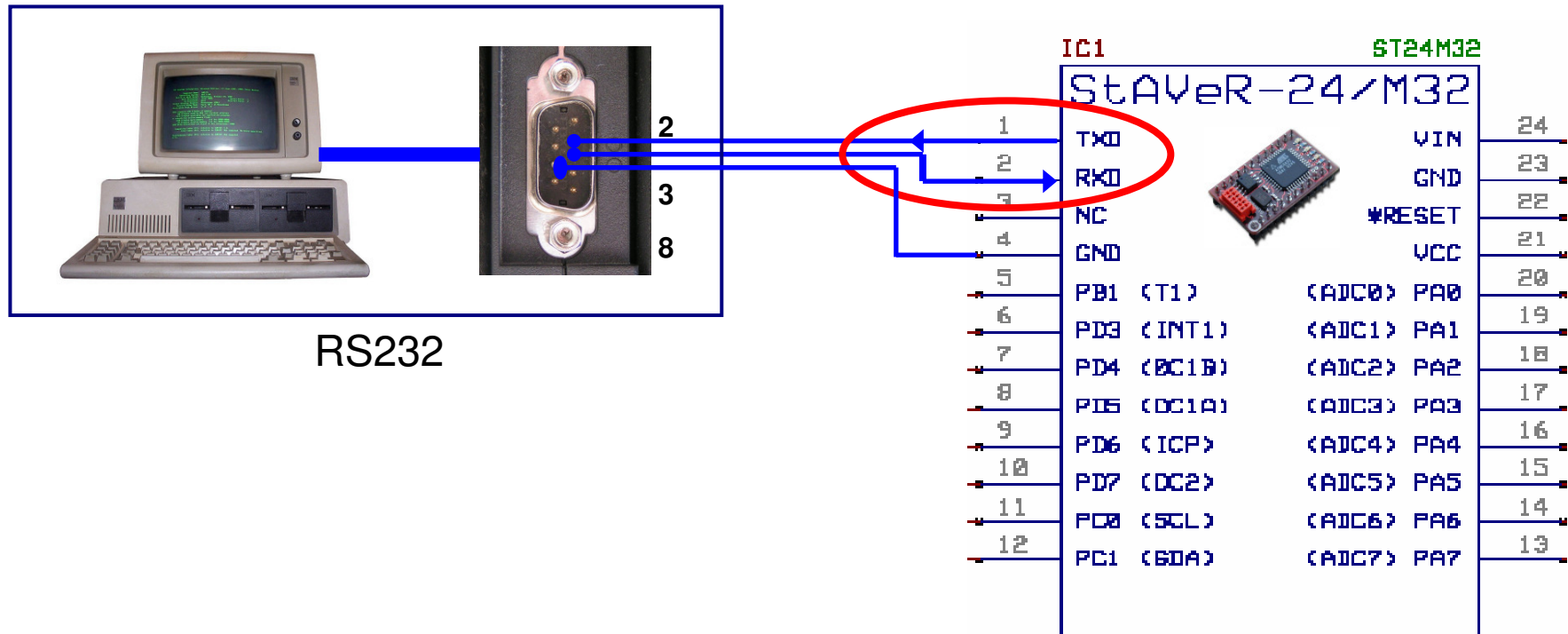
Схема подключения передатчика к модему и микропроцессору

Pin	Описание	Подсоединение к компоненту	Номер точки компонента	Описание
1	VDC		4,5 V	
2	VDC		4,5 V	
4	RSI	STAVER	20	ADC
5	ARX	Modem	через конденсатор C3 на 15	RX Signal I/P
8	TX Enable	STAVER	через транзистор на 13	PA7
9	ATX	Modem	5	TX Signal O/P
14	RF Channel A			
15	RF Channel B			
16	RF Channel C			
17	RF Channel D			
18	Data/Voice		GND	
19	GND		GND	
20	GND		GND	

Детальная схема подключения компонентов коммуникационного

- 1. PC  Microcontroller
- 2. Modem  Microcontroller
- 3. BK77  Microcontroller
- 4. Modem  BK77

PC ↔ Microcontroller

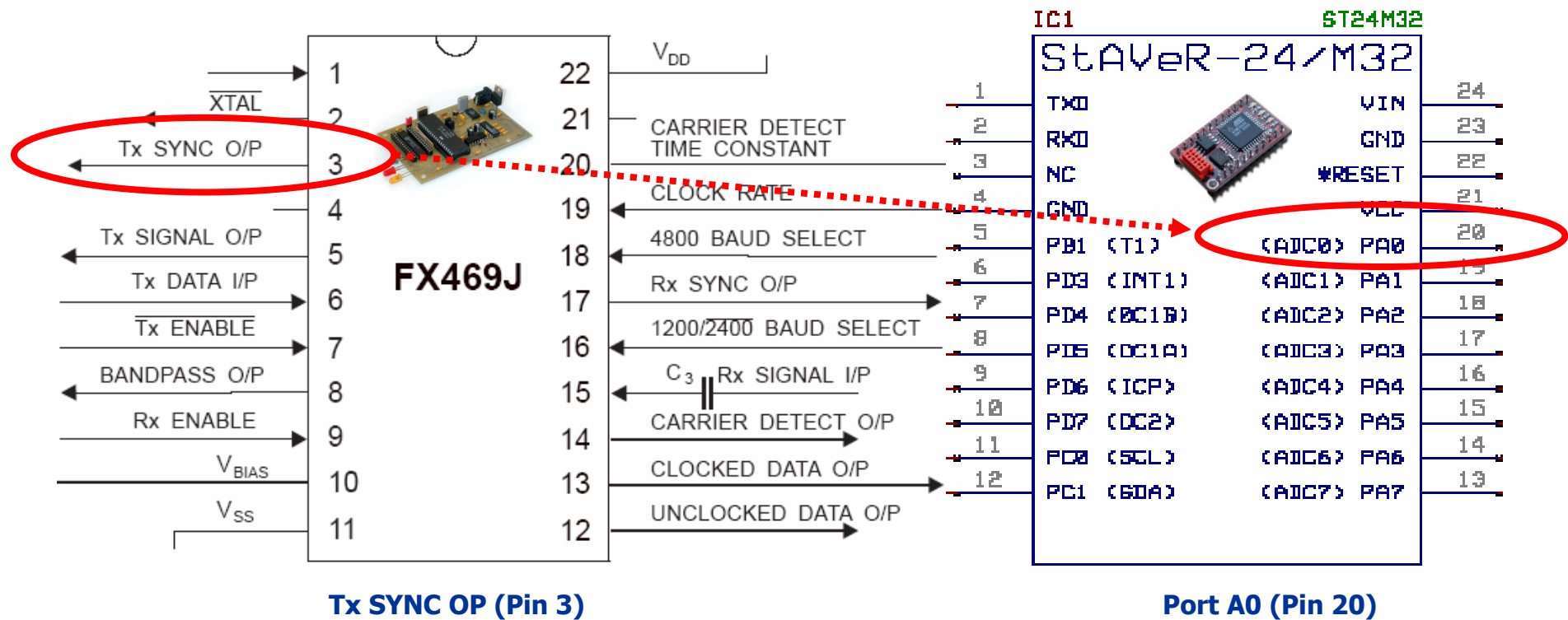


RS232 (Pin 2) - TXD (Pin 1)

RS232 (Pin 3) - RXD (Pin 2)

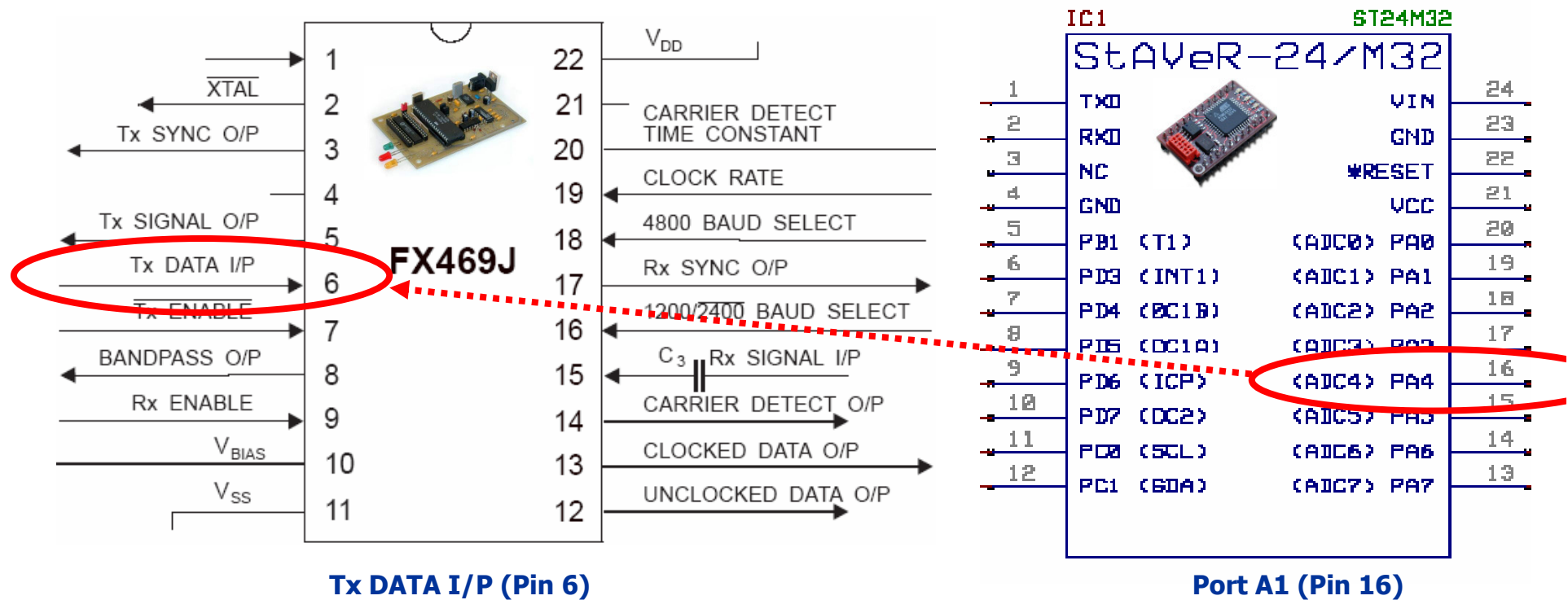
RS232 (Pin 8) - GND (Pin 4)

Modem → Microcontroller



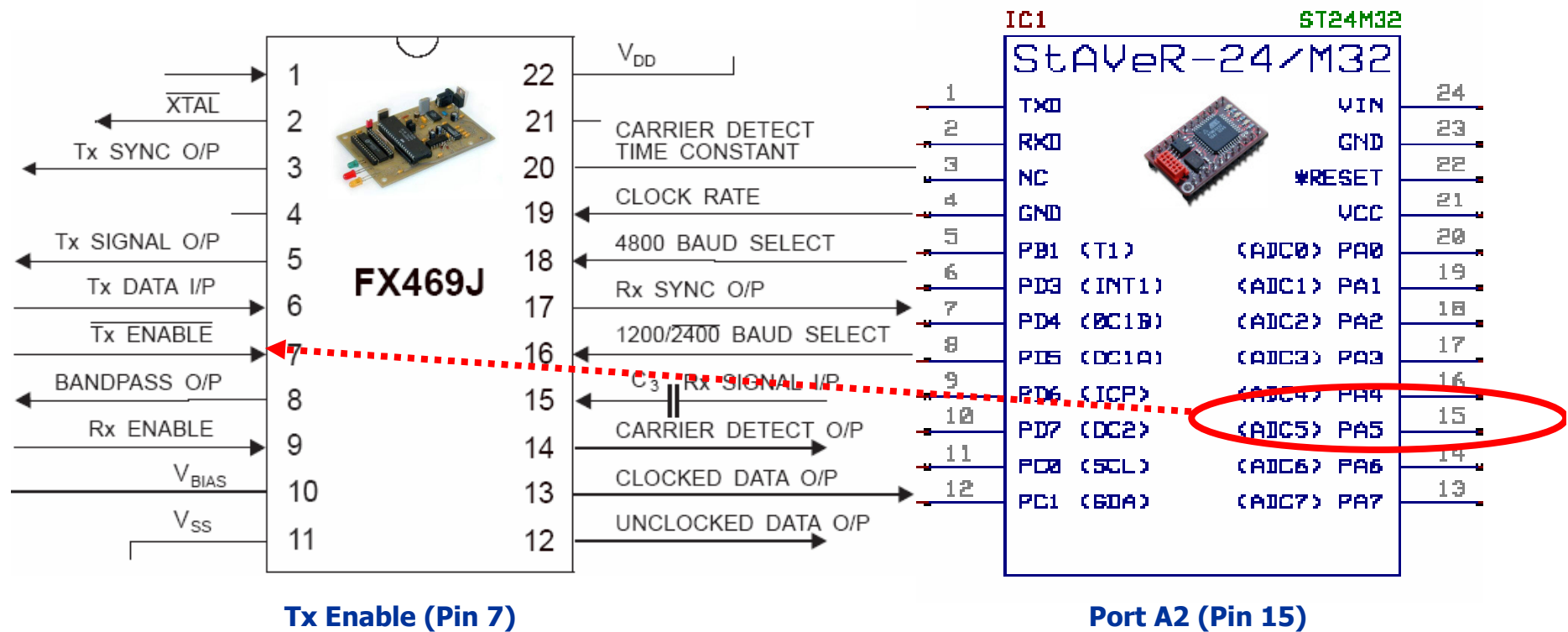
Тактовый сигнал для синхронизации линии передачи данных

Modem ← Microcontroller

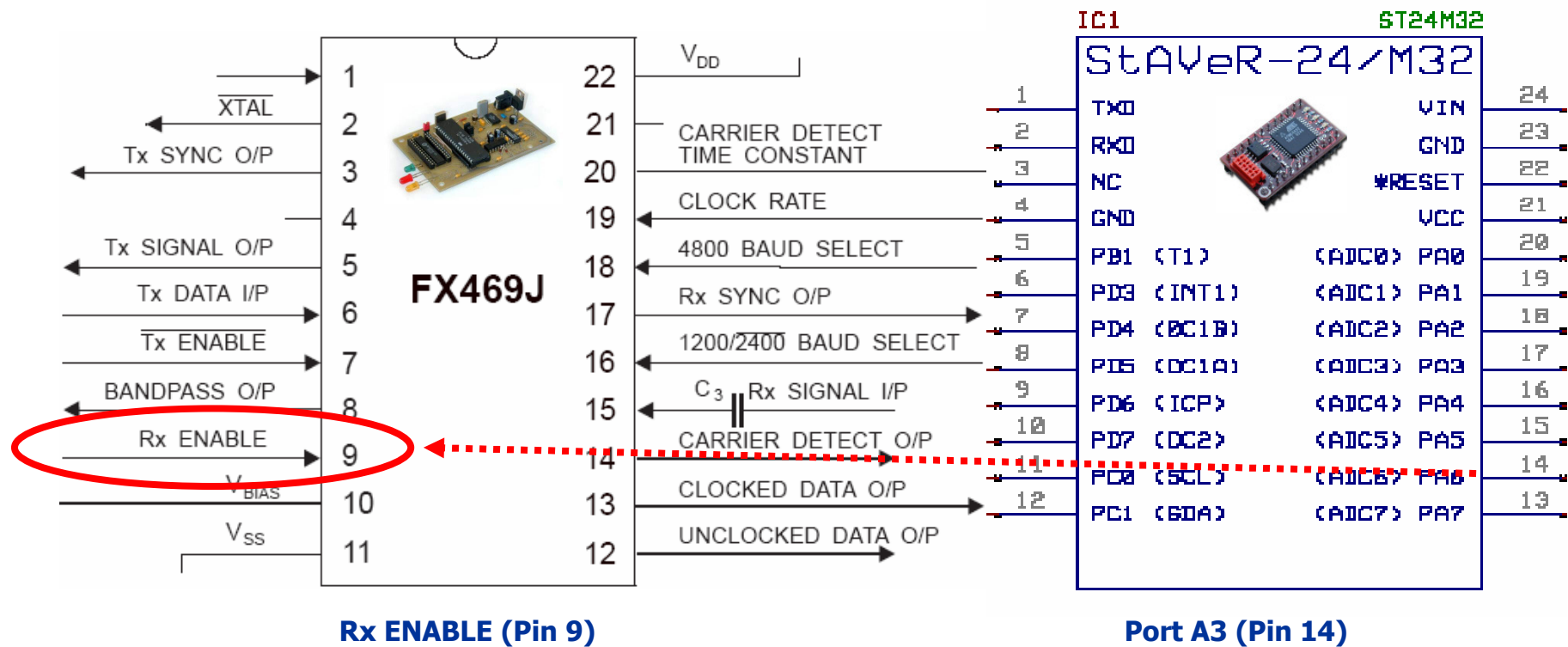


Последовательная передача данных от микроконтроллера к модему в соответствии с тактовой частотой Tx SYNC

Modem ← Microcontroller

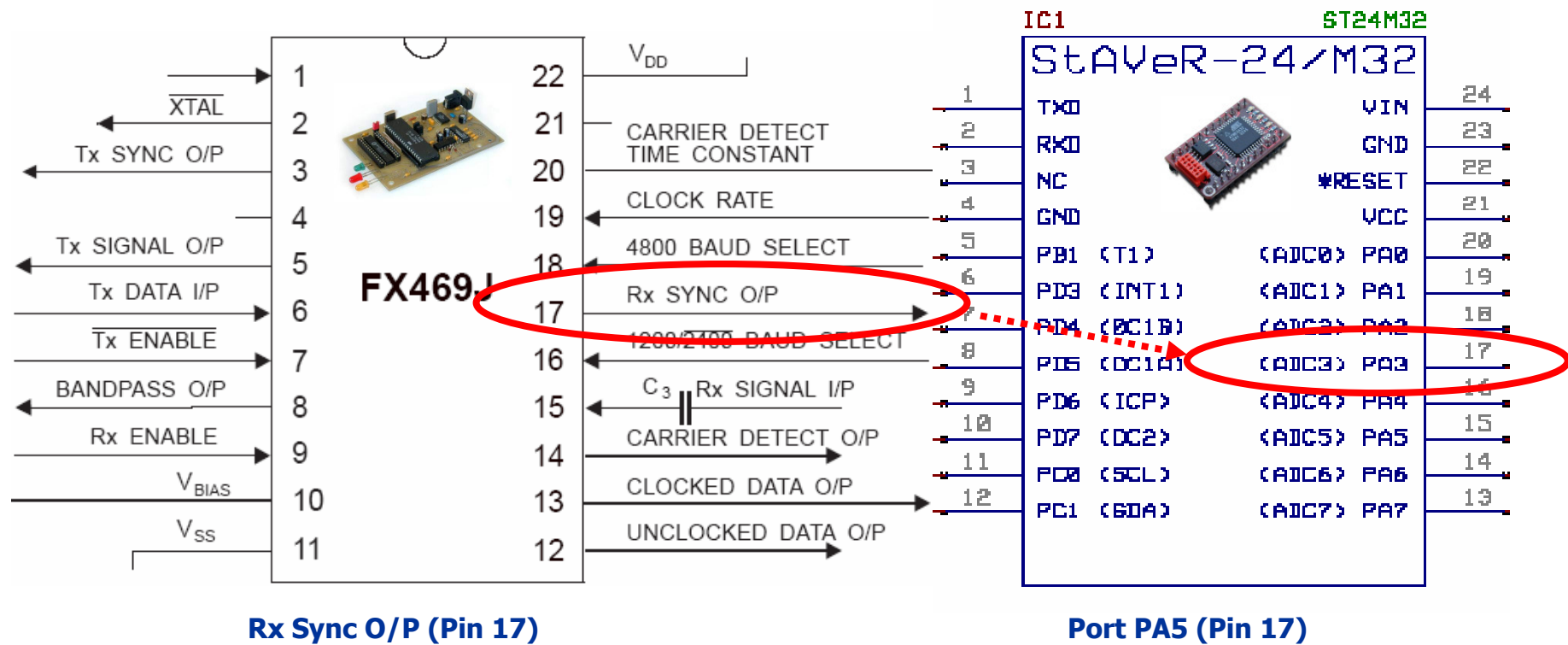


Вкл/выкл режима передачи



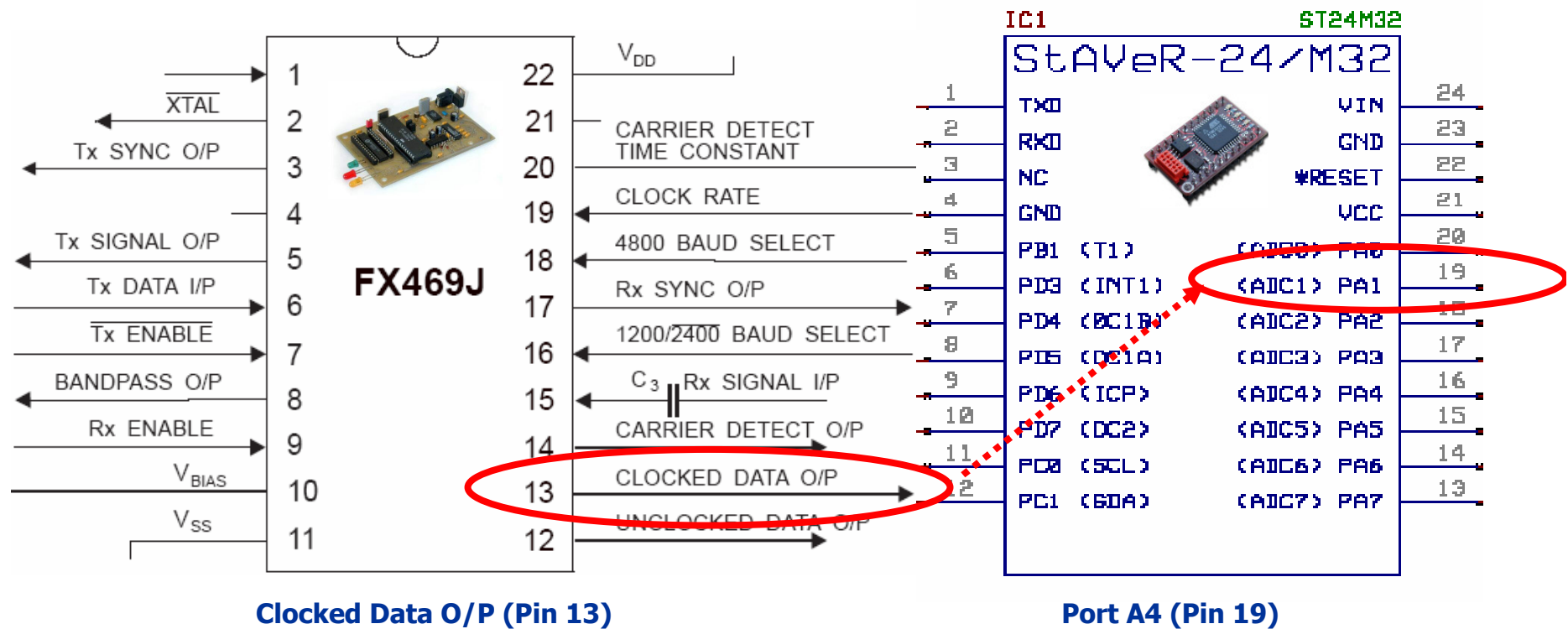
Вкл/выкл режима приема

Modem → Microcontroller



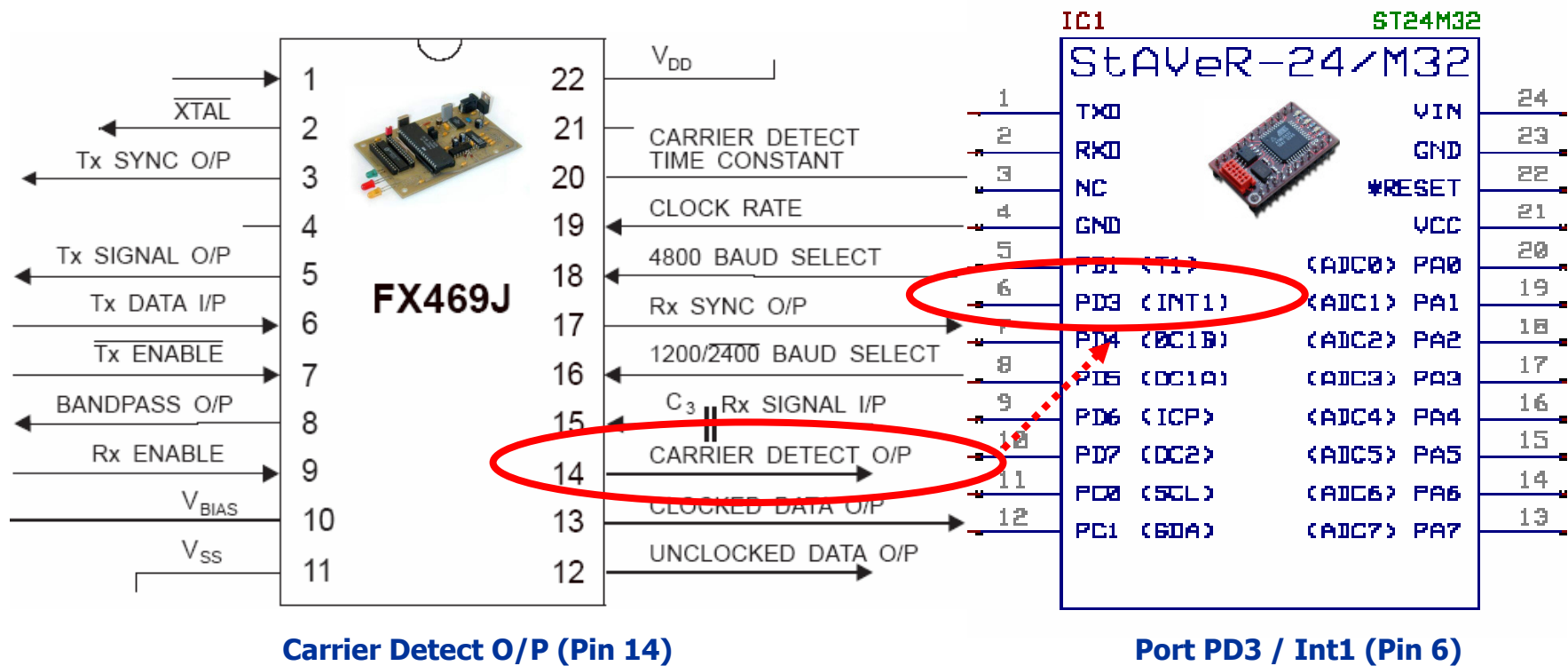
Тактовая частота для синхронизации полученных от модема данных

Modem → Microcontroller



Последовательная передача синхронизированных данных в соответствии с тактовой частотой Rx SYNC

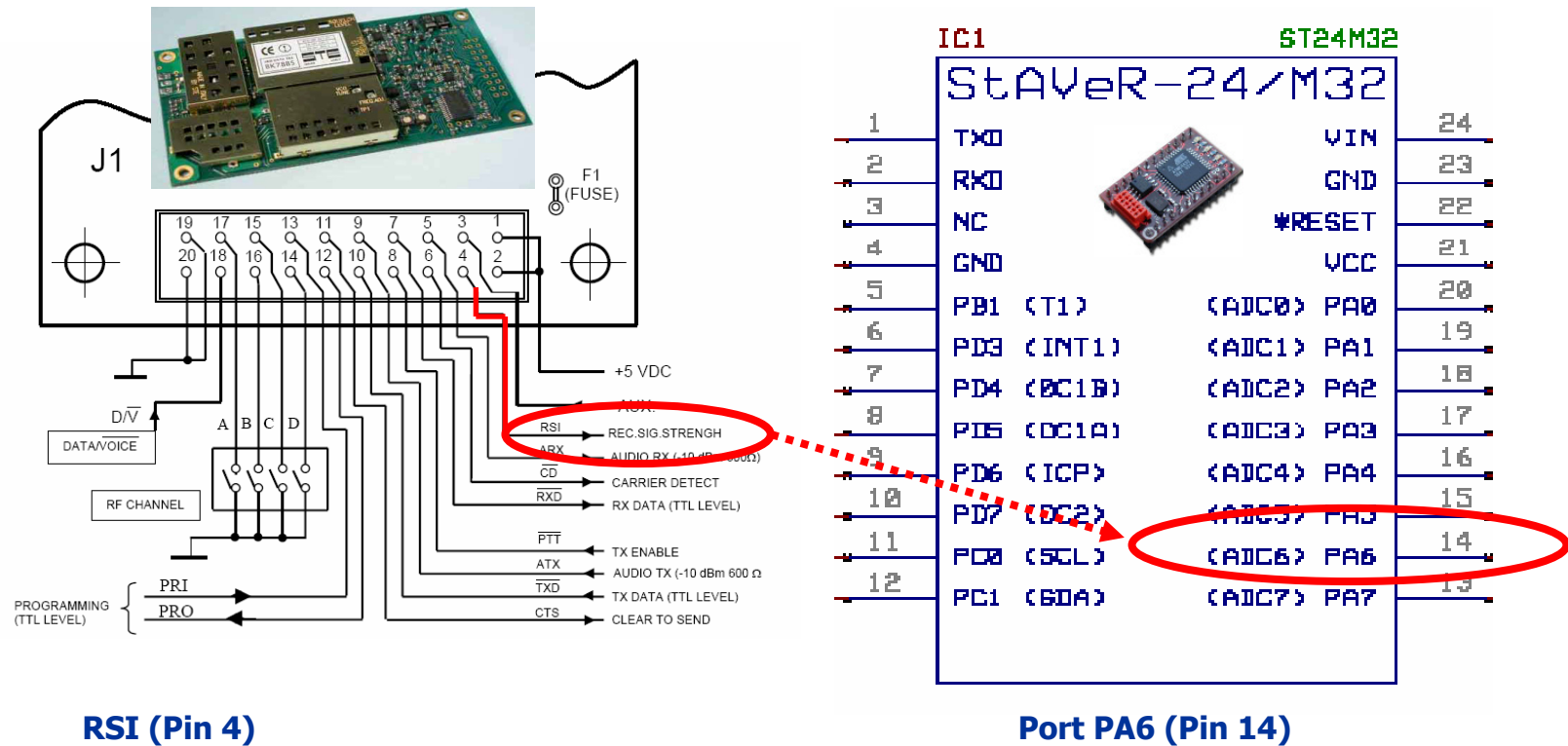
Modem → Microcontroller



Подача сигнала на микроконтроллер, которая вызывает включение прерывателя, переводящего микроконтроллер в режим приема

BK77

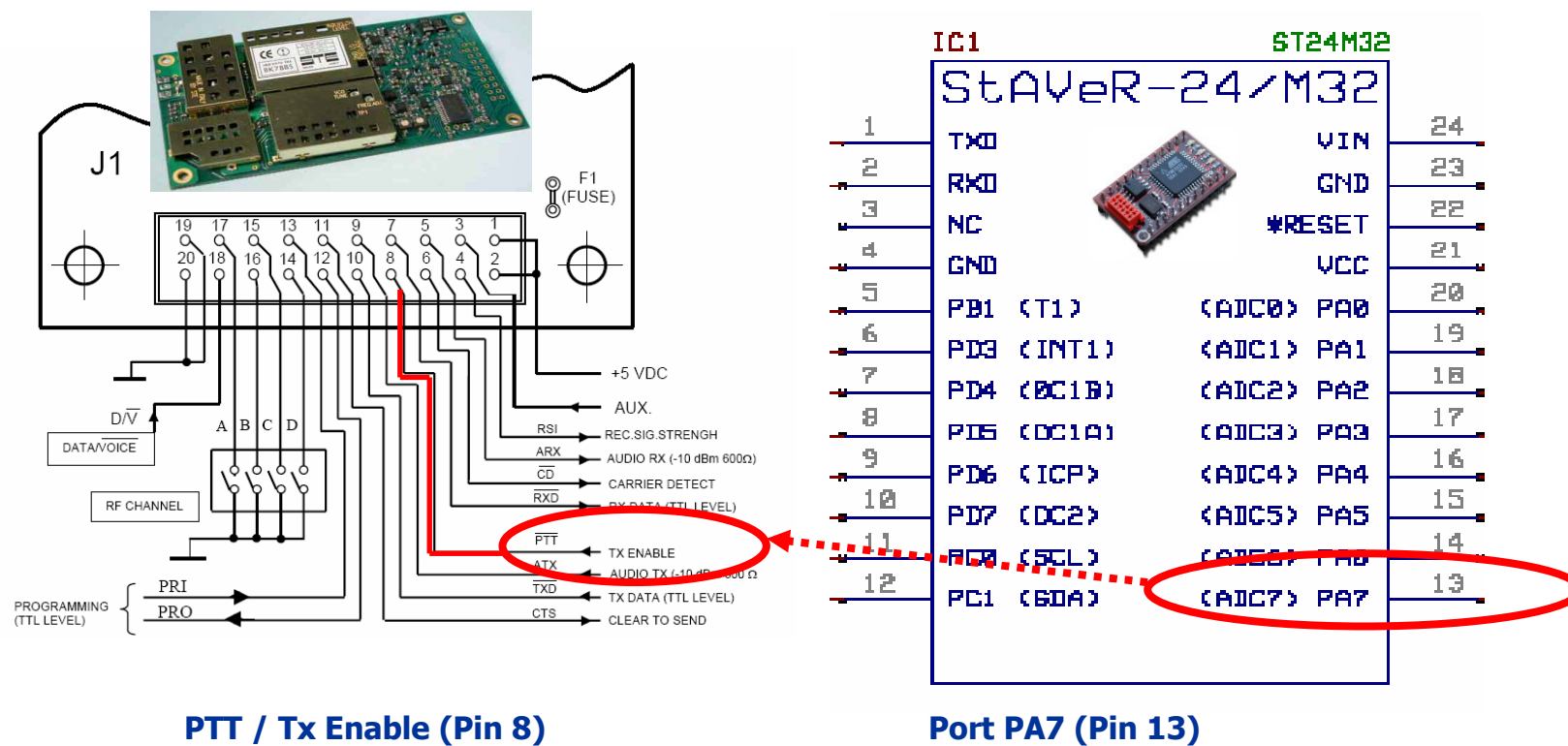
Microcontroller



Сигнал на A/D Конвертер STAVER'а, определяющий мощность входящего сигнала

BK77

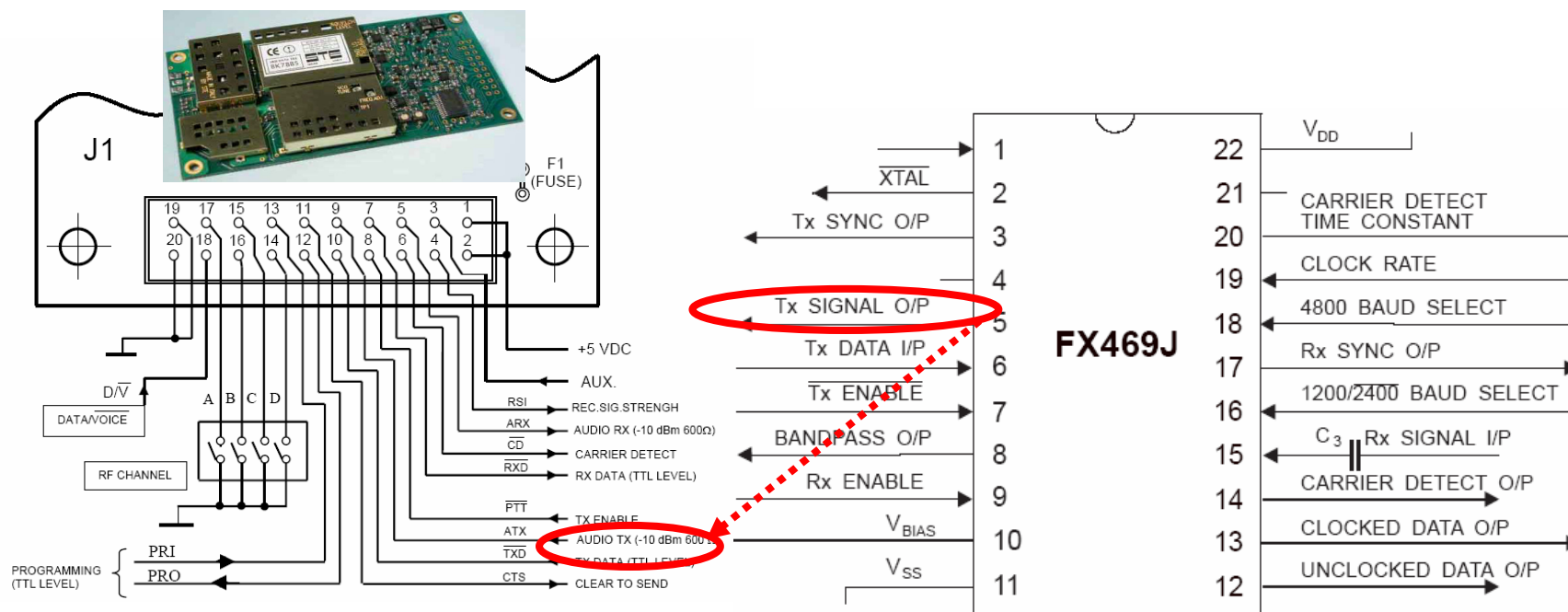
← Microcontroller



Переводит BK77 в режим передачи

BK77

Modem

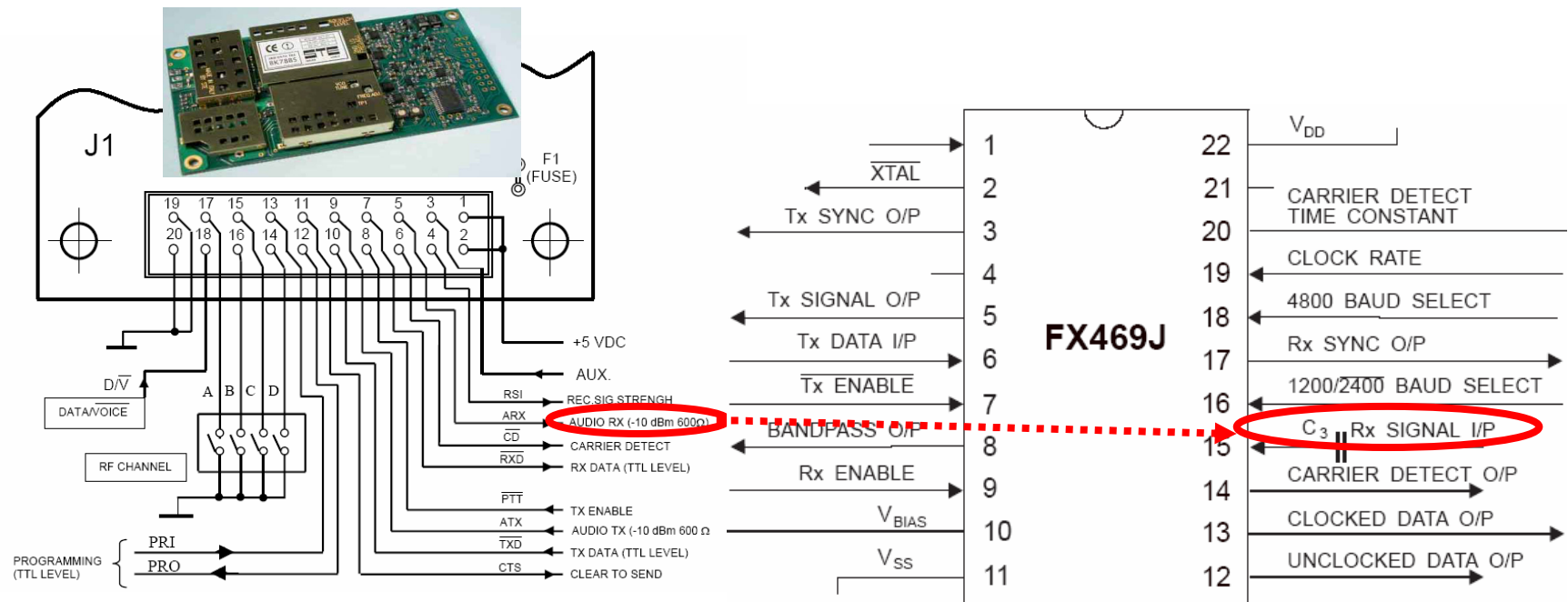


Audio Tx (Pin 9)

Port Tx SIGN (Pin 5)

Передача сигнала от модема на передатчик

BK77 → Modem



Audio Rx (Pin 5)

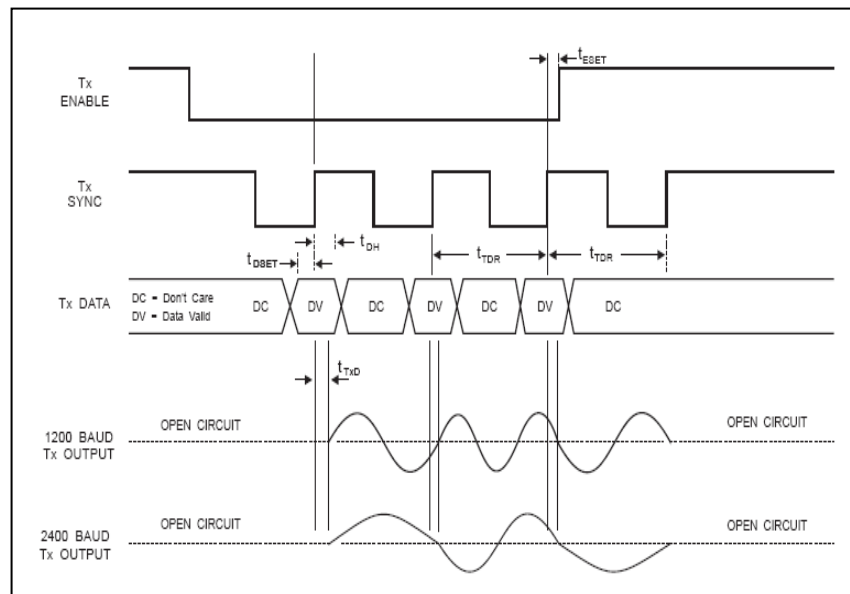
Port Rx SIGN (Pin 15)

Передача сигнала от передатчика на модем

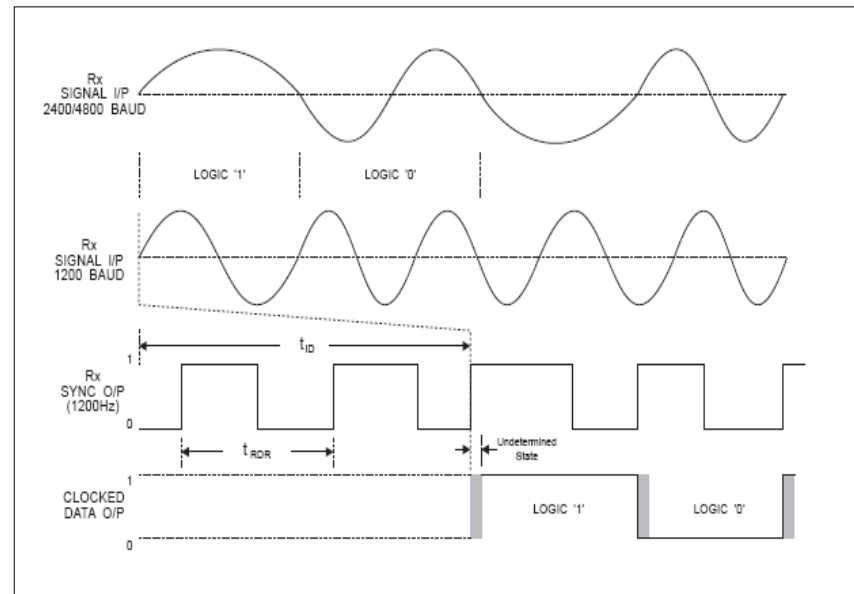
Протокол передачи данных

По-Битовая синхронизация

Данные передаются в виде синхронизированного сигнала, согласно тактовой частоте. Схема приводится ниже:



Transmitter Timing



Rx Timing Diagram

По-Байтовая синхронизация

1. После по-битовой синхронизации считываются 16 бит
2. Считанные данные сравниваются с известным 16-ти битовым шаблоном
3. Если выявлено несовпадение, то 1ый Бит отбрасывается и остальные сдвигаются на одну позицию.
4. Этот процесс повторяется до того момента, пока не достигается полное соответствие считанных битов с образцом или пока не будут считано максимально допустимое количество Бит.

Пример для создания нестандартизированного протокола связи

Преамбула				Информация		
Beginn	AB	Length	CW	Text	CRC 16	End
100 Bit	16 Bit	8 Bit	16 Bit	max. 255 Byte	16 Bit	8 Bit

Beginn: Набор символов для настройки приема

A B: Обозначение начала сообщения

Length : Длина сообщения

CW : Команда

Text : Сообщение

CRC 16: Алгоритм проверки правильности информации

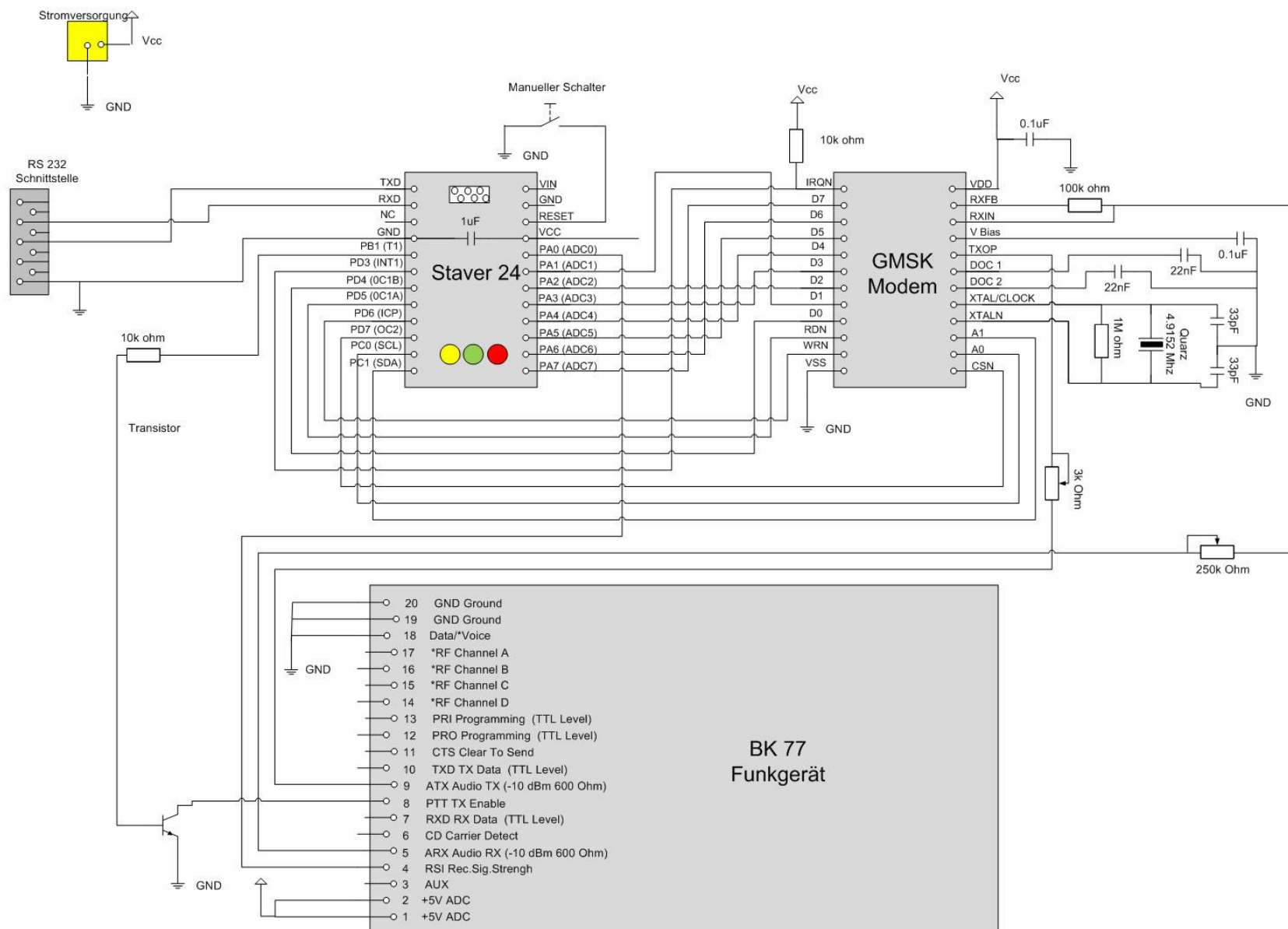
End: Произвольная информация, не подлежащая обработке

Пример программного кода в приложении

Модуль с применением модема CMX869 (GMSK)



Figure 2: Externes GMSK Modem
[www.sander-electronics.de]



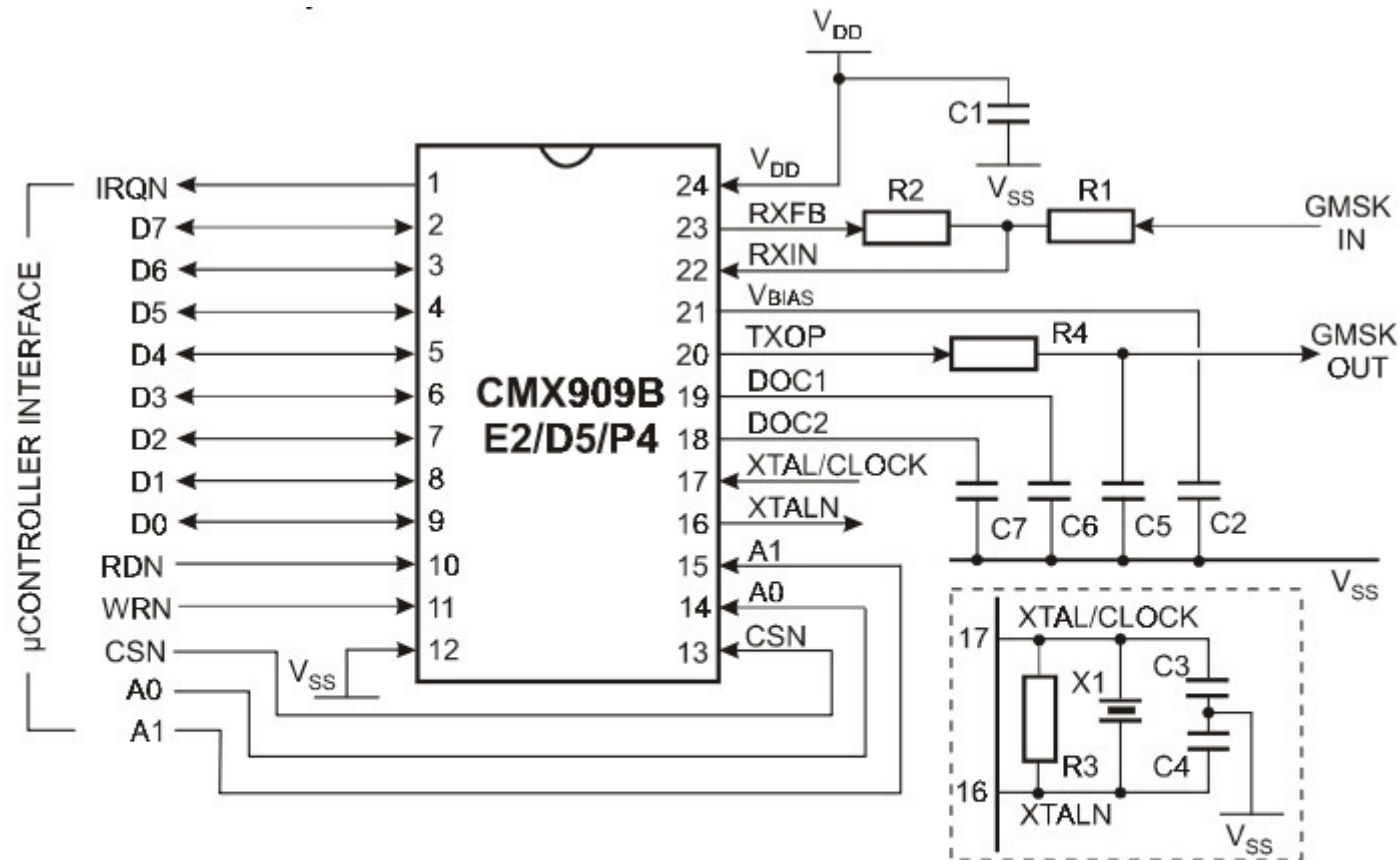


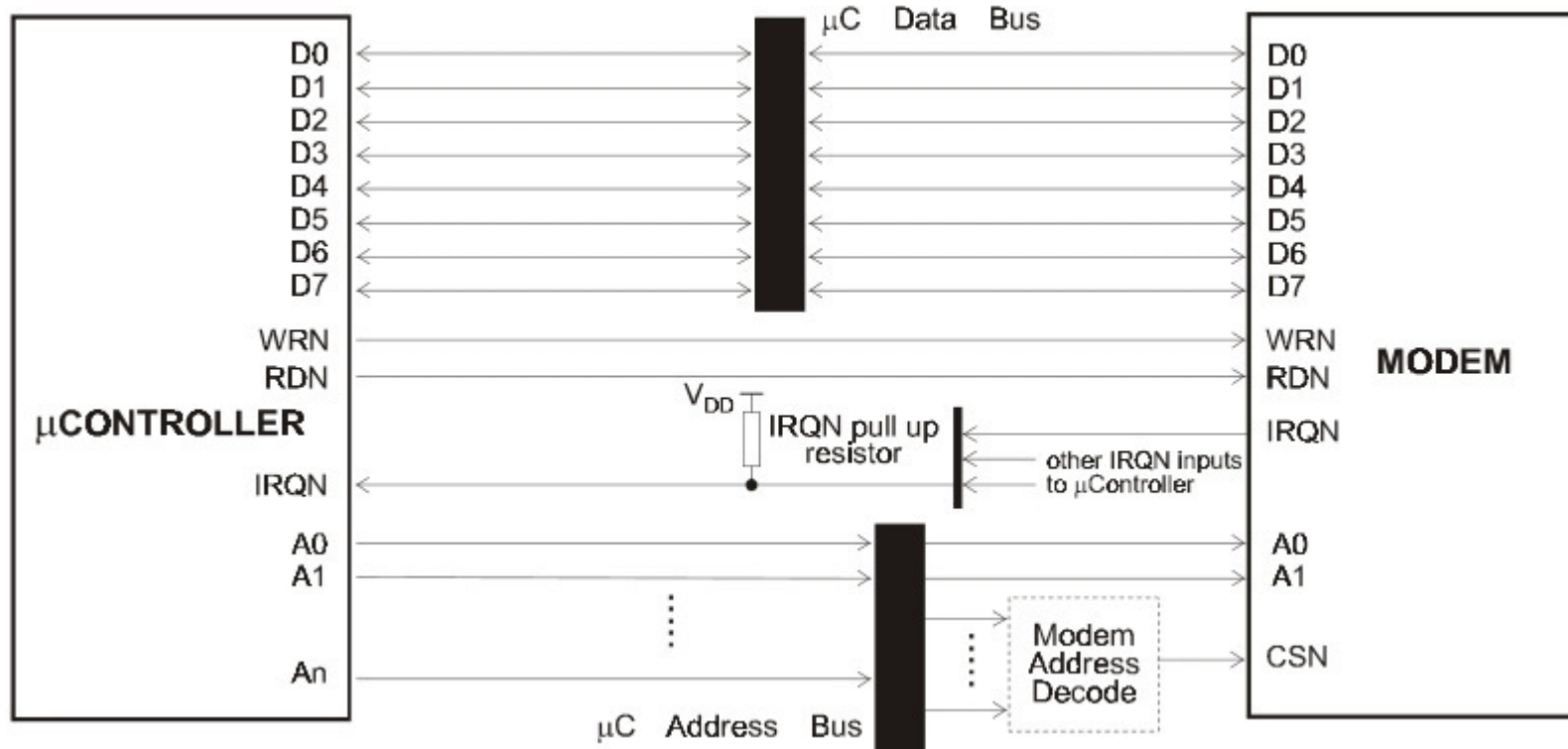
Figure 2 Recommended External Components

R1 See section 1.5.1.10
 R2 100k ohm
 R3 1M ohm *
 R4 See section 1.5.1.12

C1 0.1 μ F
 C2 0.1 μ F
 C3 See section 1.5.4.3 *
 C4 See section 1.5.4.3 *

C5 See section 1.5.1.12
 C6 See note below
 C7 See note below
 X1 See section 1.5.4.3 *

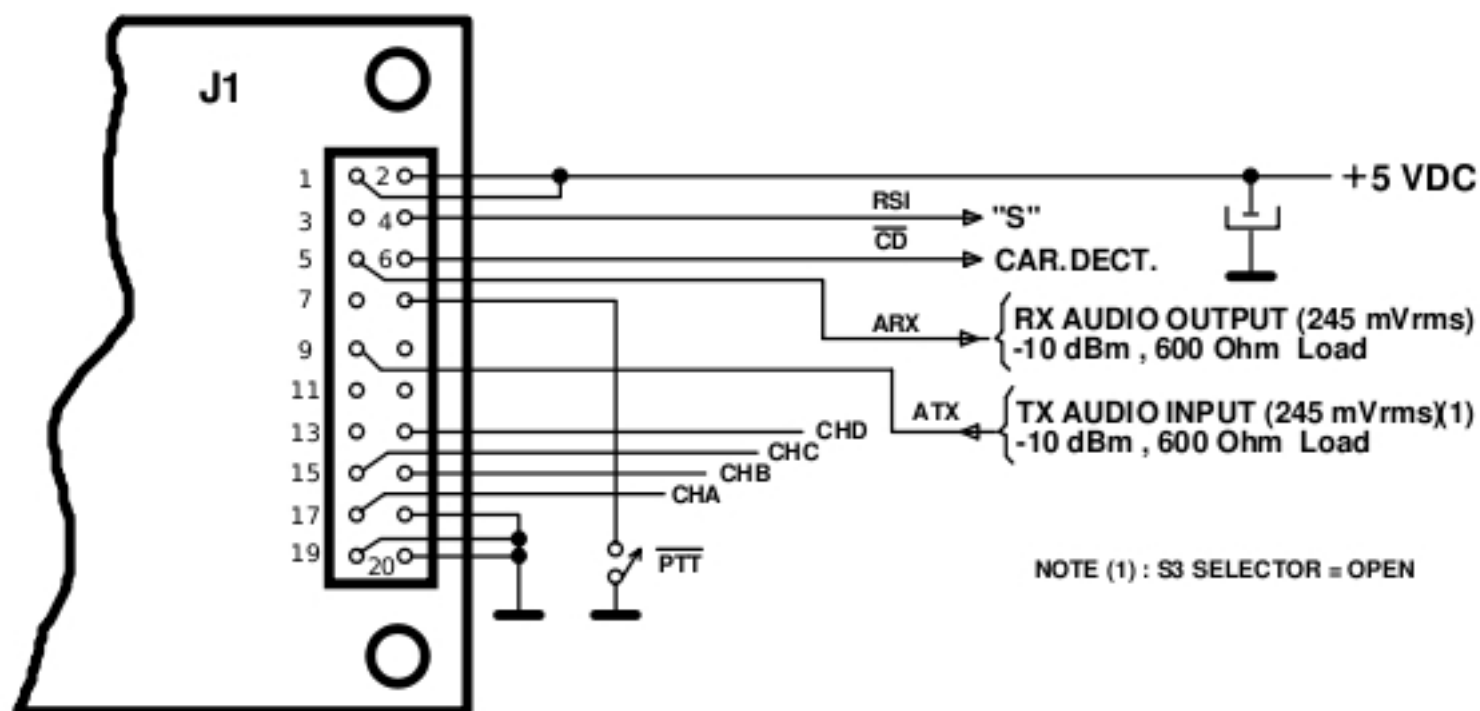
Подключение модема к микроконтроллеру



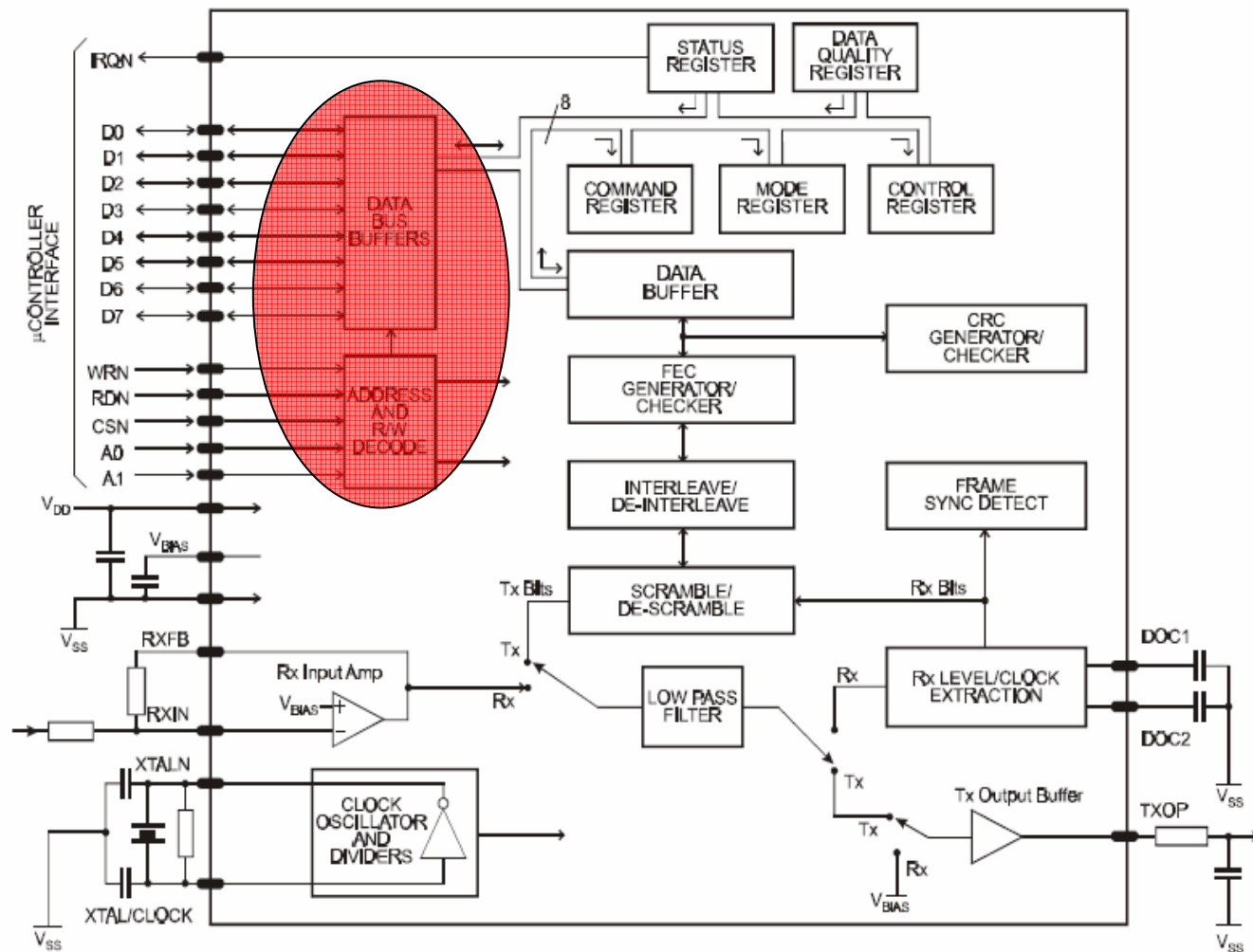
В случае модема CMX909В данные передаются параллельным способом по 8-Бит.

Такая передача данных характеризуется высокой скоростью передачи при большом количестве подключений.

Схема подключения портов BK77



Порты для подключению к микроконтроллеру



Метод ввода/вывода модема

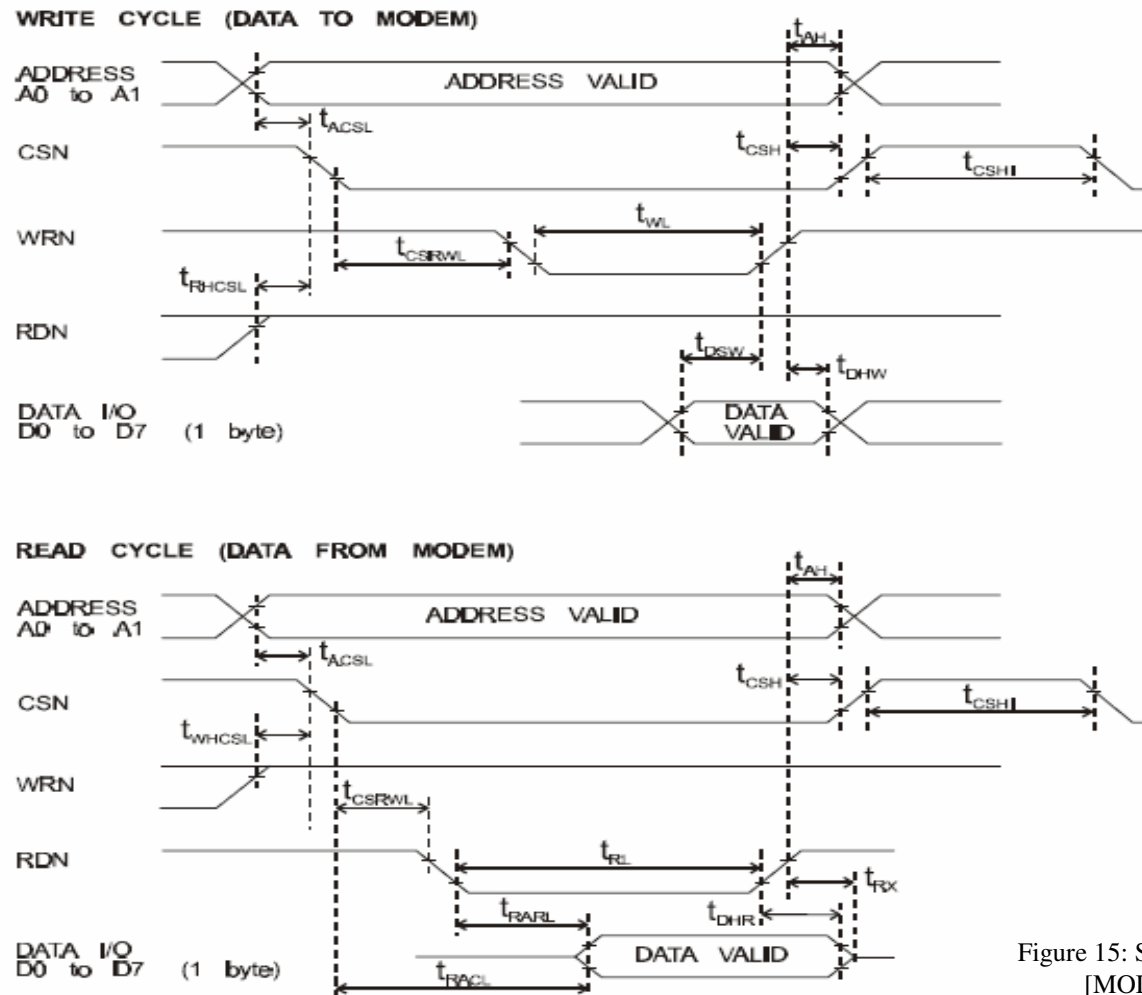
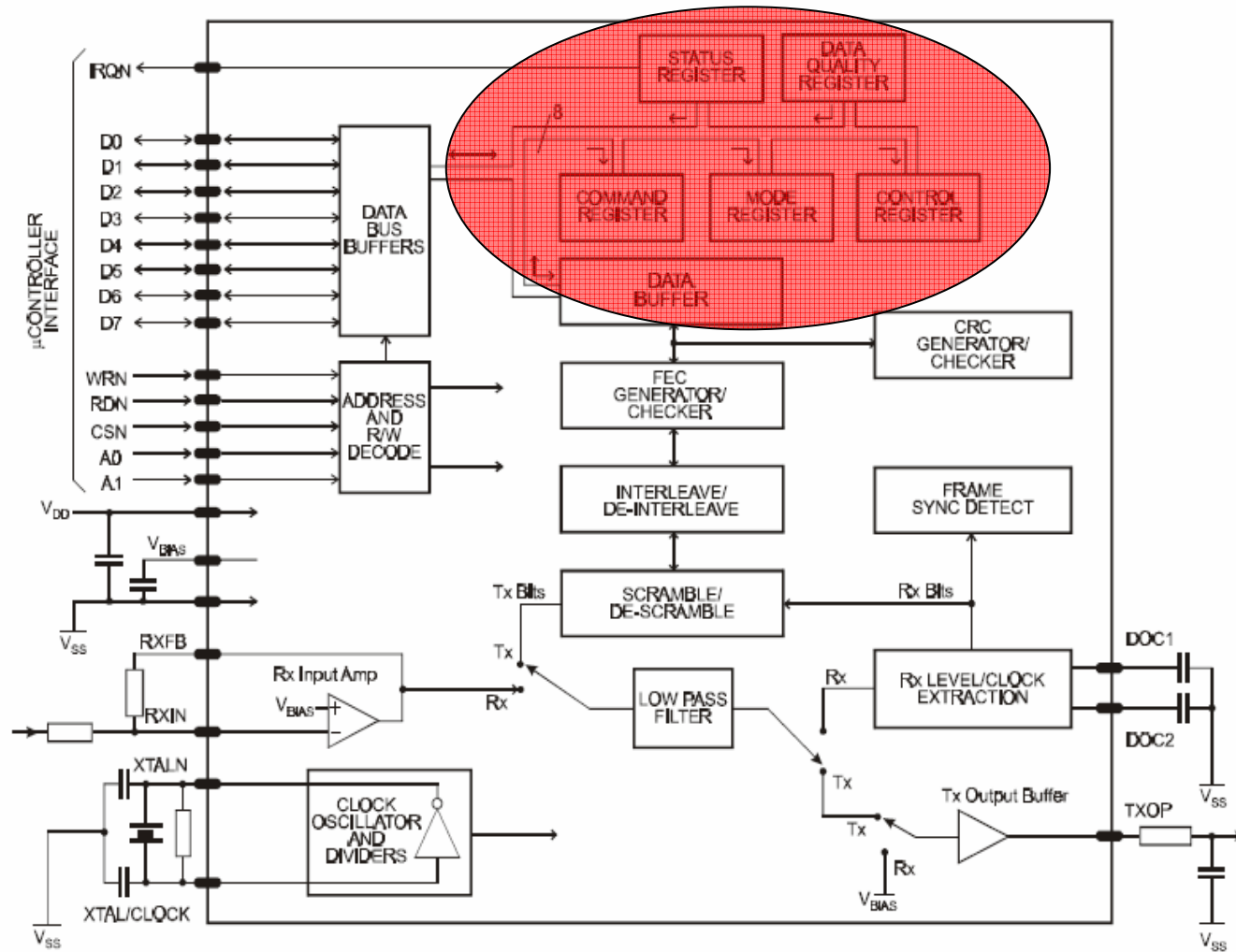
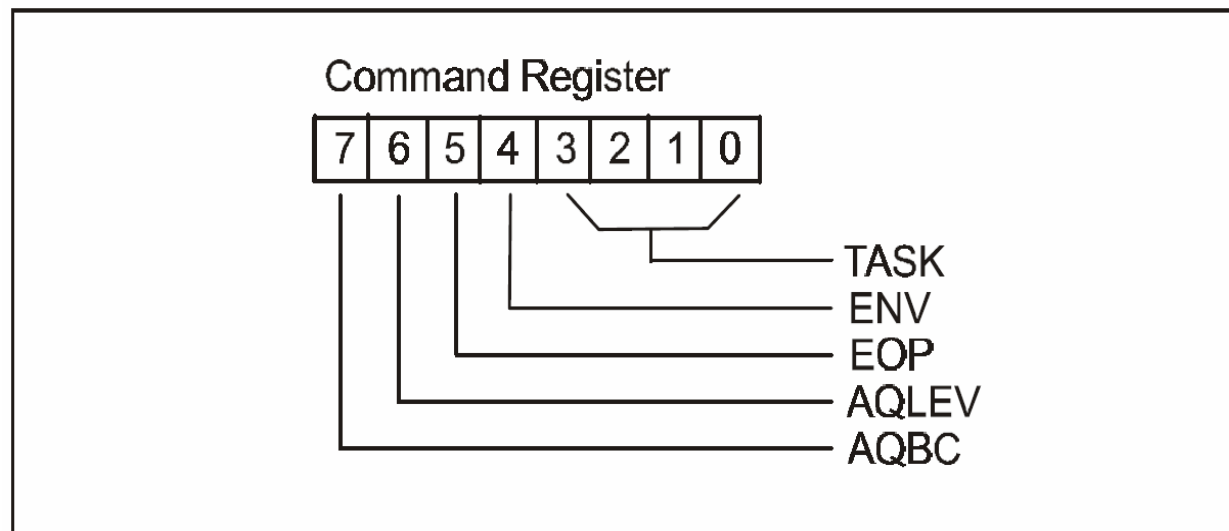


Figure 15: Schreiben und Lesen Zyklus
[MODEM_cmx909bd.pdf]

Register



Command Register



AQBC = Acquire Bit Clock

AQLEV = Acquire Receive Signal Levels

EOP = End of Packet Detector

ENV = Envelope Detector

TASK = Task

Control Register

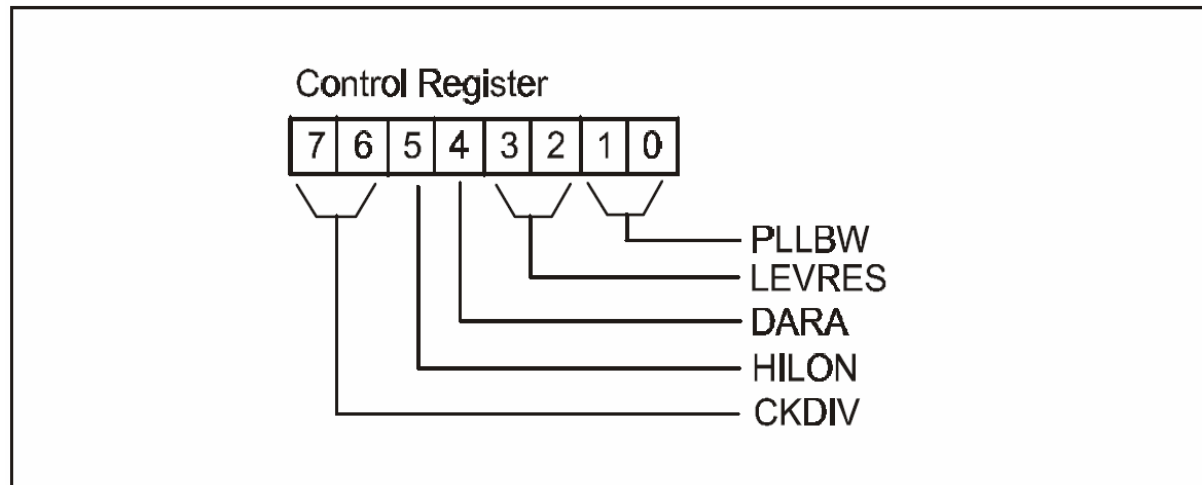
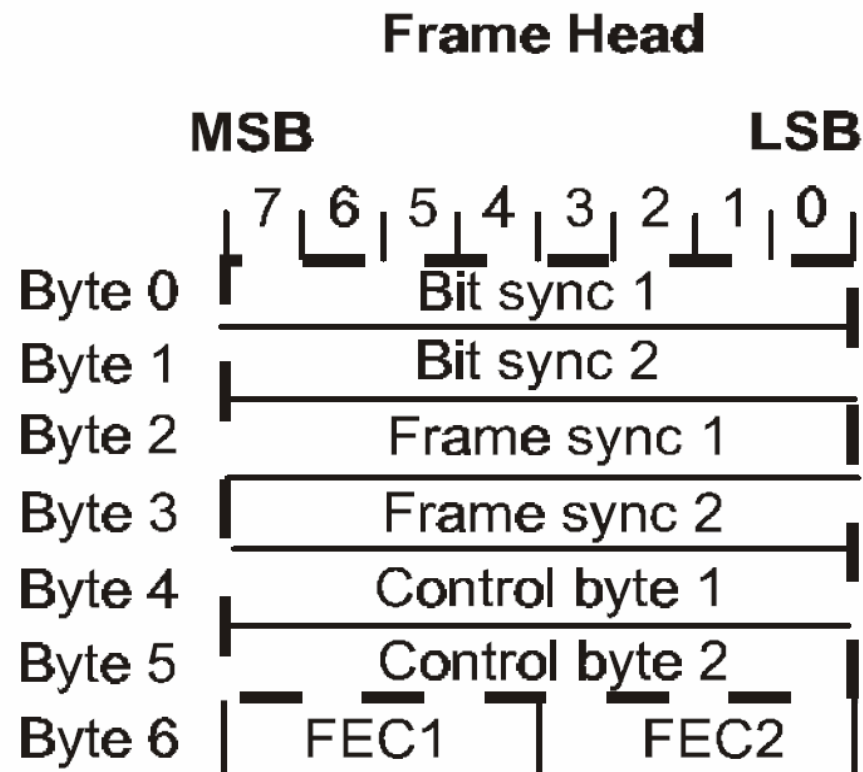


Figure 19: Control Register
[MODEM_cmx909bd.pdf]

CKDIV= Clock Division Ratio
HILON = XTAL Range Selektion
DARA = Data Rate
LEVRES = Level Measurement Response Time
PLLBW = Phase Locked Loop

Протокол передачи информации. Преамбула.



Протокол передачи информации. Данные.

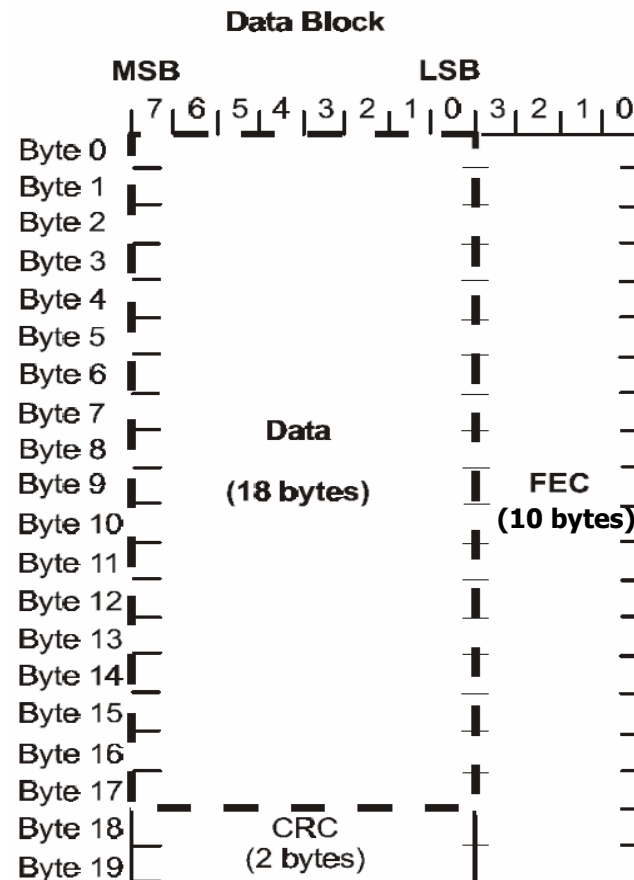
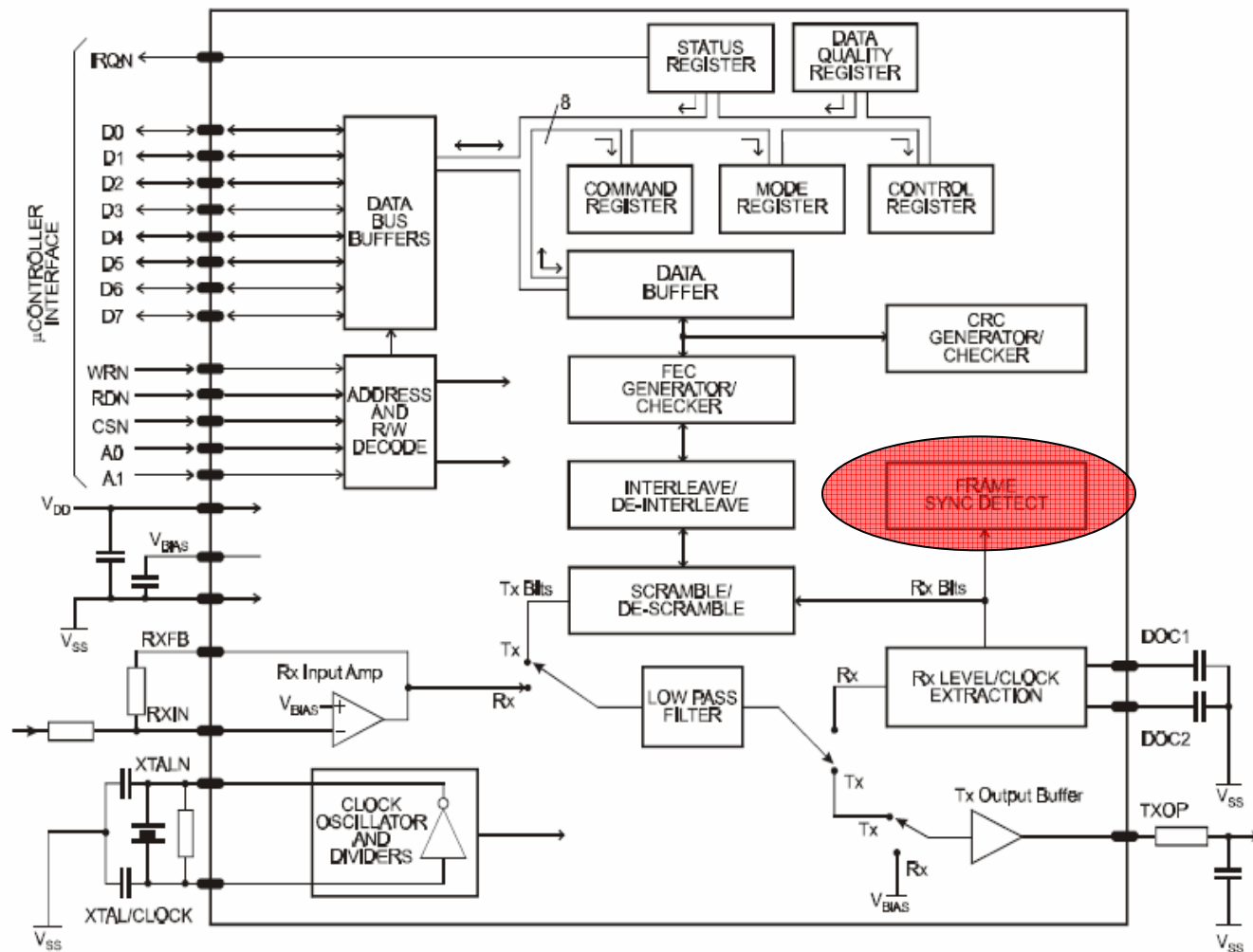
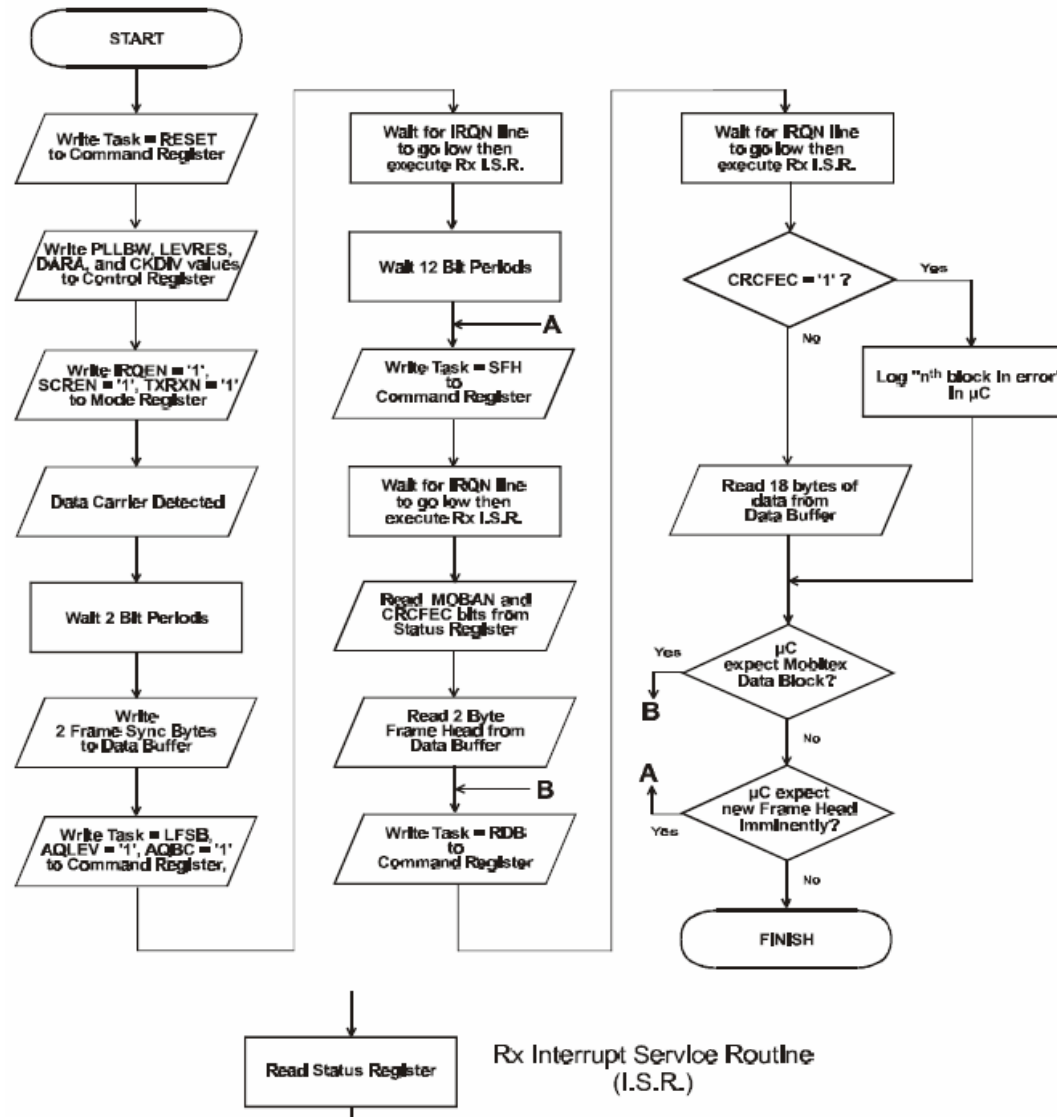
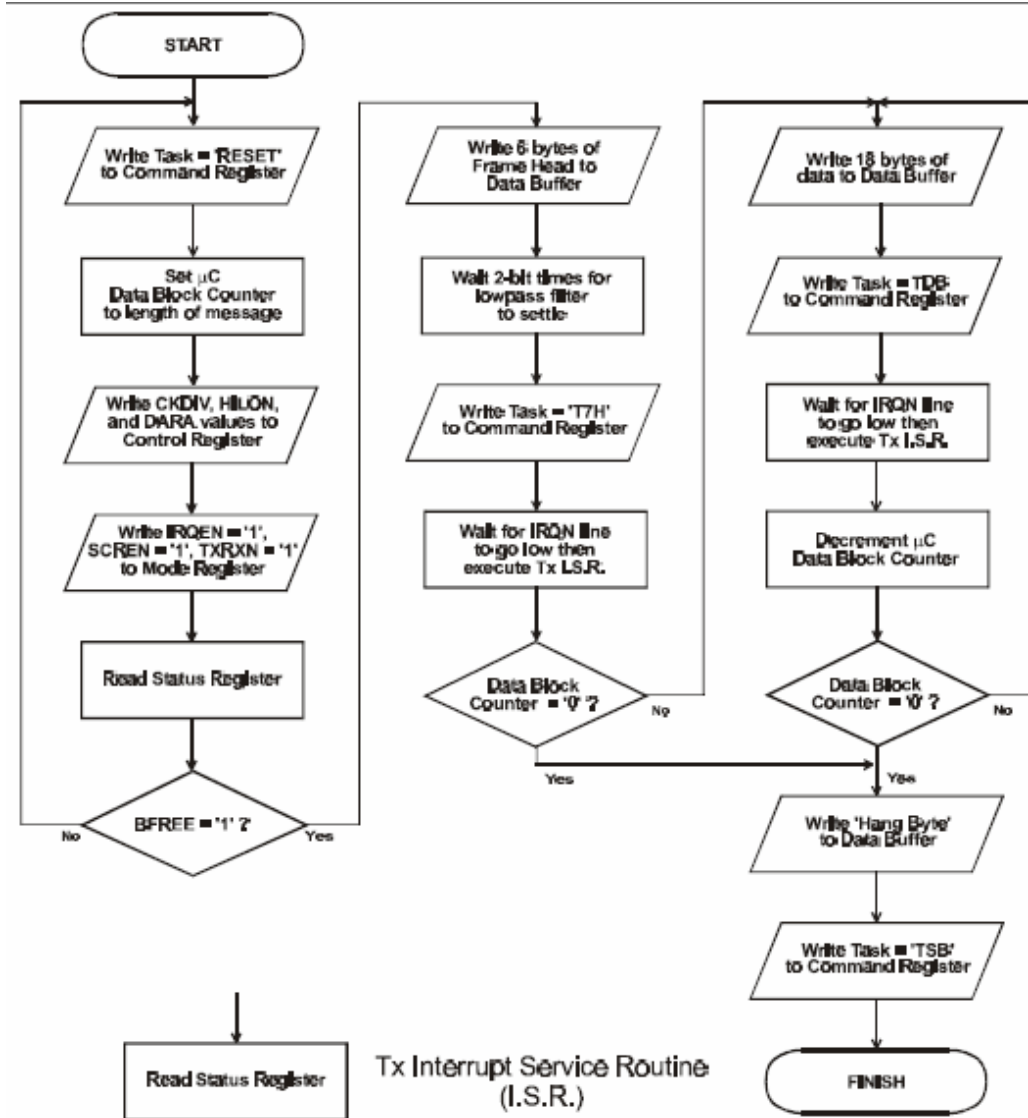


Figure 24: Externes GMSK Modem Data Block
[MODEM_cmx909bd.pdf]

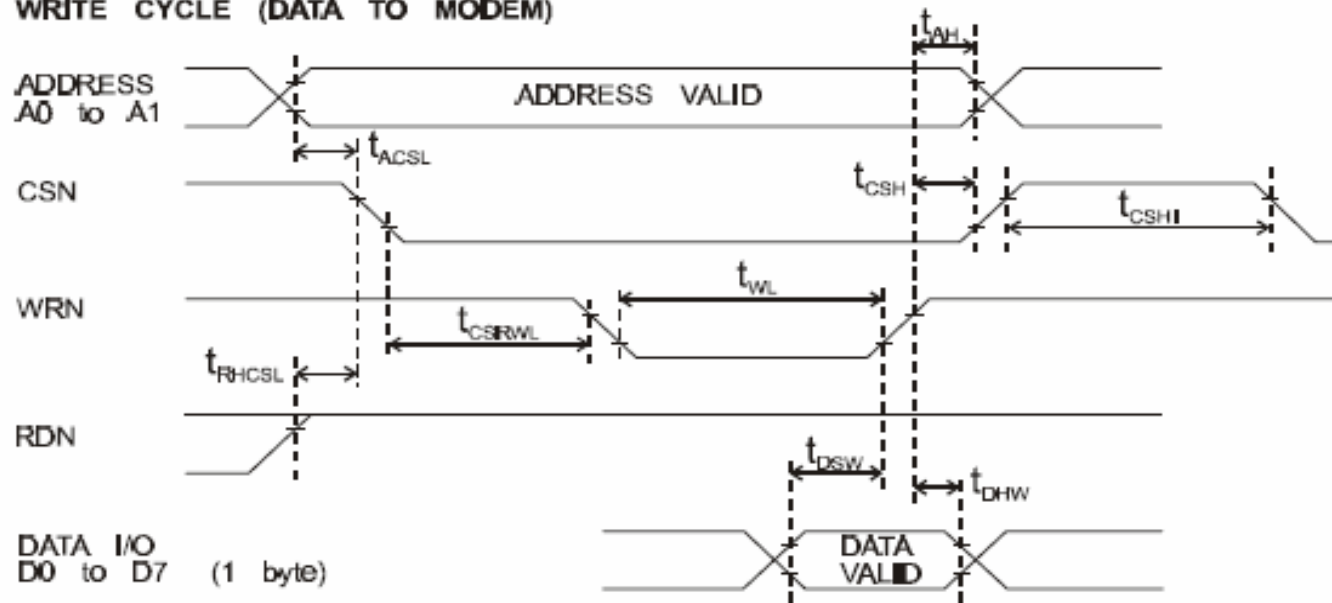
Frame Sync Detect



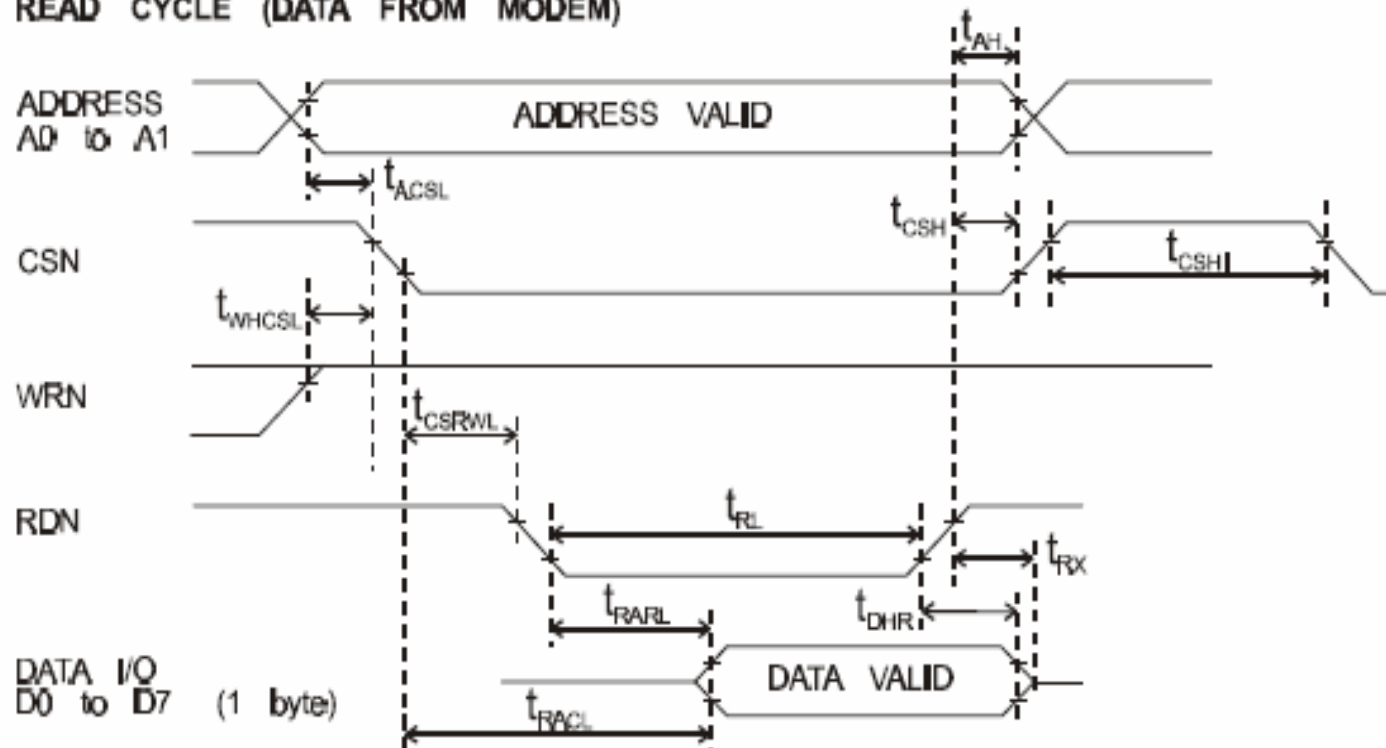




WRITE CYCLE (DATA TO MODEM)



READ CYCLE (DATA FROM MODEM)



Software

Quellcode (FFSK MODEM)

```

$regfile = "M32DEF.DAT"           ' StAveR-24M32 has the ATmega32 on board
$crystal = 14745600               ' 14.7456 MHz oscillator
$baud = 115200                    ' use baud rate
$hwstack = 32                     ' default use 32 for the hardware stack
$swstack = 10                     ' default use 10 for the SW stack
$framesize = 40                   ' default use 40 for the frame space

Enable Interrupts                  ' aktivieren von externen Interruptquellen
Config Int1 = Rising               ' auf steigende Flanke reagieren

On Int1 Modem_int                  ' Name der Subroutine
Enable Int1                        ' Interrupt freigeschaltet

On Urxc Rx232_isr                  ' Name (beliebig) aktiviert die Schnittstelle
Enable Urxc

Config Timer0 = Timer , Prescale = 1024 ' Konfiguration des Timers mit Teiler 1024
On Timer0 Timer_isr                ' Zuweisung des Interruptnamens
Enable Timer0                      ' Timer freischalten
Stop Timer0                        ' Timer stoppen

Config Adc = Single , Prescaler = Auto , Reference = Internal ' A-D Wandler deklarieren

Declare Sub Verschicken(byval Xy As Byte ) ' Deklarieren eines Unterprogramms ohne Rückgabewert zum
übermitteln eines                  ' Bytes ans Modem

Declare Function Empfangen() As Byte ' Deklarieren einer Funktion da ein Wert von der Funktion übergeben
werden muß                        ' Empfangen eines Bytes

' Deklaration der Ports als Input oder Output

Config Portc.2 = Output            ' Green
Config Portc.3 = Output            ' Yellow/Orange
Config Portc.4 = Output            ' Red
Config Portd.4 = Input             ' Txsync
Config Portd.5 = Output            ' Txdata
Config Portc.0 = Output            ' Txenable Modem
Config Portc.1 = Output            ' Rxenable Modem
Config Porta.7 = Output            ' Txenable Funkgerät
Config Porta.2 = Input             ' Rxsync
Config Porta.1 = Input             ' Clocked Data

' Festlegung der verwendeten Variablen

Dim Rs As Byte                    ' Dateneinlesen-Variable
Dim Z As Byte                     ' Zähler-Variable (256 zustände)
Dim Y As Byte                     ' Hilfszähler
Dim Daten(255) As Byte            ' Daten-Matrix zum senden (255 Byte)
Dim Datene(255) As Byte            ' Daten-Matrix zum empfangen (255 Byte)
Dim Laenge As Byte                ' Länge des Datenpakets
Dim I As Integer                  ' For-Zähler-Variablen
Dim I2 As Byte                    ' Zähler für Sub Verschicken und Empfangen
Dim I3 As Byte
Dim Timerflag As Byte              ' Timerflag für Fehlermeldungen
Dim Smeter As Word                ' Variable für Signalstärke als 16bit Wert
Dim Smeter2 As Byte               ' Variable für Signalstärke als 8bit Wert
Dim Sb2 As Byte                   ' Hilfsvariable für Sub Verschicken
Dim A As Bit                      ' Hilfsvariable für Empfangen Funktion
Dim C1 As Byte                    ' Variable für eingelesenen Byte in Empfangen Funktion
Dim Anfang(2) As Byte              ' Festlegung der Variable für den Kopf der Übertragung
Dim Empfang As Word                ' Empfängener Kopf wird als 16bit Wert eingelesen (um Anfang der
Übertragung zu finden)
Dim Kennung(2) As Byte             ' Zwei Byte als Platzhalter für die Kennung
Dim Kennunge(2) As Byte            ' Zwei Byte für die empfangene Kennung damit eigne Kennung nicht
überschrieben wird
Dim Check As Word                  ' Berechnete CRC16 Checksumme (16bit)
Dim Check1 As Byte                ' Check 1 und 2 = Check geteilt in 2 8bit Werte damit diese übertragen werden
können, wird auch für den Empfang verwendet
Dim Check2 As Byte
Dim Check3 As Word                ' Empfangene CRC16 Checksumme (16bit)
Dim Modemflag As Byte              ' wird auf 1 gesetzt wenn ein Träger erkannt wurde
Dim C2 As Word                    ' Hilfsvariable um den Kopf beim empfangen als 16bit Wert einzulesen

```

```

Dim Enter As Byte                                ' wird auf 1 gesetzt wenn Eingabe über Tastatur beendet ist

Print "BUCKAH v15 ist online!"
Print Chr(13)

' Initialisierung einiger Variablen

Z = 0
Laenge = 0
Modemflag = 0
Enter = 0
Anfang(1) = 65                                ' Definition des Kopfes mit AB
Anfang(2) = 66
Kennung(1) = 130                              ' Festlegung der Kennung
Kennung(2) = 131

' Herstellen des Empfangskopfes als 16bit Wert aus dem definierten Kopf

Empfang = Anfang(1)
Shift Empfang , Left , 8
Empfang = Empfang + Anfang(2)

' Beginn des eigentlichen Programms

Do
  Anfang:                                     ' Sprungpunkt aus späterem Programmverlauf

  ' Funkgerät und Modem wird empfangsbereit geschaltet

  Set Portc.0                                ' Txenable Modem wird auf 1 gesetzt
  Set Portc.1                                ' Rxenable Modem wird auf 1 gesetzt
  Reset Porta.7                              ' Txenable Funk wird auf 0 gesetzt (Transistorschaltung)

  ' alle LEDs aus

  Set Portc.2
  Set Portc.3
  Set Portc.4
  Idle                                     ' Ruhezustand, Programm läuft nur weiter wenn Interupt kommt

  ' Unterprogramm für zeitbegrenzte Eingabe (optional, wird zur Zeit nicht verwendet)

  'If Timerflag = 1 Then                    ' Fehlermeldung
  'Print "Error: Zeit ueberschritten!"
  'Waitms 1000
  'Z = 0
  'Laenge = 0
  'Stop Timer0
  'Timerflag = 0
  'Set Portc.4                              ' Lampe Red aus
  'End If

  ' Unterprogramm zur Datenübertragung

  If Enter = 1 Then                        ' Programmpunkt wird nur abgearbeitet wenn Daten und Enter eingegeben wurden
  'Stop Timer0                            ' hier würde der Timer bei einer zeitbegrenzten Eingabe wieder gestoppt
  If Laenge = 0 Then                      ' Ausgabe wenn keine Daten eingelesen wurden
    Print "Es wurden keine Daten eingegeben!"
    Goto Sendend                          ' Überspringen der Sendung der Daten, da keine Daten eingegeben
  End If

  ' Funkgerät und Modem wird auf sendebereit geschaltet

  Reset Portc.0                            ' Txenable Modem wird auf 0 gesetzt
  Reset Portc.1                            ' Rxenable Modem wird auf 0 gesetzt
  Set Porta.7                             ' Txenable Funk wird auf 1 gesetzt (Transistorschaltung)

  ' Die CRC16 Checksumme wird erstellt und in zwei 8bit Werte zerteilt

  Check = Crc16(daten(1) , Laenge)
  Print "CRC16: " ; Check
  Check2 = Check
  Shift Check , Right , 8
  Check1 = Check

  ' Übertragung von 150 mal 1 damit Gegenstelle empfangsbereit für die Daten ist

```

```

For I = 1 To 150
    Verschicken 1
Next

Print "Laenge: " ; Laenge          ' Ausgabe der Länge

' Übertragung des Kopfes (AB Länge Kennung1 Kennung2)

Verschicken Anfang(1)
Verschicken Anfang(2)
Verschicken Laenge
Verschicken Kennung(1)
Verschicken Kennung(2)

' Übertragung der Daten

For I = 1 To Laenge
    Verschicken Daten(i)
    Toggle Portc.3                ' gelbe LED blinkt
Next

' Übertragung der CRC16 Checksumme als 2 8bit Werte

Verschicken Check1
Verschicken Check2

' Übertragung eines Hängebytes damit eigentlich letztes Byte sauber empfangen wird

Verschicken 255

Set Portc.3                      ' Lampe Yellow aus, sendet 1, invertiert angeschlossen
Print "Done"
Print Chr(13)

Sendend:                          ' hier hin wird gesprungen wenn keine Daten eingegeben werden

' Zurücksetzen der verwendeten Variablen

Laenge = 0
Z = 0
Enter = 0
End If

' Unterprogramm zum Empfang der Daten, wird abgearbeitet wenn Modemflag 1 ist

If Modemflag = 1 Then
    Modemflag = 0

    Start Adc
    Smeter = Getadc(0)             ' auslesen der analogen Werte, Smeter Variable, 0 = Pin 0 wird ausgelesen
                                   ' Smeter ist 16 Bit Variable aber maximal 10 Bit lang von AD Wandler
                                   ' wir brauchen aber 8 Bit

    Stop Adc
    Smeter = Smeter + 2            ' aus 10 Bit Variable wird 8 Bit Variable produziert, + 2 ist für die Abrundung
    Shift Smeter , Right , 2
    Smeter2 = Smeter

    ' Einlesen der ersten Bits von Pin PA2 bis ein 16bit Wert entsteht der dem Kopf entspricht, dann wird das
    Empfangsprogramm
    ' weiter abgearbeitet oder bis maximal 500 Bits eingelesen wurden, dann wird Empfangsprogramm übersprungen da kein
    ' Kopf empfangen wurde

    For I = 0 To 500
        Bitwait Pina.2 , Set
        Bitwait Pina.2 , Reset
        A = Pina.1                 ' clocked data O/P
        Shift C2 , Left , 1
        C2 = C2 Or A
        If C2 = Empfang Then Goto A2
    Next
    Goto Anfang

A2:

Reset Portc.3                     ' einschalten der gelben Lampe als Zeichen das Kopf empfangen wurde

```

```

' Empfang von Länge und Kennung

Laenge = Empfangen()
Kennunge(1) = Empfangen()
Kennunge(2) = Empfangen()

' Empfang der eigentlichen Daten

For I = 1 To Laenge
    Datene(i) = Empfangen()
Next

' Empfang der CRC16 Checksumme als 2 8bit Werte

Check1 = Empfangen()
Check2 = Empfangen()

' Ausgabe der Signalstärke

Print "Signalstaerke: " ; Smeter2
Print Chr(13)

' Zusammenbau des empfangenen CRC16 Wertes als 16bit Wert

Check3 = Check1
Shift Check3 , Left , 8
Check3 = Check3 + Check2

' Berechnen der CRC16 Checksumme aus den empfangenen Daten

Check = Crc16(datene(1) , Laenge)

' Überprüfung ob beide Checksummen übereinstimmen

If Check3 = Check Then
    Print "Kein Fehler in empfangenen Daten gefunden."
    Print "CRC16 empfangen und berechnet: " ; Check
    Print Chr(13)
Else
    Print "Error: Es ist ein Uebertragungsfehler aufgetreten!"
    Print "CRC16 empfangen: " ; Check3
    Print "CRC16 berechnet: " ; Check
    Print Chr(13)
End If

Reset Portc.2                ' Aktivierung der grünen LED als zeichen das Datenempfang abgeschlossen

' Ausgabe der empfangenen Daten mit Kennung und Länge

Print "Kennung: " ; Chr(kennunge(1))
Print Chr(kennunge(2))
Print Chr(13)
If Laenge > 255 Then
    Print "Error: Es wurden zu viele Daten gesendet. Ein fehlerfreier Empfang kann nicht garantiert werden."
    Print Chr(13)
End If
Print "Daten: "
For I = 1 To Laenge
    Print Chr(datene(i))
Next
Print Chr(13)
Print "Laenge: " ; Laenge

Laenge = 0                ' Zurücksetzen der Länge

Print "Done"
End If

Loop
End

' Definition der Interrupts

' Timerinterrupt

Timer_isr:

```

```

Reset Portc.4
Set Portc.2
Timerflag = 1
Return
Return

' Modeminterrupt

Modem_int:
Reset Portc.4
Modemflag = 1
Return
Return

' Schnittstelleninterrupt

Rx232_isr:
Rs = Inkey()

' Wenn die Taste Enter gedrückt wurde wird die Länge der Daten bestimmt

If Rs = 13 And Z > 1 Then
    Laenge = Z - 1
    Daten(Laenge) = Rs
    Print Chr(daten(Laenge))
    Print Chr(13)
    Enter = 1
    Goto Rssubend
End If

If Z > 1 Then                                ' Datenpaket einlesen
    Y = Z - 1
    If Y = 252 Then
        Print Chr(13)
        Print "Vorletztes Zeichen eingegeben. Nur noch Platz fuer ein Zeichen jedes weitere Zeichen fuehrt zum Speicher ueberlauf."
        Print Chr(13)
    End If
    Daten(y) = Rs
    Print Chr(daten(y));
    Z = Z + 1
    Toggle Portc.3
    Goto Rssubend
End If

If Z = 1 And Rs = Anfang(2) Then              ' Auf B überprüfen
    Z = 2
    Toggle Portc.3
    Goto Rssubend
Else
    Z = 0
End If

If Z = 0 And Rs = Anfang(1) Then              ' Auf A überprüfen
    Z = 1
    Toggle Portc.3
    Goto Rssubend
End If
Rssubend:
Return
Return

'Ab hier kommt die Definiton der Subprogramme und Funktionen

Sub Verschicken(byval Xy As Byte)

For I2 = 0 To 7                                ' I2 gibt die Position des Bit im Byte an
    I3 = 7 - I2
    Bitwait Pind.4 , Set                        ' Warten auf fallende Kante (clock high)
    Bitwait Pind.4 , Reset                      ' Warten auf fallende Kante (clock low)
    Sb2 = 2 ^ I3                               ' Bewirkt das immer nur der aktuelle Bit aus dem Byte übertragen wird
    Sb2 = Xy And Sb2
    If Sb2 > 0 Then Set Portd.5 Else Reset Portd.5    ' Übertragen des Bits an Portd.5
Next
End Sub

```

Function Empfangen() As Byte

For I2 = 0 To 7

 Bitwait Pina.2 , Set

 Bitwait Pina.2 , Reset

 A = Pina.1

 Shift C1 , Left , 1

 nächste Bit

 C1 = C1 Or A

Next

Empfangen = C1

End Function

'Ende des Programms

' Es werden 8 Bit empfangen

' Warten auf fallende Kante (clock high)

' Warten auf fallende Kante (clock low)

' Einlesen des Taktes 0 oder 1

' C1 ist die Ziel Variable für das eingelesene Byte, Verschiebung um ein Bit für das

' Logische Verknüpfung

```

1 '-----
2 'Initialisierung
3 '-----
4
5 'Staver konfigurieren
6 $regfile = "M32DEF.DAT" 'Name des Chips
7 $crystal = 14745600 'Taktfrequenz des Staver in
  Hz
8 $baud = 38400 'Geschwindigkeit der
  COM-Schnittstelle in bit/s
9
10
11 'Interrupts konfigurieren
12 Enable Interrupts 'Interrupts einschalten (
  allgemein da welche benötigt werden)
13 On Urxc Com_port_int 'Urxc als COM-Interrupt
  nutzen, dabei heißt Interrupt-routine heißt "Com_port_int" (darin stehtdann was bei
  Interrupt gemacht werden soll)
14
15 Enable Urxc 'Urxc einschalten
16 On INT1 Modem_int1 'Interrupt1 (befindet sich an
  Pin PD3) soll genutzt werden, Interrupte-routine heißt: Modem_int1
17 Config INT1 = Falling 'unterbrechen wenn Flanke
  fällt
18 Enable INT1 'Int1 einschalten
19
20
21 'AD-Wandler konfigurieren
22 Config ADC = Single , Prescaler = Auto , Reference = Internal 'Single-Mode
  erlaubt auslesen mit getadc(),
23 'Prescaler teilt
  internen Takt um Aulesetakt zu erhalten, keine externer Referenzwert fü
24 'mit Reference könnte
  man das Signal mit einem externen Referenzwert abstimmen (nichtbenötigt also internal)
25 'Timer konfigurieren
26 Config Timer0 = Timer , Prescale = 1024 'Prescale multipliziert mit
  interner Taktlänge ergibt Zähldauer des Timers
27 On Timer0 Tim0_isr 'Name der Subroutine ist
  Tim0_isr (darin steht dann was gemacht werden soll wenn Timer
  abgelaufenist)
28 Enable Timer0 'Timer0 einschalten
29 Stop Timer0 'Timer0 stoppen, damit er
  nicht gleich zählt
30
31 Config Timer1 = Timer , Prescale = 1024
32 On Timer1 Tim1_isr
33 Enable Timer1
34 Stop Timer1
35
36
37 'UART zu Funkgerät konfigurieren
38 Open "comd.4:4800,8,n,1" For Output As #1 'Pin PD4 als Output nutzen (
  TXD), Geschwindigkeit 4800 bit/sec, byteweises übertragen
39 Open "comd.5:4800,8,n,1" For Input As #2 'Pin PD5 als Input nutzen (
  RXD), Geschwindigkeit 4800 bit/sec, byteweises übertragen
40
41
42 'Pins konfigurieren
43 Config PINC.2 = Output 'grüne LED für "Ausgabe"
  vorsehen
44 Config PINC.3 = Output 'gelbe LED für "Ausgabe"
  vorsehen
45 Config PINC.4 = Output 'rote LED für "Ausgabe"
  vorsehen
46 Config PINA.0 = Output 'PTT-Pin konfigurieren
  ganzen Port konfigurieren:
47 Config Port.A=Input
48
49 'Watchdog konfigurieren
50 Config Watchdog = 32 'wartet 32ms bis zum reset
51
52
53 'Variablen initialisieren
54 'bite: 1 bite variable (0 und 1 darstellbar)
55 'Byte: 1 byte variable (255 Zahlen darstellbar)
56 'Word: 2 byte variable (Zahlen bis ca. 65000 darstellbar)
57 'Long: 4 byte variable (bis einige Millionen Zahlen darstellbar)
58
59 Dim Rs As Bit 'Ein-Aus-Schalter der
  Fehlerkorrektur
60 Dim Error As Bit 'Ein-Aus-Schalter der
  Kanalstörung
61 Dim Timer0flag As Bit 'Timer0flag wird anzeigen ob
  Timer0 abgelaufen ist
62 Dim Modemflag As Bit 'Modemflag wird anzeigen ob

```

```

es ein Modeminterrupt gab
63
64 Dim A As Byte
65 Dim Beacon As Byte                                     'Beacon wird zählen wie oft
Timer1 abgelaufen ist
66 Dim Comarray(20) As Byte                             'Comarray(20) speichert die
Daten
67 Dim Bort As Byte
68 Dim Rx As Byte
69 Dim C1 As Byte
70 Dim Rscom1 As Byte
71 Dim Index As Byte
72
73 Dim Smeter As Word                                    'Smeter um AD-Wandler (10bit)
auszulesen
74 Dim Cs2 As Word
75 Dim J As Word
76 Dim K As Word
77 Dim I As Word
78 Dim I2 As Word
79
80 Dim Checksumme As Long                                'Checksumme um sehr großen
Checksummenwert speichern zu können
81
82
83 'zur Initialisierung des Fehlerkorrekturcodes springen
84 Goto Init_rs                                           'Sprung zum Label Init_r:
85 Init_rs_back:                                          'und zu diesem Label kehrt
Programm zurück wenn Init_rs abgearbeitet ist
86
87
88 'Anfangswerte festlegen
89 Rs = 0                                                  'Fehlerkorrektur ausgeschaltet
90 Error = 0                                              'Kanalstörung ausgeschaltet
91 Beacon = 0                                             'Zähler für Beacon Timer zum
Anfang 0
92 Timer0flag = 0                                         'Timer0 nicht abgelaufen
93 Set PORTC.2                                           'grüne LED erhält 5V und ist
aus
94 Set PORTC.3                                           'gelbe LED erhält 5V und ist
aus
95 Set PORTC.4                                           'rote LED erhält 5V und ist
aus
96
97
98 'Start/Reset Meldung ausgeben
99 Print "Staver gestart"                                'damit sieht man auch wenn
Watchdog abgelaufen ist
100
101
102
103 '-----
104 'Programmteil (Hauptschleife)
105 '-----
106
107 Anfang:                                              'Sprungmarke um mit
Sprungbefehl am Ende Schleife zu bilden
108
109 Idle                                                  'solange in Idle wechseln
bis Interrupt kommt (es gibt noch andere Power-Save-Mode, in denen aber noch mehr aktiv ist)
110
111 If Rscom1 = 20 Then                                    'wenn Telekommandos über COM
vollständig empfangen (Rscom1 ist Zähler, 2 Byte Header + 18 Byte Daten)
112 Stop Timer0                                           'als erstes Timer0 stoppen,
da alles rechtzeitig empfangen wurde
113 Rscom1 = 0                                             'Zähler für nächste
Übermittlung zurücksetzen
114 If Rs = 0 Then                                        'falls Fehlerkorrektur
ausgeschaltet ist an Transmit übergeben
115 Goto Transmit
116 Else
117 Goto Transmit2                                        'andernfalls an Transmit2
übergeben
118 End If
119 End If
120
121 If Timer0flag = 1 Then                                'falls Timer abgelaufen ist
122 Stop Timer0
123 Print "zu wenige Zeichen über Com1 empfangen"
124 Timer0flag = 0                                        'Timer0flag sowie Zähler
Rscom1 zurücksetzten
125 Rscom1 = 0
126 End If
127

```

128	If Beacon = 4 Then	
129	Beacon = 0	'wenn Timer1 4mal abgelaufen
	ist (dieser erhöht Zähler Beacon immer um eins)	
130	Goto Beacon2	'Zähler Beacon zurücksetzen
131	End If	'an Beacon2 übergeben, hier
	ändern falls Beacon1 gewünscht	
132		
133	If Modemflag = 1 Then	'wenn es ein Interrupt am
	Modem gab steht Modemflag auf 1	
134	Modemflag = 0	'Modemflag zurück setzen,
	damit es nächstes Interrupt wieder anzeigen kann	
135	If Rs = 0 Then	'falls Fehlerkorrektur aus
	ist an Recieve-Funktion übergeben	
136	Goto Recieve	
137	Else	'andernfalls an
	Recieve2-Funktion übergeben	
138	Goto Recieve2	
139	End If	
140	End If	
141		
142	Goto Anfang	'zurück zum Anfang der
	Hauptschleife	
143	End	'End-Befehl ist am Ende jedes
	Programmes nötig	
144		
145		
146		
147	'-----	
148	'Quasi-Funktionen (Transmit, Beacon1, Beacon2, Recieve)	
149	'-----	
150		
151	Transmit:	
152	Reset PORTC.4	'rote LED anschalten
153	Checksumme = Crc16 (comarray(1) , 18)	'mit CRC16 beginnend von 1
	Byte des Comarrays bis zum 18 Checksume bilden	
154	If Error = 1 Then	'falls die Kanalstörung
	angeschalten ist	
155	Rx = Rnd (5)	'zufällige ganze Zahl von 0
	bis 4 generieren (Anzahl der Fehler)	
156	For I = 1 To Rx	'Schleife über Anzahl der
	Fehler	
157	C1 = Rnd (18) + 1	'Fehlerstelle (1-18)
	generieren	
158	If Comarray(c1) <> 95 Then	'falls nicht schon fehler an
	dieser Stelle eingebaut wurde	
159	Print "Fehler einbaut an Stelle ";	
160	Print C1	'Fehler ins Comarray
	schreiben (95 entspricht einem Unterstrich)	
161	Comarray(c1) = 95	
162	Else	'falls doch schon Fehler an
	dieser Stelle eingebaut wurde	
163	I = I - 1	'Fehlerzähler wieder um eins
	zurück setzen	
164	End If	
165	Next	
166	End If	
167		
168	Set PORTA.0	'PTT anschalten
169	Waitms 30	'kurz warten damit
	Modem/Funkgerät starten kann	
170	Printbin #1 , 204 ; 67 ; 68	'über TXD
	Synchronisationsbyte 204 (11001100) und Framehead CD an Funkgerät schicken	
171	For K = 1 To 18	'mittels Zählschleife alle
	Elemente des Comarrays an Funkgerät schicken	
172	A = Comarray(k)	
173	Printbin #1 , A	
174	Next	
175	Printbin #1 , Checksumme ; 204	'zuletzt noch Checksumme und
	204 (11001100) als Endbyte schicken	
176	Reset PORTA.0	'PTT ausschalten
177	Set PORTC.4	'rote LED ausschalten
178	Print "Nachricht gesendet"	'zurück in die Hauptschleife
	gehen	
179	Goto Anfang	
180		
181		
182	Beacon1:	'Beacon1 lässt Funkgerät
	dreimal piepsen	
183	Set PORTA.0	'PTT anschalten
184	Waitms 30	'kurz warten damit Funkgerät
	startbereit ist	
185	Print "Beacon (dreimal piepsen)!"	'Beacon auf Bildschirm
	anzeigen	
186	For I2 = 1 To 3	'Zählschleife über

```

dreimaliges piepsen
187     Waitms 400                                     '400ms zwischen Piepen warten
188     For I = 1 To 500                               'Zählschleife um 500 mal 15 (
11110000) über TXD an Funkgerät zu schicken
189     Printbin #1 , 15                               'da wir GMSK modulieren
verringert 11110000 Übertragungsfrequenz um 4: 2,4 kHz werden zu 600Hz welche gut hörbar
sind
190     Next
191 Next
192 Reset PORTA.0                                     'PTT ausschalten
193 Print " "                                           'Zeile löschen (Beacon wird
auf Bildschirm nicht mehr angezeigt)
194 Goto Anfang                                         'zum Anfang der
Hauptschleife zurückkehren
195
196
197 Beacon2:                                           'Beacon2 morst TUBA [dit(*),
dat(-): - * * - - * * * - ]
198 Set PORTA.0                                       'PTT anschalten
199 Waitms 30                                         'kurz warten damit Funkgerät
startbereit ist
200 Print "Beacon wird gemorst!"                     'Beacon auf Bildschirm
anzeigen
201 'T                                               'erst T dann U,B,A wie oben
gezeigt morse
202 For I = 1 To 120                                 'für dit Zählschleife 40mal
ablaufen lassen
203 Printbin #1 , 15                                 'für dat Zählschleife 120mal
ablaufen lassen
204 Next
Buchstaben) 240ms
205 Waitms 240                                       'lange Pause (zwischen
Buchstaben) 80 ms
206 'U                                               'kurze Pause (in einem
207 For I2 = 1 To 2                                  '15 (11110000) über TXD an
Funkgerät schicken um hörbaren Ton zu erzeugen
208     For I = 1 To 40                              'da wir GMSK modulieren
verringert 11110000 Übertragungsfrequenz um 4: 2,4 kHz werden zu 600 Hz welche gut hörbar
sind
209     Printbin #1 , 15
210     Next
211     Waitms 80
212 Next
213 For I = 1 To 120
214     Printbin #1 , 15
215 Next
216 Waitms 240
217 'B
218 For I = 1 To 120
219     Printbin #1 , 15
220 Next
221 For I = 1 To 3
222     Waitms 80
223     For I2 = 1 To 40
224         Printbin #1 , 15
225     Next
226 Next
227 Waitms 240
228 'A
229 For I = 1 To 40
230     Printbin #1 , 15
231 Next
232 Waitms 80
233 For I = 1 To 120
234     Printbin #1 , 15
235 Next
236 Reset PORTC.3
237 Print " "                                           'Zeile löschen (Beacon wird
auf Bildschirm nicht mehr angezeigt)
238 Goto Anfang                                         'zum Anfang der
Hauptschleife zurückkehren
239
240
241 Recieve:
242 Reset PORTC.3                                     'gelbe LED anschalten wenn
etwas empfangen werden soll
243 Waitms 10
244 C1 = 0                                           'C1 zählt die ersten fünf
Bytes ab auf die gewartet wird
245 Start Watchdog                                     'Watchdog beginnt zu Zählen,
wenn er jetzt nicht immer rechtzeitig auf Null zurück gesetzt wird läuft er ab und
startet den staver neu
246 Do                                               'Beginn der Schleife zum
Empfangender ersten fünf Zeichen

```

```

247   C1 = C1 + 1                                     'Zeichen-Zähler um eins
      erhöhen
248   Rx = Waitkey(#2)                               'auf Zeichen warten (bei
waitkey() läuft das Programm nicht weiter solange kein Zeichen empfangen wurde)
249   Reset Watchdog                                'Watchdog auf Null
      zurücksetzen, um wieder Zeit zu haben nächstes Zeichen zu empfangen
250   Loop Until Rx = 67 OR C1 > 5                   'Schleife abbrechen wenn "C"
      (67) oder mehr als fünf andere Zeichen empfangen wurden
251   If Rx <> 67 Then
252       Stop Watchdog                               'wenn kein "C" (erster Teil
      vom Framehead) empfangen wurde
253       Set PORTC.3                                'Watchdog ausschalten
254       Goto Anfang                                'grüne LED ausschalten
255   End If                                           'zurück in die Hauptschleife
      wechseln
256   Reset Watchdog
257   Start ADC
258   Rx = Waitkey(#2)                               'AD-Wandler anschalten
259   Reset Watchdog                                'nächstes Zeichen am Pin
      PD5 (RXD) abwarten
260   If Rx <> 68 Then                                'sobald es empfangen wurde
      Watchdog auf Null setzen
261       Stop Watchdog                               'wenn zweites Zeichen kein
      "D" (68) ist haben wir nicht den Framehead empfangen
262       Set PORTC.3                                'Watchdog ausschalten
263       Goto Anfang                                'grüne LED ausschalten
264   End If                                           'zurück in die Hauptschleife
      wechseln
265   Reset Watchdog
266   For I = 1 To 20                                'mit Hilfe einer
      Zählschleife auf die nächsten 20 Bytes über Pin PD5 (RXD) warten
267       Comarray(i) = Waitkey(#2)
268       Reset Watchdog                              'nach jedem empfangenen
      Byte Watchdog auf Null zurück setzten
269   Next
270   Stop Watchdog                                  'Watchdog ausschalten
271   Smeter = Getadc(2)                              'am AD Wandler 2 Byte
      auslesen (Smeter ist vom Typ Word und kann beide speichern)
272   Stop ADC                                        'AD Wandler ausschalten
273   Checksumme = Crcl6(comarray(1) , 18)            'mit CRC16 beginnend von 1
      Byte des empfangenen Comarrays bis zum 18 Checksume bilden
274   Cs2 = 256 * Comarray(20)                       'gesendete Prüfsumme aus den
      zwei letzten Bytes im Comarray zusammensetzen
275   Cs2 = Cs2 + Comarray(19)                       'da Word-Variablen nur als
      zwei Byte gesendet werden können muss erstes Byte um acht bits nach links verschoben (mal
      256) und zu zweitem aufaddiert werden
276   For I = 1 To 18                                'mit Hilfe einer
      Zählschleife alle Elemente des empfangenen Comarrays ausgeben
277       A = Comarray(i)
278       Printbin A
279   Next
280   If Checksumme = Cs2 Then                         'ausgeben ob berechnete
      Prüfsumme mit gesendeter übereinstimmt
281       Print " - CRC ok";
282   Else
283       Print " - CRC falsch!";
284   End If
285   Print ", Signalstärke:";                        'Signalstärke ausgeben,
      vorher 10bit-Signal zu 8bit konvertieren
286   Smeter = Smeter + 2                             '2 addieren um Genauigkeit
      etwas zu verbessern (2 Bits werden abgeschnitten, d.h. Werte zwischen 0-4, Mittelwert 2)
287   Shift Smeter , Right , 2                        'alle Bits der
      Smeter-Variable nach rechts verschieben, dabei werden die zwei rechten abgeschnitten
288   A = Smeter
289   Print A                                           'Signalstärkewert auf
      Bildschirm ausgeben
290   Set PORTC.3                                     'gelbe LED ausschalten (Ende
      von Recieve)
291   Goto Anfang                                     'zum Anfang der
      Hauptschleife zurückkehren
292
293
294
295 '-----
296 'Interrupts (ComPort-Interrupt, Modem-Interrupt)
297 '-----
298
299 Com_port_int:
300 A = Inkey()                                       'COM-Interrupte-Puffer
      leeren (um gesendetes Byte zu erhalten und damit nächstes Byte empfangen werden kann)
301 Toggle PORTC.2                                   'Zustand gelbe LED wechseln (
      blinken zeigt dann Kommunikation an)
302 If Rscom1 = 0 Then                               'Hilfsvariable Rscom1
303     If A = 65 OR A = 97 Then                     'falls Zähler der

```

empfangenen Zeichen Rscom1 null ist	
304 Rscom1 = 1	'falls "A" (65) oder "a" (97)
empfangen geschickt wurden	
305 Goto Subende 1	'Rscom1 Zähler auf eins
setzen, da erstes Zeichen unseres Frameheads empfangen wurde	
306 End If	'zum Ende der Funktion
springen um nächstes Interrupt abzuwarten	
307 End If	
308 If Rscom1 = 1 Then	'falls schon erstes Zeichen
geschickt wurde	
309 If A = 66 OR A = 98 Then	'falls "B" (65) oder "b" (97)
empfangen geschickt wurden	
310 Rscom1 = 2	'Rscom1 Zähler auf zwei
setzen, da zweites Zeichen unseres Frameheads empfangen wurde	
311 Bort = A	'Bort speichert "B" bzw. "b"
zwischen, damit beim nächsten Interruptaufruf bekannt ob Kommando geschickt wird (nur ein	
weiteres Byte) oder Daten folgen (18 Byte)	
312 Timer0flag = 0	'Timer0flag zurücksetzten
313 Start Timer0	'Timer0 starten, bis er zu
Ende gezählt hat müssen alle Daten über COM geschickt worden sein, damit wartet man nicht	
ewig falls COM zu wenige Daten schickt	
314 Timer0 = 0	'Timer0 auf Null
zurücksetzten	
315 Goto Subende 1	'zum Ende der Funktion
springen um nächstes Interrupt abzuwarten	
316 Else	'falls kein "B" oder "b"
geschickt wurde	
317 If A <> 65 OR A <> 97 Then Rscom1 = 0	'überprüfen ob nicht
vielleicht schon wieder ein "A" oder "a" geschickt wurde	
318 Goto Subende 1	'zum Ende der Funktion
springen um nächstes Interrupt abzuwarten	
319 End If	
320 End If	
321 If Bort = 98 Then	'falls letztes Zeichen ein
"b" war, folgt ein Kommando für den Staver	
322 If Rscom1 = 2 Then	'außerdem muss gerade auf
das dritte Byte gewartet werden	
323 If A = 78 Then	'falls "N" (78) geschickt
wurde	
324 Start Timer1	'Timer1 anschalten um Beacon
anzuschalten	
325 Timer1 = 0	
326 Print "Beacon eingeschaltet"	
327 End If	
328 If A = 70 Then	'falls "F" (70) geschickt
wurde	
329 Stop Timer1	'Timer1 stoppen um Beacon
auszuschalten	
330 Print "Beacon ausgeschaltet"	
331 End If	
332 If A = 69 Then	'falls "E" (69) geschickt
wurde	
333 Error = 1	'Kanalstörung ist an wenn
Error auf 1 steht	
334 Print "Kanalstörungen eingeschaltet"	
335 End If	
336 If A = 67 Then	'falls "C" (69) geschickt
wurde	
337 Error = 0	'Kanalstörung ist aus wenn
Error auf 0 steht	
338 Print "Kanalstörungen ausgeschaltet"	
339 End If	
340 If A = 88 Then	'falls "X" (88) geschickt
wurde	
341 Rs = 1	'Fehlerkorrektur ist an wenn
Rs auf 1 steht	
342 Print "Fehlerkorrektur eingeschaltet"	
343 End If	
344 If A = 89 Then	'falls "Y" (89) geschickt
wurde	
345 Rs = 0	'Fehlerkorrektur ist an wenn
Rs auf 1 steht	
346 Print "Fehlerkorrektur ausgeschaltet"	
347 End If	
348 Stop Timer0	
349 Rscom1 = 0	'da letztes Kommandobyte
empfangen wurde wird Timer0 abgestellt	
350 End If	'Zählvariable Rscom1 wird
auf 0 zurück gesetzt werden um nächstes Empfangen zu ermöglichen	
351 End If	
352 If Rscom1 > 1 And Rscom1 < 20 Then	'falls wir ein Byte aus dem
Datenpaket erwarten	
353 Rscom1 = Rscom1 + 1	'Rscom1 Zähler um eins erhöhen
354 Index = Rscom1 - 2	'mit Rscom1 Zähler Position

```

des Comarrays bilden auf dem Byte gespeichert werden soll
355 Comarray(index) = A 'Empfangenes Byte in Comarray
    speichern
356 Goto Subende1
357 End If
358 Subende1: 'Label/Sprung mit goto
    Subende1 aufrufbar
359 Return
360 Return 'am Ende jedes Interrupts
    Return-Befehl zweimal nötig
361
362
363 Modem_int1: 'Modem Interrupte Routine
364 Modemflag = 1 'Modemflag anzeigen lassen,
    dass Modem Interrupt gesendet hat
365 Return
366 Return 'am Ende jedes Interrupts
    Return-Befehl zweimal nötig
367
368
369 '-----
370 'Timer (Timer0 für Com1, Timer1 für Beacon)
371 '-----
372
373 Tim0_isr: 'Timer0 Subroutine
374 Timer0flag = 1 'Timer0flag anzeigen lassen,
    dass Timer abgelaufen ist
375 Return
376 Return 'am Ende jeder Timers
    Return-Befehl zweimal nötig
377
378
379 Tim1_isr: 'Timer1 Subroutine
380 Beacon = Beacon + 1 'Beacon zählt wie oft Timer1
    abgelaufen ist
381 Return
382 Return 'am Ende jeder Timers
    Return-Befehl zweimal nötig
383
384
385
386
387
388
389
390
391
392
393 '-----
394 'RS(15,9) Fehlerkorrekturcode
395 '-----
396
397 Init_rs:
398 Dim Comarray2(62) As Byte
399 Dim Alpha_to(16) As Byte
400 Dim Index_of(16) As Single
401 Dim Gg(7) As Byte
402 Dim Recd(15) As Single
403 Dim Ddata(9) As Byte
404 Dim U As Byte
405 Dim Q As Byte
406 Dim Bb(6) As Byte
407 Dim Elp(48) As Single
408 Dim Dd(8) As Single
409 Dim Ll(8) As Byte
410 Dim U_lu(8) As Single
411 Dim Ss(7) As Byte
412 Dim Count As Byte
413 Dim Syn_error As Byte
414 Dim Root(3) As Byte
415 Dim Lloc(3) As Byte
416 Dim Zz(4) As Single
417 Dim Eerr(15) As Byte
418 Dim Reg(4) As Single
419 Dim Feedback As Byte
420 Dim Bbort As Single
421 Dim Rrx As Single
422 Dim O As Byte
423 Dim P As Byte
424 Dim R As Byte
425 Dim S As Byte
426 Dim T As Byte
427
428 Count = 0

```

```

429 Syn_error = 0
430 Alpha_to(1) = 1
431 Alpha_to(2) = 2
432 Alpha_to(3) = 4
433 Alpha_to(4) = 8
434 Alpha_to(5) = 3
435 Alpha_to(6) = 6
436 Alpha_to(7) = 12
437 Alpha_to(8) = 11
438 Alpha_to(9) = 5
439 Alpha_to(10) = 10
440 Alpha_to(11) = 7
441 Alpha_to(12) = 14
442 Alpha_to(13) = 15
443 Alpha_to(14) = 13
444 Alpha_to(15) = 9
445 Alpha_to(16) = 0
446 Index_of(1) = -1
447 Index_of(2) = 0
448 Index_of(3) = 1
449 Index_of(4) = 4
450 Index_of(5) = 2
451 Index_of(6) = 8
452 Index_of(7) = 5
453 Index_of(8) = 10
454 Index_of(9) = 3
455 Index_of(10) = 14
456 Index_of(11) = 9
457 Index_of(12) = 7
458 Index_of(13) = 6
459 Index_of(14) = 13
460 Index_of(15) = 11
461 Index_of(16) = 12
462 Gg(1) = 6
463 Gg(2) = 9
464 Gg(3) = 6
465 Gg(4) = 4
466 Gg(5) = 14
467 Gg(6) = 10
468 Goto Init_rs_back
469
470
471
472 Transmit2:
473 Reset PORTC.4
474 Checksumme = Crc16(comarray(1) , 18)
475
476 'formatieren zum kodieren
477 For I = 1 To 9
478     Comarray2(i + 6) = Comarray(i)
479     Comarray2(i + 21) = Comarray(i)
480     Comarray2(i + 36) = Comarray(i + 9)
481     Comarray2(i + 51) = Comarray(i + 9)
482     Shift Comarray2(i + 6) , Left , 4
483     Shift Comarray2(i + 6) , Right , 4
484     Shift Comarray2(i + 21) , Right , 4
485     Shift Comarray2(i + 36) , Left , 4
486     Shift Comarray2(i + 36) , Right , 4
487     Shift Comarray2(i + 51) , Right , 4
488 Next
489
490 'Kodierung
491 For K = 0 To 3
492     For I = 1 To 9
493         Rx = 15 * K
494         Rx = Rx + I
495         Rx = Rx + 6
496         Ddata(i) = Comarray2(rx)
497     Next
498     For I = 0 To 5
499         Bb(i + 1) = 0
500     Next
501     For I = 8 To 0 Step -1
502         Index = Ddata(i + 1) Xor Bb(6)
503         Feedback = Index_of(index + 1)
504         If Feedback <> -1 Then
505             For J = 5 To 1 Step -1
506                 If Gg(j + 1) <> -1 Then
507                     Rrx = Gg(j + 1) + Feedback
508                     Rx = Rrx
509                     Rx = Rx Mod 15
510                     O = Bb(j)
511                     P = Alpha_to(rx + 1)
512                     Bb(j + 1) = O Xor P

```

```

513         Else
514             Bb(j + 1) = Bb(j)
515         End If
516         Rrx = Gg(1) + Feedback
517         Rx = Rrx
518         Rx = Rx Mod 15
519     Next
520     Bb(1) = Alpha_to(rx + 1)
521     Else
522         For J = 5 To 1 Step -1
523             Bb(j + 1) = Bb(j)
524         Next
525         Bb(1) = 0
526     End If
527 Next
528 For I = 1 To 6
529     Rx = 15 * K
530     Rx = Rx + I
531     Comarray2(rx) = Bb(i)
532 Next
533 Next
534
535 'Formatieren Zum Senden
536 For I = 1 To 15
537     Shift Comarray2(i + 15) , Left , 4
538     Comarray2(i) = Comarray2(i) + Comarray2(i + 15)
539     Shift Comarray2(i + 45) , Left , 4
540     Comarray2(i + 15) = Comarray2(i + 30) + Comarray2(i + 45)
541 Next
542 Print "Nachricht kodiert und gesendet"
543
544 'Fehler einbauen
545 If Error = 1 Then
546     Rx = Rnd(5)
547     If Rx = 4 Then
548         Rx = 2
549     End If
550     Print Rx;
551     Print " Fehler eingebaut: ";
552     For I = 1 To Rx
553         C1 = Rnd(18) + 1
554         If C1 < 10 Then
555             If Comarray2(c1 + 6) <> 95 Then
556                 Comarray2(c1 + 6) = 95
557             Else
558                 I = I - 1
559             End If
560         Else
561             If Comarray2(c1 + 12) <> 95 Then
562                 Comarray2(c1 + 12) = 95
563             Else
564                 I = I - 1
565             End If
566         End If
567     Next
568     For I = 7 To 15
569         A = Comarray2(i)
570         Printbin A
571     Next
572     For I = 22 To 30
573         A = Comarray2(i)
574         Printbin A
575     Next
576 End If
577
578 Set PORTA.0
579 Waitms 30
580 Printbin #1 , 204 ; 67 ; 68
581 For K = 1 To 30
582     A = Comarray2(k)
583     Printbin #1 , A
584 Next
585 Printbin #1 , Checksumme ; 204
586 Reset PORTA.0
587 Set PORTC.4
588 Goto Anfang
589
590
591
592 Recieve2:
593 Reset PORTC.3
594 Waitms 10
595 C1 = 0
596 Start Watchdog

```

```

597 Do
598     C1 = C1 + 1
599     Rx = Waitkey(#2)
600     Reset Watchdog
601 Loop Until Rx = 67 OR C1 > 5
602 If Rx <> 67 Then
603     Stop Watchdog
604     Set PORTC.3
605     Goto Anfang
606 End If
607 Reset Watchdog
608 Start ADC
609 Rx = Waitkey(#2)
610 Reset Watchdog
611 If Rx <> 68 Then
612     Stop Watchdog
613     Set PORTC.3
614     Goto Anfang
615 End If
616 Reset Watchdog
617 For I = 1 To 32
618     Comarray2(i) = Waitkey(#2)
619     Reset Watchdog
620 Next
621 Stop Watchdog
622 Smeter = Getadc(2)
623 Stop ADC
624
625 Print "Nachricht empfangen, Signalstärke:";
626 Smeter = Smeter + 2
627 Shift Smeter , Right , 2
628 A = Smeter
629 Print A
630 Cs2 = 256 * Comarray2(32)
631 Cs2 = Cs2 + Comarray2(31)
632
633 Print "Nachricht wird dekodiert..."
634 'formatieren zum dekodieren
635 For I = 1 To 15
636     Comarray2(i + 45) = Comarray2(i + 15)
637     Comarray2(i + 30) = Comarray2(i + 15)
638     Comarray2(i + 15) = Comarray2(i)
639     Shift Comarray2(i) , Left , 4
640     Shift Comarray2(i) , Right , 4
641     Shift Comarray2(i + 15) , Right , 4
642     Shift Comarray2(i + 30) , Left , 4
643     Shift Comarray2(i + 30) , Right , 4
644     Shift Comarray2(i + 45) , Right , 4
645 Next
646
647 'Dekodierung
648 For K = 0 To 3
649     For I = 1 To 15
650         Rx = 15 * K
651         Rx = Rx + I
652         Recd(i) = Comarray2(rx)
653     Next
654     For I = 1 To 15
655         Rx = Recd(I) + 1
656         Recd(I) = Index_of(Rx)
657     Next
658     For I = 1 To 6
659         Ss(I + 1) = 0
660         For J = 0 To 14
661             If Recd(J + 1) <> -1 Then
662                 Rx = I * J
663                 Rx = Recd(J + 1) + Rx
664                 Rx = Rx Mod 15
665                 O = Ss(i + 1)
666                 P = Alpha_to(rx + 1)
667                 Ss(i + 1) = O Xor P
668             End If
669         Next
670         If Ss(I + 1) <> 0 Then
671             Syn_error = 1
672         End If
673         Rx = Ss(I + 1)
674         Ss(i + 1) = Index_of(rx + 1)
675     Next
676     If Syn_error <> 0 Then
677         Dd(1) = 0
678         Dd(2) = Ss(2)
679         Elp(1) = 0
680         Elp(7) = 1

```

```

681 For I = 1 To 5
682     Elp(I + 1) = -1
683     Elp(I + 7) = 0
684 Next
685 L1(1) = 0
686 L1(2) = 0
687 U_lu(1) = -1
688 U_lu(2) = 0
689 U = 0
690 Do
691     U = U + 1
692     If Dd(u + 1) = -1 Then
693         L1(u + 2) = L1(u + 1)
694         Rx = L1(U + 1)
695         For I = 0 To Rx
696             C1 = U + 1
697             C1 = C1 * 6
698             C1 = C1 + I
699             C1 = C1 + 1
700             Index = 6 * U
701             Index = Index + I
702             Index = Index + 1
703             Elp(C1) = Elp(Index)
704             Bort = Elp(Index) + 1
705             Elp(index) = Index_of(bort)
706         Next
707     Else
708         Q = U - 1
709         While Dd(Q + 1) = -1 And Q > 0
710             Q = Q - 1
711         Wend
712         If Q > 0 Then
713             J = Q
714             Do
715                 J = J - 1
716                 If Dd(J + 1) <> -1 And U_lu(Q + 1) < U_lu(J + 1) Then
717                     Q = J
718                 End If
719             Loop Until J < 1
720         End If
721         Rx = L1(Q + 1)
722         Rx = Rx + U
723         Rx = Rx - Q
724         If L1(U + 1) > Rx Then
725             L1(U + 2) = L1(U + 1)
726         Else
727             C1 = L1(Q + 1) + U
728             L1(U + 2) = C1 - Q
729         End If
730         For I = 0 To 5
731             Rx = U + 1
732             Rx = 6 * Rx
733             Rx = Rx + I
734             Elp(rx + 1) = 0
735         Next
736         C1 = L1(Q + 1)
737         For I = 0 To C1
738             Rx = 6 * Q
739             Rx = Rx + I
740             Rx = Rx + 1
741             If Elp(Rx) <> -1 Then
742                 Index = U + 1
743                 Index = Index * 6
744                 Index = Index + I
745                 Index = Index + U
746                 Index = Index - Q
747                 Bort = Q * 6
748                 Bort = Bort + I
749                 Bort = Elp(bort + 1)
750                 Bbort = Bort + Dd(u + 1)
751                 Bbort = Bbort + 15
752                 Bort = Bbort - Dd(q + 1)
753                 Bort = Bort Mod 15
754                 Elp(index + 1) = Alpha_to(bort + 1)
755             End If
756         Next
757         C1 = L1(u + 1)
758         For I = 0 To C1
759             Rx = U + 1
760             Rx = Rx * 6
761             Rx = Rx + I
762             Index = U * 6
763             Index = Index + I
764             O = Elp(rx + 1)

```

```

765         P = Elp(index + 1)
766         Bort = O Xor P
767         Elp(rx + 1) = Bort
768         Bort = Elp(index + 1) + 1
769         O = Index_of(bort)
770         Elp(index + 1) = O
771     Next
772 End If
773 Rx = Ll(u + 2)
774 U_lu(U + 2) = U - Rx
775 C1 = U + 2
776 If U < 6 Then
777     If Ss(C1) <> -1 Then
778         Index = Ss(C1) + 1
779         Dd(c1) = Alpha_to(index)
780     Else
781         Dd(C1) = 0
782     End If
783 Rx = Ll(c1)
784 For I = 1 To Rx
785     Index = U + 1
786     Index = Index * 6
787     Index = Index + I
788     If Ss(C1 - I) <> -1 And Elp(Index + 1) <> 0 Then
789         Bort = U + 1
790         Bort = Bort * 6
791         Bort = Bort + I
792         Bort = Elp(bort + 1)
793         O = C1 - I
794         O = Ss(o)
795         P = Index_of(bort + 1)
796         Bort = O + P
797         Bort = Bort Mod 15
798         O = Dd(c1)
799         P = Alpha_to(Bort + 1)
800         Bort = O Xor P
801         Dd(c1) = Bort
802     End If
803 Next
804 Bort = Dd(C1) + 1
805 Dd(c1) = Index_of(bort)
806 End If
807 Loop Until U > 5 OR Ll(u + 2) > 3
808 U = U + 1
809 C1 = U + 1
810 If Ll(c1) < 4 Then
811     For I = 0 To Ll(c1)
812         O = I + 1
813         Rx = 6 * U
814         Rx = Rx + O
815         Index = Elp(Rx) + 1
816         Elp(Rx) = Index_of(Index)
817     Next
818     For I = 1 To Ll(C1)
819         O = I + 1
820         Rx = 6 * U
821         Rx = Rx + O
822         Reg(O) = Elp(Rx)
823     Next
824 Count = 0
825 For I = 1 To 15
826     Q = 1
827     For J = 1 To Ll(C1)
828         P = J + 1
829         If Reg(p) <> -1 Then
830             Rx = Reg(p) + J
831             S = Rx Mod 15
832             Reg(p) = S
833             Index = Reg(P) + 1
834             S = Q
835             Q = S Xor Alpha_to(Index)
836         End If
837     Next
838     If Q = 0 Then
839         O = Count + 1
840         Root(O) = I
841         Lloc(O) = 15 - I
842         Count = O
843     End If
844 Next
845 If Count = Ll(c1) Then
846     For I = 1 To Ll(C1)
847         O = I + 1
848         Index = 6 * U

```

```

849 Index = Index + O
850 If Ss(O) <> -1 And Elp(Index) <> -1 Then
851     A = Ss(O) + 1
852     Bort = Elp(Index) + 1
853     S = Alpha_to(A)
854     T = Alpha_to(bort)
855     Bort = S Xor T
856     Zz(o) = Bort
857 Else
858     If Ss(O) <> -1 And Elp(Index) = -1 Then
859         Bort = Ss(O) + 1
860         Zz(O) = Alpha_to(Bort)
861     Else
862         If Ss(O) = -1 And Elp(Index) <> -1 Then
863             Bort = Elp(Index) + 1
864             Zz(O) = Alpha_to(Bort)
865         Else
866             Zz(O) = 0
867         End If
868     End If
869 End If
870 R = I - 1
871 For J = 1 To R
872     P = J + 1
873     Rx = 6 * U
874     Rx = Rx + O
875     Rx = Rx - J
876     If Ss(P) <> -1 And Elp(Rx) <> -1 Then
877         Bort = Elp(Rx) + Ss(P)
878         Bort = Bort Mod 15
879         S = Bort + 1
880         Bort = Alpha_to(S)
881         S = Zz(O)
882         T = S Xor Bort
883         Zz(o) = T
884     End If
885 Next
886 Bort = Zz(O) + 1
887 Zz(o) = Index_of(bort)
888 Next
889 For I = 0 To 14
890     O = I + 1
891     Eerr(O) = 0
892     If Recd(O) <> -1 Then
893         Bort = Recd(O) + 1
894         Recd(O) = Alpha_to(Bort)
895     Else
896         Recd(O) = 0
897     End If
898 Next
899 Rx = L1(C1) - 1
900 For I = 0 To Rx
901     O = I + 1
902     T = Lloc(O) + 1
903     Eerr(T) = 1
904     For J = 1 To L1(C1)
905         P = J + 1
906         If Zz(P) <> -1 Then
907             Index = Root(O)
908             Index = Index * J
909             Bort = Zz(p)
910             Bort = Bort + Index
911             Index = Bort Mod 15
912             Bort = Lloc(O) + 1
913             S = Index + 1
914             S = Alpha_to(S)
915             R = Eerr(Bort)
916             Eerr(Bort) = R Xor S
917         End If
918     Next
919     If Eerr(T) <> 0 Then
920         Index = Lloc(O) + 1
921         Index = Eerr(Index)
922         S = Index + 1
923         Eerr(T) = Index_of(S)
924         Q = 0
925         For J = 0 To Rx
926             P = J + 1
927             If J <> I Then
928                 Bort = Root(O) + Lloc(P)
929                 S = Bort Mod 15
930                 Bort = S
931                 S = Alpha_to(Bort + 1)
932                 Bort = 1 Xor S

```

```

933         S = Bort + 1
934         Bort = Index_of(S)
935         Q = Q + Bort
936     End If
937 Next
938 Q = Q Mod 15
939 Index = T
940 Index = Errr(Index) + 15
941 P = Index - Q
942 Index = P Mod 15
943 Bort = T
944 P = Index + 1
945 Errr(Bort) = Alpha_to(P)
946 Index = T
947 P = Recd(Bort)
948 S = Errr(Index)
949 Index = P Xor S
950 Recd(bort) = Index
951 End If
952 Next
953 Else
954     Print "zu viele Fehler in Teil ";
955     Bort = K + 1
956     Print Bort
957     For I = 0 To 14
958         O = I + 1
959         If Recd(O) <> -1 Then
960             Index = Recd(O) + 1
961             Recd(O) = Alpha_to(Index)
962         Else
963             Recd(O) = 0
964         End If
965     Next
966 End If
967 Else
968     Print "zu viele Fehler in Teil ";
969     Bort = K + 1
970     Print Bort
971     For I = 0 To 14
972         O = I + 1
973         If Recd(O) <> -1 Then
974             Index = Recd(O) + 1
975             Recd(O) = Alpha_to(Index)
976         Else
977             Recd(O) = 0
978         End If
979     Next
980 End If
981 Else
982     For I = 0 To 14
983         O = I + 1
984         If Recd(O) <> -1 Then
985             Index = Recd(O) + 1
986             Recd(O) = Alpha_to(Index)
987         Else
988             Recd(O) = 0
989         End If
990     Next
991 End If
992 For I = 1 To 15
993     Rx = Recd(i)
994     Index = 15 * K
995     Index = Index + I
996     Comarray2(index) = Rx
997 Next
998 Next
999
1000 'formatieren zum ausgeben
1001 For I = 1 To 9
1002     Shift Comarray2(i + 21) , Left , 4
1003     Comarray(i) = Comarray2(i + 6) + Comarray2(i + 21)
1004     Shift Comarray2(i + 51) , Left , 4
1005     Comarray(i + 9) = Comarray2(i + 36) + Comarray2(i + 51)
1006 Next
1007
1008 For I = 1 To 18
1009     A = Comarray(i)
1010     Printbin A
1011 Next
1012 Checksumme = Crc16(comarray(1) , 18)
1013 If Checksumme = Cs2 Then
1014     Print " - CRC ok";
1015 Else
1016     Print " - CRC falsch!";

```

```
1017 End If
1018 Set PORTC.3
1019 Goto Anfang
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
```