PVP SIDDHARTHA INSTITUTE OF TECHNOLOGY, KANURU, VIJAYAWADA-7

DEPARTMENT OF E.C.E.

DIGITAL COMMUNICATIONS LAB MANUAL

AUTONOMOUS PVP-12

Department of Electronics & Communication engineering
Prasad V. Potluri Siddhartha Institute of Technology
Affiliated to JNTU Kakinada,
Approved by AICTE, New Delhi Accredited By NBA,
ISO9001:2008 Certified Institute
(Sponsored by: Siddhartha Academy of General & Technical Education)
Kanuru, Vijayawada -520007

LIST OF EXPERIMENTS

1. **FREQUENCY SHIFT KEYING**
2. **PHASE SHIFT KEYING**
3. **TIME DIVISION MULTIPLEXING & DEMULTIPLEXING**
4. **DIFFERENTIAL PHASE SHIFT KEYING**
5. **PULSE CODE MODULATION & DEMODULATION**
6. **DELTA MODULATION & DEMODULATION**
7. **PHASE SHIFT KEYING MODULATION USING MATLAB**
8. **FREQUENCY SHIFT KEYING MODULATION USING MATLAB**
9. **DIRECT SEQUENCE SPRED SPECTRUM USING MATLAB**
10. **IMPLEMENTATION OF SHANNON FANO CODING USING MATLAB**
11. **IMPLEMENTATION OF HUFFMAN CODING ALGORITHM USING MATLAB**
12. **IMPLEMENTATION OF CYCLIC CODE ENCODER USING MATLAB**
13. **IMPLEMENTATION OF CONVOLUTIONAL CODE ENCODER USING MATLAB**
14. **COMPANDING USING MATLAB**

FSK MODULATION & DEMODULATION

Aim: To study the Modulation and Demodulation Techniques of FSK

Equipment Required:

1. DCS Kit
2. Power Supply
3. 20 MHz CRO
4. CRO Probes
5. Patch cords

Theory:

In this type of modulation, the modulated output shifts between tow frequencies for all the 'one' to 'zero' transitions.

Let the carrier frequencies be represented by w1 and w2. Thus, we have,

$M(t) = A(t)Cosw_1t$ If data is 'One'

$A(t)Cosw_2t$ If data is 'Zero'

here A (t) = Time varying amplitude of the sine wave

M (t) = Modulated carrier

FSK Demodulator employs PLL logic for the recovery of data. FSK describes the modulation of a carrier (or two carriers) by using a different frequency for 1 or 0. The resultant modulated signal may be regarded as the sum of two amplitude modulated signals of different carrier frequency.

$$s(t) = f_1(t)\sin(2\pi fc_1t + \emptyset) + f_2(t)\sin(2\pi fc_2t + \emptyset)$$

Procedure for DCL-01 Kit:

- Ensure that the group **4 (GP4)** clock is selected in the Clock Generation section. Selection is done with the help of switch S1 and observes the corresponding LED indication.
- Observe the transmitter clock of frequency **250 KHz** at **TXCLK** post.
- Set the data pattern using switch S4 as per the given block diagram.
- Observe the 8-bit data pattern at **S DATA** post.
- Observe the carrier sine wave of frequencies **500 KHz** at **SIN1** post and **1MHz** at **SIN3** post in the carrier section.
- Connect the **SIN1** post to the **IN3** post and **SIN3** post to the **IN2** post of the Carrier modulator section.
- Connect **SDATA** to **IN16** post and **TXCLK** to **CLK2** post of the Encoded Data section.
- Select **NRZ-L** data with the help of the switch S3 and observe the corresponding LED indication in the Encoded Data section.
- Connect **OUT10** post of the Encoded Data section to **IN4** post as a control input for the carrier modulator section.
- Observe the FSK modulated signal at the **OUT2** post of the Carrier modulator section.
- For the demodulation of the FSK modulated data, connect the **OUT2** post of the carrier modulator to the **IN28** post of the FSK demodulator section.
- Observe the FSK demodulated data at **OUT24** post of the FSK demodulator section.
- Verify the recovered data with the **S DATA.**

Observations:

- **INPUT NRZ-L DATA AT IN 4**
- Carrier frequency **SIN1** and **SIN3**

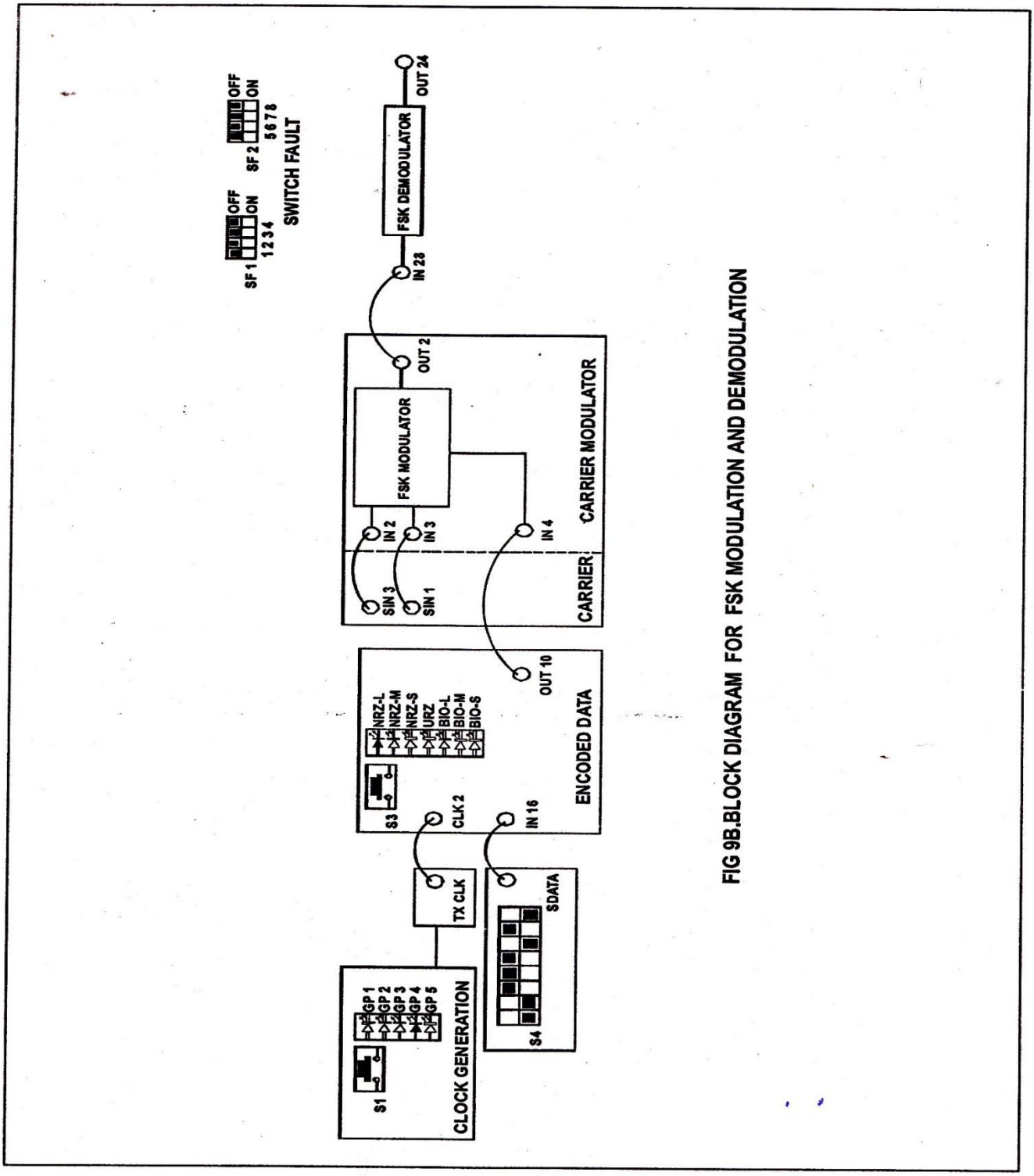


FIG 9B.BLOCK DIAGRAM FOR FSK MODULATION AND DEMODULATION

- FSK modulated signal at **OUT 2**
- **FSK DEMODULATED SIGNAL AT OUT 24**
- Observe output of **PHASE DETECTOR**, **LPF**, and **VCO** on test points provided.

I/P Sin 1 – 2.8Vpp, 526 KHz
I/P Sin 2 – 3.4Vpp, 1.06 MHz

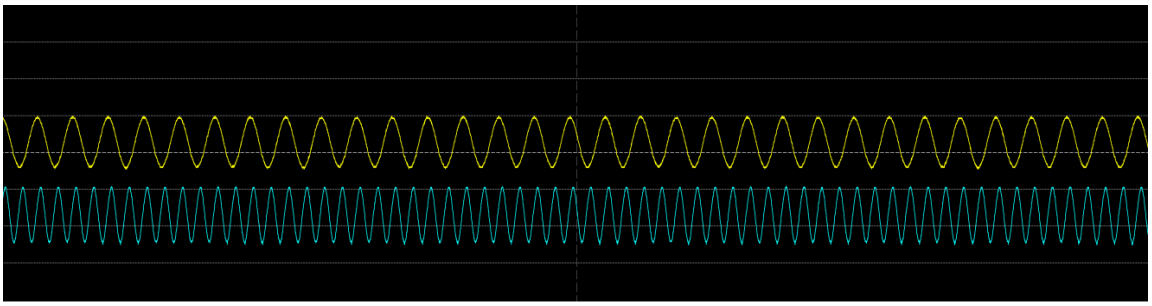


Figure 1: Input Signals

Data I/P – 9.4Vpp, 32.9 KHz

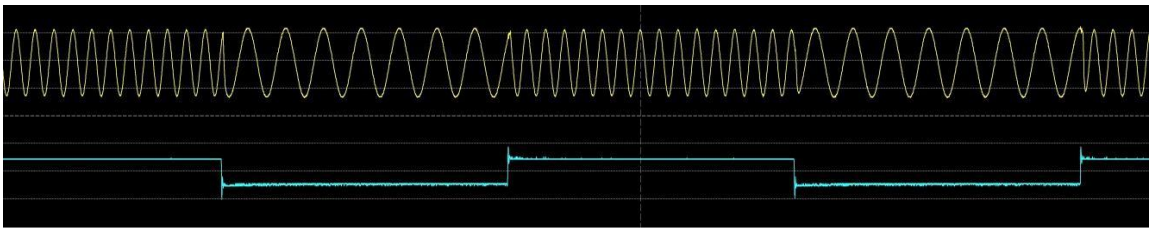


Figure 2: I/P data & Modulated o/p

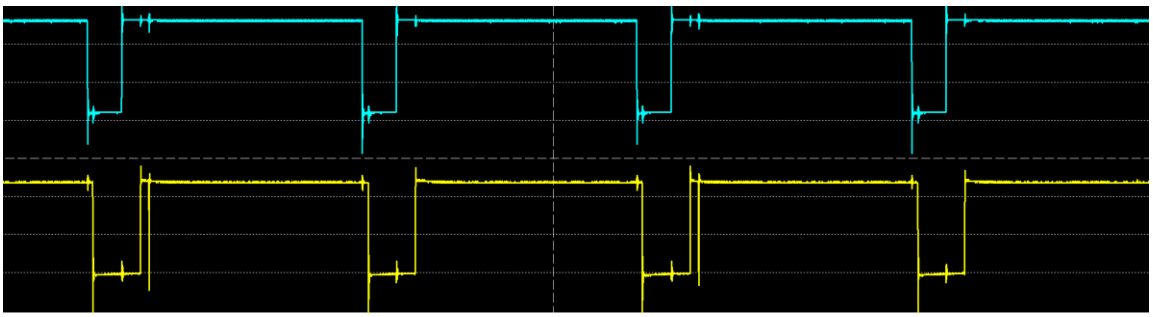


Figure 3: Input data & Recovered O/P

FREQUENCY SHIFT KEYING MODULATION & DEMODULATION MODEL WAVEFORMS

-----*****-----*****-----*****-----*****-----*****-----

Procedure for DCL-06 Kit:

- Refer to the connecting diagram and carry out the following connections and switch settings
- Connect power supply in proper polarity to the kit DCL-06 and switch it ON.
- Connect SERIAL DATA generated on board to CONTROL IN of CARRIER MODULATOR.

- Select FSK modulation using switch S1, the FSK LED will glow.
- Observe the waveforms at SINE1, SINE 2 and MOD OUT.
- Connect FSK modulated signal MODOUT to the FSK IN of the FSK DEMODULATOR.
- Observe various waveforms as mentioned below.

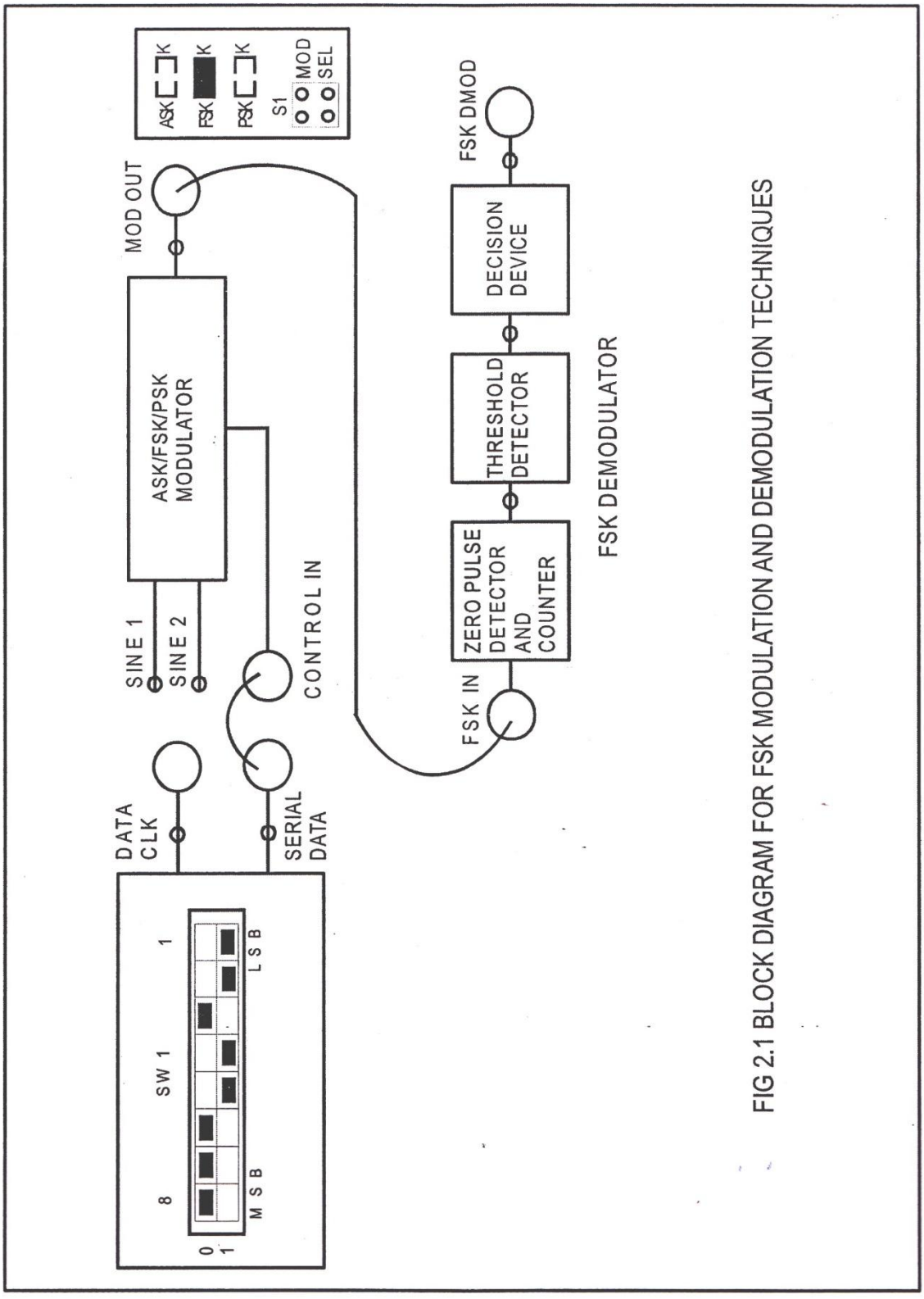


FIG 2.1 BLOCK DIAGRAM FOR FSK MODULATION AND DEMODULATION TECHNIQUES

Observations:

Observe the following signal on the oscilloscope and plot it on the paper.

- **SERIAL DATA with respect to DATACLK**
- **SINE 1 (1.024MHZ) and SINE 2 (512KHz)**
- **FSK modulated signal MODOUT with respect to CONTROL IN.**
- **Output of Zero Pulse Detector at its test point with respect to FSK IN**
- **Output of Threshold Detector at its test point with respect to FSK IN**
- **FSK DMOD with respect to CONTROL IN.**

Sin-1: 3Vpp, 513 KHz
Sine-2: 1.9Vpp, 1MHz

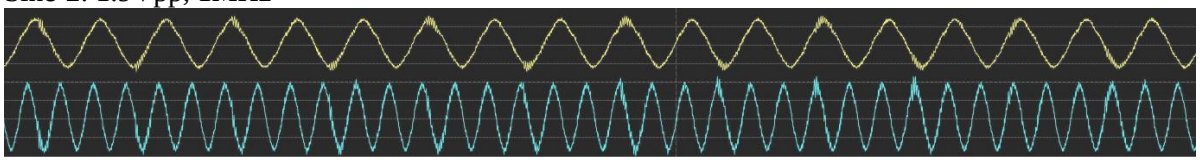


Figure 1: Sin I/P's

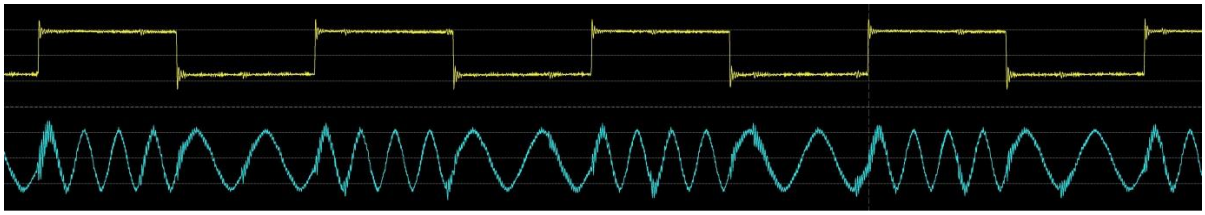


Figure 2: Data & Modulation O/P

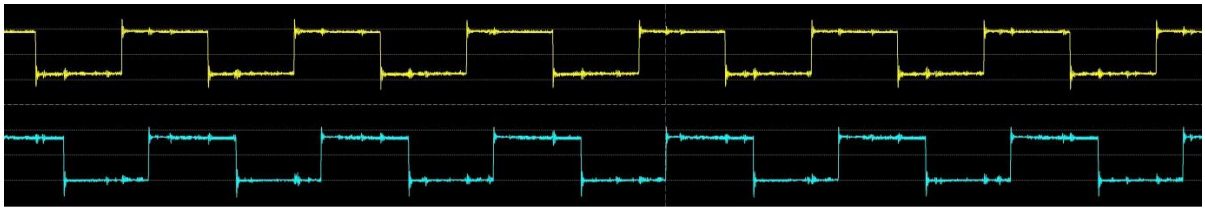


Figure 3: Data & Demodulated O/P

FSK Waveforms for DCL-06 Kit

Conclusion: A small phase lag exists between the modulating data and the recovered data because of the limitation of the tracking ability and the time response of PLL.

PSK MODULATION & DEMODULATION

Aim: To study the Modulation and Demodulation Techniques of PSK

Equipment Required:

1. DCS Kit
2. Power Supply
3. 20 MHz CRO
4. CRO Probes
5. Patch cords

Theory: In the PSK modulation or phase shift keying, for all the 'one' to 'zero' transitions of the modulating data, the modulated output switches between the in phase and out of phase components of the modulating frequency. If the modulated carrier is represented by,

$M(t)$ = $A(t)Cos(wt + phase)$

Where $A(t)$ = Time varying amplitude

wt = Time varying angle

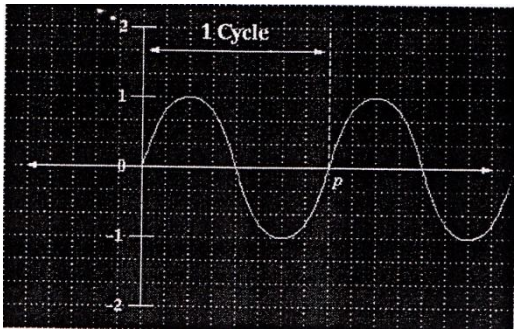
$M(t)$ = Modulated carrier

PSK describes the modulation technique that alters the phase of the carrier. Mathematically,

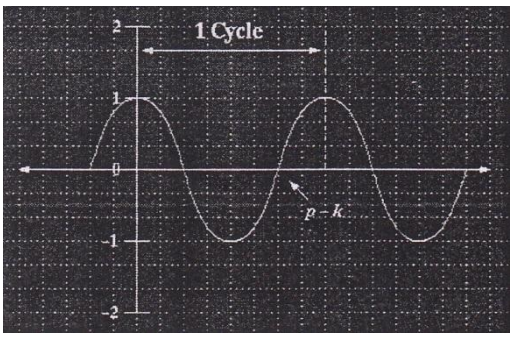
$$s(t) = \sin\left(2\pi f_c + \emptyset(t)\right)$$

1. In angle modulation, the phase of the carrier is discretely varied in relation either to a reference phase or to the phase of the immediately preceding signal element, in accordance with data being transmitted.
2. In a communication system, the representation of characters, such as bits or quaternary digits, is done by a shift in the phase of an electromagnetic carrier wave with respect to a reference, by an amount corresponding to the symbol being encoded. Note1 : For example, while encoding bits, the phase shift could be$0^0$for encoding a "0", and 180^0 for encoding a "1", or the phase shift could be -90^0 for "0" and $+90^0$ for a "1", thus making the representations for "0" and "1" a total of 180^0 apart. Note 2: PSK systems are designed so that the carrier can assume only two different phase angles, each change of phase carries one bit of information, i.e., the bit rate equals the modulation rate. If the number of recognizable phase angles is increased to 4, then 2 bits of information can be encoded into each signal element; likewise, 8 phase angles can encode 3 bits in each signal element.

Phase shift keying is a technique which shifts the period of a wave. This wave 1 has a period of p, noted above. Also notice that the start of the wave's period is at 0.



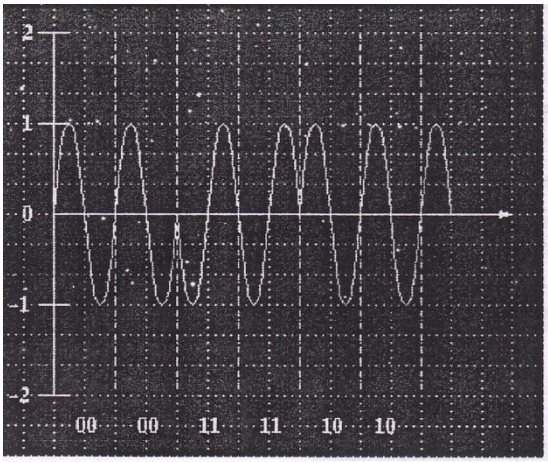
WAVE-1



WAVE-2

Wave 2 is the same wave as the first, but its phase has been **shifted**. Notice that the period starts at the wave's highest point (1). So what's the point? Such a behavior is seen because we have shifted this wave by **one quarter** of the wave's full period. We can shift it to another quarter, if we want to, so the original wave would be shifted by **half** its period. And, we could do it one more time, so that it would be shifted **three quarters** of its original period.

This means that, we have 4 separate waves. So why not let each wave stand for some binary value? Since there are 4 waves, we can let each wave signify 2 bits (00, 01, 11).



Bit value	Amount of shift
00	None
01	1/4
10	1/2
11	3/4

This technique of allowing each shift of a wave represent some bit value is **phase shift keying**. But, the real key is to shift each wave relative to the wave that came before it. An example can be seen in the above diagram.

Please note that when binary values were chosen randomly, for each wave, the values shown are incorrect. Thus, the correct pattern should be: 00 01 10 00 10 00.

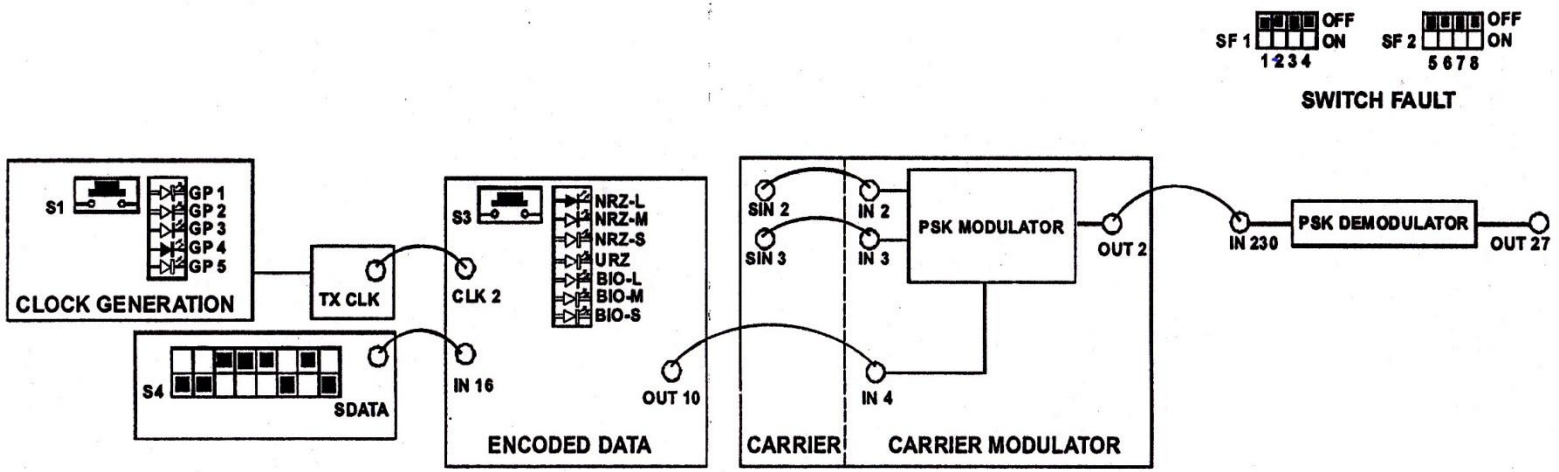


FIG 9C.BLOCK DIAGRAM FOR PSK MODULATION AND DEMODULATION

I/P Sin 1 – 3.4Vpp, 1.04 MHz
I/P Sin 2 – 3.4Vpp, 1.04 MHz

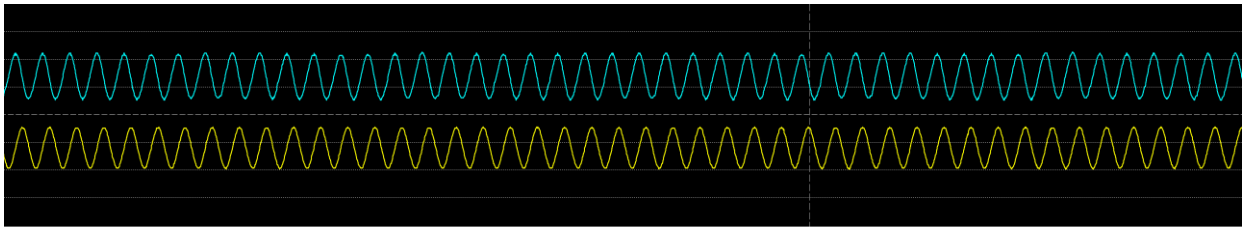


Figure 1: Input Signals

Data I/P – 9.4Vpp, 33.1 KHz

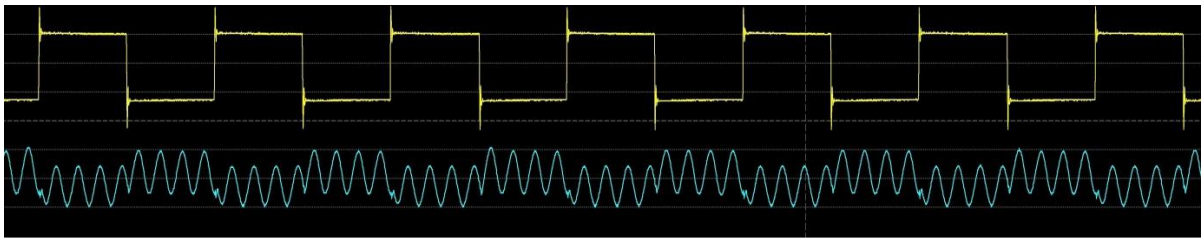


Figure 2: I/P data & Modulated O/P

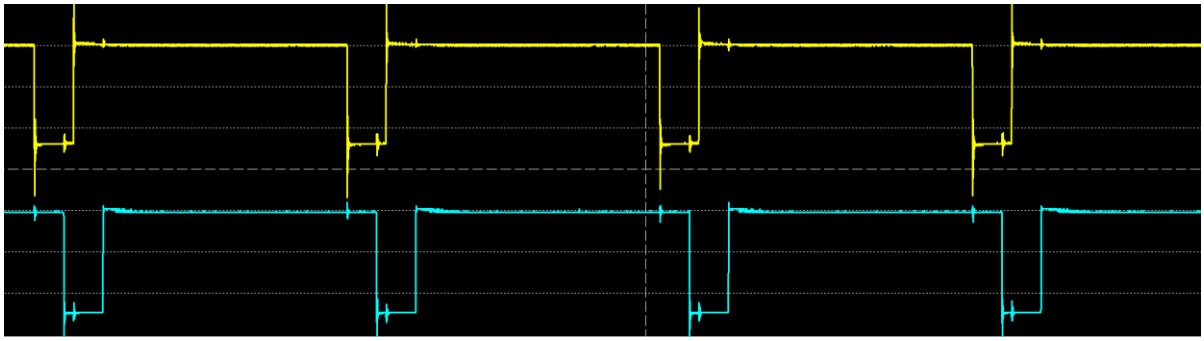


Figure 3: I/P Data & Recovered O/P

PHASE SHIFT KEYING MODULATION & DEMODULATION MODEL WAVEFORMS

Procedure for DCL-01 Kit:

- Ensure that the group **4 (GP4)** clock is selected in the Clock Generation section. Selection is done with the help of switch S1 and observes the corresponding LED indication.
- Observe the transmitter clock of frequency **250 KHz** at **TXCLK** post.
- Set the data patter-using switch S4 as per the given block diagram.

- Observe the 8-bit data pattern at **S DATA** post.
- Observe the carrier sine waves of frequencies **1MHZ** at **SIN2** post and **1MHz** with **180^0** phase at **SIN3** post in the carrier section.
- Connect the **SIN2** post to the **IN2** post and **SIN3** post to the **IN3** post of the Carrier modulator section.
- Connect **SDATA** to **IN16** post and **TXCLK** to **CLK2** post of the Encoded Data section.
- Select **NRZ-L** data with the help of the switch **S3** and observe the corresponding LED indication in the Encoded Data section.
- Connect **OUT10** post of the Encoded Data section to IN4 post as a control input for the carrier modulator section.
- Observe the **PSK** modulated signal at the **OUT2** post of the Carrier modulator section.
- For the demodulation of the **PSK** modulated data, connect the **OUT2** post of the carrier modulator to the **IN30** post of the PSK demodulator section.
- Observe the PSK demodulated data at **OUT27** post of the PSK demodulator section.
- Verify the recovered data with the **S DATA**.

Observations:

- Input **NRZ-L** Data at **IN 4**
- Carrier frequency **SIN 2** and **SIN 3**
- PSK modulated signal at **OUT 2**
- PSK Demodulated signal at **OUT 27**
- Observe the output of the **SINE TO SQUARE CONVERTOR**, **SQUARING LOOP**, and **DIVIDE BY 2** on the test points provided.

-----*****-----*****-----*****-----*****-----

Procedure for DCL-06 Kit:

- Refer to the Figure and carry out the following connections and switch settings.
- Connect power supply in proper polarity to the kit DCL-06 and switch it ON.
- Connect SERIAL DATA generated on board to CONTROL INPUT of CARRIER MODULATOR.
- Select PSK modulation using switch S1, the PSK LED will glow.
- Observe the waveforms at SNE 1, SINE 2 and MOD OUT.
- Connect PSK modulated signal MODOUT to the PSK IN of the PSK DEMODULATOR
- Observe various waveforms as mentioned below.

Observations:

Observe the following signal on the oscilloscope and plot it on the paper.

- SERIAL DATA with respect to DATA CLK
- SINE 1 (1.024 MHz, 0^0) and SINE 2 (1.02MHz, 180^0)
- PSK modulated signal MOD OUT with respect to CONTROL IN.
- MID OUT with respect to PSK IN.
- PSK DMOD with respect to CONTROL IN.

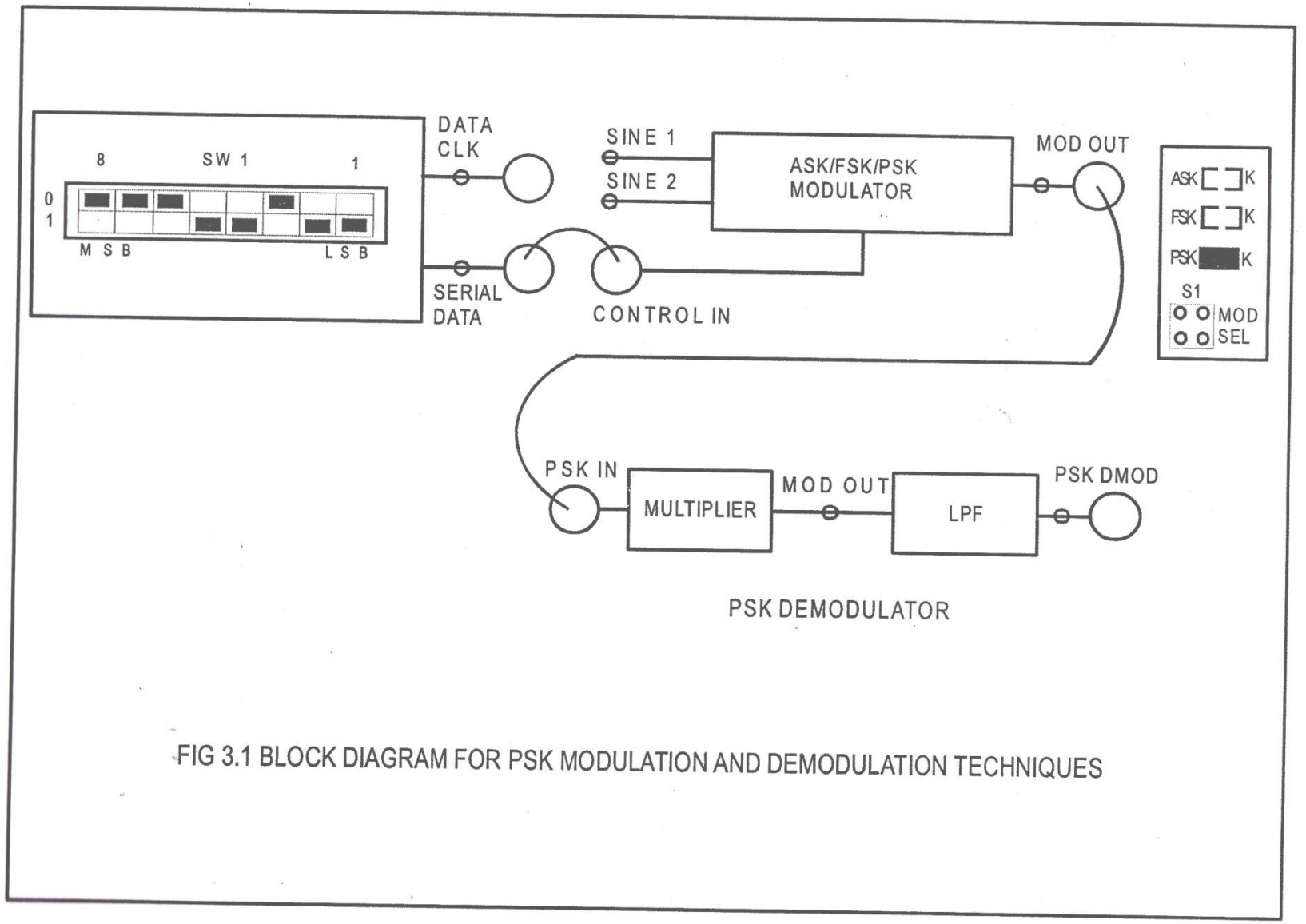


FIG 3.1 BLOCK DIAGRAM FOR PSK MODULATION AND DEMODULATION TECHNIQUES

Sin-1: 3Vpp, 513 KHz
Sine-2: 1.9Vpp, 1MHz

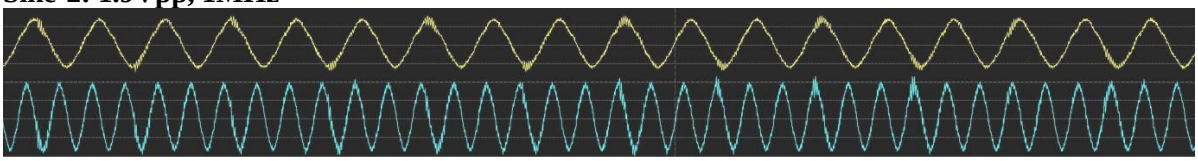


Figure 1: Sin I/P's

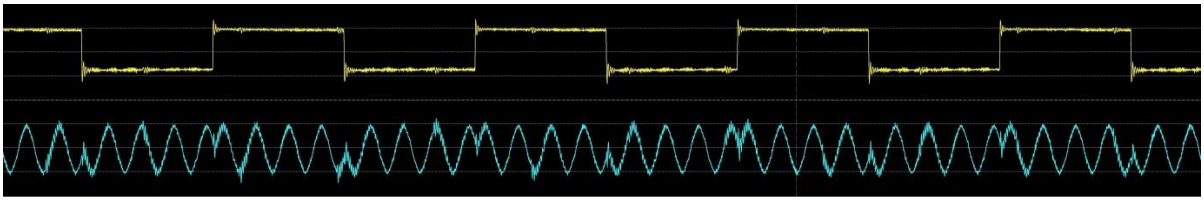


Figure 2: Data & Modulated O/P

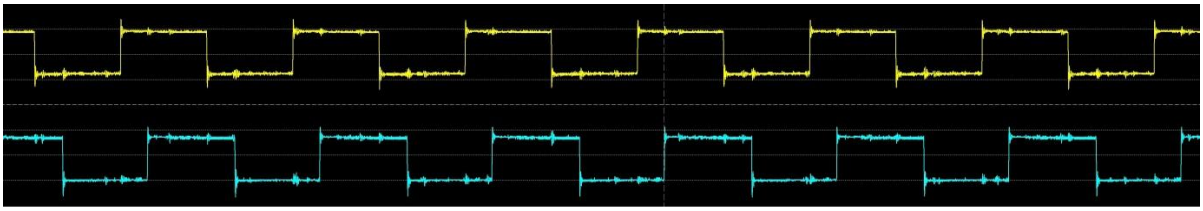


Figure 3: Data & Demodulated O/P

PSK Waveforms for DCL-06 Kit

Conclusion: It is observed that the successful operation of the PSK detector is fully dependent on the phase components of the transmitted modulated carrier. If the phase reversal of the modulated carrier, along with the rising and falling edges of the data is not proper, then the efficient detection of data from PSK modulated carrier becomes impossible.

TIME DIVISION MULTIPLEXING & DEMULTIPLEXING

Aim: To verify the operation of Time Division Multiplexing

Equipment Required:

1. DCS Kit
2. Power Supply
3. 20 MHz CRO
4. CRO Probes
5. Patch cords

Theory: Time-division multiplexing (TCM) is a method of putting multiple data streams in a single signal by separating the signal by separating the signal into many segments, each having a very short duration. Each individual data stream is reassembled at the receiving end based on the timing.
To maintain proper positions of Sample Pulses in the Multiplexer, it is necessary to synchronize the Sampling Process. Because the sampling operations are usually electronic, there is typically a Clock Pulse Train. That serves as a reference for all samples. At the receiving Station, a similar Clock Synchronization can be derived from the received waveforms by observing the Pulse Sequence over many pulses and averaging the pulses (in a closed loop with the Clock derived on the Voltage Controlled Oscillator).
Clock Synchronization does not guarantee that the proper sequence of samples is synchronized. Proper alignment of the Time Slot Sequence requires Frame Synchronization. Hence one or more Time Slots per Frame may be used to send Synchronization Information. For example, by placing a Special Pulse with larger amplitude than the largest expected Message Amplitude in TIME SLOT-1, the start of a Frame can easily be identified using a suitable Threshold Circuit.
In time-division multiplexing (TCM), the transmission between the multiplexers is provided by a single high-speed digital transmission line. Each connection produces a digital information flow, which is then inserted into the high-speed line. For example in Figure 1(a) each connection generates a signal that produces one unit of information every 3T seconds. This unit of information could be a bit, a byte, or a fixed-size block of bits. Typically, the transmission line is organized into frames that in turn are divided into equal-sized slots. For example, in Fig1 (b) the transmission line can send one unit of information every T seconds, and the combined signal has a frame structure that consists of three slots, one for each user. During connection setup, each connection is assigned a slot that can accommodate the information produced by the connection.
TDM was introduced in the telephone network in the early 1960's. The T1 carrier system that carries 24 digital telephone connections is shown in Figure 2.

Procedure:

- Ensure that group **2 (GP2)** clock is selected in the clock generation section. Selection is done with the help of switch S1.
- The selected clock of frequency 32 KHz, 80% duty cycle will be available at the **TXCLK** post.
- The timing signals required for multiplexing four analog channels are derived internally from **TXCLK** and are available at the test points **TP1**, **TP2**, **TP3**, and **TP4**.
- Feed four sine waves of frequency **250 Hz**, **500 Hz**, **1 KHz** and **2 KHz** having amplitude around **1Vpp** from the Function Generator to the four channel inputs **IN7**, **IN8**, **IN9**, and **IN10** of the analog multiplexer.

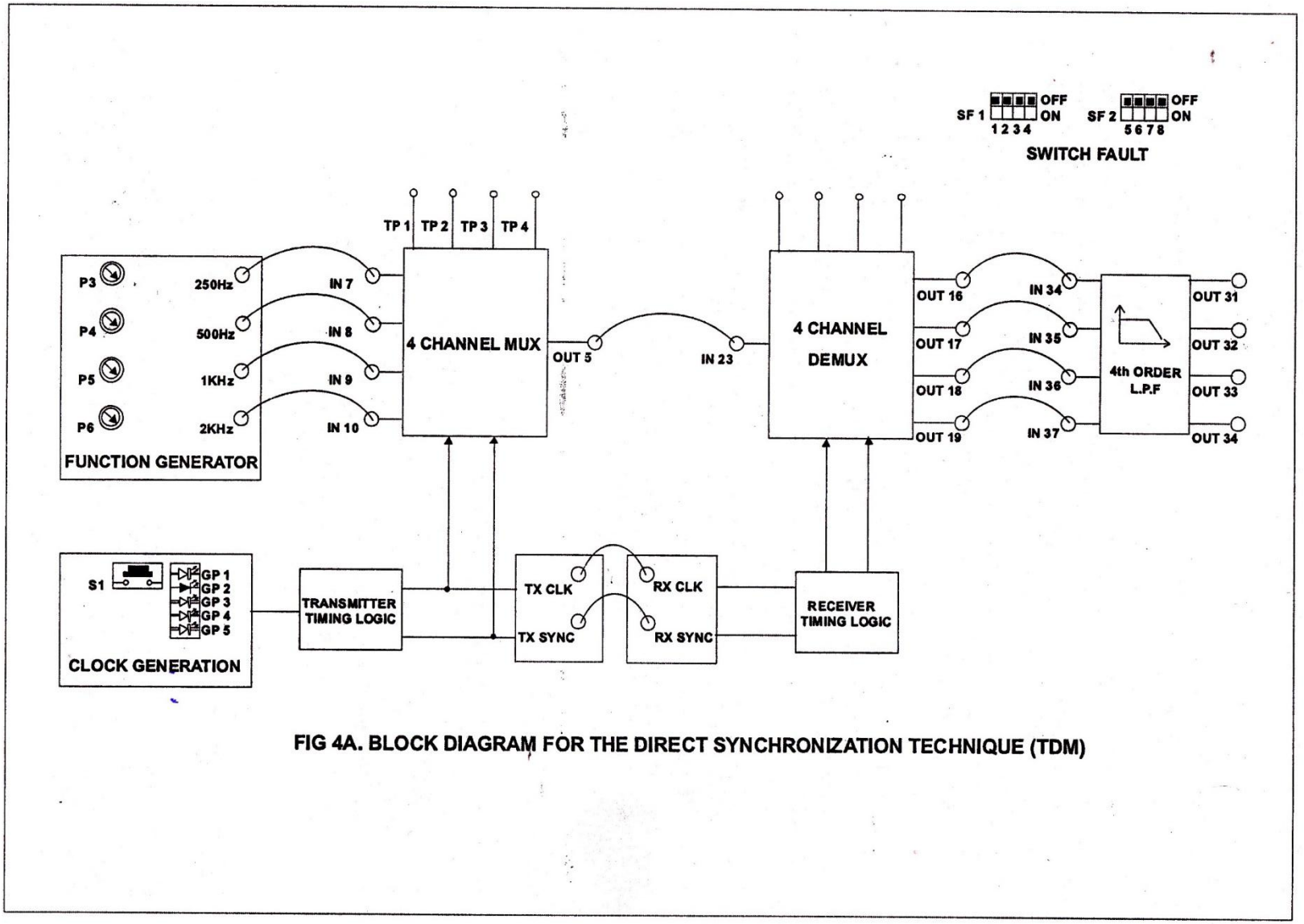


FIG 4A. BLOCK DIAGRAM FOR THE DIRECT SYNCHRONIZATION TECHNIQUE (TDM)

Sin 1: 2.6Vpp, 2 KHz
Sin 2: 1.5Vpp, 1 KHz
Sin3: 3.12Vpp, 500Hz

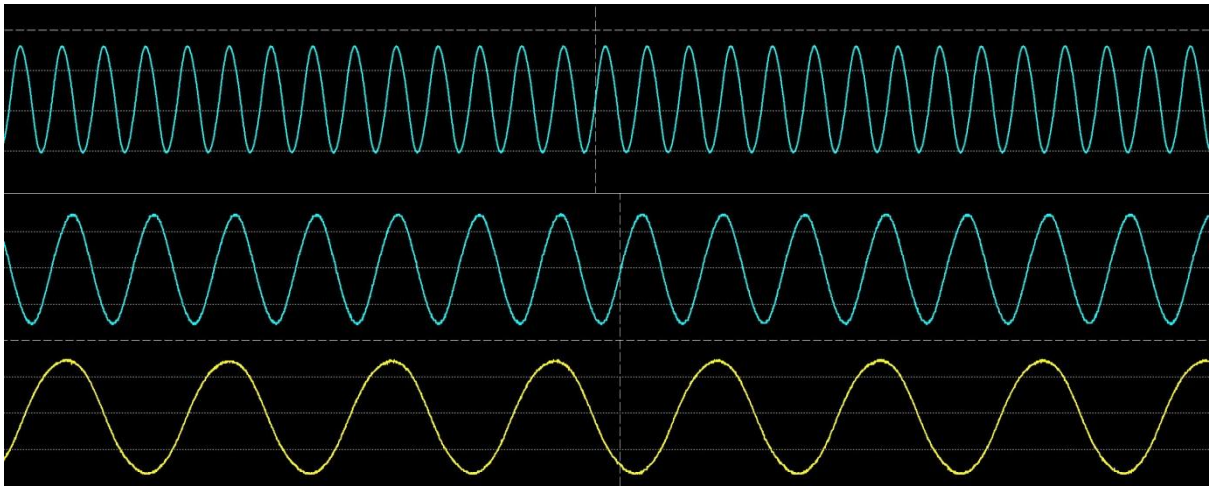


Figure 1: Sin Inputs 1, 2, 3

TP1: 6Vpp, 7.9 KHz
TP2: 8.8Vpp, 7.9 KHz
TP3: 7Vpp, 7.9 KHz

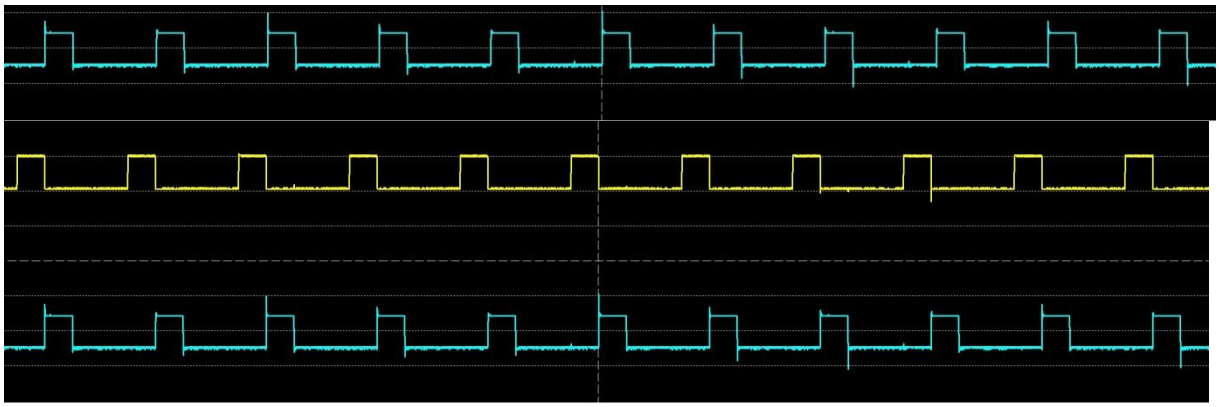


Figure 2: TP1, TP2, TP3

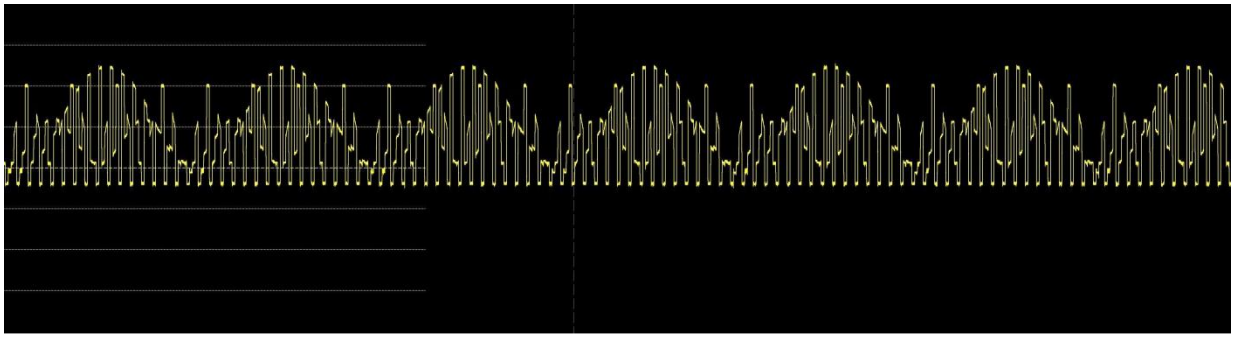


Figure 3: TDM Modulation

Sin 1: 2.6Vpp, 2 KHz
Sin 2: 1.5Vpp, 1 KHz
Sin3: 3.12Vpp, 500Hz

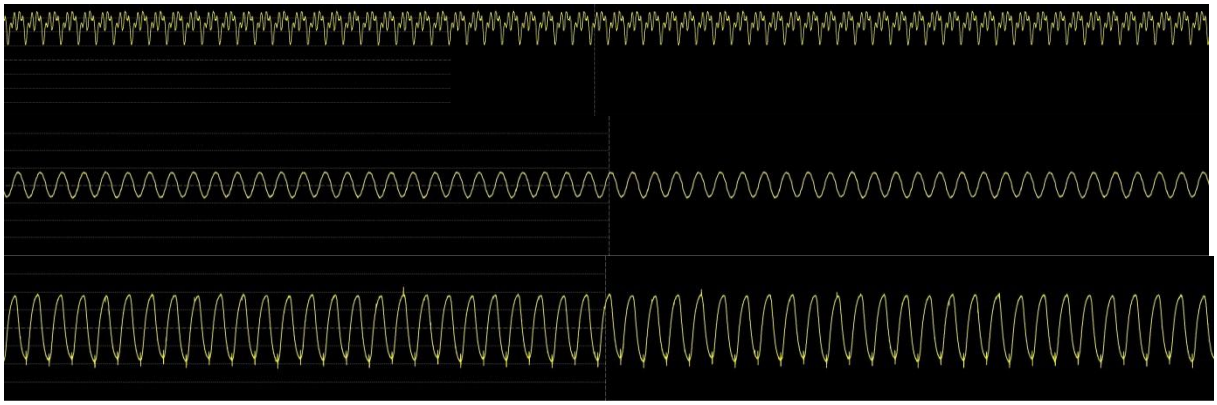


Figure 4: Demodulated O/P for Sin1, Sin2, Sin3

- Observe the **TDM** output at the **OUT5** post of the 4-channel Mux section.
- To recover the individual analog signals from the multiplexed data at the **OUT5** post, connect it to **IN23** post of the 4-channel Demux section.
- To provide the timing and synchronization information to the receiver section, connect the **TX CLK** post to the **RX CLK** post and **TX SYNC** post to the **RX SYNC** post. Here, we are connecting the transmitter clock to the receiver clock and transmitter sync to the receiver sync physically.
- The timing signals required for de-multiplexing the four analog channels are derived internally from the **RX CLK** and are available at the test points.
- The four de-mulitplexed signals are available at the posts **OUT16**, **OUT17**, **OUT18**, and **OUT19** of the 4-Channel Demux section.
- The multiplexed sine waves are not pure sine waves. Therefore it needs to be filtered. Connect **OUT16** post to **IN34** post, **OUT17** post to **IN35** post, **OUT18** post to **IN36** post and **OUT19** to **IN37** post of the 4^{th} order LPF.
- Observe the filtered output at posts **OUT31**, **OUT32**, **OUT33** and **OUT34** respectively.
- Observe the sequence of the recovered signals, which is in exact compliance with that of the transmitter inputs sequence. Similarly, observe the frequencies of the recovered signals.

Observations:

- Input signals **IN7**, **IN8**, **IN9**, and **IN10**
- Channel Selection Signal at **TP1**, **TP2**, **TP3**, **TP4**.
- **TX CLK** and **RX CLK**.
- Channel Identification Signal **TX SYNC** and **RX SYNC**.
- Multiplexer Output **OUT5**
- De-Multiplexer output **OUT16**, **OUT17**, **OUT18**, and **OUT19**.
- Reconstructed signal **OUT31**, **OUT32**, **OUT33**, **OUT34**.

Conclusion: In this experiment, the transmitter clock and the channel identification clock (Sync) are directly linked to the receiver section. Hence, the transmitter and the receiver are synchronized and proper reconstruction of the signal is achieved.

DIFFERENTIAL PHASE SHIFT KEYING

Aim: To study carrier modulation techniques Differential Phase Shift Keying

Equipment Required:

1. ADCL -01 kit
2. Power Supply
3. Patch cords
4. 20 MHz Dual trace CRO
5. CRO Probes.

Theory: In BPSK communication system, the demodulation is made by comparing the instant phase of the BPSK signal to an absolute reference phase locally generated in the receiver. The modulation is called in this case BPSK absolute. The greatest difficulty of these systems lies in the need to keep the phase of the regenerated carrier always constant. This problem is solved with the PSK differential modulation, as the information is not contained in the absolute phase of the modulated carrier but in the phase difference between two next modulation intervals.

Fig.3.2 a & b shows the block diagram of DPSK modulation and demodulation system. The coding is obtained by comparing the output of an EX-OR, delayed of a bit interval, with the current data bits. As total result of operation, the DPSK signal across the output of the modulator contains 180 deg. Phase variation at each data bit "1". The demodulation is made by a normal BPSK demodulator, followed by a decision device supplying a bit "1" each time there is a variation of the logic level across its input.

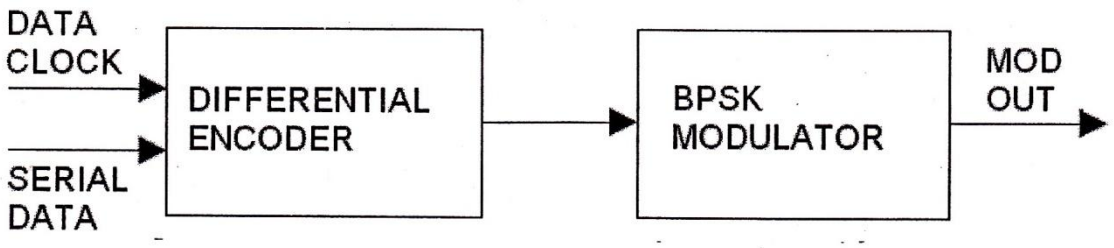


Fig. 3.2 (a) DPSK MODULATOR

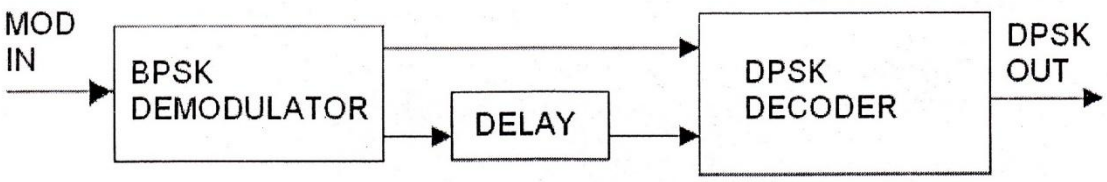


Fig. 3.2 (b) DPSK DEMODULATOR

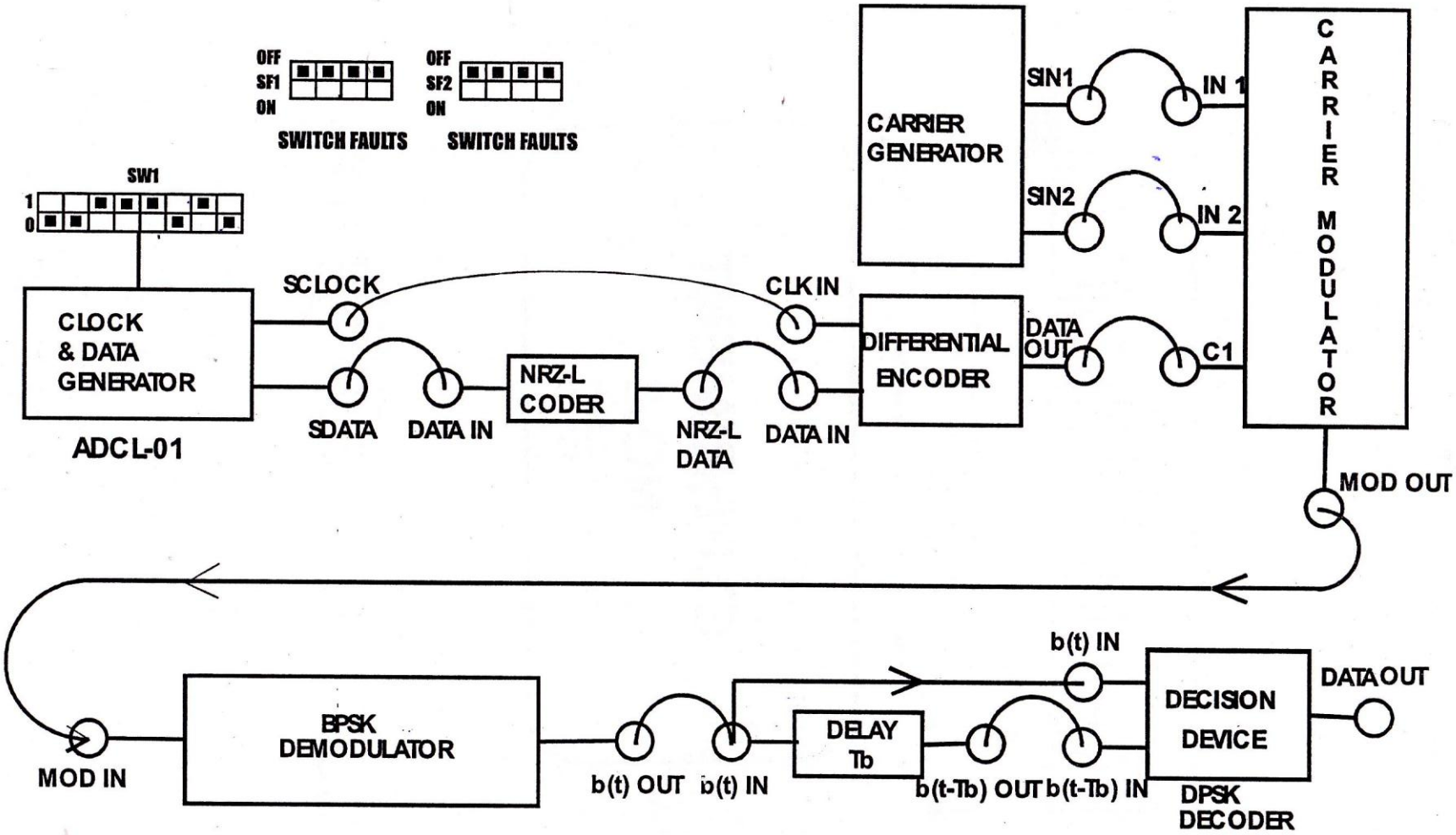


FIG. 3.1 BLOCK DIAGRAM FOR
DIFFERENTIAL PHASE SHIFT KEYING MODULATION TECHNIQUE

The DPSK system explained above has a clear advantage over the BPSK system in that the former avoids the need for complicated circuitry used to generate a local carrier at the receiver. To see the relative disadvantage of DPSK in comparison with PSK, Consider that during some bit interval the received signal is so contaminated by noise that in a PSK system an error would be made in the determination of whether the transmitted bit was a 1 or 0. In DPSK a bit determination is made on the basis of the signal received in two successive bit intervals. Hence noise in one bit interval may cause errors to two-bit determination. The error rate in DPSK is therefore greater than in PSK, and, as a matter of fact, there is a tendency for bit errors to occur in pairs. It is not inevitable however that error occur in pairs. Single errors are still possible.

Procedure:

- Refer to the block diagram (Fig.3.1) and carry out the following connections and switch settings.
- Connect power supply in proper polarity to the kit **ADCL-01** and switch it on.
- Select Data pattern of simulated data using switch SW1
- Connect **DATA** generated to **DATA IN** of **NRZ-L CODER**.
- Connect the **NRZ-L DATA** output to the **DATA IN** of the **DIFFERENTIAL ENCODER**.
- Connect the clock generated **SCLOCK** to **CLK IN** of the **DIFFERENTIAL ENCODER**.
- Connect differentially encoded data to control input C1 of **CARRIER MODULATOR**.
- Connect carrier component **SIN1** to **IN1** and **SIN 2** to **IN2** of the Carrier Modulator Logic.
- Connect DPSK modulated signal **MOD OUT** to **MOD IN** of the **BPSK DEMODULATOR**.
- Connect output of **BPSK** demodulator **b (t) OUT** to input of **DELAY SECTION b (t) IN** and one input **b (t) IN** of decision device.
- Connect the output of delay section **b (t-Tb) OUT** to the input **b (t-Tb) IN** of decision device.
- Compare the **DPSK** decoded data at **DATA OUT** with respect to input **SDATA**.
- Observe various waveforms as mentioned below (Fig.3.3), if recovered data mismatches with respect to the transmitter data, then use **RESEST** switch for clear observation of data output.

Observations:

Observe the following waveforms on CRO and plot it on the paper.

ON Kit ADCL-01

- Input **NRZ-L** Data at **DATA IN** of **DIFFERENTIAL ENCODER**.
- Differentially encoded data at **DATA OUT** of **DIFFERENTIAL ENCODER**.
- Carrier frequency **SIN1** and **SIN 2**.
- **DPSK** modulated data at **MOD OUT**.
- **DPSK DEMODULATED** signal at **b (t) OUT** of **BPSK DEMODULATOR**.
- Delayed data by one bit interval at **b (t-Tb) OUT** of **DELAY SECTION**.
- **DPSK** decoded data at **DATA OUT** of **DPSK DECODER**.

Conclusion: The differential coding of data to be transmitted makes the bit “1” to be transformed onto carrier phase variation. In this way the receiver recognizes one bit “1” at a time which detects a phase shift of the modulated carrier, independently from its absolute phase. In this way the BPSK modulation, which can take to the inversion of the demodulated data, is overcome

Sin-1: 1.32Vpp, 510 KHz
Sin-2: 1.42Vpp, 510 KHz

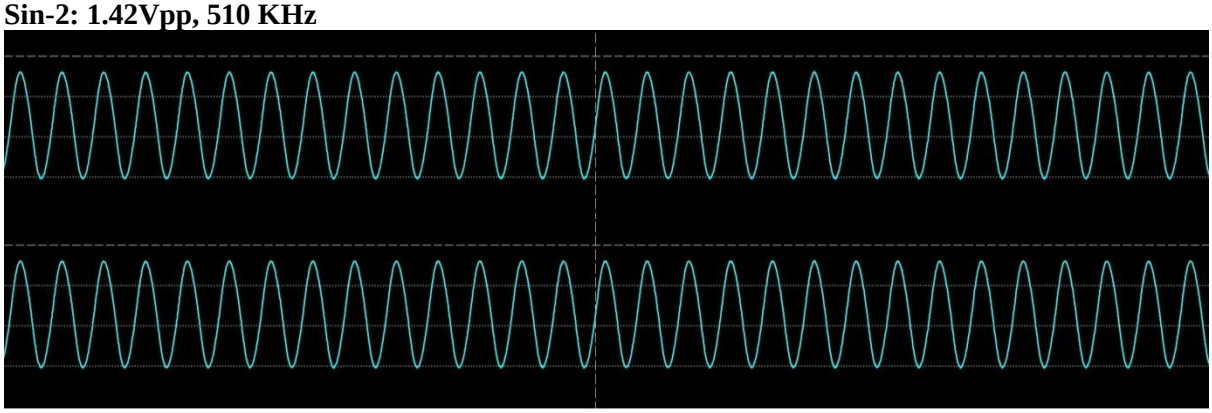


Figure 1: Sin I/P's

Clock: 10.5Vpp, 256 KHz
S Data: 10.5Vpp, 51 KHz

Figure 2: Clock & Serial Data

Differential Data: 5Vpp, 85 KHz
Mod O/P: 1.6Vpp, 521 KHz

Figure 3: Modulated O/P & Differential Data

D. Data & Demodulated

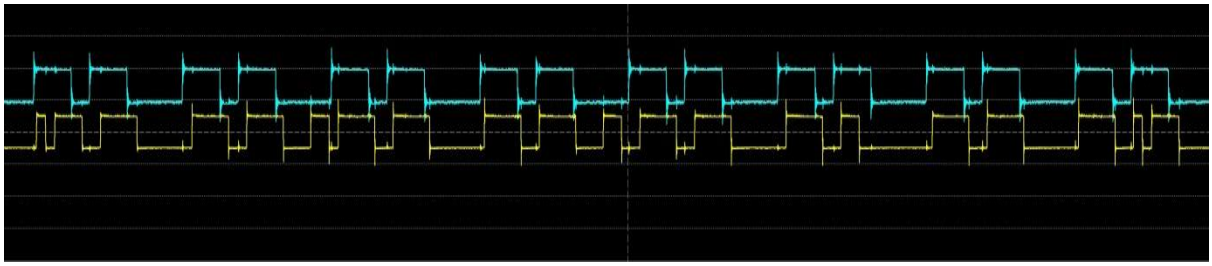


Figure 4: Demodulated O/P

PULSE CODE MODULATION

Aim: To study 2-channel Time Division Multiplexing and Sampling of analog signal, and its pulse code modulation in none parity mode in the transmitter section and to study the demultiplexing and the reconstruction of the analog signal in the receiver section.

Equipment Required:

1. Experiment kits DCL-03 & DCL-04
2. Connecting chords
3. Power supply
4. 50 MHz Digital Storage Oscilloscope
5. CRO Probes

Note: Keep the switch faults in off position.

Theory:

The sine waves (analog signal) of frequency 500Hz and 1 KHz and DC Signal DC1 and DC2 whose amplitude can be varied accordingly are generated onboard on DCL-03. These signals are fed to the input of the Sampling logic Cho & CH1 and their samples are multiplexed by interleaving them properly in their assigned time slots.

The crystal oscillator generates a clock of 6.4MHz from which all the transmitter data and timing signals are derived. For fast mode operation the transmitter clock is 240 KHz, and sampling clock is 16 KHz. For slow mode operation depending on 0.088 Hz or 0.044 Hz i.e. the sampling rate per channel is 11 or 22 seconds and serial data transmission rate is 813 milliseconds or 1.6 seconds.

The multiplexed data is Pulse Code Modulated before transmission. At the receiver after the Pulse Code Demodulation, the recovered multiplexed data is sent to De-multiplexing logic. The two demultiplexed samples are fed to reconstruction unit. Which consist of 4^{th} order Low Pass Butterworth Filter, where frequency components are filtered out to recover the original base band signal at the receiver output CH0 and CH1.

Procedure:

- Refer to the Block Diagram (Fig.1.1) & Carry out the following connections.
- Connect power supply in proper polarity to the kits **DCL-03** and **DCL-04** and switch it on.
- Connect sine wave of frequency **500Hz** and **1 KHz** to the input **CH0** and **CH1** of the sample and hold logic.
- Connect **OUT 0** to **CH0 IN & OUT 1** to **CH1 IN**.
- Set the speed selection switch **SW1** to **FAST** mode.
- Select parity selection switch to **NONE** mode on both the kit **DCL-03** and **DCL-04** as shown in switch setting diagram (Fig. A).
- Connect **TXDATA**, **TXCLK** and **TXSYNC** of the transmitter section **DCL-03** to the corresponding **RXDATA**, **RXCLK**, and **RXSYNC** of the receiver section **DCL-04**.
- Connect posts **DAC OUT** to **IN** post of de-multiplexer section on **DCL-04**.

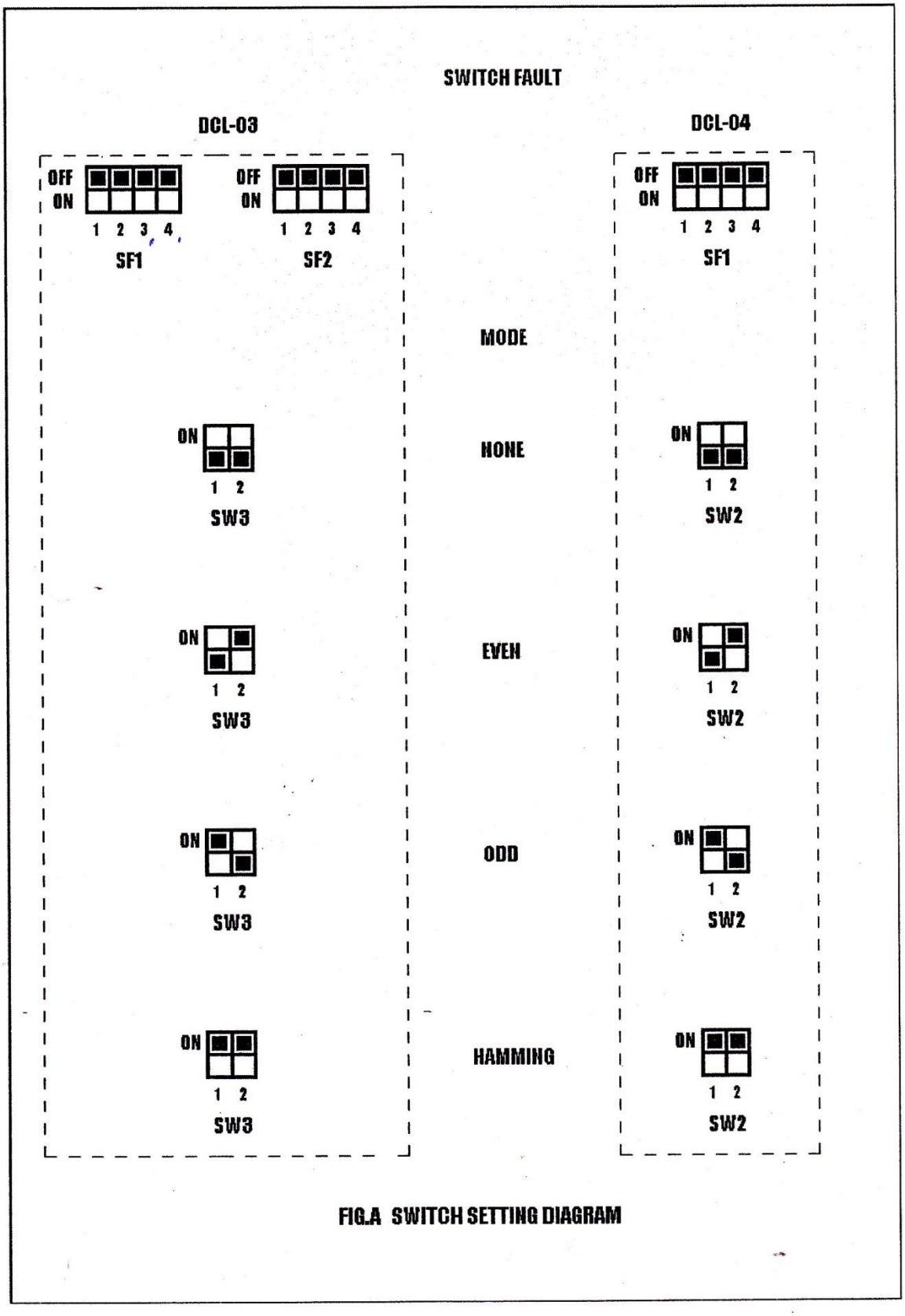


FIG.A SWITCH SETTING DIAGRAM

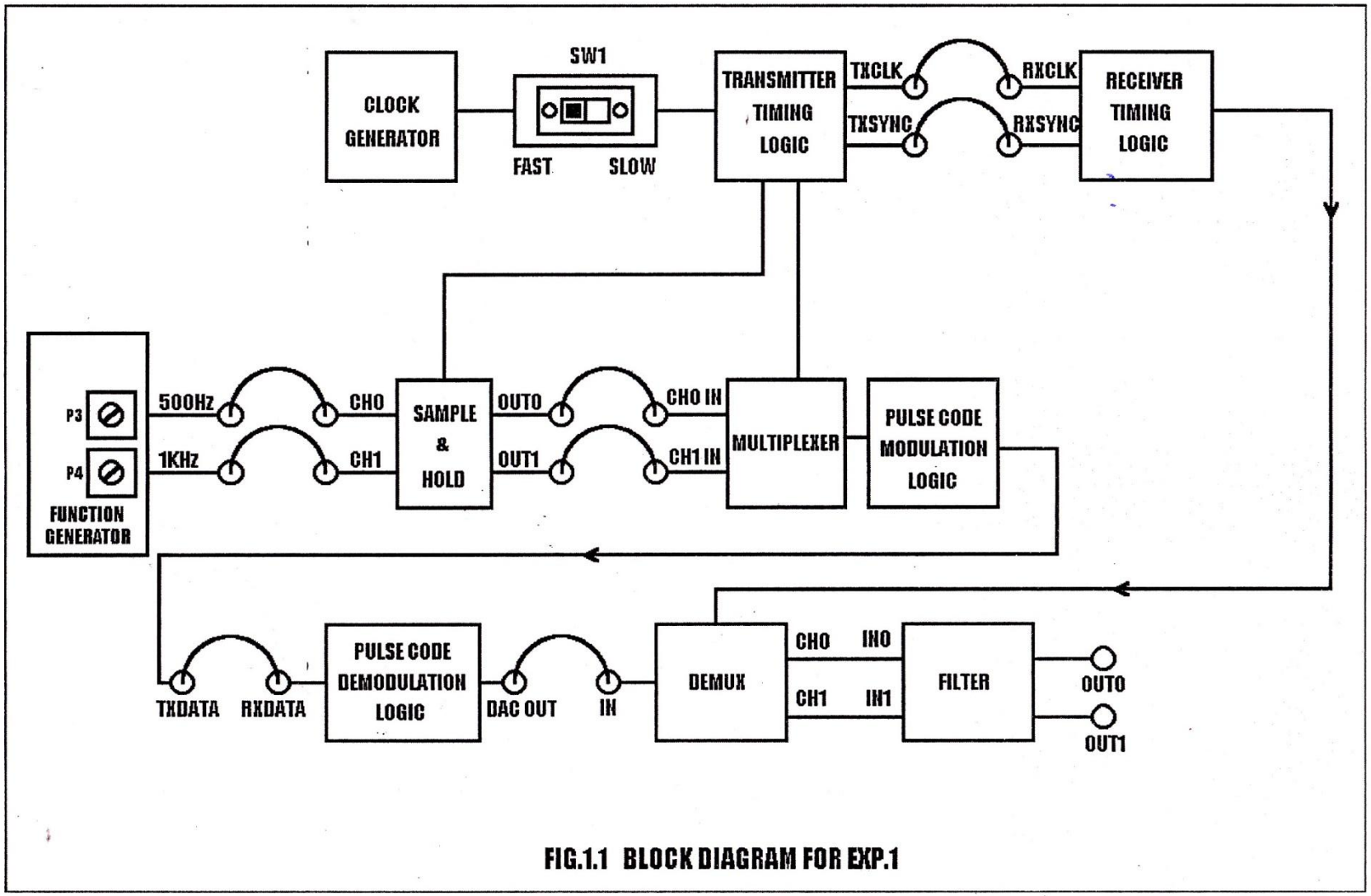


FIG.1.1 BLOCK DIAGRAM FOR EXP.1

CH0- 2.12Vpp, 500 KHz
CH1- 2.5Vpp, 1 KHz

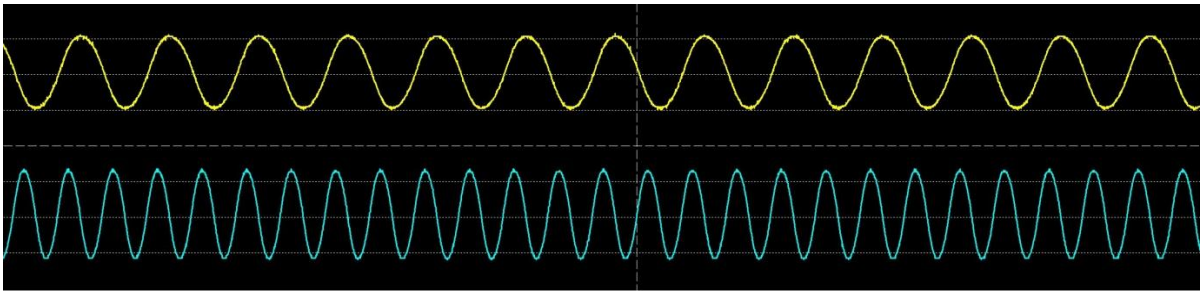


Figure 1: Sin I/P's to CH0 & CH1

TXSYNC- 5.6Vpp, 7.6 KHz

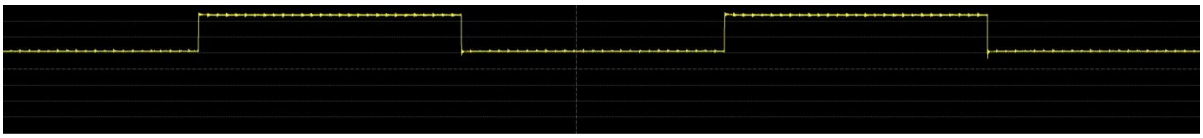


Figure 2: Signal at TXSYNC

PRBS- 5.4Vpp, 42.4 KHz
TXDATA- 5.8Vpp, 71.4 KHz

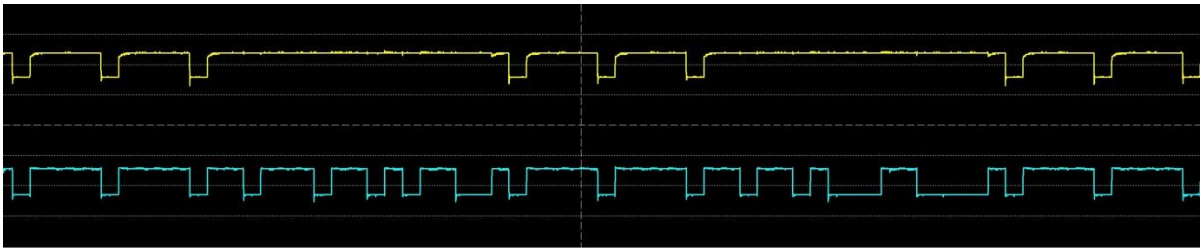


Figure 3: Signals at PRBS OUT & TXDATA

CH0- 500 KHz
CH1- 1 KHz

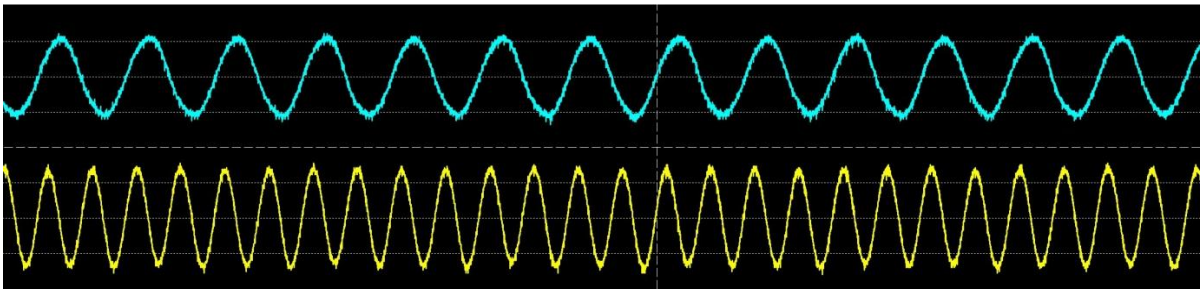


Figure 4: Demodulation

PULSE CODE MODULATION & DEMODULATION WAVEFORMS

- Ensure that **FAULT SWITCH SF1** as shown in switch setting diagram (Fig. A) introduces no fault.
- Take the observations as mentioned below.
- Repeat the above experiment with DC Signal at the inputs of the Channel **CH0** and **CH1**.
- Connect ground points of both the kits with the help of connecting chord provided during all the experiments

Observation:

Observe the following signal on oscilloscope and plot it on the paper.

ON KIT DCL-03 (Fig. 1.2) & (Fig. 1.3)

- Input signal **CH0** and **CH1**.
- Sample and Hold output **OUT0** and **OUT1**
- Multiplexer clock **CLK 1** and **CLK2**
- Multiplexed data **MUX OUT**.
- PCM Data **TXDATA**, **TSCLK**, **TXSYNC**

ON KIT DCL-04 (Fig. 1.4) & (Fig.1.5)

- **RXCLK**, **RXSYNC**, **RXDATA**
- **DAC OUT**
- Demultiplexer clock **CLK1** and **CLK2**
- Demultiplexed Data **CH0** and **CH1**
- Received signal **OUT0** and **OUT1**.

Conclusion:

We conclude that at the transmitter side sampling for 500Hz and 1 KHz signals is done by using 16 KHZ sampling clock, hereby satisfying the Nyqist criterion. Similarly the multiplexed output observed in the oscilloscope shows the proper alignment of samples in their respective time slots.

At the receiver side the 4^{th} order low pass butter-worth filter is used as reconstruction unit, which reproduce the signals (sine wave and DC signal levels) same as that of the transmitter side. It is observed in this case, that the reconstructed sine wave has good linearity.

DIFFERENTIAL PULSE CODE MODULATION

Aim: Study of differential pulse code modulation technique

Equipment Required:

1. ADCL-07 Kit
2. Patch cords
3. Power Supply
4. 50MHz Digital Storage Oscilloscope
5. CRO Probes.

Theory: DPCM is a good way to reduce the bit rate for voice transmission. However it causes some other problems that deal with voice quality. DPCM quantizes and encodes the difference between a previous sample input signal and a current sample input signal. DPCM quantizes the difference signal using uniform quantization. Uniform quantization generates an SNR that is small for small input sample signals and large for large input sample signals. Therefore, the voice quality is better at higher signals.

The first part of DPCM works exactly like PCM (that is why it is called differential PCM). The input signal is sampled at a constant sampling frequency (more than the input frequency). Then these samples are modulated. At this point, the DPCM process takes over. The sampled input signals are stored in what is called a predictor. The predictor takes the stored sample signal and sends it through a differentiator. The differentiator compares the previous sample signal and sends its difference to the quantizing and coding phase of PCM (this phase can be a uniform quantizing or Companding with A-law or μ-law). After quantizing and coding, the difference signal is transmitted to its final destination. At the receiving end of the network, everything is reversed. First the difference signal is decoded and dequantized. This difference is added to the sample signal stored in the predictor and send through a low-pass filter that reconstructs the original input signal.

Procedure:

- Refer to the block diagram (Fig.1) and carry out the following connections and switch settings.
- Connect power supply in proper polarity to the kit ADCL-07 and switch it ON.
- Keep the clock frequency at 512 KHz, by changing the jumper position of JP1 in the clock generator section.
- Keep the amplitude of the onboard sine wave, of frequency 500Hz to 1Vpp.

DPCM Modulation:

- Connect the 500Hz sine wave to the IN post of Analog Buffer.
- Connect OUT post of Analog Buffer to IN post of DPCM modulator section.
- Observe the sample output at the given test point. The input signal is sampled at the clock frequency of 16 KHz.
- Observe the linear predictor output at the PREDICTED OUT post of the linear predictor in the DPCM modulator section.
- Observe the differential pulse code modulated data (DPCM) at the DPCM OUT post of the DPCM modulator section.
- Observe the DPCM data at DPCM OUT post by varying input signal from 0 to2V.

DPCM Demodulation:

- Connect the DPCM modulated data from the DPCM OUT post of the DPCM modulator to the IN post of the DPCM demodulator.

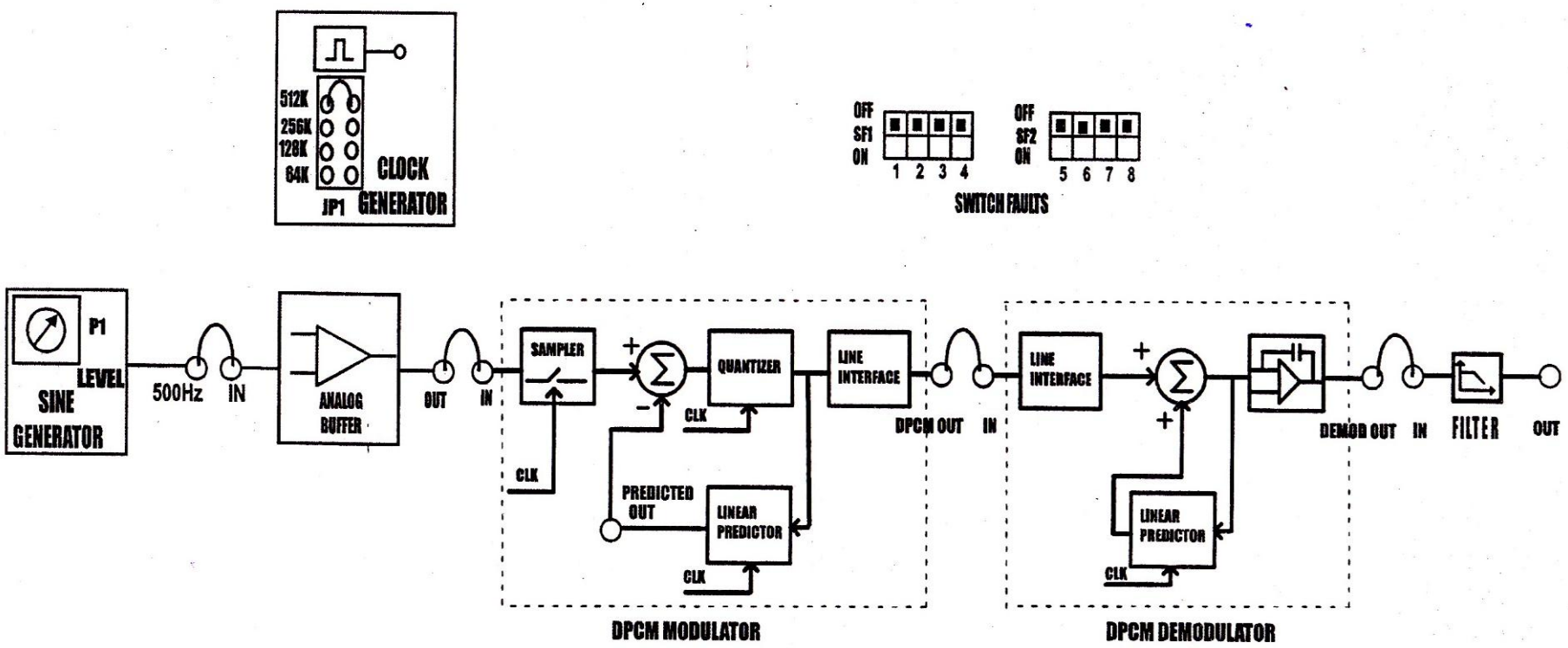


FIG. 1 BLOCK DIAGRAM FOR STUDY OF DIFFERENTIAL PULSE CODE MODULATION AND DEMODULATION

Sin I/P: 1Vpp, 500Hz
Predicted O/P: 3.3Vpp, 272 Hz

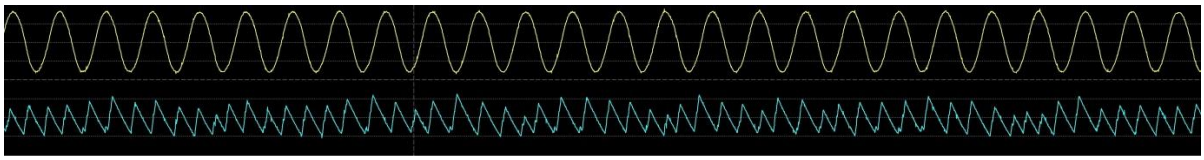


Figure 1: sin I/P & Predicted O/P

Clock: 7.2Vpp, 128 KHz

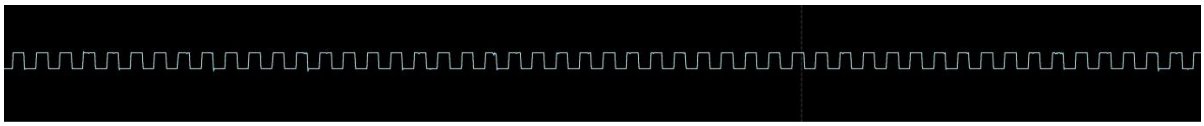


Figure 2: Clock

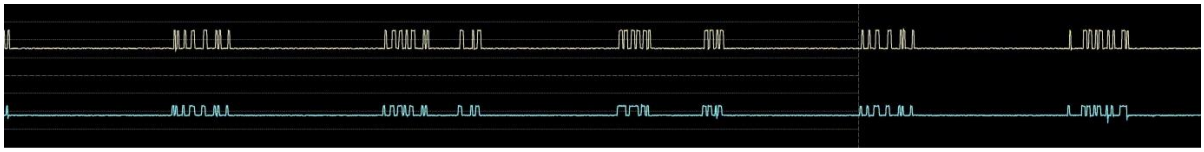


Figure 3: DPCM O/P & Summation O/P

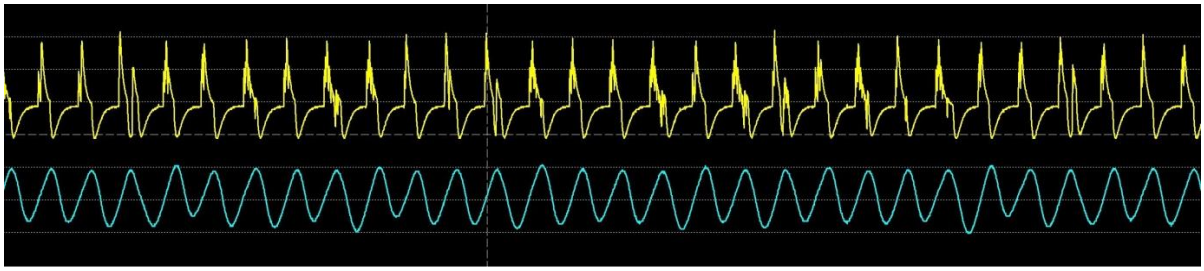


Figure 4: Demodulation O/P & Filter O/P

DPCM WAVEFORMS

- Observe the demodulated data at the output of summation block.
- Observe the integrated demodulated data at the DEMOCOUT post of the DPCM demodulator to the IN post of the low-pass filter.
- Observe the reconstructed signal at the OUT post of the filter. Use RST switch for clear observation of output.
- Now, simultaneously reduce the clock frequencies from 512 KHz to 256 KHz, 128 KHz and 64 KHZ by changing the jumper position of JP1 and observe the difference in the DPCM modulated and

demodulated data. As the frequency of clock decreases, DPCM demodulated data at DEMOC OUT becomes distorted.

- Observe various waveforms as mentioned below (Fig.1.2).

Observations:

ON KIT ADCL-07
Observe the following waveforms on the oscilloscope and plot on the paper.

- 500Hz, 1Vpp input sine wave.
- Sampled out at the provided test point SAMPLER OUT.
- Linear predictor out at PREDICTED OUT post.
- DPCM data at DPCM OUT post.
- Line interface out at the given output test point of line interface block in DPCM demodulator.
- Demodulated DPCM data at the output test point of summation block in DPCM demodulator.
- Integrated demodulated data at the DEMOC OUT post of the DPCM demodulator.
- Reconstructed sine wave at the OUT post of the filter.
- Observe the data at different clock rates.

Conclusion:

DELTA MODULATION

Aim: (a) To verify the Encoding process of Linear Delta Modulator and corresponding waveforms.

(b) To verify the operation of the Linear Delta Demodulator.

Apparatus:

1. Experimental kit of Delta modulation & demodulation.
2. 20MHz Dual trace Oscilloscope
3. Patch chords & CRO probes.

Procedure:

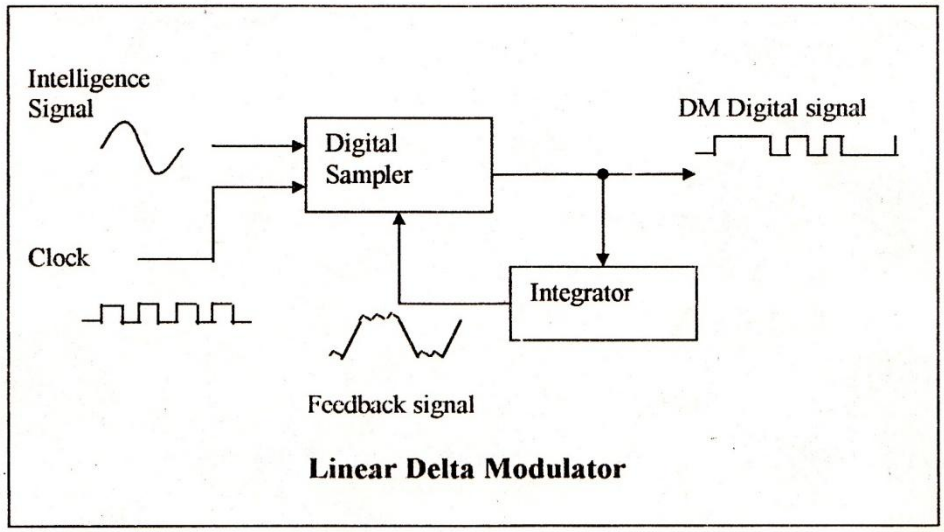
Modulation:

1. Connect PLA1 to PLAA.
2. Connect channel 1 of CRO to TPA1/TPAA. Adjust VR1 to minimum to get zero level signals.
3. Connect channel-1 to TP1 and channel-2 to TPB1 and adjust VR2 to obtain square wave half the frequency of the clock rate selected (Output at TP1).
4. Connect channel 1 to TP2 and set voltage/div of channel 1 to mV range and observe a triangle waveform which is output of integrator. It can be observed that as the clock rate is increased, amplitude of triangle waveform decreases. This is called minimum step size (Clock rate can be changed by depressing SW1 switch).
5. Connect channel 1 to TPA1/TPAA. Adjust VR1 in order to obtain a 1 KHz sine wave of 500 mV_{p-p} approximately.
6. Signal approximating 1 KHz is available at the integrator output (TP2). This signal is obtained by integrating the digital output resulting from delta modulation.
7. Connect channel 1 to TP2 and channel 2 to TPBa. It can be observed that the digital high makes the integrator output to go upwards and digital low makes the integrator output to go downwards.
8. With an oscilloscope displaying three traces, it is possible to simultaneously observe the input signal of the modulation, the digital output of the modulator and the signal obtained by the integration from the modulator digital output. Notice that, when the output (Feedback signal) is lower than the analog input the digital output is high, whenever it is low when the analog input is lower than the integrated output.
9. Increases the amplitude of 1 KHz sine wave by rotating VR1 to 1Vp-p observe that the integrator output follows the input signal.
10. Increase that amplitude of 1 KHz sine wave further high, and observe that the integrator output cannot follow the input signal.
11. Repeat the above mentioned procedures with different signal sources and selecting different clock rates and observe the response of the Linear Delta Modulator.

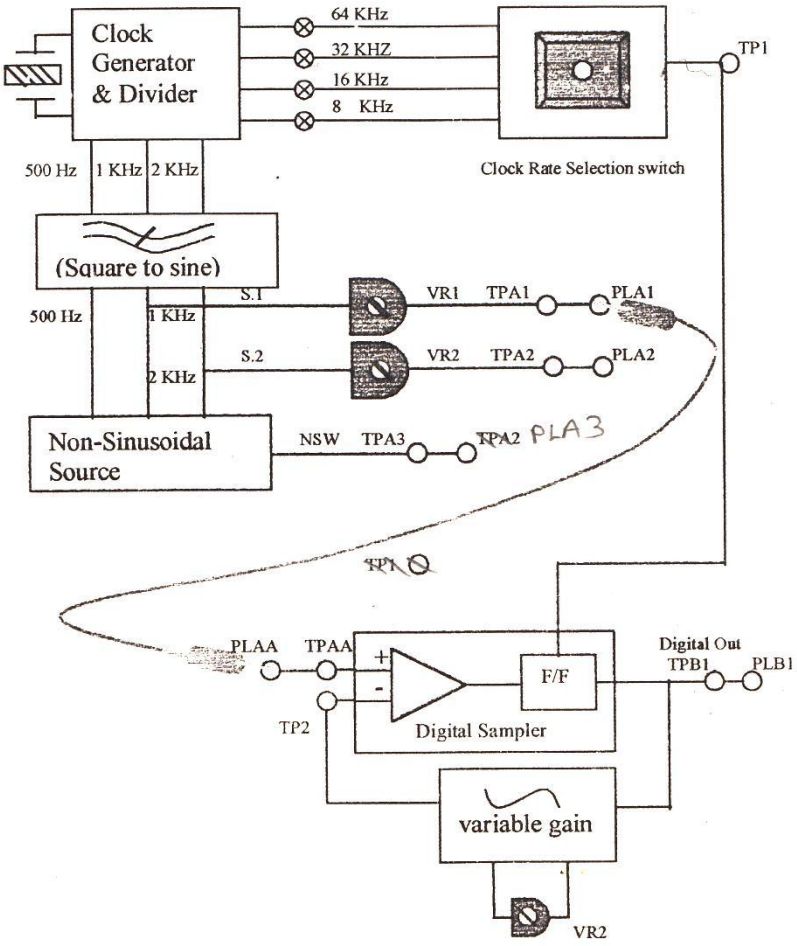
Demodulation:

1. Prearrange the connections of Linear Delta Modulator.
2. Connect PLB1 (Digital output of Delta Modulator) to PLBB (input of Linear Delta modulator).
3. Connect PLC1 (Linear Delta Demodulator output) to either PLCA (input of fourth order LPF) or PLCB (input of Second order LPF).

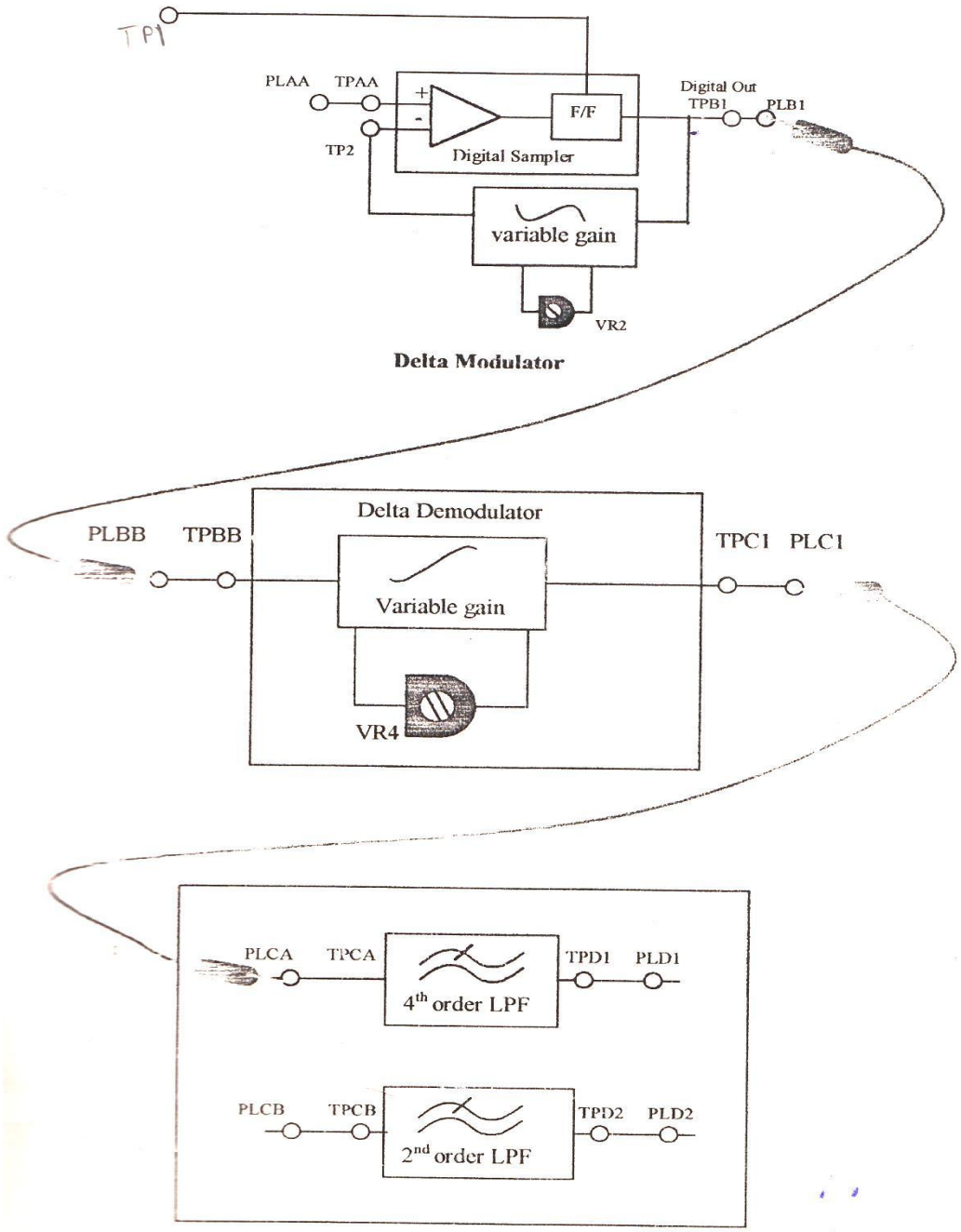
Block Diagram:



EXPERIMENTAL SETUP FOR DELTA MODULATION



EXPERIMENTAL SETUP FOR DELTA DEMODULATION



Observations:

Observe the reconstructed output of the fourth order Low Pass Filter at TD1 and also observe the output of the second order filter at TPD2.

Precautions:

1. Do not make any interconnections with power switched ON.
2. Measure the delta modulated signal corresponding to the input analog signal by keeping the CRO in dual mode.
3. Verify the loose connections before observing the output on CRO.

Sin I/P: 3Vpp, 977Hz
Clock: 4.4Vpp, 7.9 KHz

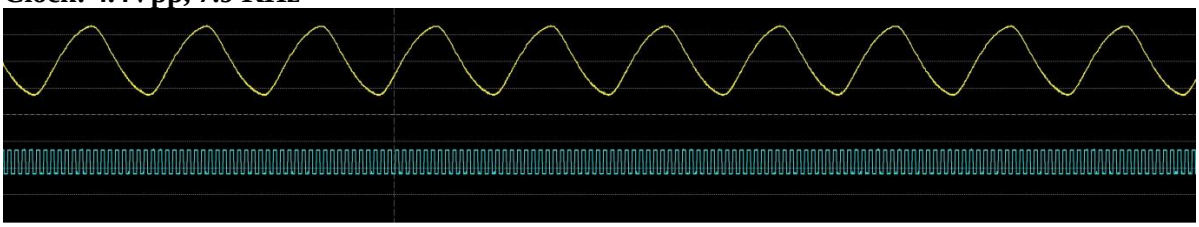


Figure 1: Sin I/P & Clock

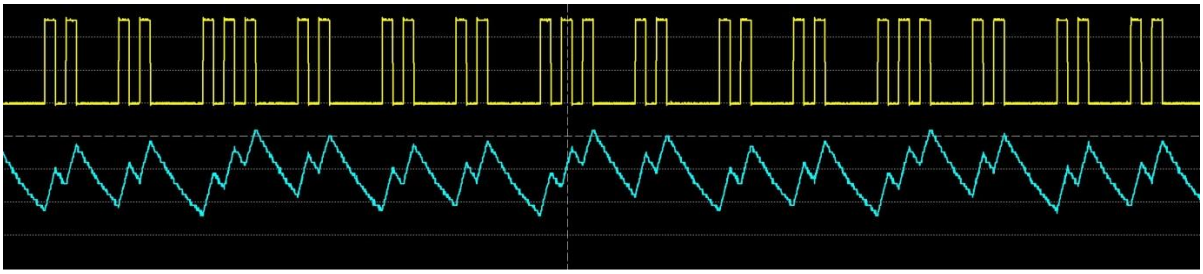


Figure 2: Modulated O/P- Digital & Integrator o/P

Clock: 2.4Vpp, 7.6 KHz

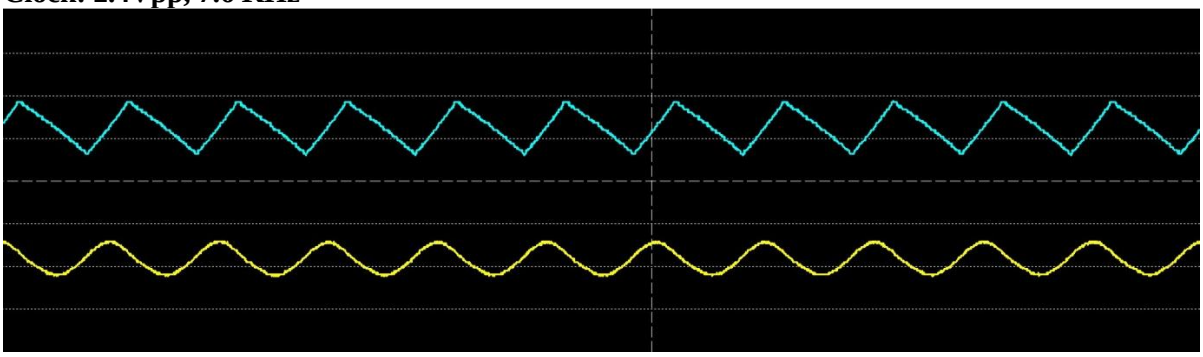


Figure 3: Demodulated O/P & Filter O/P

DELTA MODULATION WAVEFORMS

Conclusion:

PHASE SHIFT KEYING (MATLAB)

Aim: To perform phase shift keying in MATLAB

Experimental requirements: PC loaded with MATLAB software

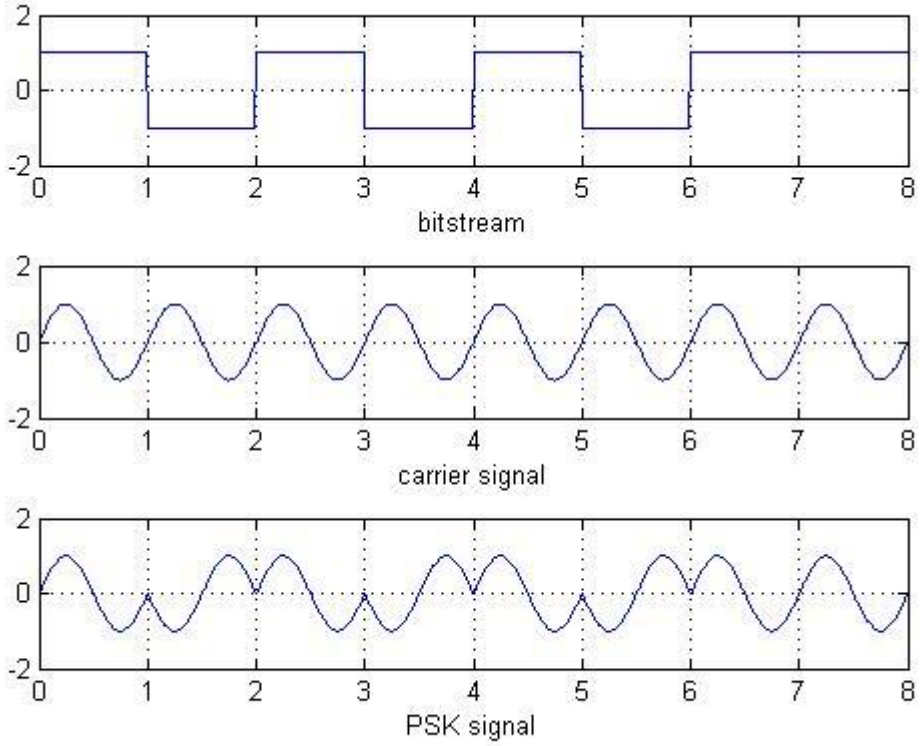
Procedure:

1. Run MATLAB
2. Open a new script file
3. Write the code for phase shift keying
4. Run the code for execution and obtain the necessary results

MATLAB script

```
clear;
clc;
b = input('Enter the Bit stream \n '); %b = [0 1 0 1 1 1
0];
n = length(b);
t = 0:.01:n;
x = 1:1:(n+1)*100;
for i = 1:n
if (b(i) == 0)
b_p(i) = -1;
else
b_p(i) = 1;
end
for j = i:.1:i+1
bw(x(i*100:(i+1)*100)) = b_p(i);
end
end
bw = bw(100:end);
sint = sin(2*pi*t);
st = bw.*sint;
subplot(3,1,1)
plot(t,bw)
gridon ; axis([0 n -2 +2])
subplot(3,1,2)
plot(t,sint)
gridon ; axis([0 n -2 +2])
subplot(3,1,3)
plot(t,st)
gridon ; axis([0 n -2 +2])
```

Waveforms



Conclusion:

FREQUENCY SHIFT KEYING (MATLAB)

Aim: To perform Frequency shift keying in MATLAB

Experimental requirements: PC loaded with MATLAB

Procedure:

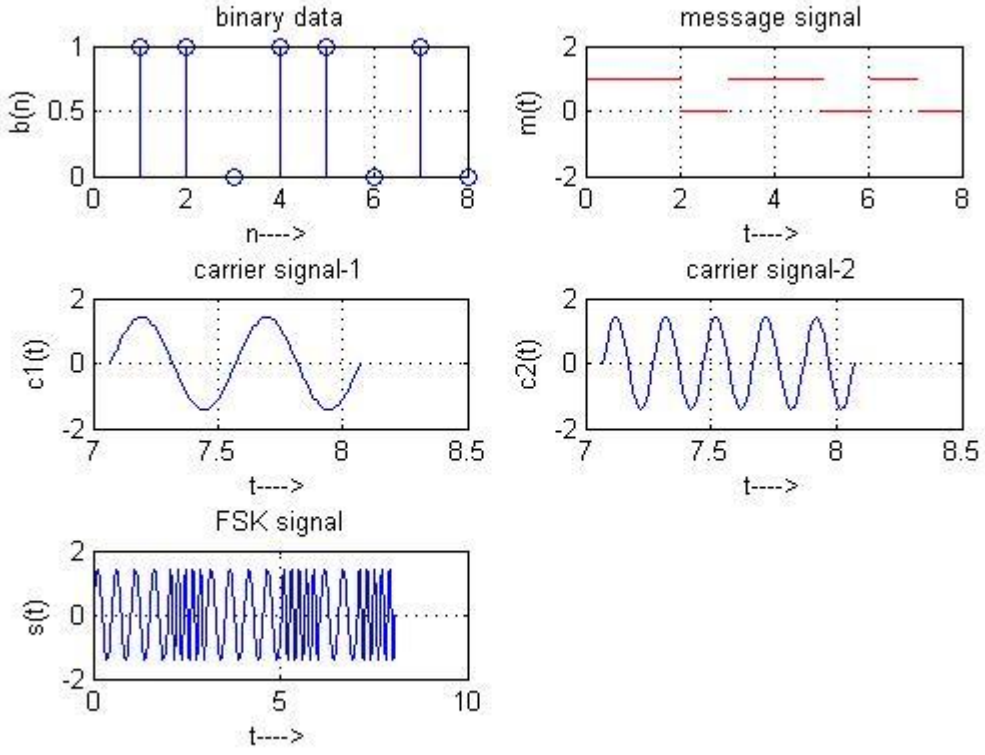
1. Run MATLAB
2. Open a new script file
3. Write the code for frequency shift keying
4. Run the code for execution and obtain the necessary results

MATLAB script

```
clc;
clearall;
closeall;
%GENERATE CARRIER SIGNAL
Tb=1; fc1=2;fc2=5;
t=0:(Tb/100):Tb;
c1=sqrt(2/Tb)*sin(2*pi*fc1*t);
c2=sqrt(2/Tb)*sin(2*pi*fc2*t);
%generate message signal
N=8;
m=rand(1,N);
t1=0;t2=Tb
for i=1:N
t=[t1:(Tb/100):t2]
if m(i)>0.5
m(i)=1;
m_s=ones(1,length(t));
invm_s=zeros(1,length(t));
else
m(i)=0;
m_s=zeros(1,length(t));
invm_s=ones(1,length(t));
end
message(i,:)=m_s;
%Multiplier
fsk_sig1(i,:)=c1.*m_s;
fsk_sig2(i,:)=c2.*invm_s;
fsk=fsk_sig1+fsk_sig2;
%plotting the message signal and the modulated signal
```

```
subplot(3,2,2);axis([0 N -2 2]);plot(t,message(i,:),'r');
title('message signal');
xlabel('t---->');ylabel('m(t)');grid on;hold on;
subplot(3,2,5);plot(t,fsk(i,:));
title('FSK signal');
xlabel('t---->');ylabel('s(t)');grid on;hold on;
t1=t1+(Tb+.01); t2=t2+(Tb+.01);
end
holdoff
%Plotting binary data bits and carrier signal
subplot(3,2,1);stem(m);
title('binary data');xlabel('n---->'); ylabel('b(n)');grid
on;
subplot(3,2,3);plot(t,c1);
title('carrier signal-1');
xlabel('t---->');ylabel('c1(t)');grid on;
subplot(3,2,4);plot(t,c2);
title('carrier signal-2');
xlabel('t---->');ylabel('c2(t)');grid on;
```

Waveforms:



Conclusion:

DIRECT SEQUENCE SPREAD SPECTRUM (MATLAB)

Aim: To implement direct sequence spread spectrum in MATLAB

Experimental requirements: PC loaded with MATLAB

Theory: **Direct-sequence spread spectrum** (**DSSS**) is a modulation technique. As with other spread spectrum technologies, the transmitted signal takes up more bandwidth than the information signal that is being modulated. The name 'spread spectrum' comes from the fact that the carrier signals occur over the full bandwidth (spectrum) of a device's transmitting frequency.

Direct-sequence spread-spectrum transmissions multiply the data being transmitted by a "noise" signal. This noise signal is a pseudorandom sequence of 1 and −1values, at a frequency much higher than that of the original signal, thereby spreading the energy of the original signal into a much wider band.

The resulting signal resembles white noise, like an audio recording of "static". However, this noise-like signal can be used to exactly reconstruct the original data at the receiving end, by multiplying it by the same pseudorandom sequence (because 1 × 1 = 1, and −1 × −1 = 1). This process, known as "de-spreading", mathematically constitutes a correlation of the transmitted PN sequence with the PN sequence that the receiver believes the transmitter is using.

Procedure:

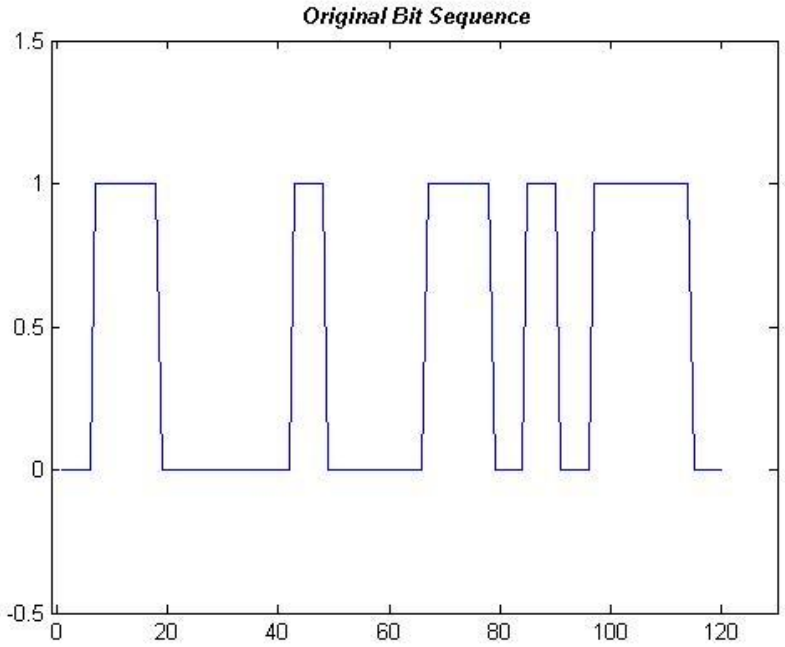
1. Run MATLAB
2. Open a new script file
3. Write the code for Direct sequence spread spectrum technique.
4. Run the code for execution and obtain the necessary results

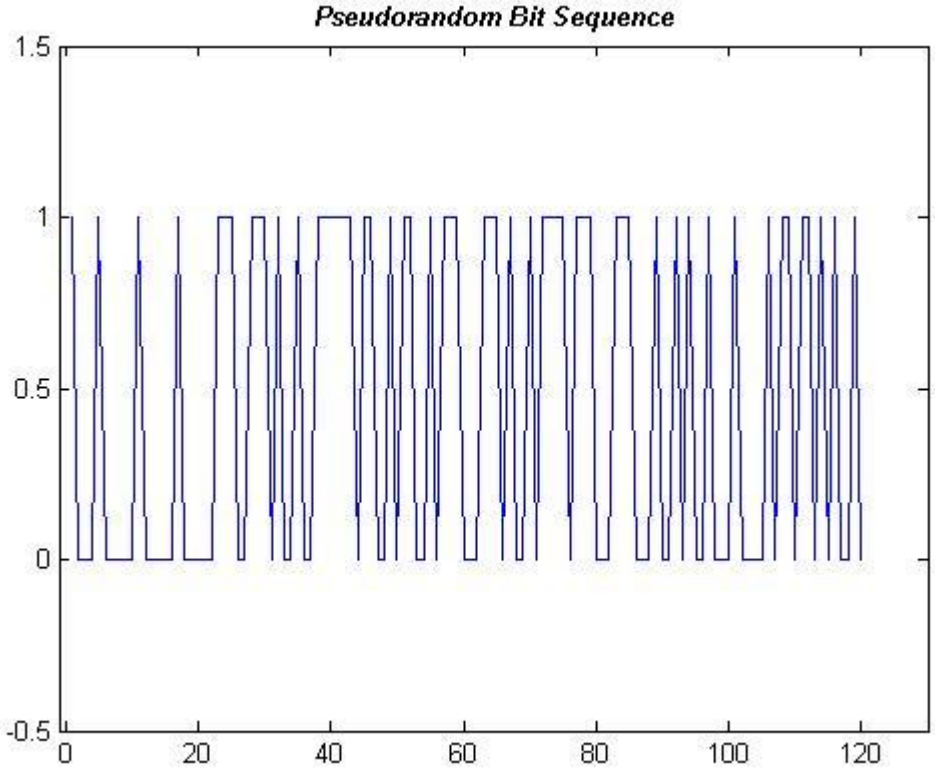
MATLAB script:

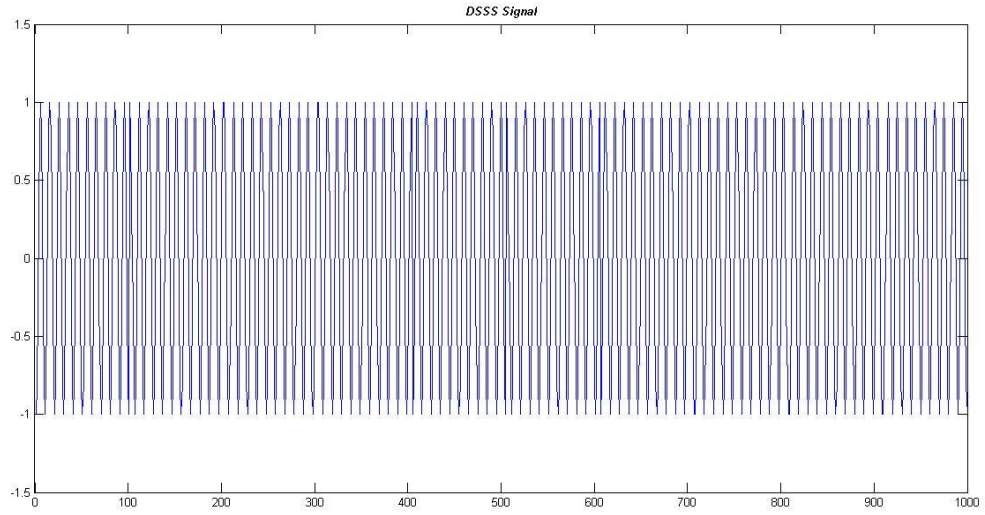
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Direct Sequence Spread Spectrum
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
clear

% Generating the bit pattern with each bit 6 samples long
b=round(rand(1,20));
pattern=[];
for k=1:20
if b(1,k)==0
sig=zeros(1,6);
else
sig=ones(1,6);
end
pattern=[pattern sig];
end
```

```
plot(pattern);
axis([-1 130 -.5 1.5]);
title('\bf\it Original Bit Sequence');

% Generating the pseudo random bit pattern for spreading
spread_sig=round(rand(1,120));
figure,plot(spread_sig);
axis([-1 130 -.5 1.5]);
title('\bf\it Pseudorandom Bit Sequence');

% XORing the pattern with the spread signal
hopped_sig=xor(pattern,spread_sig);

% Modulating the hopped signal
dsss_sig=[];
t=[0:100];
fc=.1;
c1=cos(2*pi*fc*t);
c2=cos(2*pi*fc*t+pi);
for k=1:120
if hopped_sig(1,k)==0
        dsss_sig=[dsss_sig c1];
else
        dsss_sig=[dsss_sig c2];
end
end
figure,plot([1:12120],dsss_sig);
axis([-1 1000 -1.5 1.5]);
title('\bf\it DSSS Signal');

% Plotting the FFT of DSSS signal
figure,plot([1:12120],abs(fft(dsss_sig)))
```

Waveforms:


Original Bit Sequence
1.5
1
0.5
0
-0.5
0
20
40
60
80
100
120


Pseudorandom Bit Sequence
1.5
1
0.5
0
-0.5
0
20
40
60
80
100
120


DSSS Signal
1.5
1
0.5
0
-0.5
-1
-1.5
0
100
200
300
400
500
600
700
800
900
1000

Conclusion:

IMPLEMENTATION OF SHANNON- FANO CODING USING MATLAB

Aim: To implement Shannon Fano coding using MATLAB

Experimental requirements: PC loaded with MATLAB software

Theory: In Shannon–Fano coding, the symbols are arranged in order from most probable to least probable, and then divided into two sets whose total probabilities are as close as possible to being equal. All symbols then have the first digits of their codes assigned; symbols in the first set receive "0" and symbols in the second set receive "1". As long as any sets with more than one member remain, the same process is repeated on those sets, to determine successive digits of their codes. When a set has been reduced to one symbol, of course, this means the symbol's code is complete and will not form the prefix of any other symbol's code.

The algorithm works, and it produces fairly efficient variable-length encodings; when the two smaller sets produced by a partitioning are in fact of equal probability, the one bit of information used to distinguish them is used most efficiently. Unfortunately, Shannon–Fano does not always produce optimal prefix codes; the set of probabilities {0.35, 0.17, 0.17, 0.16, 0.15} is an example of one that will be assigned non-optimal codes by Shannon–Fano coding.

Procedure:

1. Run MATLAB
2. Open a new script file
3. Write the code for Shannon fano encoding technique.
4. Run the code for execution and obtain the necessary results

MATLAB script:

```
aa=fopen('C:\Users\vvinod1\Desktop\DC lab
programs\shannonfanoencoder\FF7.txt');
bb=fread(aa);
cc=unique(bb); %find used characters
taas=length(bb);
for n=1:length(cc)
d(n,1)=length(find(bb==cc(n))); %count the occurence of a
character

end
ss=transpose(d)./taas;
ss=sort(ss,'descend');
siling=ceil(log2(1/ss(1)));
sf=0;
fano=0;
n=1;Hs=0;
```

```
for iii=1:length(ss)
   Hs=Hs+ ss(iii)*log2(1/ss(iii));
end
shano=ss(1);
for o=1:length(ss)-1
fano=fano+ss(o);
sf=[sf 0]+[zeros(1,o) fano];
siling=[siling 0]+[zeros(1,o) ceil(log2(1/ss(o+1)))];
end

for r=1:length(sf)
esf=sf(r);

for p=1:siling(r)

esf=mod(esf,1)*2;
h(p)=esf-mod(esf,1);

end
hh(r)=h(1)*10^(siling(r)-1);
for t=2:siling(r)
hh(r)=hh(r)+h(t)*10^(siling(r)-t);
end
end
c={'0','1'};
for i=1:length(hh)
    u=1;
for t=siling(i):-1:1
       f=floor(hh(i)/10^(t-1));
hh(i)=mod(hh(i),10^(t-1));
if f==1
if u==1
                d=c{2};
else
                d=[d c{2}];
end
else
if u==1
                d=c{1};
else
                d=[d c{1}];
end
end
codex{i,:}={d};
       u=u+1;
end
```

```
end
tao=siling(1)*ss(1);
for u=1:length(ss)-1
tao=tao+siling(u+1)*ss(u+1);
end
T=tao/n;
B=[flipud(rot90(ss)),flipud(rot90(siling)),flipud(rot90(sf
))];
disp(['       s','        Li','        Pk'])
disp(B)
disp('Code')
disp('codex')
disp(['Hs = ',num2str(Hs)])
disp(['T = ',num2str(T),'bits/symbol'])
disp([num2str(Hs),' <= ',num2str(T),' <= ',num2str(Hs+1)])


function[codex,T]=sfencoderkasan(ss)
%made by Jamil Kasan from Manila, Philippines
%input = row matrix of occurrences or probabilities e.g.
ss=[1 3 4 5] or
%ss[0.4 0.3 0.2 0.1]
%outputs = string of codewords,average codeword length
ss=ss./sum(ss); %if occurrences are inputted,
probabilities are gained
ss=sort(ss,'descend');  %the probabilities are sorted in
descending order
siling=ceil(log2(1/ss(1))); %initial length is computed
sf=0;

fano=0;
%initializations for Pk
n=1;Hs=0; %initializations for entropy H(s)
for iii=1:length(ss)
   Hs=Hs+ ss(iii)*log2(1/ss(iii)); %solving for entropy
end

for o=1:length(ss)-1
fano=fano+ss(o);
sf=[sf 0]+[zeros(1,o) fano]; %solving for Pk for every
codeword
siling=[siling 0]+[zeros(1,o) ceil(log2(1/ss(o+1)))];
%solving for length every codeword
end
```

```
for r=1:length(sf)
esf=sf(r);

for p=1:siling(r)
esf=mod(esf,1)*2;
h(p)=esf-mod(esf,1); %converting Pk into a binary number
end
hh(r)=h(1)*10^(siling(r)-1); %initializtion for making the
binary a whole number
for t=2:siling(r)
hh(r)=hh(r)+h(t)*10^(siling(r)-t);    %making the binary a
whole number
end%e.g. 0.1101 ==> 1101
end
c={'0','1'};
for i=1:length(hh)
    u=1;                                          %converting
the codes into a string
for t=siling(i):-1:1
       f=floor(hh(i)/10^(t-1));                   %1001 ==>1
(getting the first highest unit of a number)
hh(i)=mod(hh(i),10^(t-1));                 %1001
==>001(eliminating the first highest unit of a number)
if f==1
if u==1
                d=c{2};                           %conversion
part (num(1001) to str(1001))
else
                d=[d c{2}];
end
else
if u==1
                d=c{1};
else
                d=[d c{1}];
end
end
codex{i,:}={d};
       u=u+1;
end
end
tao=siling(1)*ss(1); %initialization for codeword length
for u=1:length(ss)-1 %computing for codeword length
tao=tao+siling(u+1)*ss(u+1);
end
T=tao/n; %computing for average codeword length
```

```
B=[flipud(rot90(ss)),flipud(rot90(siling)),flipud(rot90(sf
))];
disp(['       s','        Li','        Pk'])
disp(B)
disp('Code')
disp(codex)
disp(['Hs = ',num2str(Hs)])
disp(['T = ',num2str(T),'bits/symbol'])
disp([num2str(Hs),' <= ',num2str(T),' <= ',num2str(Hs+1)])
```

Conclusion:

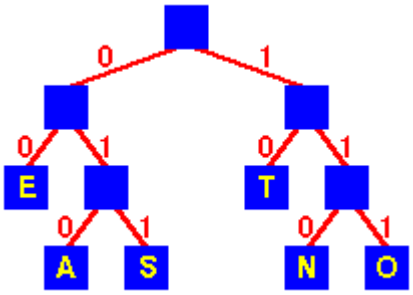
IMPLEMENTATION OF HUFFMAN CODING USING MATLAB

Aim: To implement Huffman coding using MATLAB

Experimental requirements: PC loaded with MATLAB software

Theory:

Huffman's scheme uses a table of frequency of occurrence for each symbol (or character) in the input. This table may be derived from the input itself or from data which is representative of the input. For instance, the frequency of occurrence of letters in normal English might be derived from processing a large number of text documents and then used for encoding all text documents. We then need to assign a variable-length bit string to each character that unambiguously represents that character. This means that the encoding for each character must have a unique prefix.
If the characters to be encoded are arranged in a binary tree:



Encoding tree for ETASNO

An encoding for each character is found by following the tree from the route to the character in the leaf: the encoding is the string of symbols on each branch followed.

For example:

String	Encoding
TEA	10 00 010
SEA	011 00 010
TEN	10 00 110

Notes:

1. As desired, the highest frequency letters - E and T - have two digit encodings, whereas all the others have three digit encodings.
2. Encoding would be done with a lookup table.

A divide-and-conquer approach might have us asking which characters should appear in the left and right subtrees and trying to build the tree from the top down. As with the optimal binary search tree, this will lead to an exponential time algorithm.

Procedure:

1. Run MATLAB
2. Open a new script file
3. Write the code for Huffman coding technique.
4. Run the code for execution and obtain the necessary results

MATLAB script:

```
lc;
clearall;
```

```
s=input('Enter symbols- ')      %format
['a','b','c','d','e','f'];
p=input('Enter value of probabilty- ')  %format
[0.22,0.20,0.18,0.15,0.13,0.12];

if length(s)~=length(p)
error('Wrong entry.. enter again- ')
end

i=1;
for m=1:length(p)
for n=1:length(p)
if(p(m)>p(n))
          a=p(n);  a1=s(n);
p(n)=p(m);s(n)=s(m);
p(m)=a;    s(m)=a1;
end
end
end
display(p) %arranged prob. in descending order.

tempfinal=[0];
sumarray=[];
w=length(p);
lengthp=[w];
b(i,:)=p;

while(length(p)>2)
tempsum=p(length(p))+p(length(p)-1);
sumarray=[sumarray,tempsum];
 p=[p(1:length(p)-2),tempsum];
 p=sort(p,'descend');
 i=i+1;
 b(i,:)=[p,zeros(1,w-length(p))];
 w1=0;
lengthp=[lengthp,length(p)];

for temp=1:length(p)
if p(temp)==tempsum;
       w1=temp;
end
end
tempfinal=[w1,tempfinal];  % Find the place where tempsum
has been inserted
display(p);
end
```

```
sizeb(1:2)=size(b);
tempdisplay=0;
 temp2=[];

for i= 1:sizeb(2)
  temp2=[temp2,b(1,i)];
end
sumarray=[0,sumarray];
var=[];
  e=1;
for ifinal= 1:sizeb(2)
  code=[s(ifinal),'      ']
for j=1:sizeb(1)
tempdisplay=0;

for i1=1:sizeb(2)
if( b(j,i1)==temp2(e))
tempdisplay=b(j,i1);
end
if(tempdisplay==0 & b(j,i1)==sumarray(j))
tempdisplay=b(j,i1);
end
end
var=[var,tempdisplay];
if tempdisplay==b(j,lengthp(j))        %assign 0 & 1
code=[code,'1'];
elseif tempdisplay==b(j,lengthp(j)-1)
code=[code,'0'];
else
code=[code,''];
end
temp2(e)=tempdisplay;
end
display(code) %display final codeword
    e=e+1;
end
```

Conclusion:

IMPLEMENTATION OF CYCLIC CODE ENCODER USING MATLAB

Aim: To implement cyclic code encoder using MATLAB

Experimental requirements: PC loaded with MATLAB software

Theory: Cyclic codes are of interest and importance because

- They possess rich algebraic structure that can be utilized in a variety of ways.
- They have extremely concise specifications.
- They can be efficiently implemented using simple *shift registers.*
- Many practically important codes are cyclic.

Convolution codes allow to encode streams of data (bits).

Definition A code *C* is cyclic if

(i) *C* is a linear code;

(ii) Any cyclic shift of a codeword is also a codeword, i.e. whenever $a_0,\ldots a_{n-1} \hat{I} C$, then also $a_{n-1}a_0 \ldots a_{n-2} \hat{I} C$.

Procedure:

1. Run MATLAB
2. Open a new script file
3. Write the code for cyclic code technique.
4. Run the code for execution and obtain the necessary results

MATLAB script:

```
clc;
clearall;
sym=input('Enter the Message--- ');
l=length(sym);
x=sort(sym);

a=seqwordcount(sym,'a');      %counting no.of character
h=seqwordcount(sym,'h');   i=seqwordcount(sym,'i');
m=seqwordcount(sym,'m');   s=seqwordcount(sym,'s');
b=seqwordcount(sym,'_');

prob_a=a./l;                        %Finding probability of
each symbol
prob_h=h./l; prob_i=i./l;
```

```
prob_m=m./l; prob_s=s./l; prob_b=b./l;

msg=[1,2,3,4,5,6];
prob=[prob_a,prob_h,prob_i,prob_m,prob_s,prob_b];
dict = huffmandict(msg,prob);          %arrange symbols
according to probability

code_a = huffmanenco(1,dict);         %Huffman coding of
symbols
code_a = (code_a)';
code_h = huffmanenco(2,dict);
code_h= (code_h)';
code_i = huffmanenco(3,dict);
code_i = (code_i)';
code_m = huffmanenco(4,dict);
code_m = (code_m)';
code_s = huffmanenco(5,dict);
code_s = (code_s)';
code_b = huffmanenco(6,dict);
code_b = (code_b)';

for i=1:length(prob)
z(i)=prob(i).*log2(1./prob(i));     %Calculation of
entropy
end
entropy=sum(z)

code_array= [code_i code_b code_a code_m code_b code_a
code_s code_h code_i code_s code_h]
sz= size(code_array);


n=35;   %code vector to b transmitted
k=29;  %message-code_array
p=n-k;  %parity bit
pol=cyclpoly(n,k);
[parmat,genmat]=cyclgen(n,pol);
msg= code_array;
codeword=mod(msg*genmat,2)
trt = syndtable(parmat);    % Produce decoding table.
recd= input('enter the recieved vector- ');
syndrome = rem(recd*parmat',2);
syndrome_de = bi2de(syndrome,'left-msb');     % Convert to
decimal.
errorvect = trt(1+syndrome_de,:);
correctedcode = rem(errorvect+recd,2);
```

```
if recd == codeword
disp('No Error- ')
disp(sym)
else
disp('Error in message....')
end
```

Conclusion:

IMPLEMENTATION OF CONVOLUTIONAL ENCODER USING MATLAB

Aim: To implement convolutional encoder using MATLAB

Experimental requirements: PC loaded with MATLAB software

Theory: A **convolutional code** is a type of error-correcting code in which

- Each *m*-bit information symbol (each *m*-bit string) to be encoded is transformed into an *n*-bit symbol, where *m*/*n* is the code *rate* ($n \geq m$) and
- The transformation is a function of the last *k* information symbols, where *k* is the constraint length of the code.

To convolutionally encode data, start with *k* memory registers, each holding 1 input bit. Unless otherwise specified, all memory registers start with a value of 0. The encoder has *n* modulo-2 adders (a modulo 2 adder can be implemented with a single Boolean XOR gate, where the logic is: 0+0 = 0, 0+1 = 1, 1+0 = 1, 1+1 = 0), and *n*generator polynomials — one for each adder (see figure below). An input bit m_1 is fed into the leftmost register. Using the generator polynomials and the existing values in the remaining registers, the encoder outputs *n* bits. Now bit shift all register values to the right (m_1 moves to m_0, m_0 moves to m_{-1}) and wait for the next input bit. If there are no remaining input bits, the encoder continues output until all registers have returned to the zero state.

Procedure:

1. Run MATLAB
2. Open a new script file
3. Write the code for convolutional coding technique.
4. Run the code for execution and obtain the necessary results

MATLAB script:

```
% This m-file allows the user to input a source code to be
encoded and
% input the values of the generator polynomials.
% It outputs the encoded data bits, where 1/n is the
% code rate.

% the rate is 1/n
% K is the constraint length
% m is the amount of memory
clear
g=[1 1 1;1 0 1];%generator polynomials
[n,K] = size(g);
m = K-1;%number of registers
state = zeros(1,m);%set registers to zero
```

```
inputx=[0 1 0 1 1 1 0 0 1 0 1 0 0 0 1];%encoder input
source code
[trash,h]=size(inputx);
outputy=[];
for x=1:h%h=number of input bits
input=inputx(1,x);
for i=1:n
output(i) = g(i,1)*input;
for j = 2:K
       z=g(i,j)*state(j-1);
output(i) = xor(output(i),z);
end;
end
state = [input, state(1:m-1)];
outputy=[outputy,output];%new element added to sequence
end
outputy%final encoder output in command window
```

Conclusion:

COMPANDING (MU-LAW) IMPLEMENTATION USING MATLAB

Aim: To implement Companding (mu-law) using MATLAB

Experimental requirements: PC loaded with MATLAB software

Theory: **companding**(occasionally called **compansion**) is a method of mitigating the detrimental effects of a channel with limited dynamic range. The name is a portmanteau of**comp**ressing and ex**panding**.

While the compression used in audio recording and the like depends on a variable-gain amplifier, and so is a locally linear process (linear for short regions, but not globally), companding is non-linear and takes place in the same way at all points in time. The dynamic range of a signal is compressed before transmission and is expanded to the original value at the receiver.

The electronic circuit that does this is called a **compandor** and works by compressing or expanding the dynamic range of an analog electronic signal such as sound. One variety is a triplet of amplifiers: a logarithmic amplifier, followed by a variable-gain linear amplifier and an exponential amplifier. Such a triplet has the property that its output voltage is proportional to the input voltage raised to an adjustable power. Compandors are used in concert audio systems and in some noise reduction schemes such as dbx and Dolby NR (all versions).

Procedure:

1. Run MATLAB
2. Open a new script file
3. Write the code for mu-law Companding technique.
4. Run the code for execution and obtain the necessary results

MATLAB script:

```
clearall
closeall
clc
 M=input('enter the signal') %enter the signal with time
like sin(2*pi*[0:0.01:1]
 Mmax=max(M)
 Mn=M/Mmax
 u=input('enter the u value') %default value is 255
 Vn=log(1+u*Mn)/(log(1+u))
figure(1)
plot(Mn)
figure(2)
plot(Vn)
figure(3)
plot(Mn,Vn)
```

Conclusion: