



ACS COLLEGE OF ENGINEERING

(Approved by AICTE, New Delhi, Govt. of Karnataka, Affiliated to
Visvesvaraya Technological University, Belagavi)

#207, Kambipura, Mysore Road, Bengaluru – 560074



DEPARTMENT OF AERONAUTICAL ENGINEERING

AIRCRAFT SYSTEMS AND AVIONICS LAB MANUAL

(BAE701)

(Prescribed for VII – Semester Aeronautical Engineering)



NAME OF THE STUDENT : _____

USN : _____

BRANCH : AERONAUTICAL ENGINEERING

SEMESTER & YEAR : VII & IV

ACADEMIC YEAR : 2025-26 (ODD)

Sl.No.	Experiments	Page No.
1.	Realize basic logic functions using universal gates (7400, 7402, 7408, 7432, 7486, 7404, 7411,7410,7420)	
2.	Design half adder and full adder using basic logic gates and verify the truth table	
3.	Design half subtractor and full subtractor using basic logic gates and verify the truth table.	
4.	Design and implement the encoder and decoder and to verify the truth table.	
5.	Design and implement multiplexer and demultiplexer and to verify the truth table.	
6.	Realize the following shift registers using IC7474/7495 (i) SISO (ii) SIPO (iii) PISO(iv))PIPO (v) Ring (vi) Johnson counter	
7.	Compute Indicated Airspeed for Pitot-Static Airspeed Indicator for cessana aircraft by using MATLAB	
8.	Six-DOF mathematical modelling and simulation of an aircraft and avionics integration	
9.	Study of MIL-STD-1553 B Data Bus	
10.	Study of Pulse Amplitude Modulation (PAM) and Demodulation.	
11.	Study of HAM Radio	
12.	Study of Flip flops	

EXP.NO.	Realize basic logic functions using universal gates (7400, 7402, 7408, 7432, 7486, 7404, 7411,7410,7420)
DATE:	

Aim:

To realize and verify basic logic functions using universal gates by implementing and testing the truth tables of ICs 7400 (NAND), 7402 (NOR), 7408 (AND), 7432 (OR), 7486 (XOR), 7404 (NOT), 7411 (3-input AND), 7410 (3-input NAND), and 7420 (4-input NAND), and to demonstrate the realization of NOT, AND, OR, and XOR functions using only NAND and NOR gates.

Requirements:

S. No	Components	IC number	Quantity
1.	AND gate	IC 7408	1
2.	OR gate	IC 7432	1
3.	NOT gate	IC 7404	1
4.	2-i/pNAND gate	IC 7400	1
5.	NOR gate	IC 7402	1
6.	X-OR gate	IC 7486	1
7.	3i/p NAND gate	IC 7410	1
8.	IC trainer kit	--	1
9.	Connecting wires	--	Required

Concepts involved:**Logic gate:**

The circuit which takes the logic decision is called as logic gates. It has seven types such as OR, AND, NOT, NAND, NOR, XOR, XNOR gates. NAND, NOR, XOR, XNOR are the universal logic gates. Universal gates are the one derived from basic gates. All logic gates have two or more inputs and only one output. But NOT gate has only one input.

AND gate:

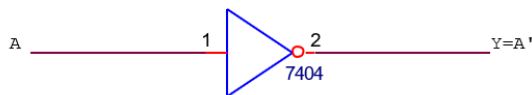
The AND gate executes the logical multiplication known as AND operation. It has HIGH output when all the inputs are HIGH. When any of the input is LOW, then the output

NOT gate:

The NOT gate is called an inverter. It has one input and one output. The output is HIGH when the input is LOW. The output is LOW when the input is HIGH.

If A represents the input and Y represents the output, then

$$Y = \bar{A}$$

NOT gate:**Table 1.3: Truth table of NOT gate**

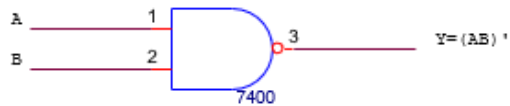
Input	Output
A	Y=A'
0	1
1	0

Fig 1.3: Symbol of NOT gate**NAND gate:**

NAND is a construction of the NOT-AND gates. When all the inputs are HIGH, the output is LOW. If any one or both the inputs are LOW, then the output is HIGH.

If A and B are the input variables of an NAND gate and Y is its output, then

$$Y = \overline{A \cdot B}$$

2- input NAND gate:**Fig 1.4: Symbol of 2-input NAND gate****Table 1.4: Truth table of 2-input NAND gate**

Input		Output
A	B	$Y=(A.B)'$
0	0	1
0	1	1
1	0	1
1	1	0

3- input NAND gate:**Fig 1.5: Symbol of 3-input NAND gate****Table 1.5: Truth table of 3-input NAND gate**

Input			Output
A	B	C	$Y=(A.B.C)'$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

NOR gate:

NOR is a contraction of NOT-OR gates. It has two or more inputs and only one output. The output is HIGH only when all the inputs are Low. If any one or both the inputs are HIGH, then the output is LOW.

If A and B are the input variables of an NOR gate and Y is its output,

$$\text{then } Y = A+B$$

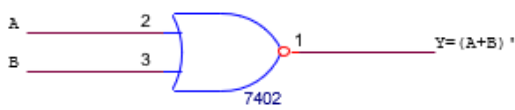


Fig 1.6: Symbol of NOR gate

Table 1.6: Truth table of NOR gate

Input		Output
A	B	$Y=(A+B)'$
0	0	1
0	1	0
1	0	0
1	1	0

XOR Gate:

XOR stands for Exclusive-OR gate. It has two or more inputs and only one output. The output is HIGH when the number of HIGH inputs is odd. For two inputs, the output is HIGH only when one input is HIGH and the other is LOW. If both inputs are LOW or both are HIGH, the output is LOW.

If A and B are the input variables of an XOR gate and Y is its output, then

$$Y=A\oplus B=\bar{A}B+A\bar{B}$$

XOR gate:

Fig 1.7: Symbol of XOR gate

Table 1.7: Truth table of XOR gate

Input		Output
A	B	$Y=A'B+AB'$
0	0	0
0	1	1
1	0	1
1	1	0

7411 – Triple 3-Input AND Gate

7411 is a **Triple 3-input AND Gate** IC. It has three independent AND gates, each with three inputs and one output. The output is HIGH only when **all three inputs are HIGH**. If any one or more inputs are LOW, the output becomes LOW.

If **A, B, and C** are the input variables of a 3-input AND gate and **Y** is its output, then:

$$Y=A\cdot B\cdot C$$

A	B	C	Y = A·B·C
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

7420 – Dual 4-Input NAND Gate

7420 is a **Dual 4-input NAND Gate** IC. It has two independent NAND gates, each with four inputs and one output. The output is LOW only when **all four inputs are HIGH**. If any one or more inputs are LOW, the output is HIGH.

If **A, B, C, and D** are the input variables of a 4-input NAND gate and **Y** is its output, then:

$$Y = \overline{A \cdot B \cdot C \cdot D}$$

A	B	C	D	Y = (A·B·C·D)'
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

RESULT:

Basic logic functions were successfully verified using the given ICs, and NOT, AND, OR, and XOR were correctly realized with universal gates.

EXP.NO.	HALF ADDER AND FULL ADDER
DATE:	

AIM

To design and implement Half Adder and Full Adder using basic logic gates and verify their truth tables.

APPARATUS REQUIRED

Digital IC trainer kit	-	1
IC7408 (AND Gate)	-	1
IC7486 (XOR Gate)	-	1
IC7432 (OR Gate)	-	1
Connecting Wires	-	as required

THEORY

An adder is a digital circuit that performs addition of numbers. In many computers and other kinds of processors adders are used in the arithmetic logic units or ALU. They are also utilized in other parts of the processor, where they are used to calculate addresses, table indices, increment and decrement operators, and similar operations.

The half adder adds two single binary digits A and B. It has two outputs, sum (S) and carry (C).

The carry signal represents an overflow into the next digit of a multi-digit addition.

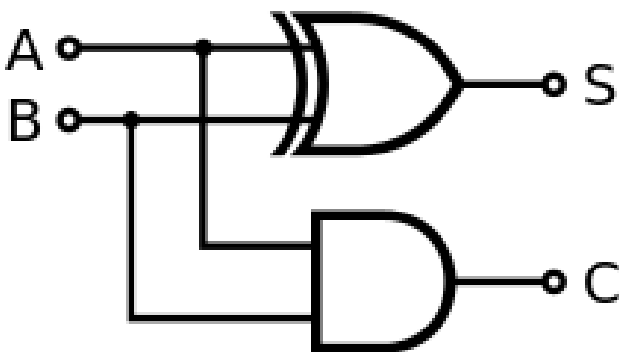
A full adder adds binary numbers and accounts for values carried in as well as out. A one-bit full adder adds three one-bit numbers, often written as A, B, and C_{in} ; A and B are the operands, and C_{in} is a bit carried in from the previous less-significant stage. The full adder is usually a component in a cascade of adders, which add 8, 16, 32, etc. bit binary numbers.

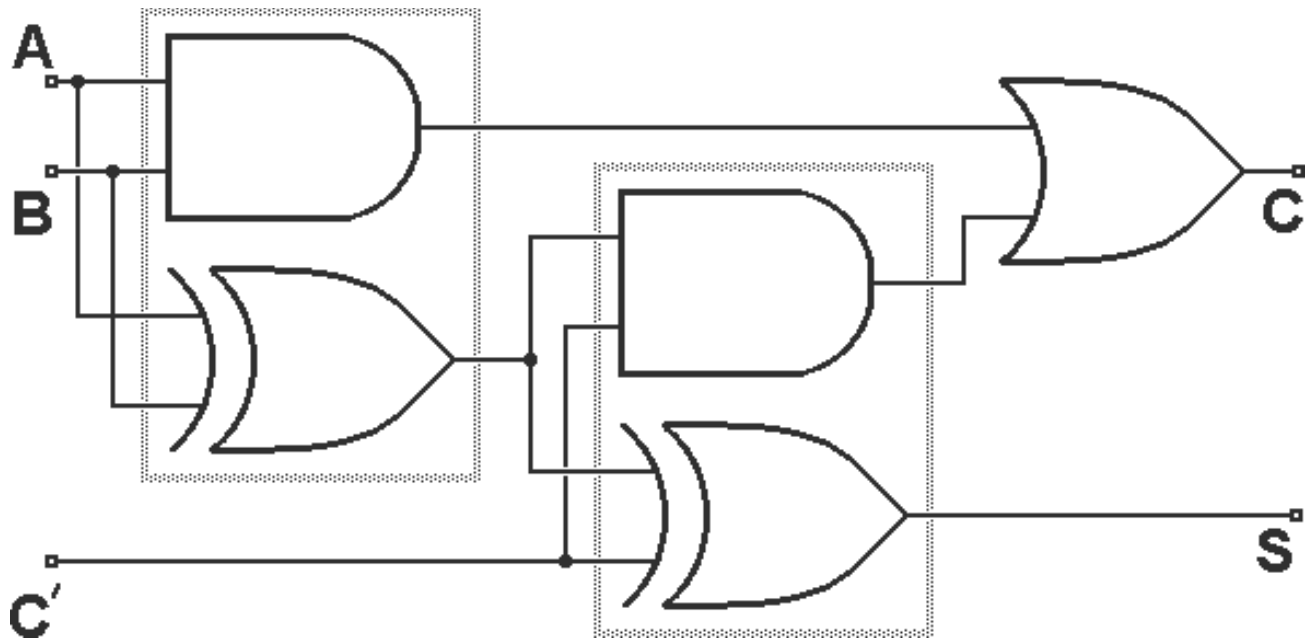
PROCEDURE**For Half Adder:**

- Identify the required ICs (e.g., 7486 – XOR, 7408 – AND) and fix them on the digital IC trainer kit.
- Make the connections as per the circuit diagram using patch cords.
- Connect Vcc (+5V) and GND pins of all ICs properly.
- Apply the first input combination ($A = 0, B = 0$) using the input switches of the trainer kit.
- Observe the outputs at the Sum (XOR gate output) and Carry (AND gate output) pins using LEDs or logic probe. Record the values.
- Change the inputs systematically for all possible combinations of A and B (00, 01, 10, 11).
- Note down the corresponding Sum and Carry values in the observation table.
- Compare the observed results with the theoretical truth table of the Half Adder.

For Full Adder:

- Identify the required ICs (e.g., 7486 – XOR, 7408 – AND, 7432 – OR) and place them on the kit.
- Connect the circuit as per the Full Adder circuit diagram (using two Half Adders and one OR gate).
- Provide power supply connections (Vcc and GND) to all ICs.
- Apply the first input combination ($A = 0, B = 0, C_{in} = 0$) through the input switches.
- Observe the outputs of Sum and Carry. Record the values.
- Vary the inputs systematically to cover all 8 possible input combinations of A, B, and C_{in} .
- Note down the Sum and Carry outputs for each case.
- Compare the observed values with the theoretical truth table of the Full Adder and verify correctness.

CIRCUIT DIAGRAM:**HALF ADDER**

FULL ADDER**TRUTH TABLE****HALF ADDER**

A	B	SUM, S	CARRY, C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

FULL ADDER

A	B	Cin	SUM, S	CARRY, C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

RESULT

Thus the Half Adder and Full Adder were successfully designed and the outputs matched the expected truth tables.

EXP.NO.	HALF SUBTRACTOR AND FULL SUBTRACTOR
DATE:	

AIM

To design half subtractor and full subtractor using basic logic gates and to verify the truth table.

APPARATUS REQUIRED

Digital IC trainer kit	-	1
IC7408 (AND Gate)	-	1
IC7486 (XOR Gate)	-	1
IC7432 (OR Gate)	-	1
IC7404 (NOT Gate)	-	1
Connecting Wires	-	as required

THEORY

A subtractor is a digital circuit that performs subtraction of numbers. In many computers and other kinds of processors subtractors are used in the arithmetic logic units or ALU. They are also utilized in other parts of the processor, where they are used to calculate addresses, table indices, increment and decrement operators, and similar operations.

The half Subtractor subtracts two single binary digits A and B. It has two outputs, difference (D) and borrow (B₀). The borrow signal represents an overflow into the next digit of a multi-digit subtraction.

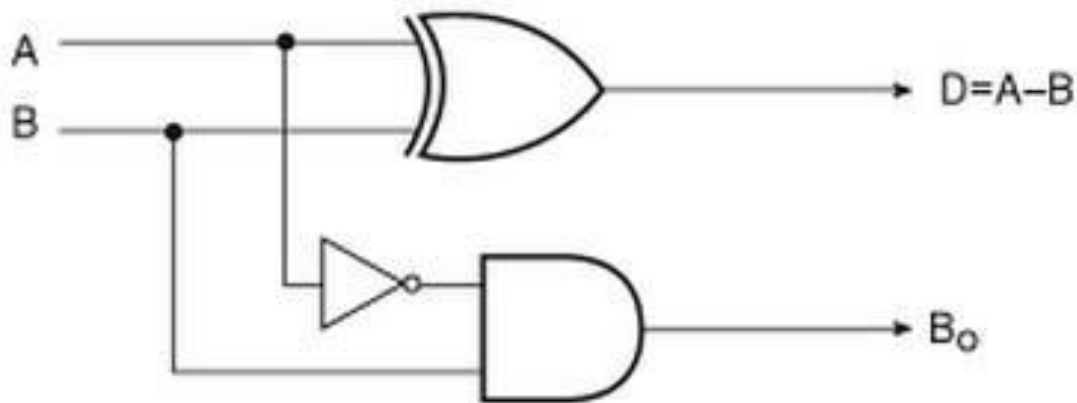
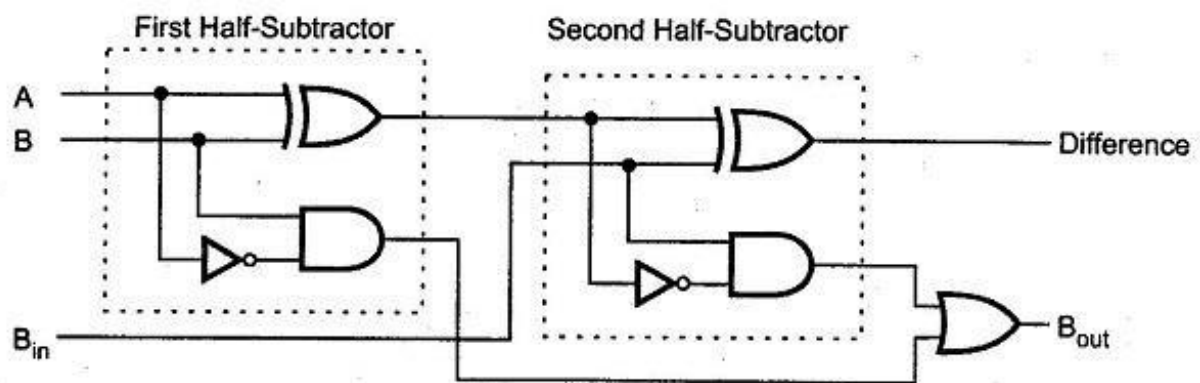
A full Subtractor subtracts binary numbers and accounts for values borrowed in as well as out. A one-bit full Subtractor subtracts three one-bit numbers, often written as A, B, and C_{in}; A and B are the operands, and C_{in} is a bit carried in from the previous less-significant stage. The full subtractor is usually a component in a cascade of subtractors, which subtract 8, 16, 32, etc. bit binary numbers.

PROCEDURE**For Half Subtractor:**

- Identify the required ICs (e.g., 7486 – XOR, 7404 – NOT, 7408 – AND) and fix them on the digital IC trainer kit.
- Make the connections as per the Half Subtractor circuit diagram using patch cords.
- Connect Vcc (+5V) and GND pins of all ICs properly.
- Apply the first input combination ($A = 0, B = 0$) using the input switches of the trainer kit.
- Observe the outputs at the Difference (XOR gate output) and Borrow ($\bar{A} \cdot B$) pins using LEDs or logic probe. Record the values.
- Change the inputs systematically for all possible combinations of A and B (00, 01, 10, 11).
- Note down the corresponding Difference and Borrow values in the observation table.
- Compare the observed results with the theoretical truth table of the Half Subtractor.

For Full Subtractor:

- Identify the required ICs (e.g., 7486 – XOR, 7404 – NOT, 7408 – AND, 7432 – OR) and place them on the kit.
- Connect the circuit as per the Full Subtractor circuit diagram (using two Half Subtractors and an OR gate).
- Provide power supply connections (Vcc and GND) to all ICs.
- Apply the first input combination ($A = 0, B = 0, Bin = 0$) through the input switches.
- Observe the outputs of Difference and Borrow. Record the values.
- Vary the inputs systematically to cover all 8 possible input combinations of A, B, and Bin.
- Note down the Difference and Borrow outputs for each case.
- Compare the observed values with the theoretical truth table of the Full Subtractor and verify correctness.

CIRCUIT DIAGRAM - HALF SUBTRACTOR**FULL SUBTRACTOR****TRUTH TABLE :****HALF SUBTRACTOR**

A	B	DIFFERENCE, D	BORROW, B0
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

FULL SUBTRACTOR

A	B	Cin	DIFFERENCE, D	BORROW, B0
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

RESULT

Thus, the half subtractor and full subtractor were designed using basic logic gates and the output was verified.

EXP.NO.	ENCODER AND DECODER
DATE:	

AIM

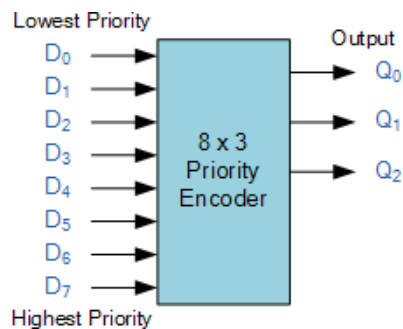
To study the encoder and decoder and their operation

APPARATUS REQUIRED

Encoder & Decoder trainer kit	-	1
Connecting Wires	-	as required

THEORY**ENCODER**

An encoder is a digital circuit that performs inverse operation of a decoder. An encoder has 2^n input lines and n output lines. In encoder the output lines generate the binary code corresponding to the input value. In octal to binary encoder it has eight inputs, one for each octal digit and three outputs that generate the corresponding binary code. In encoder it is assumed that only one input has a value of one at any given time otherwise the circuit is meaningless. Here, when all inputs are zero the outputs

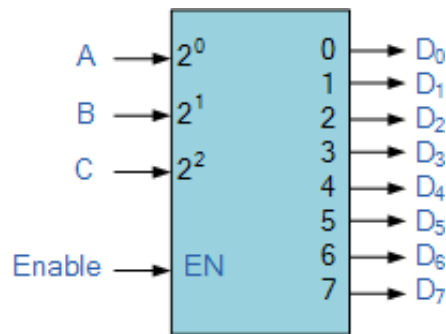


are zero. The zero outputs can also be generated when $D_0 = 1$.

DECODER

A decoder is a multiple input multiple output logic circuit which converts coded input into coded output where input and output codes are different. The input code generally has fewer bits than the output code. Each input code word produces a different output code word i.e there is one to one mapping can be expressed in truth table. In the block diagram of decoder circuit the encoded information is present as n input producing 2^n possible outputs. 2^n output values are from 0 through

output 2^n-1

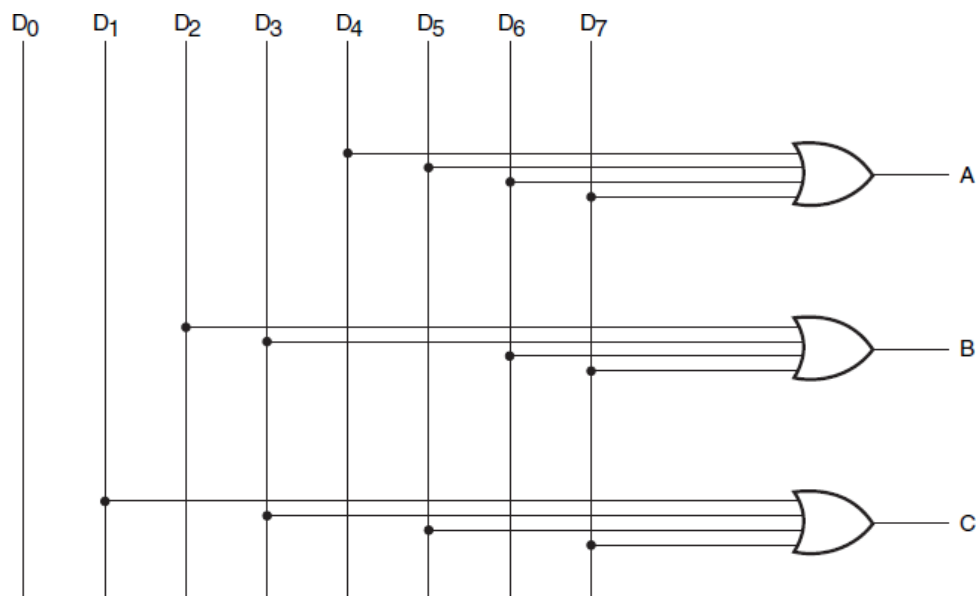


PROCEDURE

1. Make the connections as per the circuit diagram using the Universal Shift Registers trainer kit and connecting wires
2. Switch ON the power supply
3. Note down the output values for various input combinations
4. Verify the output

CIRCUIT DIAGRAM

ENCODER



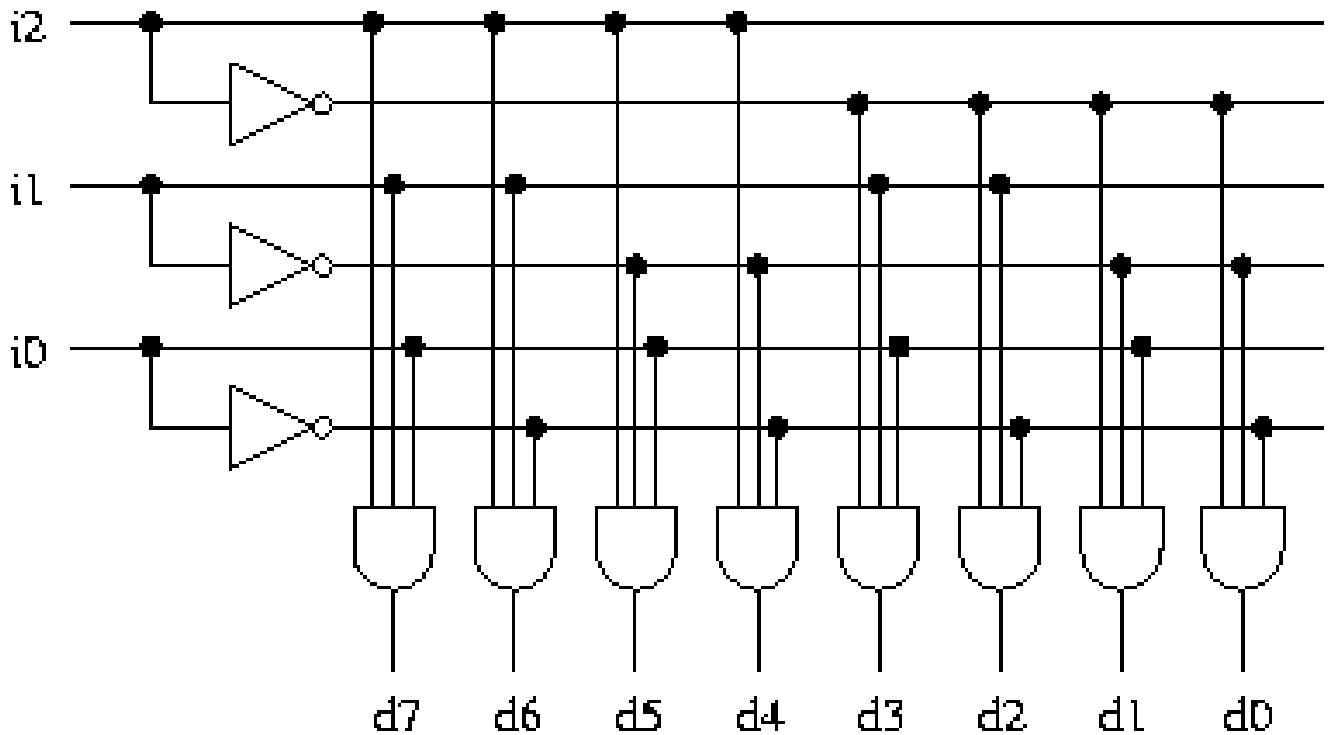
DECODER

TRUTH TABLE

ENCODER

INPUT								OUTPUT		
D7	D6	D5	D4	D3	D2	D1	D0	Q2	Q1	Q0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	X	0	0	1
0	0	0	0	0	1	X	X	0	1	0
0	0	0	0	1	X	X	X	0	1	1
0	0	0	1	X	X	X	X	1	0	0
0	0	1	X	X	X	X	X	1	0	1
0	1	X	X	X	X	X	X	1	1	0
1	X	X	X	X	X	X	X	1	1	1

DECODER



INPUT			OUTPUT							
Q2	Q1	Q0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

RESULT

Thus, the encoder and decoder were studied and the output was verified.

EXP.NO.	MULTIPLEXER AND DEMULTIPLEXER
DATE:	

AIM

To design and implement multiplexer and demultiplexer and to verify the truth table.

APPARATUS REQUIRED

Digital IC trainer kit	-	1
IC74151 (MUX)	-	1
IC74138 (DEMUX)	-	1
Mono Pulse Generator	-	1
Address Generator	-	1
Connecting Wires	-	as required

THEORY

The multiplexer, shortened to “MUX” or “MPX”, is a combinational logic circuit designed to switch one of several input lines through to a single common output line by the application of a control signal. Multiplexers operate like very fast acting multiple position rotary switches connecting or controlling multiple input lines called “channels” one at a time to the output.

Multiplexers, or MUX’s, can be either digital circuits made from high speed logic gates used to switch digital or binary data or they can be analogue types using transistors, MOSFET’s or relays to switch one of the voltage or current inputs through to a single output.

In digital electronics, multiplexers are also known as data selectors because they can “select” each input line, are constructed from individual Analogue Switches encased in a single IC package as opposed to the “mechanical” type selectors such as normal conventional switches and relays.

They are used as one method of reducing the number of logic gates required in a circuit design or when a single data line or data bus is required to carry two or more different digital signals.

The demultiplexer takes one single input data line and then switches it to any one of a number

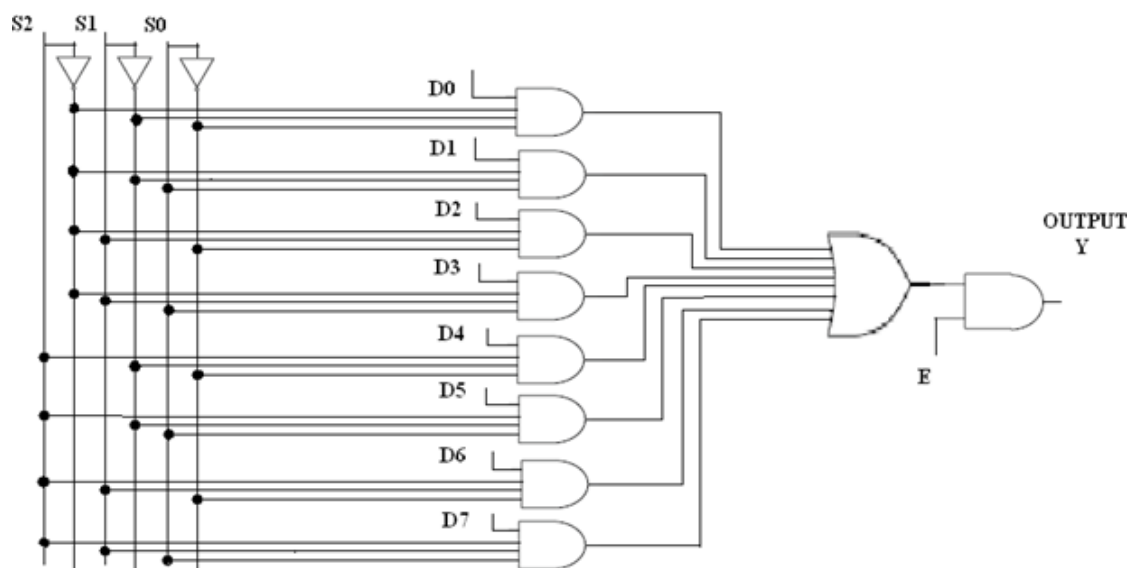
of individual output lines one at a time. The demultiplexer converts a serial data signal at the input to a parallel data at its output lines.

PROCEDURE

1. Make the connections as per the circuit diagram using the digital IC trainer kit and connecting wires
2. Switch ON the power supply
3. Note down the output values for various input combinations
4. Verify the truth table

CIRCUIT DIAGRAM

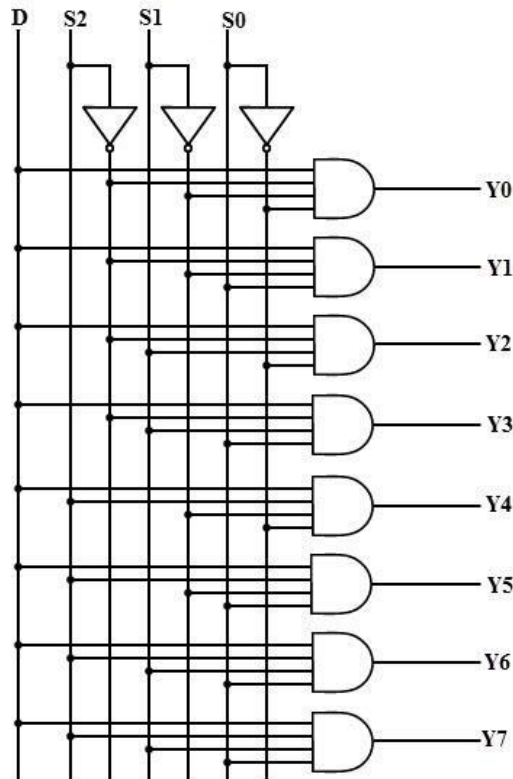
MULTIPLEXER



DEMULTIPLEXER

TRUTH TABLE

MULTIPLEXER



S2	S1	S0	OUTPUT Y
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

DEMULTIPLEXER

Data Input	Select Inputs			Outputs							
D	S ₂	S ₁	S ₀	Y ₇	Y ₆	Y ₅	Y ₄	Y ₃	Y ₂	Y ₁	Y ₀
D	0	0	0	0	0	0	0	0	0	0	D
D	0	0	1	0	0	0	0	0	0	D	0
D	0	1	0	0	0	0	0	0	D	0	0
D	0	1	1	0	0	0	0	D	0	0	0
D	1	0	0	0	0	0	D	0	0	0	0
D	1	0	1	0	0	D	0	0	0	0	0
D	1	1	0	0	D	0	0	0	0	0	0
D	1	1	1	D	0	0	0	0	0	0	0

RESULT

Thus, MUX and DEMUX were designed and implemented and their output was verified.

EXP.NO.	COMPUTATION OF INDICATED AIRSPEED USING MATLAB
DATE:	

Aim

To compute the Indicated Airspeed (IAS) of a Cessna aircraft using pitot-static pressure values by implementing a MATLAB script.

Apparatus Required

- Computer system with MATLAB installed
- Standard atmosphere reference data (ISA table)
- Sample pitot and static pressure values
- Calculator or coding environment for validation

Theory

In aviation, airspeed is a critical parameter used for flight control, safety, and navigation. The airspeed that is directly measured by an aircraft's pitot-static system and displayed on the airspeed indicator is known as Indicated Airspeed (IAS). This value does not account for instrument errors, altitude, or compressibility effects, but is essential for maintaining proper aircraft handling characteristics.

The pitot-static system measures two types of pressure:

- Pitot Pressure: Total pressure from the aircraft's forward motion
- Static Pressure: Ambient atmospheric pressure

The difference between the two is the Dynamic Pressure (ΔP):

$$\Delta P = P_{\text{pitot}} - P_{\text{static}}$$

According to Bernoulli's principle, the relationship between dynamic pressure and velocity is: ‘

$$IAS = \sqrt{\frac{2 \times q}{\rho}}$$

Where:

- ΔP is the dynamic pressure (in Pascals)

- ρ is the air density (in kg/m^3), typically 1.225 at sea level
- IAS is the indicated airspeed in meters per second (m/s)

This method assumes incompressible flow and standard atmospheric conditions. While True Airspeed (TAS) may differ significantly at higher altitudes, IAS is still used for structural limits, stall speed monitoring, and aircraft certification. It is also the basis for maneuvering speed (V_a), approach speeds, and more.

Procedure

- Open MATLAB software on your system.
- Click on "**New Script**" from the toolbar.
- Define the input values in the script:
 - Pitot Pressure = 101325 Pa
 - Static Pressure = 100000 Pa
 - Air Density (ρ) = 1.225 kg/m^3
- Calculate the dynamic pressure using the formula:

$$\Delta P = \text{Pitot Pressure} - \text{Static Pressure} = 1325 \text{ Pa}$$
- Compute the Indicated Airspeed (IAS) using the formula:

$$\text{IAS} = \sqrt{(2 \times \Delta P) / \rho}$$
- Write the code to perform the above calculations.
- Display the IAS result using the disp() function.
- Convert the IAS value from m/s to knots using the conversion factor:

$$1 \text{ m/s} = 1.94384 \text{ knots}$$
- Display the IAS value in both m/s and knots.

Sample MATLAB Code

% Indicated Airspeed Calculation using Pitot-Static Pressures

```

rho = 1.225; % Air density in kg/m3
Pitot = 101325; % Pitot pressure in Pa
Static = 100000; % Static pressure in Pa

dp = Pitot - Static; % Dynamic pressure
IAS = sqrt((2 * dp) / rho); % Compute IAS in m/s
IAS_knots = IAS * 1.94384; % Convert IAS to knots

% Display Results

disp(['Indicated Airspeed (m/s): ', num2str(IAS)]);
disp(['Indicated Airspeed (knots): ', num2str(IAS_knots)]);

```

- Go to **File > Save As**
- Save it with a name like calculate_IAS.m
- Click the green "**Run**" button at the top.
- The results will appear in the **Command Window**.

Expected Output

If Pitot = 101325 Pa and Static = 100000 Pa, the output might be:

Indicated Airspeed (m/s): 46.5051

Indicated Airspeed (knots): 90.3877

This output helps validate the accuracy of the MATLAB implementation and confirms the understanding of the underlying aerodynamic principle.

Observation / Output Table

Parameter	Value	Unit
Pitot Pressure	101325	Pa
Static Pressure	100000	Pa
Dynamic Pressure (ΔP)	1325	Pa
Air Density (ρ)	1.225	kg/m ³
IAS (calculated)	46.5051	m/s
IAS (converted)	90.3877	knots

Result

The indicated airspeed of the Cessna aircraft was successfully computed in MATLAB using pitot-static data, confirming Bernoulli's principle and reinforcing key concepts in aerodynamics and instrumentation.

Viva Questions

- What is the difference between Indicated Airspeed and True Airspeed?
- Why is Indicated Airspeed used for structural limitations of aircraft?
- Explain the role of the pitot-static system.
- What assumptions are made in the IAS formula?
- How does air density affect IAS?
- Why is MATLAB used in such computations?
- What happens to IAS as altitude increases (assuming constant true airspeed)?
- Define dynamic pressure.
- What are the limitations of the pitot-static system?
- How is IAS converted to knots?

EXP.NO.	STUDY OF PAM
DATE:	

Generation of PAM

Pulse amplitude modulation is the basic form of pulse modulation in which the signal is sampled at regular and each sample is made proportional to the amplitude of the modulating signal at the sampling instant.

The Fig1 shows the generation of PAM signal from the sampler which has two inputs i.e. modulating signal and sampling signal or carrier pulse.

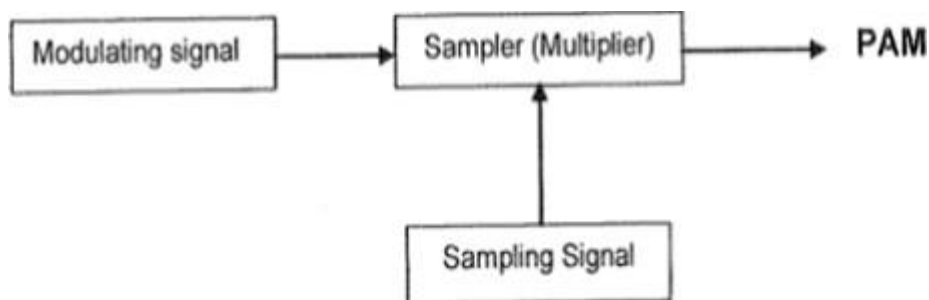


Fig1. Generation of PAM signal

Thus the amplitude of the signal is proportional to the modulating signal through which information is carried. This is Pulse amplitude modulation signal.

Fig2 shows the spectrum of pulse amplitude modulated signal along with the message signal and the sampling signal which is the carrier train of pulses with the help of the waveform plotted in time domain.

Pulse Modulation may be used to transmitting analog information, such as continuous speech signal or data.

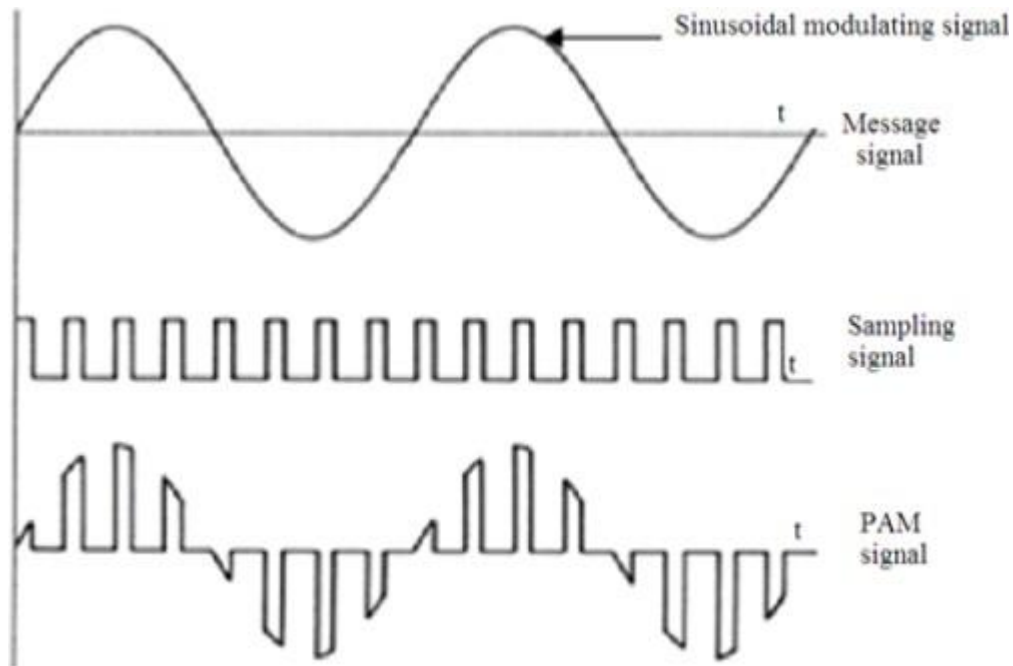


Fig2. Spectrum of PAM signal

Demodulation of PAM

For Demodulation of the Pulse Amplitude Modulated signal, PAM is fed to the low pass filter as shown in Fig3 below.

The low pass filter eliminates high frequency ripples and generates the demodulated signal which has its amplitude proportional to PAM signal at all time instant.

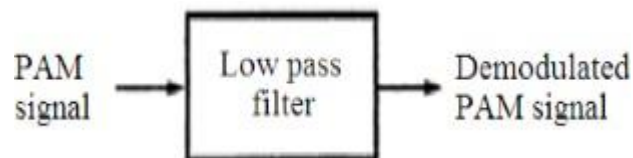


Fig3. PAM detector

This signal is then applied to an inverting amplifier to amplify its signal level to have the demodulated output with almost equal amplitude with the modulating signal.

The Fig4 below shows the modulated and demodulated PAM signal.

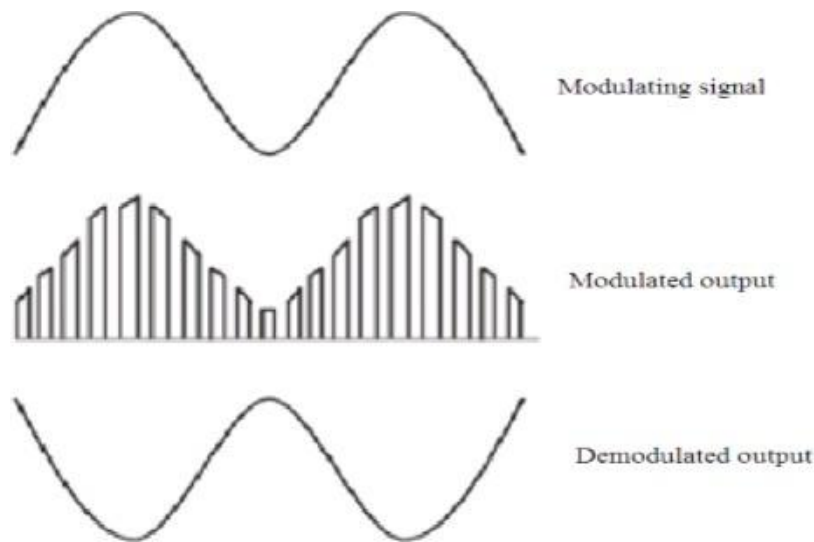


Fig4. Modulation and demodulation of PAM

Generation of PWM

PWM signal can be generated by using a comparator, where modulating signal and sawtooth signal form the input of the comparator. It is the simplest method for PWM generation.

The PWM generation is explained with the help of the Fig5 given below.

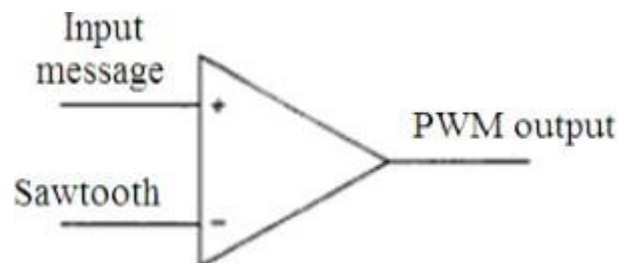


Fig5. PWM generation by a comparator

As shown in the figure, one input of the comparator is fed by the input message or modulating signal and the other input by a sawtooth signal which operates at carrier frequency.

Considering both \pm ve sides, the maximum of the input signal should be less than that of sawtooth signal.

The comparator will compare the two signals together to generate the PWM signal at its output as shown in the third waveform of Fig6.

The rising edges of the PWM signal coincides with the falling edge of the sawtooth signal.

When the sawtooth signal is at the minimum value which is less than the minimum of the input signal, then the positive input of the comparator is at higher potential which gives the comparator output as positive.

When the sawtooth signal rises and is at the maximum value, the negative input of the comparator is at higher potential, which will produce the comparator output to be negative.

Thus the input signal magnitude determines the comparator output and its potential, which then decides the width of the pulse generated at the output.

In other words we can say that the width of the pulse generated signal is directly proportional to the amplitude of the modulating signal.

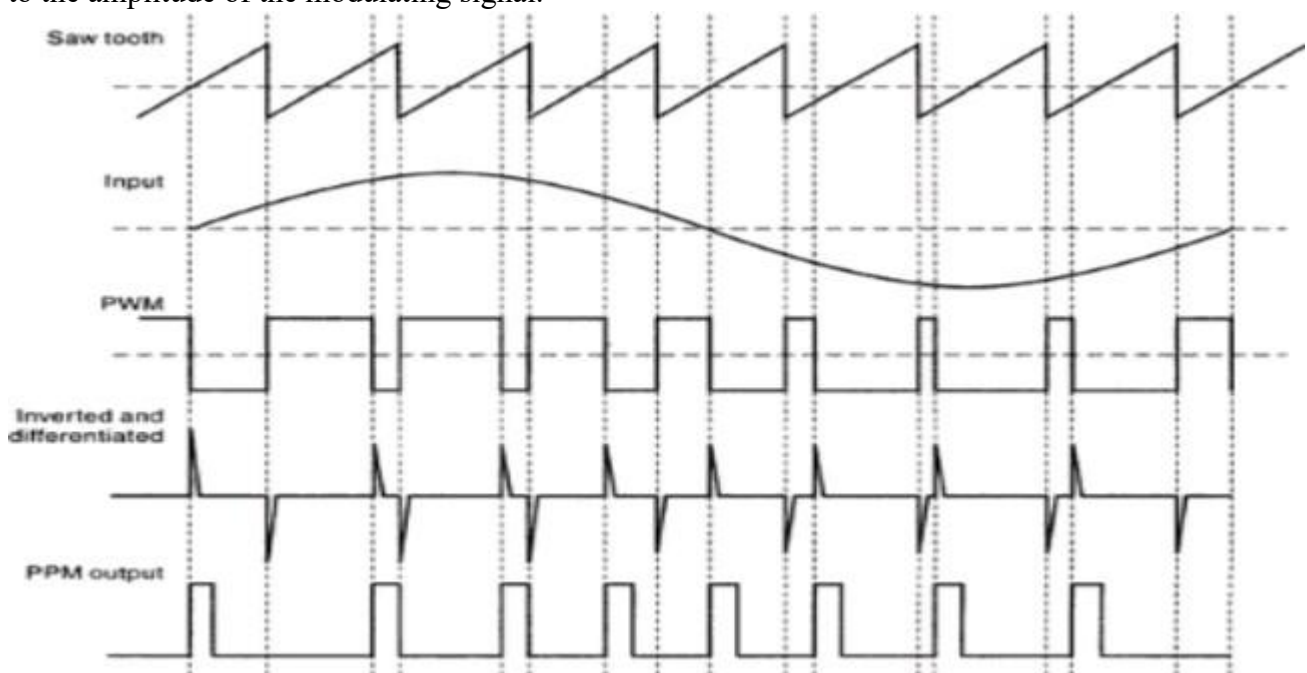


Fig6. PWM and PPM signal generation

Generation of PPM

PPM signal can be generated with the help of PWM as shown in Fig7 below.

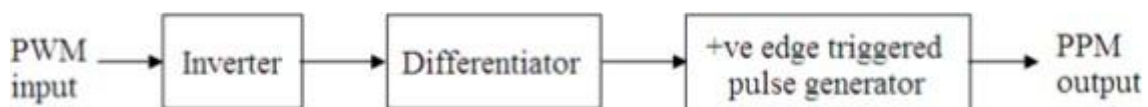


Fig7. PPM generation from PWM

The PWM signal generated above is sent to an inverter which reverses the polarity of the pulses.

This is then followed by a differentiator which generates +ve spikes for PWM signal going from High to Low and -ve spikes for Low to High transistion. The spikes generated are shown in the fourth waveform of Fig8.

These spikes are then fed to the positive edge triggered pulse generator which generates fixed width pulses when a +ve spike appears, coinciding with the falling edge of the PWM signal.

Thus PPM signal is generated at the output which is shown in the fifth waveform of Fig8. where pulse position carry the message information.

Demodulation of PWM and PPM

For PWM demodulation, put a ramp at the +ve edge which will stop at the arrival of -ve edge.

The ramp will attain different heights in each cycle since the widths are different and the heights attained are directly proportional to the pulse width and in turn the amplitude of the message signal.

This is then passed through a low pass filter where it will follow the envelop i.e. the message signal, which produces the demodulated signal at the output.

For PPM demodulation, ramp is used which starts at the +ve edge of the one pulse and stops at the +ve edge of the next pulse.

Thus the height of the generated ramp is determined by the delay between the pulses which indirectly follows the amplitude of the modulating signal.

This is then passed through a low pass filter which filters the envelop information as the demodulated signal.

The modulation and demodulation waveforms of PWM and PPM signals are shown in Fig8.

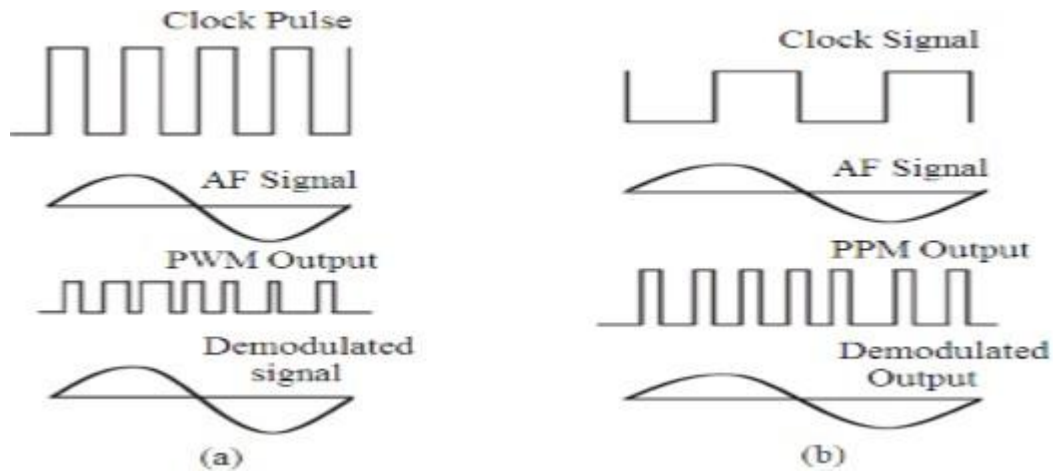


Fig8. Modulation and Demodulation of (a) PWM and (b) PPM

EXP.NO.	STUDY OF MIL STD 1553B DATA BUS
DATE:	

MIL STD 1553B Data Bus

MIL-STD-1553 is a military standard that defines the electrical and protocol characteristics for a data bus. A data bus is used to provide a medium for the exchange of data and information between various systems. It is similar to what the personal computer and office automation industry has dubbed a Local Area Network (LAN).

MIL-STD-1553B defines the term Time Division Multiplexing (TDM) as “the transmission of information from several signal sources through one communications system with different signal samples staggered in time to form a composite pulse train.” This means that data can be transferred between multiple avionics units over a single transmission media, with the communications between the different avionics boxes taking place at different moments in time, hence time division.

Elements of MIL STD 1553B Data Bus

They are:

- The transmission media.
- Remote terminals.
- Bus controllers.
- Bus monitors.

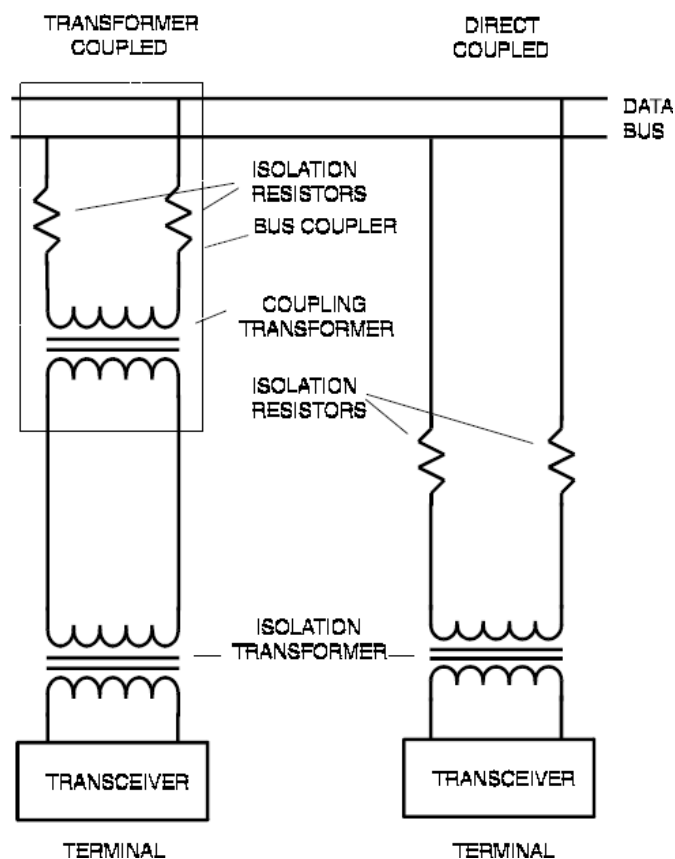
Transmission Media

The transmission media, or data bus, is defined as a twisted shielded pair transmission line consisting of the main bus and a number of stubs. There is one stub for each terminal connected to the bus. The main data bus is terminated at each end with a resistance equal to the cable's characteristic impedance (plus or minus two percent). This termination makes the data bus behave electrically like an infinite transmission line.

Coupling Method

There are two types of coupling, (i) direct coupling and (ii) transformer coupled

The primary difference between the two is that the transformer coupled method uses an isolation transformer for connecting the stub cable to the main bus cable. In both methods, two isolation resistors are placed in series with the bus. In the direct-coupled method, the resistors are typically located within the terminal, whereas in the transformer-coupled method, the resistors are typically located with the coupling transformer in boxes called data bus couplers.



Another difference between the two coupling methods is the length of the stub. For the direct-coupled method, the stub length is limited to a maximum of one foot. For the transformer-coupled method, the stub can be up to a maximum of twenty feet long. Therefore for direct-coupled systems, the data bus must be routed in close proximity to each of the terminals, whereas for a transformer-coupled system, the data bus may be up to twenty feet away from each terminal.

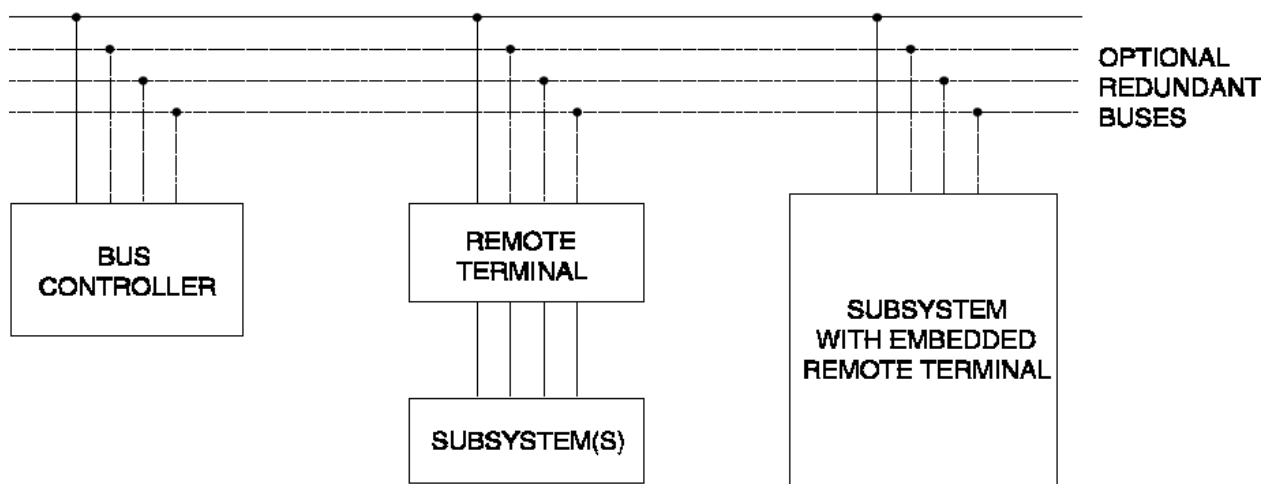
Remote Terminals

Remote terminals are defined within the standard as “All terminals not operating as the bus controller or as a bus monitor”. Therefore if it is not a controller, monitor, or the main bus or stub, it must be a remote terminal. The remote terminal comprises the electronics necessary to transfer data between the data bus and the subsystem.

A remote terminal typically consists of a transceiver, an encoder/decoder, a protocol controller, a buffer or memory, and a subsystem interface. In a modern black box containing a computer or processor, the subsystem interface may consist of the buffers and logic necessary to interface to the computer's address, data, and control buses.

But a remote terminal must be more than just a data formatter. It must be capable of receiving and decoding commands from the bus controller and responding accordingly. It must also be capable of buffering a message worth of data, detecting transmission errors and performing validation tests upon the data, and reporting the status of the message transfer.

A remote terminal must follow the protocol defined by the standard. It can only respond to commands received from the bus controller (i.e., it speaks only when spoken to). When it receives a valid command, it must respond within a very small, closely defined amount of time. If a message



doesn't meet the validity requirements defined, then the remote terminal must invalidate the message and discard the data (not allow it to be used by the subsystem). In addition to reporting status to the bus controller, most remote terminals today are capable of providing some level of status information to the subsystem.

Bus Controller

The bus controller is responsible for directing the flow of data on the data bus. While several

terminals maybe capable of performing as the bus controller, only one bus controller may be active at a time. The bus controller is the only one allowed to issue commands onto the data bus.

The commands may be for the transfer of data or the control and management of the bus (referred to as mode commands). Typically, the bus controller is a function that is contained within some other computer, such as a mission computer, a display processor, or a fire control computer.

There are three types of bus controller architectures,

- Word Controller – transfers one word at a time (old)
- Message Controller – transfers one message at a time
- Frame Controller – capable of processing multiple messages in a sequence specified by a hostcomputer (latest)

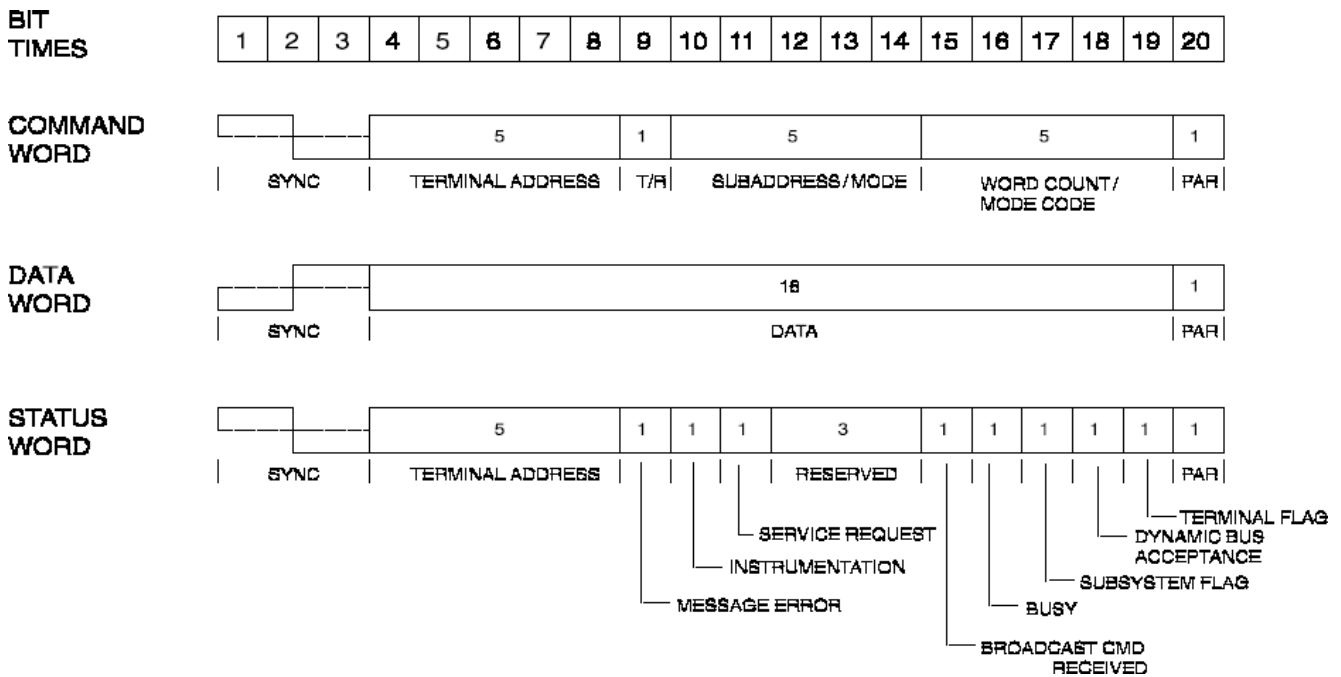
Bus Monitor

A bus monitor is a terminal that listens (monitors) to the exchange of information on the data bus. The standard strictly defines how bus monitors may be used, stating that the information obtained by a bus monitor be used “for off-line applications (e.g., flight test recording, maintenance recording or mission analysis) or to provide the back-up bus controller sufficient information to take over as the bus controller.”

Word Types

- Command Word
- Data Word

- Status Word



Command Word

The Command Word (CW) specifies the function that a remote terminal is to perform. Only the active buscontroller transmits this word. The word begins with command sync in the first three bit times. Five bit Terminal Address (TA) field (bit times 4-8) states to which unique remote terminal the command is intended (no two terminals may have the same address).

The next bit (bit time 9) makes up the Transmit/Receive (T/R) bit. This defines the direction of information flow and is always from the point of view of the remote terminal. A transmit command (logic 1) indicates that the remote terminal is to transmit data, while a receive command (logic 0) indicates that the remote terminal is going to receive data. The only exceptions to this rule are associated with mode commands.

The next five bits (bit times 10-14) make up the Subaddress (SA)/Mode Command bits. Logic 00000B or 11111B within this field is decoded to indicate that the command is a Mode Code Command.

The next five bit positions (bit times 15-19) define the Word Count (WC) or Mode Code to be performed. If the Subaddress/Mode Code field is 00000B or 11111B, then this field defines the mode code to be performed. If not a mode code, then this field defines the number of data words to be received or transmitted depending on the T/R bit. A word count field of 00000B is decoded as 32 data words.

The last bit (bit time 20) is the word parity bit. Only odd parity is used.

Data Word

The Data Word (DW) contains the actual information that is being transferred within a message. The first three-bit time contains data sync. This sync pattern is the opposite of that used for command and status words and therefore is unique to the word type. Data words can be transmitted by either a remote terminal (transmit command) or a bus controller (receive command).

The next sixteen bits of information are left to the designer to define. The last bit is parity.

Status Word

A remote terminal in response to a valid message transmits only the status word (SW). The status word is used to convey to the bus controller whether a message was properly received or to convey the state of the remote terminal (i.e., service request, busy, etc.).

The first five bits (bit times 4-8) of the information field are the Terminal Address (TA).

The next bit (bit time 9) is the Message Error (ME) bit. This bit is set by the remote terminal upon detection of an error in the message or upon detection of an invalid message (i.e. Illegal Command) to the terminal.

The Instrumentation bit (bit time 10) is provided to differentiate between a command word and a status word (remember they both have the same sync pattern). The instrumentation bit in the status word is always set to logic "0". If used, the corresponding bit in the command word is set to logic 1.

The Service Request bit (bit time 11) is provided so that the remote terminal can inform the bus controller that it needs to be serviced. This bit is set to logic "1" by the subsystem to indicate that servicing is needed.

Bit times 12-14 are reserved for future growth of the standard and must be set to logic "0". The bus controller should declare a message in error if the remote terminal responds with any of these bits set in its status word.

The Broadcast Command Received bit (bit time 15) indicates that the remote terminal received a valid broadcast command. On receiving a valid broadcast command, the remote terminal sets this bit to logic "1" and suppresses the transmission of its status words.

The Busy bit (bit time 16) is provided as a feedback to the bus controller as to when the remote terminal is unable to move data between the remote terminal electronics and the subsystem in

compliance to a command from the bus controller.

The Dynamic Bus Control Acceptance bit (bit time 18) informs the bus controller that the remote terminal has received the Dynamic Bus Control Mode Code and has accepted control of the bus. The bus controller, on receiving the status word from the remote terminal with this bit set, ceases to function as the bus controller and may become a remote terminal or bus monitor.

The Terminal Flag bit (bit time 19) informs the bus controller of a fault or failure within the remote terminal circuitry (only the remote terminal). A logic “1” shall indicate a fault condition.

The last bit (bit time 20) is used for parity.