Scenix Semiconductor, Inc.
V.23 Originate Mode Reference Design

Document V. 0.9

June 1, 1999
Chris Fogelklou
Applications Engineer
Scenix Semiconductor
http://www.scenix.com

TABLE OF CONTENTS

INTRODUCTION

The subject of this overview is the Scenix V.23 modem reference design. Scenix Semiconductor, Inc., has completed a design for a V.23 modem, operating in originate mode, to suit the needs of some of its customers. This document describes the operation of each block of hardware, in the circuit schematics, and the operation of most of the software modules.

Although modem standards have advanced enormously since V.23 was first developed in the 1960's, the standard still exists today. Typically, these applications utilizing V.23 do not require the high data rates provided by modern modems, and significant cost savings can be achieved by choosing a lower-speed, lower-cost design.

The goal of the Scenix V.23 reference design is to allow the end user, typically an engineer, the capability to quickly embed the Scenix solution into any design that requires the use of a low-speed modem. Typical applications that would benefit from a low-cost, low-speed modem include credit-card readers, remote monitoring equipment, alarm systems, set-top-boxes, etc.

SPECIFICATIONS

This V.23 reference design is designed to meet these specifications:

Universal Asynchronous Receiver/Transmitter

- 1200 bps
- No Parity
- 8 Data Bits
- 1 Stop Bit
- Hardware Flow Control (CTS, RTS)

Compact AT command set

- 64-byte command buffer
- Dial: "ATDTxxxxxxxxx…"
- Switch from data mode to command mode: "+++"
- Switch from command mode to data mode: "ATO"
- Hang up: "ATH"
- Initialize: "ATZ"
- Hybrid Optimization "ATY"

Dual-Tone Multiple-Frequency Generation for Dialing

- Tones generated: 697Hz, 770Hz, 852Hz, 941Hz, 1209Hz, 1336Hz, 1477Hz,1633Hz, ± 0.5Hz
- On time = 100ms
- Off time = 100ms
- Off-hook delay time before dialing = 4 s
- D/A conversion provided by filtered PPM output

Data transmission and modulation

- FSK transmission data rate at 75bps
- Hardware flow control, 16-byte buffer, and 75bps asynchronous transmitter for data rate conversion from 1200bps to 75bps
- Logic '1' (mark) modulated by 390 Hz
- Logic '0' (space) modulated by 450 Hz
- Transmission power = -15dB
- D/A conversion provided by filtered PPM output

Data reception and demodulation

- FSK reception data rate at 1200bps
- Logic '1' (mark) demodulated from 1300Hz carrier
- Logic '0' (space) demodulated from 2100Hz carrier
- Carrier detection
- Timed-Zero-Cross algorithm
- Carrier Detection

D/A conversion

- Pulse Position Modulation with maximum output frequency of 154kHz

Filtering

- Low pass filter on PPM output
- High pass filter on FSK input

Hybrid

- Four settings provided for automatic hybrid adjustment for various line impedance's
- Hybrid adjusted by outputting signal onto line and measuring fed-back signal with a low-resolution sigma-delta A/D converter

HARDWARE

Hardware Requirements

According to the V.23 Specification, the originating (dialing) modem transmits a data rate of 75bps using a carrier of 450Hz and 390Hz. The answering modem transmits a data rate of 1200bps using a carrier of 2100Hz and 1300Hz. The block diagram in Figure 1 shows the functional blocks required to perform V.23 origination.

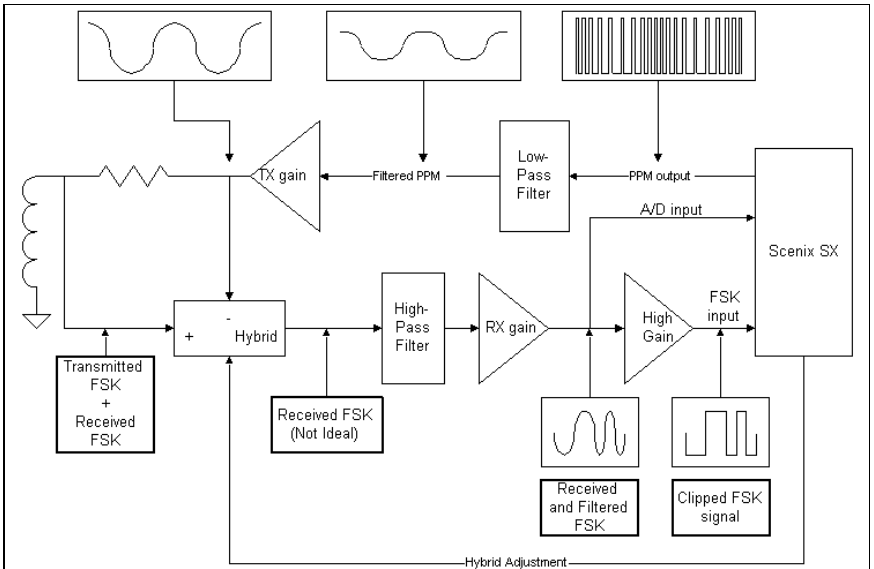


Figure 1. Functional blocks required to perform V.23 origination

A serial connection to a terminal or PC is required to send commands and data to the modem and to receive data and responses from the modem. Figure 2 is a functional block diagram of the circuitry and serial lines to connect.

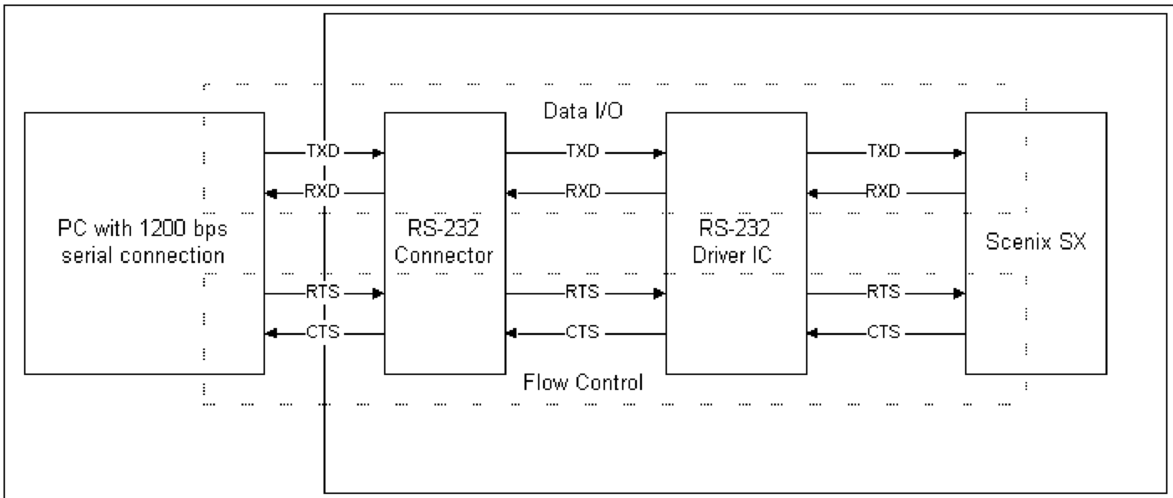


Figure 2: Serial lines required: TXD, RXD, RTS, CTS

The serial connection settings on the terminal or PC should be 1200bps, No Parity, 8 Data Bits, and 1 Stop Bit. Hardware flow control must be enabled, since the V.23 modem's transmit rate is only 75bps, and the modem needs a way to force the PC to stop transmitting characters until the modem's buffer is empty.

HARDWARE

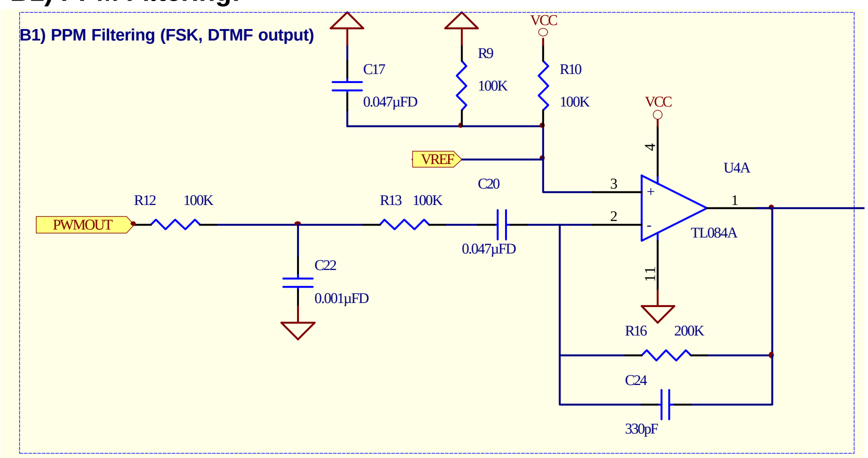
Modem Schematics

See Appendix A for the complete schematics for the Scenix Modem Demonstration board. Since this application note only covers V.23 origination, several of the components on this schematic are not necessary. The relevant parts of this schematic for V.23 origination include the output filtering, the 4 hybrid resistors, the transformer, the portion of the opto-isolator that allows the modem to go on- or off-hook, the high-pass filtering on the input, and the amplification circuitry on the input. Performing only origination eliminates these components:

- R17, R16, C25, R15, C18, D5, for ring detection
- U6A and its surrounding discretes for Caller-ID detection (B5)
- U4B and U4C and their discretes for low-pass filtering FSK in V.23 answer-mode (B3)
- R6, C9, and C10 for DTMF and Call-Progress detection (A2)
- To simplify the circuit even further, the automatic hybrid adjustment may be removed (A2 + B8) and replaced with a fixed resistor value to ground.

Block-by-Block Schematic Descriptions

B1) PPM Filtering:



The filtering on the pulse-position-modulation output of the SX creates the D/A converter for generating FSK and DTMF signals. The cut-off frequency of the filters should be no higher than the highest frequency to generate.

This is a dual stage filter. The cutoff frequency for the first stage is

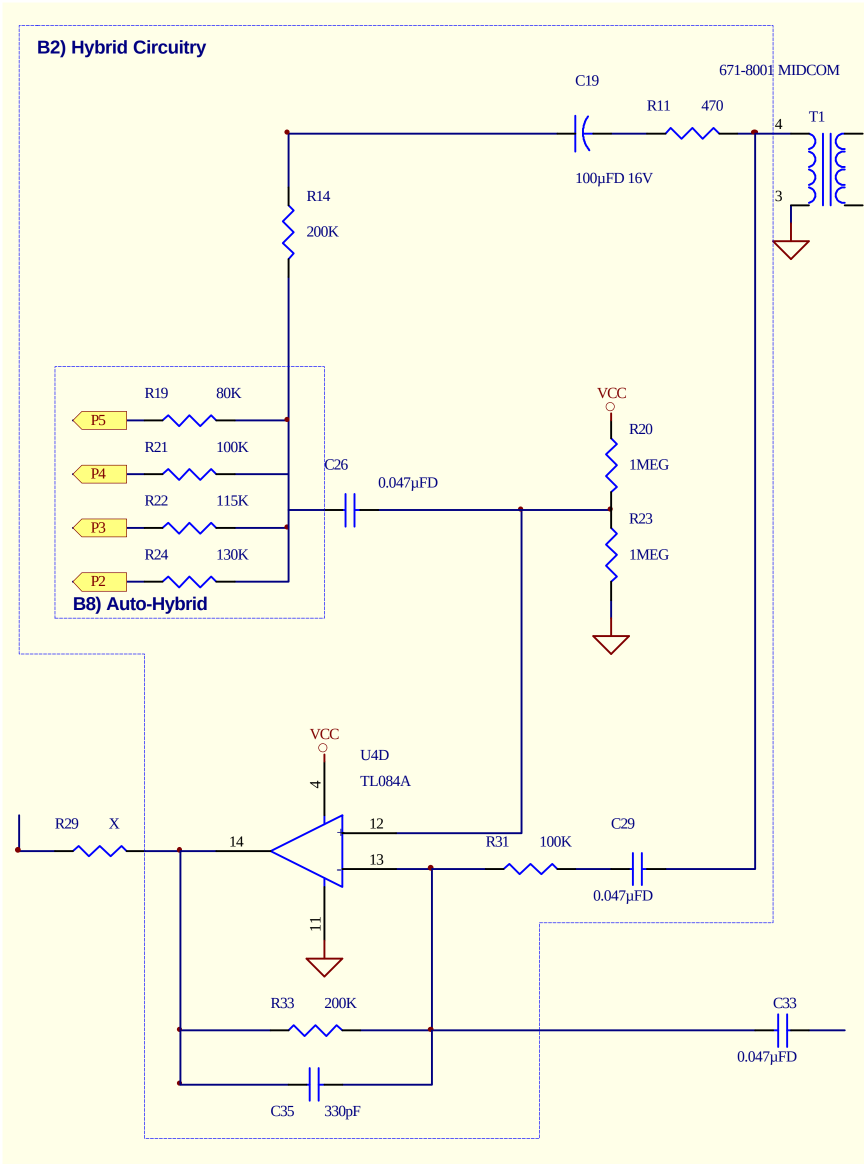
$$Fc = (2 * \pi * ((R12^{-1} + R13^{-1})^{-1}) * C22)^{-1}$$

The cutoff frequency for the second stage is

$$Fc = (2 * \pi * R16 * C24)^{-1}$$

HARDWARE

B2) Hybrid Circuitry

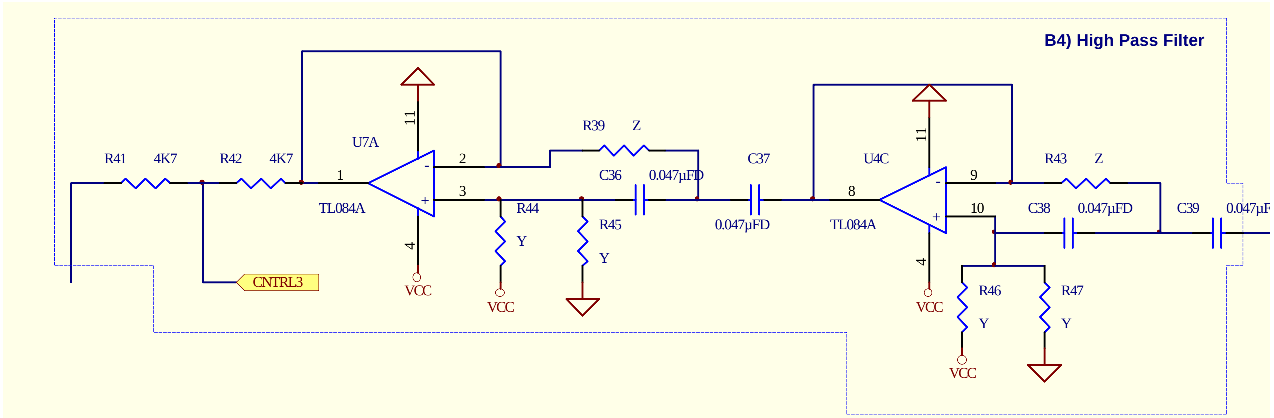


The hybrid circuitry removes some of the transmitted signal from the received signal. R19, R21, R22, and R24 choose the appropriate resistor ratio for the line impedance. To match the hybrid to a specific line impedance, use these resistor values:

R19 = 450 ohms
R21 = 600 ohms
R22 = 750 ohms
R24 = 900 ohms

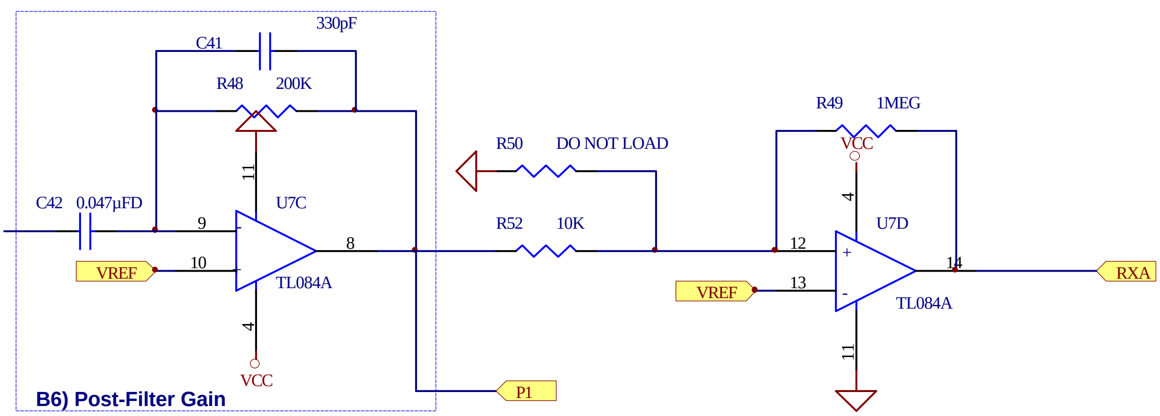
For instance, to match the hybrid to a 450-ohm impedance, set P5 as a 0V output and tristate P4, P3, and P2.

B3 and B4) Input Filtering:



The high-pass filtering removes the remaining low-frequency transmitted signal from the received signal. For V.23 origination, only the high-pass filter is enabled. (Setting CNTRL3 as a tristate pin enables the output of the filter, setting CNTRL3 as an output disables the output of the filter.)
The cut-off frequency for the filter in V.23 originate mode is set to ≅ 1200Hz, as calculated by
$f_c = (2\pi RC)^{-1}$
For this circuitry, these values were chosen:
Y = 5.62kΩ
Z = 2.8kΩ
C36, C37, C38, C39 = 0.047µF
For a calculated cut-off frequency of:
$[2\pi(2.8k\Omega)(0.047\mu F)]^{-1} = 1209Hz$.
Notice that the two 'Y' resistors are double the value of the 'Z' resistors, since they are effectively in parallel and their impedance when combined is ½(Z), producing the same cut-off frequency.

B6 and B7) FSK Amplification:



Since the algorithm for receiving the FSK signal is a zero-cross algorithm, the analog signal is transformed into a digital (+5V and GND) signal by amplifying the received FSK signal into a comparator. The amplification circuitry has a gain of ≅ 20. C41 provides low-pass filtering to eliminate high-frequency noise. R49 adds hysterises to the comparator, reducing the effect of noise on the zero-cross signal. R50 can be used raise the zero-cross point on the input signal, but this is not necessary and is left out of this reference design.

SOFTWARE:

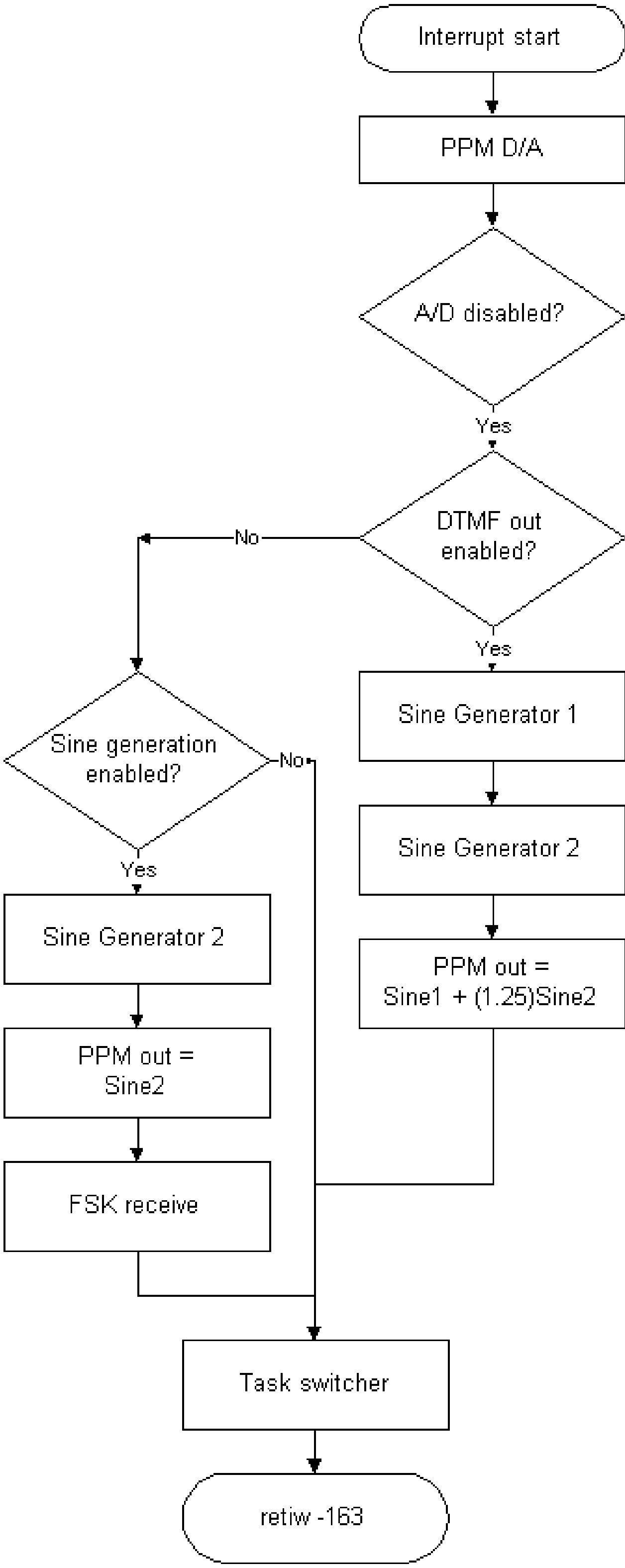
The software of the V.23 origination modem is completed, although a few minor improvements have been suggested:

- Once dialing has been initiated, modem should cancel dialing if the user presses a key.
- Once carrier is detected, modem should delay another few seconds, and check again, to ensure no false carrier was detected.

Aside from these improvements, the modem software has been tested to comply with all of the original specifications. (See the specifications at the beginning of this document)

SOFTWARE

Interrupt Service Routine:



All of the Virtual Peripherals are integrated into the interrupt service routine of the modem. This flowchart shows that the ISR splits off to a different routine if the A/D is enabled. This is because the A/D needs to run at a faster rate to get accurate results. The regular ISR services these tasks:

- DTMF generation
- FSK generation
- 1200bps RS-232 transmitter
- 1200bps RS-232 receiver
- 75bps FSK transmitter
- PPM D/A
- 16-bit Timer

With a retiw value of –163, the interrupt occurs every 3.26µs, giving an interrupt rate of 307kHz. This interrupt rate is exact, and all calculations can be made from this rate. Care should be taken that the ISR does not take longer than 3.26µs, or an entire interrupt period will be missed.

SOFTWARE

Interrupt Service Routine:

Pulse Position Modulation D/A:

This is the assembly code for the PPM D/A, which runs on every pass of the ISR.

```
;*****************************************************************************
PPM_output
               bank    PPM_bank               ; Update the PPM pin
               clc
               add     PPM0_acc,PPM0_out
               snc
               setb    PPM_pin
               sc
               clrb    PPM_pin
;*************************************************************************
```

A simple Pulse Position Modulator is used to perform the Digital to Analog conversion. The resolution of the PPM modulator is 3.26 microseconds, resulting in a maximum output frequency of 154kHz. A low-pass filter with a cutoff of 1.6kHz or greater is used to filter the PPM signal.

On every interrupt, the PPM ISR adds the pwm0_out register to the accumulator register, and the carry flag is moved directly to the PPM pin. A large pwm0_out will cause more frequent carries, producing a larger analog voltage, whereas a small value will produce less frequent carries, producing a smaller analog voltage. An external low-pass filter removes the high-frequency components of the PPM output, producing a steady analog output.

FSK generation and DTMF generation

```
;***************************************************************************
sine_generator1                                     ;(Part of interrupt service routine)
; This routine generates a sine wave with values from the sine table
; at the end of this program.  Frequency is specified by the counter.  To set
; the frequency, put this value into the 16-bit freq_count register:
; freq_count = FREQUENCY * 6.83671552 (@50MHz)
;***************************************************************************
       bank    sine_gen_bank

               clc
               add     freq_acc_low,freq_count_low
               add     freq_acc_high,freq_count_high
               sc
               jmp     :no_change
               inc     sine_index
               mov     w,sine_index
               and     w,#$1f
               call    sine_table
               mov     curr_sine,w           ;1

:no_change
```

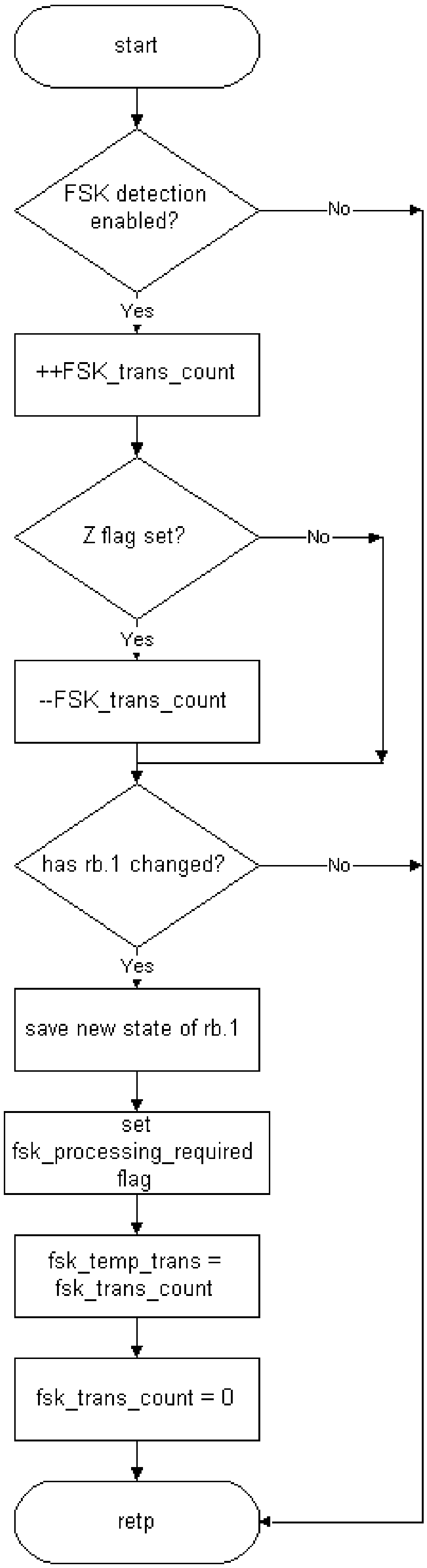
The sine generators use 16-bit phase accumulators, in addition to a table index, to produce a sine wave from a table in the EEPROM of the SX. On each pass of the ISR, the freq_count registers are added to the freq_acc registers, and the table index is incremented if a carry occurs. For very low freq_count values, carries will be less frequent and the index will move through the table at a lower rate. For higher freq_count values, carries will be more frequent and the index will move through the table at a higher rate.

Only one sine generator is used to generate FSK. To generate DTMF, two sine generators are used and their outputs are summed in the PPM_out register.

SOFTWARE

Interrupt Service Routine:

Receiving FSK:



Data reception and demodulation

- FSK reception data rate at 1200bps
- Logic '1' (mark) demodulated from 1300Hz carrier
- Logic '0' (space) demodulated from 2100Hz carrier
- Carrier detection
- Timed-Zero-Cross algorithm
- Carrier Detection

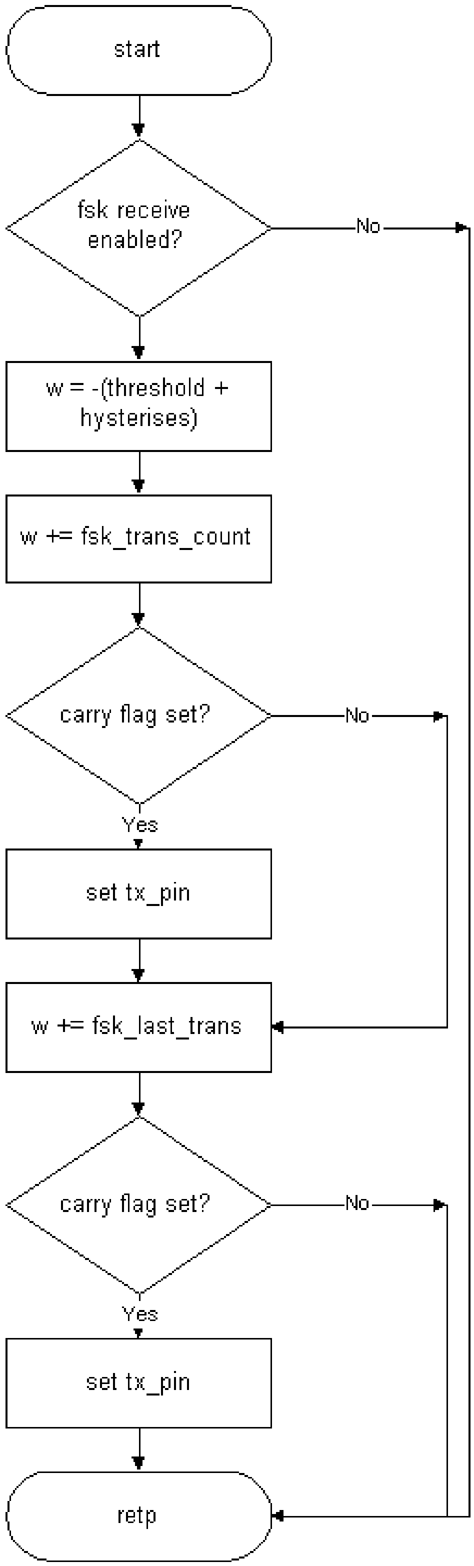
The FSK receive portion of the modem software is performed completely by the Interrupt Service Routine, with no logic required in the mainline routine. FSK receive is enabled via the fsk_rx_en flag in the global flags register. Once enabled, the FSK receive algorithm sets or clears the RS-232 transmit pin, depending on the incoming FSK signal. A timed zero-cross algorithm is used to demodulate the incoming FSK signal. The FSK signal is converted to a square wave by the analog circuitry, and the time between transitions is compared to a threshold. The threshold is raised if a low frequency was just detected, and lowered if a high frequency was just detected. This reduces the effects of jitter caused by noise.

The flowchart to the left is for the main part of the FSK-receive algorithm. It runs on every pass of the ISR as long as FSK reception is enabled. This code counts the time between transitions on rb.1. When a transition is detected, the transition count is saved in a temporary register, and a flag is set that indicates to the processing routines that there is data to process. The transition count is re-started from zero.

SOFTWARE

Interrupt Service Routine:

Receiving FSK:

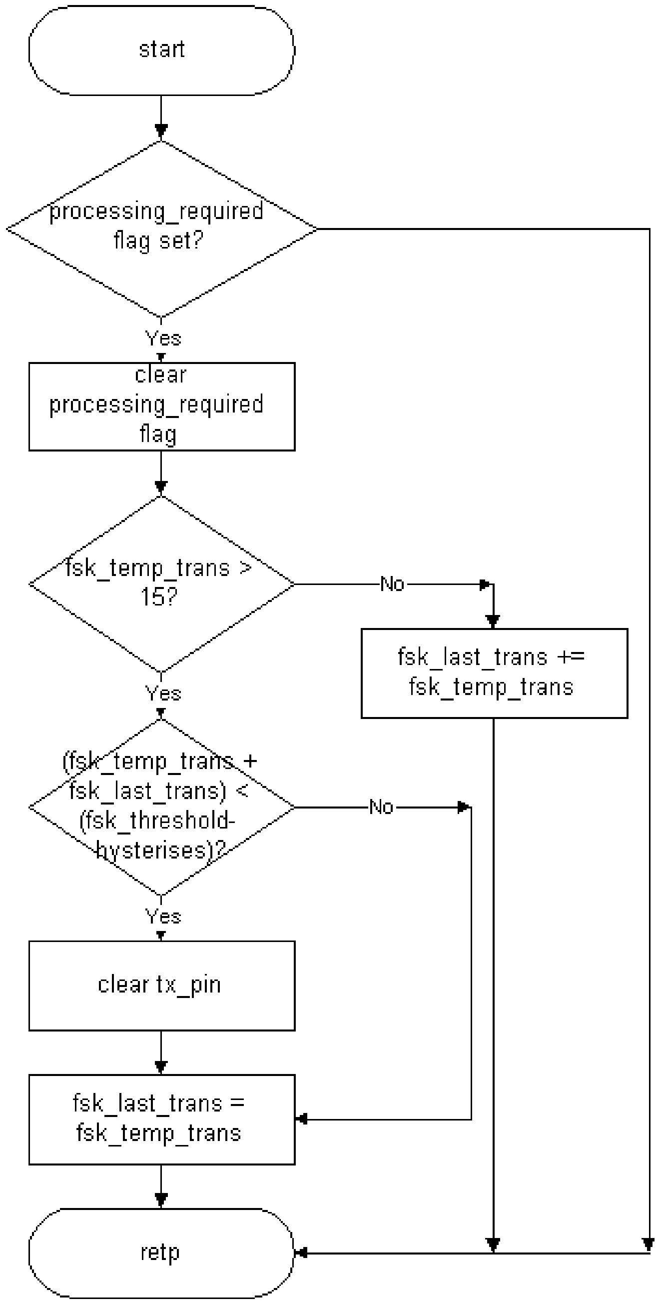


Another part of the FSK receive algorithm is the FSK_receive_main_2 subroutine, which also runs in the ISR. This routine watches for transition counts that are well over the high frequency/low frequency threshold, and automatically sets the RS-232 transmit pin high the instant that this threshold is exceeded. This routine runs in the task switcher.

SOFTWARE

Interrupt Service Routine:

Receiving FSK:



When a transition occurs on the FSK receive pin (rb.1), the fsk_processing_required flag gets set. The fsk_receive_processing routine (left) then checks the latest transition count for a high frequency. If the current transition count, when added to the previous transition count, does not exceed the threshold, then the current input frequency is high and the rs-232 tx_pin is cleared.

SOFTWARE

Interrupt Service Routine:

Multitasking the Interrupt Service Routine:

To save on processing time, a task manager runs any tasks that don't need to run at the full interrupt speed. The task manager runs one task per interrupt from a table. These tasks include the RS-232 transmitters and receivers, the 75bps FSK synchronizer, the 16-bit timers, and some of the FSK detection processing. The task_switcher variable is a global variable used to keep track of the next task to run.

```
;***************************************************************************
task_manager
; This portion of the ISR allows 1 of 16 separate tasks to run in each
; interrupt.
;***************************************************************************
               inc     task_switcher
               mov     w,task_switcher
               and     w,#$0f
               clc
               jmp     pc+w

               ;*** TASKS ***
               jmp     fsk_receive_main_2     ;0
               jmp     transmit               ;1
               jmp     receive                ;2
               jmp     fsk_transmit_uart      ;3
               jmp     fsk_receive_main_2     ;4
               jmp     transmit_fsk           ;5
               jmp     do_timers              ;6
               jmp     fsk_receive_processing1 ;7
               jmp     fsk_receive_main_2     ;8
               jmp     carrier_detect         ;9
               retp                           ;10
               retp                           ;11
               jmp     fsk_receive_main_2     ;12
               retp                           ;13
               retp                           ;14
               retp                           ;15
               jmp     fsk_receive_main_2     ;16
               retp                           ; (just in case)

;***************************************************************************
```

SOFTWARE

Interrupt Service Routine:

Universal Asynchronous Receiver/Transmitter:

- 1200 baud
- No Parity
- 8 Data Bits
- 1 Stop Bit
- Hardware Flow Control (CTS, RTS)

The UART is integrated into the Interrupt Service Routine of the software, which runs every 3.26us. The UART runs on every 16^{th} pass of the ISR, or every 52.16 microseconds. The bit time for a 1200bps UART is 83.33 milliseconds. Dividing 83.33 milliseconds by 52.16us gives a result of 15.97, or 16, an easy divide ratio in a binary system.

```
;*****************************************************************************
transmit
; This is an asynchronous RS-232 transmitter
; INPUTS:
;       tx_divide.baud_bit  -  Transmitter only executes when this bit is = 1
;       tx_high             -  Part of the data to be transmitted
;       tx_low              -  Some more of the data to be transmitted
;       tx_count            -  Counter which counts the number of bits transmitted.
; OUTPUTS:
;       tx_pin              -  Sets/Clears this pin to accomplish the transmission.
;*****************************************************************************
               bank    serial
               clrb    tx_divide.baud_bit      ;clear xmit timing count flag
               inc     tx_divide               ;only execute the transmit routine
               STZ                             ;set zero flag for test
               SNB     tx_divide.baud_bit      ; every 2^baud_bit interrupt
               test    tx_count                ;are we sending?
               snz
               retp                           ;if not, go to :receive
               clc                             ;yes, ready stop bit
               rr      tx_high                 ; and shift to next bit
               rr      tx_low                  ;
               dec     tx_count                ;decrement bit counter
               movb    tx_pin,/tx_low.6        ;output next bit
               retp
;*****************************************************************************
```

```
;*****************************************************************************
receive
; This is an asynchronous receiver for RS-232 reception
; INPUTS:
;       rx_pin             -    Pin which RS-232 is received on.
; OUTPUTS:
;       rx_byte            -    The byte received
;       rx_flag            -    Set when a byte is received.
;*****************************************************************************
               bank    serial
               movb    c,rx_pin                ;get current rx bit
               test    rx_count                ;currently receiving byte?
               jnz     :rxbit                  ;if so, jump ahead
               mov     w,#9                    ;in case start, ready 9 bits
               sc                              ;skip ahead if not start bit
               mov     rx_count,w              ;it is, so renew bit count
               mov     rx_divide,#start_delay  ;ready 1.5 bit periods
:rxbit         djnz    rx_divide,:rxdone       ;middle of next bit?
               setb    rx_divide.baud_bit      ;yes, ready 1 bit period
               dec     rx_count                ;last bit?
               sz                              ;if not
               rr      rx_byte                 ;  then save bit
               snz                             ;if so
               setb    rx_flag                 ;  then set flag
:rxdone
               retp
;*****************************************************************************
```

SOFTWARE

Interrupt Service Routine:
Transmitting FSK:

The timer for the FSK transmitter uses the same timing scheme used by the RS-232 transmitter, but it divides the timers by 16 again to accomplish 75bps transmission.

```
;*****************************************************************************
fsk_transmit_uart
; This is an asynchronous RS-232 transmitter
; INPUTS:
;       tx_divide.baud_bit  -  Transmitter only executes when this bit is = 1
;       tx_high             -  Part of the data to be transmitted
;       tx_low              -  Some more of the data to be transmitted
;       tx_count            -  Counter which counts the number of bits transmitted.
; OUTPUTS:
;       tx_pin              -  Sets/Clears this pin to accomplish the transmission.
;*****************************************************************************
               bank    fsk_serial_bank
               sb      fsk_answering
               inc     fsk_tx_divide_2
               and     fsk_tx_divide_2,#$0f       ; Divide the 1200bps UART by 16 to
                                                  ; achieve 75bps
               sz
               retp
               clrb     fsk_tx_divide.baud_bit      ;clear xmit timing count flag
               inc      fsk_tx_divide               ;only execute the transmit routine
               STZ                                ;set zero flag for test
               SNB      fsk_tx_divide.baud_bit      ; every 2^baud_bit interrupt
               test     fsk_tx_count                ;are we sending?
               snz
               retp                              ;if not, go to :receive
               clc                                ;yes, ready stop bit
               rr       fsk_tx_high                 ; and shift to next bit
               rr       fsk_tx_low                  ;
               dec      fsk_tx_count                ;decrement bit counter
               movb     fsk_tx_bit,/fsk_tx_low.6        ;output next bit
               retp
;*****************************************************************************
```

This routine is tied into the transmit_fsk routine, which loads the sine generator's registers with a high frequency when the fsk_tx_bit is low, and vice-versa for a high fsk_tx_bit.

```
;*****************************************************************************
transmit_fsk
;*****************************************************************************
               bank    fsk_transmit_bank
               sb      fsk_tx_en
               retp
               jb      fsk_answering,transmit_answer_tones
transmit_originate_tones
               jnb     fsk_tx_bit,:low_bit
:high_bit
               bank    sine_gen_bank
               mov     freq_count_high2,#f390_h
               mov     freq_count_low2,#f390_l
               retp
:low_bit
               bank    sine_gen_bank
               mov     freq_count_high2,#f450_h
               mov     freq_count_low2,#f450_l
               retp
transmit_answer_tones
               jnb     fsk_tx_bit,:low_bit
:high_bit
               bank    sine_gen_bank
               mov     freq_count_high2,#f1300_h
               mov     freq_count_low2,#f1300_l
               retp
:low_bit
               bank    sine_gen_bank
               mov     freq_count_high2,#f2100_h
               mov     freq_count_low2,#f2100_l
               retp
```

SOFTWARE

Main Program:

Compact AT command set:

- 64-byte command buffer
- Dial: "ATDTxxxxxxxxx…"
- Switch from data mode to command mode: "+++"
- Switch from command mode to data mode: "ATO"
- Hang up: "ATH"
- Initialize: "ATZ"
- Hybrid Optimization "ATY"

The AT-commands were chosen to provide enough functionality for a very simple modem design. Since the SX originate-only modem can only originate a data call, no answer functions are implemented. Incoming AT-commands are stored in a 64-byte buffer, and compared to software lookup tables on reception of a carriage return.
The AT-Commands are stored in a series of jump tables, like the above. The last table entry is a jump to the routine that handles the command.

```
;***************************************************************************
command_1                                    ; Dial command
               mov     w,pop_index
               add     PC,w
               retw    'A'
               retw    'T'
               retw    'D'
               retw    'T'
               jmp     DIAL_MODE
;***************************************************************************
```

Hybrid:

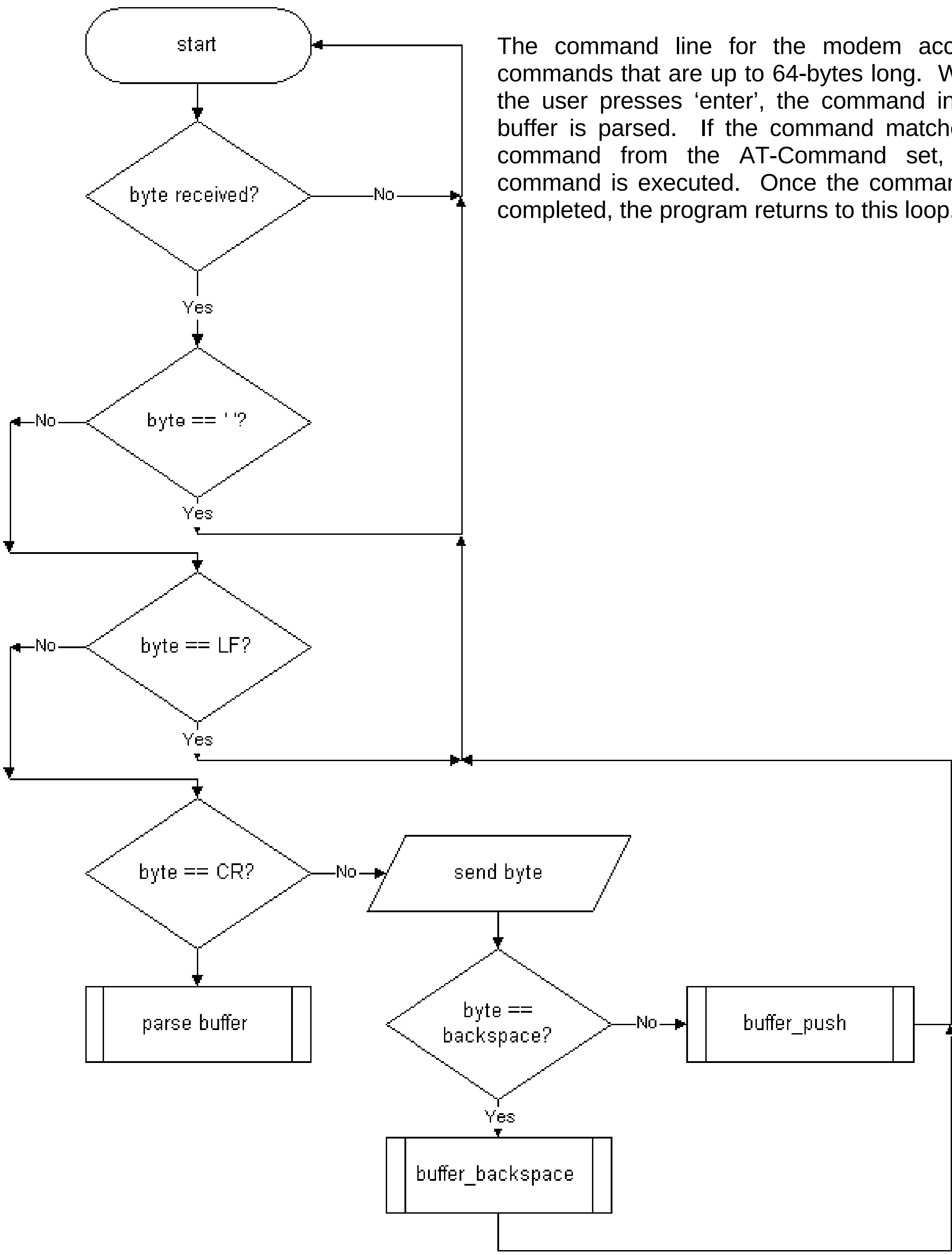
- Four settings provided for automatic hybrid adjustment for various line impedance's
- Hybrid adjusted by outputting signal onto line and measuring fed-back signal with a low-resolution sigma-delta A/D converter

Because of the high attenuation at the filtering stages, it is not necessary for the hybrid to be perfectly matched to the line impedance. Four impedance-match settings are provided by the V.23 reference design. On initialization, the modem outputs a DTMF digit to quiet the line. It then outputs a 2100Hz tone to disable the line equalizers, and measures the amplitude of the signal being fed-back. Each setting is tried, and the setting that produces the most attenuated feedback is saved and used. This allows the SX reference design to be optimized in software for each individual telephone line. The command to optimize the hybrid is "ATY." The optimization process takes about 10 seconds. Optimization needs to be performed each time the modem is powered down, since there is no NVRAM on the board to remember the result of the last optimization.

SOFTWARE

Main Program:

Modem Command Line:

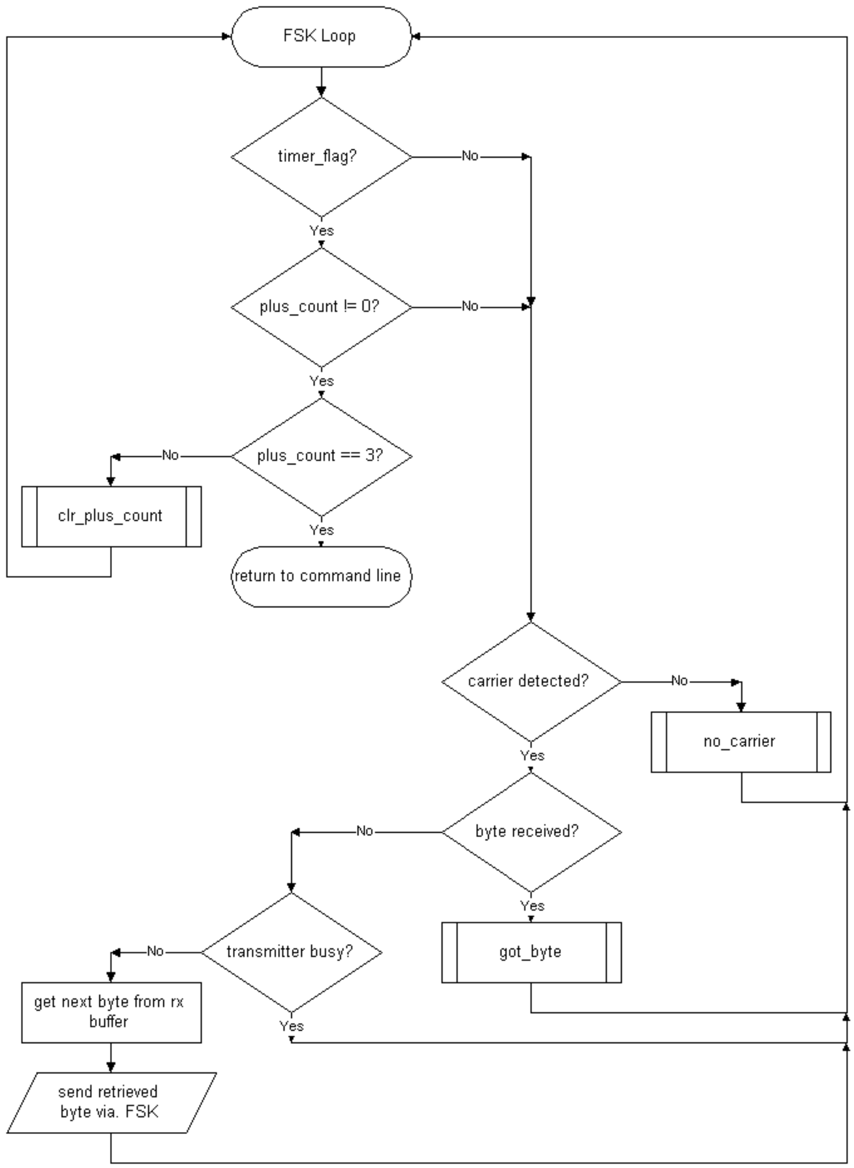


The command line for the modem accepts commands that are up to 64-bytes long. When the user presses 'enter', the command in the buffer is parsed. If the command matches a command from the AT-Command set, this command is executed. Once the command is completed, the program returns to this loop.

SOFTWARE

Main Program:

Main Loop:

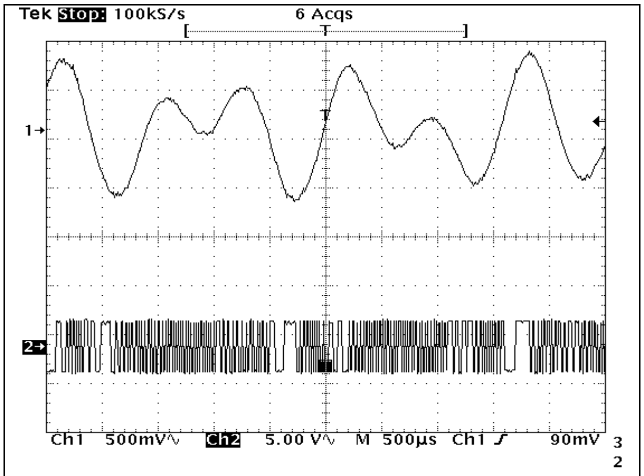


This a flowchart of the operation of the main program loop while the V.23 modem is in it's data mode. In data mode, the modem is taking care of these tasks:

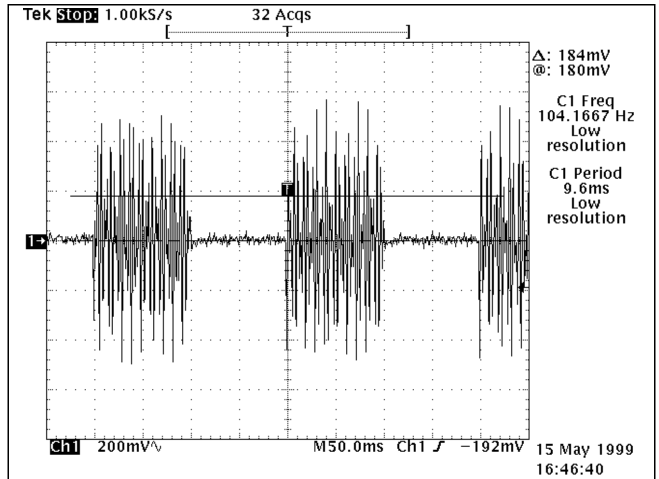
- Receiving RS-232
- Buffering rs-232 characters
- Hardware flow control
- Transmitting FSK
- Receiving the exit string ("+++")
- Timing for the exit string (> 1 second from last data byte before, < 1 second between '+'s, and >1s after final '+' before next received byte)

See the source code for documentation on how each of the subroutines shown here works (e.g. no_carrier, got_byte, etc.…)

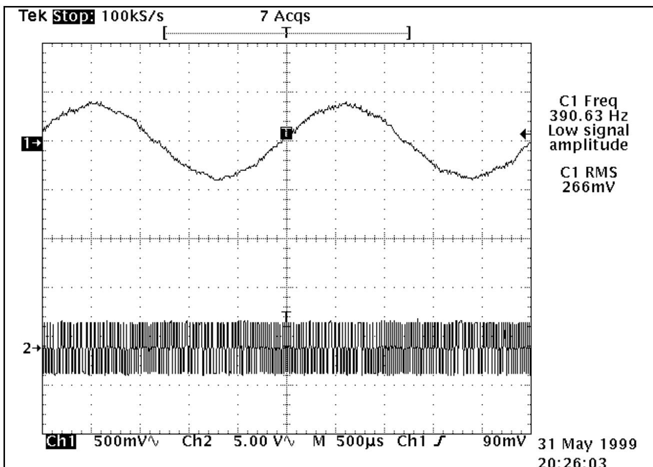
Appendix A: SIGNALS IN THE MODEM



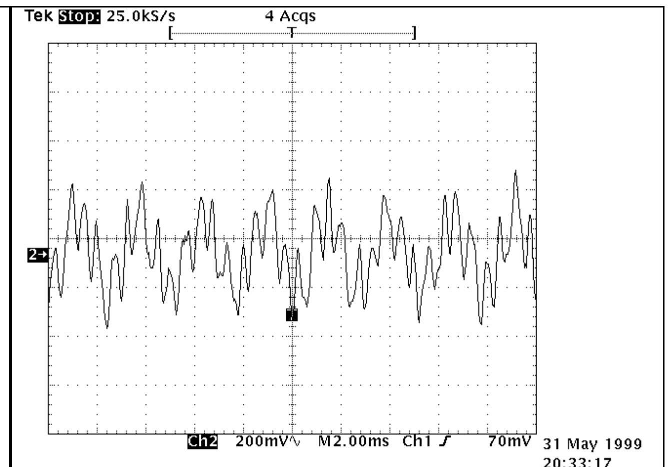
Top:	U4 pin 1	Filtered DTMF output ('1')
Bottom:	U1 pin 6	PPM output



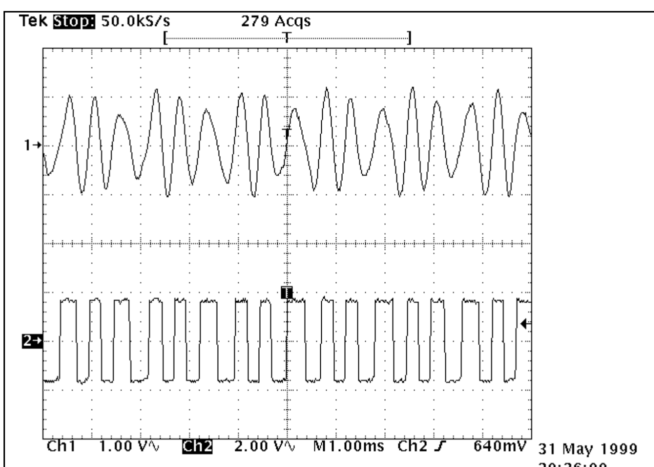
Dial Times: 100ms ON
100ms OFF



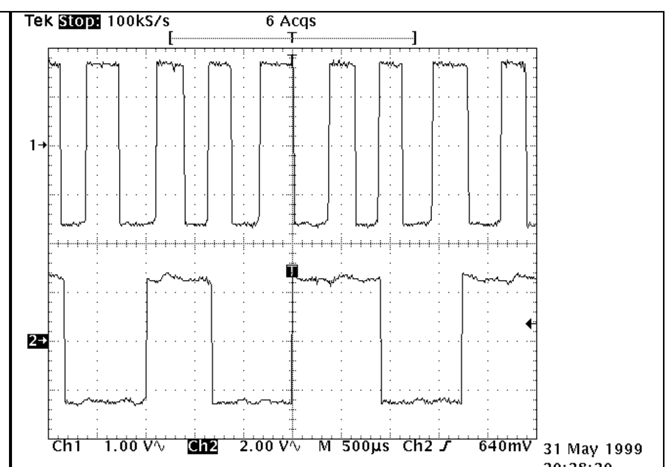
Top:	U4 pin 1	Filtered 390Hz output
Bottom:	U1 pin 6	PPM output



Hybrid Output: U4 pin 14
1300Hz received on 390Hz transmitted.



Top:	U7 pin 8	Filtered FSK input
Bottom:	U7 pin 14	Zero-cross FSK input



Top:	U7 pin 14	Zero-cross FSK input
Bottom:	U1 pin 8	De-modulated data stream

APPENDIX B: CD-R Contents

Quickstart.PDF : Quick guide to running the modem demonstration

read_me.txt : The CD-R's index

max232.pdf : MAX232 RS-232 Driver datasheet

st_tl084cn.pdf : TL084 Op-Amp datasheet

ts117.pdf : TS117 Multifunction Telecom Switch datasheet

sx_datasheet.pdf : SX18AC/SX28AC datasheet

sx28_addendum.pdf : Addendum to sx_datasheet.pdf

SX_User's_Manual.pdf :User's manual for Scenix SX devices

ar40eng.exe : Adobe Acrobat Reader V.4.0

SXKey28L.exe : Parallax Assembler for SX28L devices

V_23_Schematic_2_2.pdf : .PDF's of the modem schematics

V.23 Source Code\ : Folder containing all V.23 source code up to June 1, 1999.

I.D.C\ : Folder containing all files provided to Scenix from I.D.C., including ORCAD schematics, PCB layouts, netlists, Bills of Material, etc. Some component values may differ, but the netlist and layout is the same as the final design.

Protel Stuff\Protel Trial Version\Setup.exe : Trial version of Protel 99 (Also downloadable from Protel's website). Opens all Protel files included in the Protel Stuff directory.

Protel Stuff\Scenix2.DDB : The Scenix version of the V.23 modem schematic. Includes all changes made after I.D.C. handed the design over. This file can be opened with the trial version of Protel 99.

Protel Stuff\Scenix2_Cache : Protel '98 format parts cache

Protel Stuff\Scenix2_Library : Protel '98 format parts library

Protel Stuff\SCHEMATIC1 : Protel '98 format master schematic (links page 1 and page 2 of schematic)

Protel Stuff\Page1_2.sch : Protel '98 format page 1 of schematic

Protel Stuff\Page2_2.sch : Protel '98 format page 2 of schematic

Protel Stuff\SCHEMATIC1_BOM.CSV : Bill of Materials with most up-to-date component values (June 1, 1999)

APPENDIX C: V.23 MODEM DEMO SCHEMATICS

APPENDIX D: v_23_originate_1_35_rev_2_1.src