Project #4: Binary Phase-Shift Keying Modem (Transmitter)

In this project, we will build a crude modem which will allow us to transmit and receive data over the audio cables. While not following a particular CCITT standard, our modem will employ many of the techniques used in real modem design including asynchronous operation, BPSK modulation, frame synchronization, matched filtering, start/stop/parity bits, etc….

Introduction to Modems

Definition: A *symbol* is a group of k bits. The symbol's size is the number of bits in the group.

Definition: For $k = 1$ (symbol size of 1) the system is called *binary*.

Definition: The set of all $M = 2^k$ symbols is called the *symbol set* or *alphabet*.

Example. Let $k = 2$ for a particular group. Then $M = 4$ and the symbol set is given by, {00, 01, 10, 11}. Here two data bits are transmitted with each symbol. Some state-of-the-art modems have $k = 8$, i.e. 8 bits/symbol.

Digital modulation is the process by which digital symbols are transformed into waveforms that are compatible with the characteristics of a channel, such as a telephone line. Typically, the desired information signal modulates a sinusoid called a carrier.

Figure: Digital modulator

A device that can MOdulate and DEModulate is called a modem. In general, modems can be categorized by their modulation type–either amplitude modulation, frequency modulation, and/or phase modulation. In digital communications, these modulation types are also known as amplitude shift keying, frequency shift keying, and phase shift keying where the term "keying" is a remnant of telegraph transmission terminology.

Amplitude Shift Keying (ASK)

The general form of amplitude modulation is

$$s(t) = a_i \cos(2\pi ft + \varphi)$$

where frequency, f and phase, ϕ are fixed and amplitude, a_i has a different value for each symbol being transmitted.

Example. For a simple binary ASK (one bit per symbol), we have $k = 1$ and thus $M = 2$. We can choose $a_0 = 0$ and $a_1 = 1$ as the two amplitude levels leading to the following waveform.

Figure: Binary ASK with $a_0 = 0$ and $a_1 = 1$

If we choose $a_0 = 1/2$ and $a_1 = 1$ as the two amplitude levels we have the following waveform

Figure: Binary ASK with $a_0 = 1/2$ and $a_1 = 1$

If the symbol takes on only one of two possible values (as in the above figures)

$$a_i \in \{a_0, a_1\}$$

then this form of binary ASK (BASK) is known as On Off Keying (OOK).

Frequency Shift Keying (FSK)

The general form of frequency modulation is

$$s(t) = a\cos(2\pi f_i t + \varphi)$$

where amplitude, a and phase, ϕ are fixed and frequency, f_i has a different value for each symbol being transmitted.

Example. For a simple binary FSK (one bit per symbol), we have $k = 1$ and thus $M = 2$. We can choose $f_0 = 1/T_s$ and $f_1 = 2/T_s$ as the two frequencies where T_s is the symbol time. This leads to the following waveform.

Figure: Binary FSK with $f_0 = 1/T_s$ and $f_1 = 2/T_s$

Phase Shift Keying (PSK)

The general form of phase modulation is

$$s(t) = a\cos(2\pi f t + \varphi_i)$$

where amplitude, a and frequency, f are fixed and phase, ϕ_i has a different value for each symbol being transmitted.

Example. For a simple binary PSK (one bit per symbol), we have $k = 1$ and thus $M = 2$. We can choose $\phi_0 = 0$ and $\phi_1 = \pi$ radians as the two phase values leading to the following waveform. This modulation is known as BPSK.

Figure: Binary PSK with $\phi_0 = 0$ and $\phi_1 = \pi$

Definition: The *baud rate* is defined as the number of transmitted symbols per second.

Definition: The *data rate*, R_b in bits per second (bps) is related to the baud rate by

$$R_b = \text{(baud rate)(symbol size)}.$$

Note that if the symbol size is 1 (binary system) then the baud rate is the same as the data rate in bps. Unfortunately, as larger symbol sets have evolved, some sources still refer to the data rate as the baud rate which is incorrect.

Example: The Bell Type 103 modem was one of the earliest modems in the marketplace. It consisted of a FSK modulator/demodulator transmitting 300 symbols per second (300 baud) with 1 bit per symbol for 300bps. The standard consisted of

1070Hz (‘0’ or ‘space’) and 1270Hz (‘1’ or ‘mark’) for the originate modem
2025Hz (‘0’ or ‘space’) and 2225Hz (‘1’ or ‘mark’) for the answer modem.

Since the tones are in the audio band, this is sometimes referred to as Audio FSK or AFSK. Finally, the FSK modem is sometimes referred to as an FM modem.

Combined Modulation Schemes

It is possible to increase the data rate by combining modulation schemes in conjunction with an increase in the symbol set.

Example: The combination of the BASK and BPSK [APK (amplitude phase keying)] is described by

$$s(t) = a_i \cos(2\pi ft + \varphi_j)$$

where frequency, f is fixed and amplitude, a_i and phase, ϕ_j have a different value for each bit being transmitted. For this case a_i and ϕ_j each take on a bit value and so each pair can represent one of four possible two bit combinations

Symbol	Amplitude Scaling	Phase Shift
00	1.0	0
01	1.0	π
10	0.5	0
11	0.5	π

Figure: APK with $a_0 = 1.0$ and $a_1 = 0.5$ and $\phi_0 = 0$ and $\phi_1 = \pi$

In this case our symbol size, $M = 4$. If the baud rate were 2400 symbols/s then the data rate would be 4800bps.

Theorem (Shannon's Information Capacity Theorem): The information capacity [in bits per second (bps)], C of a continuous channel of bandwidth B hertz, perturbed by additive white Gaussian noise of power spectral density (PSD) $N_0 / 2$ (noise power is constant over frequency) and limited in bandwidth to B is given by

$$C = B\log_2\left(1 + \frac{P}{N_0 B}\right)$$

where P is the average transmitted power.

Comments:

1) The above formula implies that for a given average transmitted power, P and channel bandwidth B, we can transmit information at a rate of C bits per second with an arbitrarily small probability of error (given a complex coding scheme). It is not possible to transmit at a rate higher than C bits per second without a definite probability of error. The channel capacity theorem defines the fundamental limit on the rate of error-free transmission for a power-limited, band-limited Gaussian channel. To approach this limit, however, the transmitted signal must have statistical properties approximating those of white Gaussian noise.
2) The available bandwidth of the telephone channel (voice band) is from 300 to 3,400Hz and given typical SNR and transmitted power (regulated by the FCC), today's 56K modems (v.90) are near the limit.
3) New digital communications methods such as the Digital Subscriber Line (DSL) use out-of-band frequencies for higher data rates over twisted pair (TP) but require special terminating hardware at the telephone company's central office and home.

Definition: A simplex connection is a one-way link, i.e. transmissions are made from terminal A to terminal B.

Definition: A half-duplex connection is a link whereby transmission may be made in either direction but not simultaneously.

Definition: A full-duplex connection is a two-way link where transmissions may proceed in both directions

simultaneously.

Definition: An asynchronous modem does not require a global clock between the transmitter and receiver.

In an asynchronous modem, in order to let the receiver know that information is being transmitted, a "start" bit is sent first followed by the data bits and a parity bit (used for error detection/control), and then finally one or two "stop" bits. This group of bits is sometimes called a character. In order to illustrate what the serial bit stream would look like at the input to the originating modem, consider the 7-bit ASCII code for the letter "S" embedded in an 11 bit frame.

Figure (9.3b Radio Shack Telephone Electronics)

Figure (CT waveform)

The parity bit takes on a value of zero or one as needed to insure that the summation (using modulo 2 arithmetic) of all the bits in the data word yields an even or odd result. If the added parity bit is designed to yield an even result, the method is termed *even parity*, and if designed to yield an odd result, it is termed *odd parity*.

At the receiving terminal, the decoding procedure consists of testing that the modulo-2 sum of the data bits yields a zero result (even parity). If the result is found to be one instead of zero, the data word is known to be in error.

Design Specifications for DSP-Based Modulator (Motorola-based Transmitter)

In this project, we will assume the carrier will be a sinusoid with frequency 2400 Hz, digitally synthesized using a 9600Hz sampling rate. This suggests a simple 4-point lookup table with a table increment, $\Delta = 1$:

$$f_{FUND} = \frac{f_s}{L} = 2400$$

We further assume an asynchronous, simplex modem operating at a 300 symbols/s or in this case, 300 bps data rate using a BPSK modulation (this may change for the current EE442/EE592 class). With the assumptions we have the following parameters

4 samples/carrier cycle = 9600 samples/s / 2400 cycles/s
8 cycles/symbol = 2400 cycles/s / 300 bits/s
32 samples/bit = 9600 samples/s / 300 bits/s

Design Specifications for DSP-Based Modulator (TI-based Transmitter)

In this project, we will assume the carrier will be a sinusoid with frequency 2000 Hz, digitally synthesized using a 8000 Hz sampling rate. This suggests the usual 256-point lookup table with a table increment, $\Delta = 64$:

$$f = \frac{f_s}{L}\Delta = \frac{8000}{256}\cdot 64 = 2000$$

or a simple 4-point lookup table with a table increment, $\Delta = 1$:

$$f = \frac{f_s}{L}\Delta$$
$$= \frac{8000}{4}\cdot 1$$
$$= 2000$$

We further assume an asynchronous, simplex modem operating at a 250 symbols/s or in this case, 250 bps data rate using a BPSK modulation (this may change for the current EE442/EE592 class). With the assumptions we have the following parameters

4 samples/carrier cycle = 8000 samples/s / 2000 cycles/s
8 cycles/symbol = 2000 cycles/s / 250 bits/s
32 samples/bit = 8000 samples/s / 250 bits/s

We will assume the data word to be transmitted is a 7-bit American Standard Code for Information Interchange (ASCII) character. Combined with a single start bit of '0', two stop bits of '1 1', and an odd parity bit. The data frame format will look something like (using the ASCII character 'S' = 83 = %1010011)

Figure Data frame with 7-bit ASCII character

ASCII tables and calculators are widely available on the Internet. For example see

`http://www.asciitable.com/`

Transmitter Algorithm

```
MAIN1 (once per sample period; for 5 seconds)
      SYNTHESIZE_WAVEFORM for BIT '1'
      Goto MAIN1

INITIALIZE
      Set DATA_FLAG

MAIN2 (once per sample period)
      If DATA_FLAG
            GET_ASCII_CODE
            CALCULATE_PARITY BIT
            BUILD_FRAME (see Figure)
            Set BIT_FLAG
      End
      If BIT_FLAG
            GET_BIT
      End
      SYNTHESIZE_WAVEFORM for current BIT
      Decrement SAMP_COUNTER
      CHECK_SAMP_COUNTER
      CHECK_BIT_COUNTER
```

```
      Goto MAIN2
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
GET_ASCII_CODE
      Read next ASCII code
      Clear DATA_FLAG
      BIT_COUNTER = 11
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
GET_BIT
      Read next BIT in frame
      Clear BIT_FLAG
      SAMP_COUNTER = SAMPS_PER_BIT
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
SYNTHESIZE_WAVEFORM
      COMPUTE_SINUSOID_SAMPLE (use SINGENID.ASM or lookup_waveIntDelta.c), x
      If (DATA_BIT == '1')
            x = -x (phase change)
      End
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
CHECK_SAMP_COUNTER
      If (SAMP_COUNTER == 0)
            Set BIT_FLAG
            Decrement BIT_COUNTER
      End
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
CHECK BIT COUNTER
      If (BIT_COUNTER == 0)
            Set DATA_FLAG
      End
```

Note: computation of sinusoid samples should maintain continuous phase. Do *not* reset lookup table pointer to base address of the table for each bit. You must maintain coherency.

PASS.DAT (Motorola-based Transmitter)

In the project4.dat file, we can get ASCII characters into memory as follows.

```
        org x:$00000a
CHARS   dsm   10
   org x:CHARS
      dc    'A','B','C','D','E','F','G','H','I','J'
```

The contents of memory will look like this:

```
      X:CHARS:    %0000 0000 0000 0000 0100 0001
      X:CHARS+1:  %0000 0000 0000 0000 0100 0010
```

and so on.

PASS.DAT (TI-based Transmitter)

In C, we can get ASCII characters into memory as follows.

user_data.h

```
...

extern char characters[];

...
```

initialize_program.c

```
...

char characters[] = {'A', 'B', ‘C’, ‘D’, ‘E’, ‘F’, ‘G’, ‘H’, ‘I’, ‘J’};

...
```

The contents of the array (in binary) will look like this:

```
characters[0]:    %0000 0000 0100 0001
characters[1]:    %0000 0000 0100 0010
```

and so on.