In [12]:
```
import numpy as np #importing libraries
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.axes as axis
```

In [13]:
```
def normalize(fold_data,icount_data):                                          #creatin
g function for normalizing folded pulse data

    norm_data = np.zeros_like(fold_data)                                       #initial
izing array for normalized data

    for i in range(len(fold_data[:,:,:])):                                     #looping
over how ever many itterations within the folded data necessary to fill nor
m_data
        norm_data[:,:,:,i] = fold_data[:,:,:,i]/icount_data[:,:,:] #normali
zing data

    return norm_data
```

In [14]:
```
def new_norm(fold_data,icount_data):                        #creating function fo
r normalizing folded pulse data

    norm_data = np.zeros_like(icount_data)                  #initializing array f
or normalized data
    norm_data = fold_data[:,:,:,0]/icount_data[:,:,:] #normalizing data

    return norm_data
```

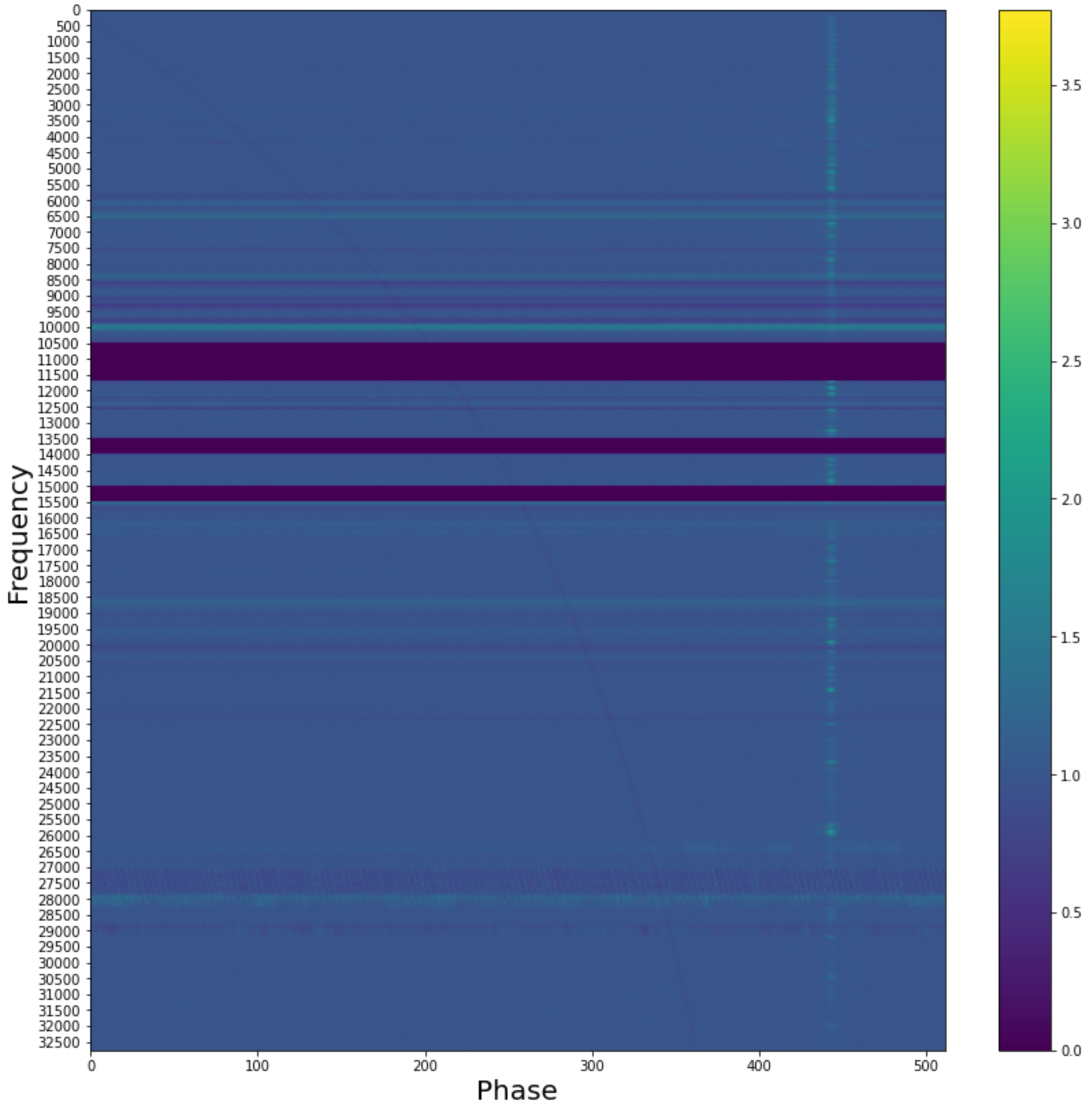
In [15]:
```
def plotting(folded_data, norm_data,j,plot_size):
    for i in range(len(folded_data)):                                         #itterati
ng over time values in data file
        plt.figure(figsize=(plot_size,plot_size))                             #creating
figure
        plt.imshow(norm_data[i,:,:],cmap='viridis',aspect='auto') #plotting
data
        plt.yticks(range(0,32768,500))                                        #setting
additional 'ticks' on y axis such that it is easier to erad off frequency b
ands
        plt.colorbar()                                                        #showing
colour band
        plt.xlabel('Phase', size='20')                                        #setting
plot labels
        plt.ylabel('Frequency', size='20')
        plt.savefig('Figure_'+str(j)+'.'+str(i)+'.png')                       #saving p
lot
```

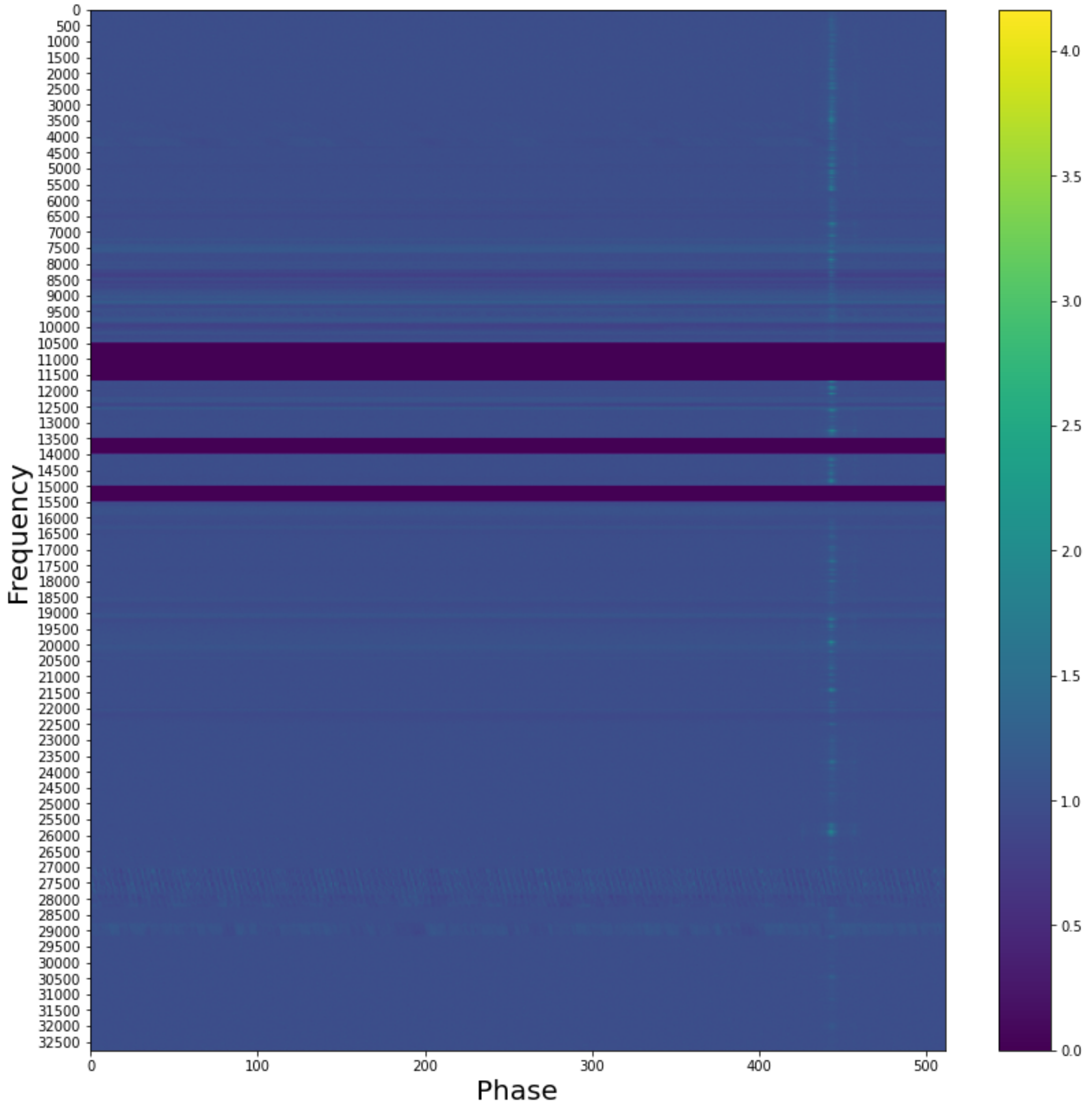
In [16]:
```
def mask_bands(data):
    i = 0                                    #initializing variables to lo
op over
    l = 0
    m = 0
    for h in range(len(data[:,0,0])):         #for each entry of time
        for k in range(len(data[0,0,:])):     #for each entry of phase
            for j in range(len(data[0,:,0])): #for each entry of frequency
                if (i >= 10500 and i <= 11700) or (i >= 13500 and i <= 1400
0) or (i >= 15000 and i <= 15500): #conditional for noisy frequency bands
                    data[m,i,l] = 0           #weighing down the RFI freque
ncies
                    i = i+1
                else:
                    i = i+1
            i=0
            l=l+1
        i=0
        l=0
        m = m+1
    return data

def mask_bands_specific(data,low_1,high_1,low_2,high_2,low_3,high_3):
    i = 0                                    #initializing variables to lo
op over
    l = 0
    m = 0
    for h in range(len(data[:,0,0])):         #for each entry of time
        for k in range(len(data[0,0,:])):     #for each entry of phase
            for j in range(len(data[0,:,0])): #for each entry of frequency
                if (i >= low_1 and i <= high_1) or (i >= low_2 and i <= hig
h_2) or (i >= low_3 and i <= high_3): #conditional for noisy frequency band
s
                    data[m,i,l] = 0           #weighing down the RFI freque
ncies
                    i = i+1
                else:
                    i = i+1
            i=0
            l=l+1
        i=0
        l=0
        m = m+1
    return data
```

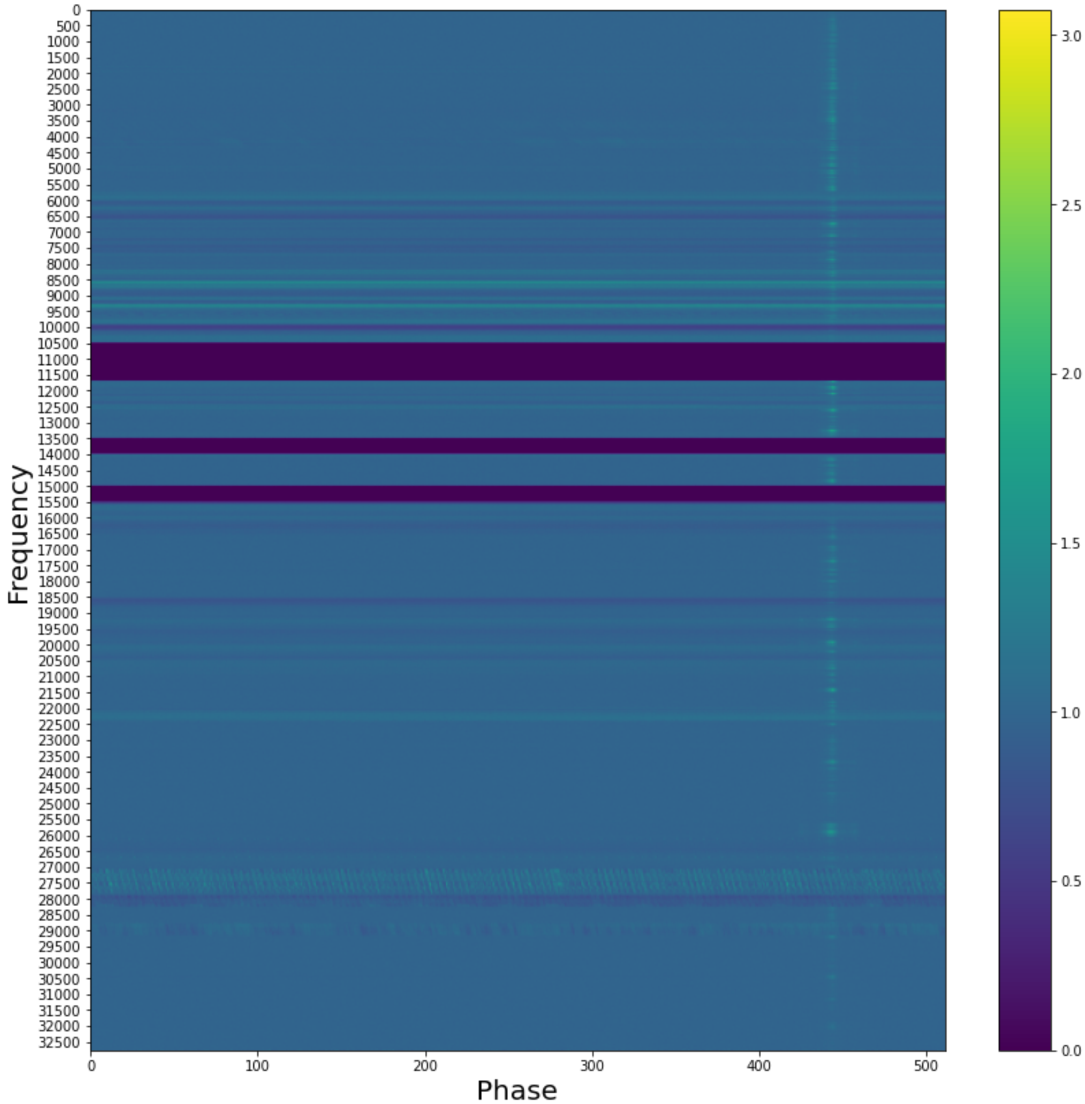
In [17]:
```
def average(norm_data):
    # averaged_data[axis=(1)] = norm_data[axis=(1)]/avg[axis=(1)]
    return (norm_data/np.mean(norm_data,axis=(0,2),keepdims=True)) #taking
the average of the time and phase axis for the normalized data, and dividin
g the normalized data by the resuling array
```

In [18]:
```
start = "arochime-invpfbB0329+54_32768chan3ntbin" #setting strings to make
calling files easier/shorter, since most of them share the same name
fold = "foldspec_2018-08-16T10:"
icount = "icount_2018-08-16T10:"
end = ".000+30.000000000000004sec"

data1 = np.load(start+fold+str(38)+":"+str(30)+end+".npy")
data2 = np.load(start+icount+str(38)+":"+str(30)+end+".npy")
```

In [19]:
```
data_fold_1 = np.load(start+fold+str(47)+":"+str(30)+end+".npy")    #loadin
g data from folded pulses
data_count_1 = np.load(start+icount+str(47)+":"+str(30)+end+".npy") #loadin
g data from icount file
norm_data_1 = new_norm(data_fold_1,data_count_1)                    #normal
izing data
avg_data = average(norm_data_1)                                     #dividi
ng normalized data by time and phase averaged array
masked_data = mask_bands(avg_data)                                  #weighi
ng down RFI frequencies
plotting(data_fold_1,masked_data,2,14)
```


Frequency
Phase
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
5500
6000
6500
7000
7500
8000
8500
9000
9500
10000
10500
11000
11500
12000
12500
13000
13500
14000
14500
15000
15500
16000
16500
17000
17500
18000
18500
19000
19500
20000
20500
21000
21500
22000
22500
23000
23500
24000
24500
25000
25500
26000
26500
27000
27500
28000
28500
29000
29500
30000
30500
31000
31500
32000
32500
0
100
200
300
400
500
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5


Frequency
Phase



In [ ]:

In [ ]: