# Double Layer Winding Scheme (Code)

In [1]:
```
# Import some important libraries.
import math

# Define Global Variables and take inputs from the user as of now.
number_of_phases=3
number_of_slots = int(input("Enter the number of slots: "))
number_of_poles = int(input("Enter the number of poles: "))
```

```
Enter the number of slots: 48
Enter the number of poles: 8
```

In [2]:
```
# Calculate internal parameters
slot_pitch_mech = 360 / number_of_slots
number_of_slots_per_pole_per_phase = number_of_slots/(number_of_poles*number_

# Calculate coil offset
for q in range(1,1000):
    coil_offset= (2 / 3) * (number_of_slots/number_of_poles) * (1+3*q)
    if coil_offset.is_integer():
        break
```

In [3]:
```
# conditions for existance of double layer winding
if (number_of_phases%3 != 0 or number_of_slots % 3 != 0 or number_of_slots_pe
        coil_offset.is_integer()==False):
    print("Double layer winding is not feasible for the given number of poles
    exit()
```

In [4]:
```
# Define some paramters and their calculations.
# calculation of slot pitch in electrical and mechenical degrees.
slot_pitch_mech = 360 / number_of_slots
slot_pitch_elec = (number_of_poles / 2) * slot_pitch_mech

# calculation of coil pitch and coil span
coil_span = int(number_of_slots / number_of_poles)

# if coil_span is zeo
if coil_span ==0:
    coil_span = 1
coil_pitch_mech = coil_span * slot_pitch_mech
coil_pitch_elec = coil_span * slot_pitch_elec


# calculation of chording angle
chording_angle = (180 - coil_pitch_mech) / 2
number_of_coils = number_of_slots
```

In [5]:

```
# make a list for coil number
coil_number = [i for i in range(1,number_of_slots+1)]
```

In [6]:

```
# Initialize lists for for In and Out connection of all slots
slotin = [0] * number_of_slots
slotout = [0] * number_of_slots
theta = [0] * number_of_slots
```

In [7]:

```
# Calculate relative angle theta for each slot
for i in range(0,number_of_slots):
    theta[i] = (i) * (number_of_poles/number_of_slots)*180
```

In [8]:

```
# Fill slotin and slotout lists according to rule base of double layer winding
for i in range(0, number_of_slots):
    slotin[i] = i + 1

for i in range(0, number_of_slots):
    slotout[i] = i + 1 + coil_span
    if slotout[i] > number_of_slots:
        slotout[i] -= number_of_slots


# convert slot angle between -180 to +180 degrees
for i in range(0,number_of_slots):
    theta[i] = ((theta[i]+180)%360)-180

# round-off theta to nearest integer
for i in range(len(theta)):
    theta[i] = math.ceil(theta[i])

# Step 23: Check theta and swap slots if necessary
for i in range(0, number_of_slots):
    if theta[i] >= 90 or theta[i]<-90:
        slotin[i],slotout[i] = slotout[i],slotin[i]

# In case the magnitude of coil angle is greater than 90 and the sign of the c
#     then perform the operation coil angle-180, if the coil angle is greater
#     then perform coil angle+180
for i in range(0,number_of_slots):
    if theta[i]>90:
        theta[i] = theta[i]-180
    elif theta[i]<-90:
        theta[i] = theta[i]+180
    elif theta[i] == 90 or theta[i] == -90:
        theta[i] = theta[i]
```

In [9]:

```
    # initialize a list for storing relative slot angle for phase A
    theta1 = []

    # take out positive slot angles
    for i in range(0,number_of_slots):
        if theta[i] >= 0:
            theta1.append(theta[i])

    # Now sort the positive relative slot angles
    theta1.sort()
    coil_number1=[]
```

In [10]:

```
# Final step to select the phases.
# Phase A selection
slotin1 = []
slotout1 = []
set1=  [False] * number_of_slots
for i in range(len(theta1)):

    for j in range(number_of_slots):
        if(len(slotin1)== number_of_slots//3):
            break
        else:
            if theta[j]== theta1[i]:
                if set1[j] ==False:
                    coil_number1.append(coil_number[j])
                    slotin1.append(slotin[j])
                    slotout1.append(slotout[j])
                    set1[j]=True
```

In [11]:

```
# make four lists for other remaining two phases also.
slotin2=[0]*len(slotin1)
slotin3=[0]*len(slotin1)
slotout2=[0]*len(slotin1)
slotout3=[0]*len(slotin1)
```

In [12]:

```
# make 2 lists for handling of coil numbers
coil_number2=[0]*len(slotin1)
coil_number3=[0]*len(slotin1)
```

In [13]:

```
#Phase B and Phase C selection
for i in range(len(slotin1)):
    slotin2[i]=slotin1[i]+coil_offset
    while slotin2[i]>number_of_slots:
        slotin2[i]  -= number_of_slots

    slotout2[i]=slotout1[i]+coil_offset
    while slotout2[i]>number_of_slots:
        slotout2[i]  -= number_of_slots

    coil_number2[i]=coil_number1[i]+coil_offset
    while coil_number2[i]>number_of_slots:
        coil_number2[i]  -= number_of_slots

    slotin3[i]=slotin1[i]+ 2*coil_offset
    while slotin3[i]>number_of_slots:
        slotin3[i]  -= number_of_slots

    slotout3[i]=slotout1[i]+ 2*coil_offset
    while slotout3[i]>number_of_slots:
        slotout3[i]  -= number_of_slots

    coil_number3[i]=coil_number1[i]+2*coil_offset
    while coil_number3[i]>number_of_slots:
        coil_number3[i]  -= number_of_slots
```

In [14]:

```
# function to convert a value to its nearest integer
def mapp(arr):
    for i in range(len(arr)):
        arr[i] = math.ceil(arr[i])
    return arr

# Call the above functions on desierd lists
slotin2=mapp(slotin2)
slotout2=mapp(slotout2)
slotin3=mapp(slotin3)
slotout3=mapp(slotout3)
coil_number2=mapp(coil_number2)
coil_number3=mapp(coil_number3)
```

In [15]:

```
# calculation of coil pitch or coil span
coil_pitch = number_of_slots//number_of_poles

# calculation of slot pitch in electrical degrees
slot_pitch_elec = (number_of_poles/2)*slot_pitch_mech

# calculation of coil pitch in electrical as well as mechanical
coil_pitch_elec = coil_pitch*slot_pitch_elec
coil_pitch_mech = coil_pitch*slot_pitch_mech

# check for full pitched or short pitched winding
def CheckForFullPitchedWinding(coil_pitch_elec):
    if coil_pitch_elec == 180:
        print("Winding is Full Pitched")
    elif coil_pitch_elec > 180:
        print("Winding is over pitched")
    else:
        print("Winding is Short Pitched")

# calculation for pitch factor // give angles in radians
pitch_factor = math.cos((math.pi/180)*chording_angle/2)

# angular displacement between slots
beta = (180*number_of_poles)/number_of_slots

# calculation of distribution factor
distribution_factor = math.sin((math.pi/180)*number_of_slots_per_pole_per_pha
                        # Only for visibility purpose, we wrote in next line.
                        #
# calculation of winding factor
winding_factor = pitch_factor*distribution_factor
```

In [16]:
```
# Print Required Paramters and values......................................
print("Coil offset is: ",coil_offset)
print("Number of slots per pole per phase is: ",number_of_slots_per_pole_per_
print('coil pitch in number of slots : ',coil_span)
print('Slot pitch in mechanical degrees: ',slot_pitch_mech)
print('Slot pitch in electrical degrees: ',slot_pitch_elec)
print('coil pitch in mechanical degrees: ',coil_pitch_mech)
print('coil pitch in electrical degrees: ',coil_pitch_elec)
CheckForFullPitchedWinding(coil_pitch_elec)
print('Chording angle is : ',chording_angle)
print('Pitch factor is: ',pitch_factor)
print('Distribution factor is: ',distribution_factor)
print('Winding factor is: ', winding_factor)
print('---------In & Out Connections for Phase A---------------')

print('In',slotin1)
print('Out',slotout1,'\n')
print('--------In & Out Connections for Phase B--------------- ')

print('In',slotin2)
print('Out',slotout2,'\n')
print('--------In & Out Connections for Phase C----------------')

print('In',slotin3)
print('Out',slotout3)
```

```
Coil offset is:  16.0
Number of slots per pole per phase is:  2.0
coil pitch in number of slots :  6
Slot pitch in mechanical degrees:  7.5
Slot pitch in electrical degrees:  30.0
coil pitch in mechanical degrees:  45.0
coil pitch in electrical degrees:  180.0
Winding is Full Pitched
Chording angle is :  67.5
Pitch factor is:  0.8314696123025452
Distribution factor is:  0.9659258262890683
Winding factor is:  0.8031379722975873
---------In & Out Connections for Phase A---------------
In [1, 13, 13, 25, 25, 37, 37, 1, 2, 14, 14, 26, 26, 38, 38, 2]
Out [7, 7, 19, 19, 31, 31, 43, 43, 8, 8, 20, 20, 32, 32, 44, 44]

--------In & Out Connections for Phase B---------------
In [17, 29, 29, 41, 41, 5, 5, 17, 18, 30, 30, 42, 42, 6, 6, 18]
Out [23, 23, 35, 35, 47, 47, 11, 11, 24, 24, 36, 36, 48, 48, 12, 12]

--------In & Out Connections for Phase C----------------
In [33, 45, 45, 9, 9, 21, 21, 33, 34, 46, 46, 10, 10, 22, 22, 34]
Out [39, 39, 3, 3, 15, 15, 27, 27, 40, 40, 4, 4, 16, 16, 28, 28]
```