In [1]:
```
from zero import Circuit
from zero.components import Resistor, Capacitor, Inductor
from zero.analysis import AcNoiseAnalysis, AcSignalAnalysis
import numpy as np
from pyliso import round_to_nearest_value as Eval
from plotting import plotTFs
from matplotlib.backends.backend_pdf import PdfPages       #For saving figures t
figlist = []
#*****************************************************************************
#Setting RC Parameters for figure size and fontsizes
import matplotlib.pylab as pylab
params = {'legend.fontsize': 'xx-large',
          'figure.figsize': (20, 10),
         'axes.labelsize': 'xx-large',
         'axes.titlesize':'xx-large',
         'xtick.labelsize':'xx-large',
         'ytick.labelsize':'xx-large'}
pylab.rcParams.update(params)
#*****************************************************************************
```

Converting to frequency actuation units

Sensor conversion slope

Incoming signal at RF input is in volts after mixer. This is related to frequency error by PDH discriminant (Black et al. Am. J. Phys., 69, 1 (2001) (https://aapt.scitation.org/doi/pdf/10.1119/1.1286663?class=pdf) (Eq 4.2):

$$D = -\frac{8\sqrt{P_C P_S}}{\delta\nu} \quad \frac{W}{Hz}$$

where $P_C = J_0^2(\beta)$ is carrier power, $P_S = J_1^2(\beta)$ is the sideband power, β being modulation index, and $\delta\nu = \Delta\nu_{FSR}/\mathcal{F}$ is cavity linewidth, $\Delta\nu_{FSR} = c/2L$ being free spectral range and $\mathcal{F}$ being the cavity finesse. This approximates to:

$$D = -\frac{4P_0\beta}{\frac{c}{2L}}\mathcal{F} \quad \frac{W}{Hz}$$

Where P_0 is incident power and L is cavity length. The RFPDs gain and transimpedance converts this into V/Hz. So our sensors frequency error to V conversion is:

$$H_{Sensor} = -Z_{TI}\mathcal{R}\frac{8P_0\beta L}{c}\mathcal{F} \quad \frac{V}{Hz}$$

where Z_{TI} is transimpedance gain of RFPD amplifier and $\mathcal{R}$ is the photodiode's responsivity.

PZT actuation slope

PZT actuation slope is directly marked on the laser as 1 MHz/V. So

$$H_{PZT} = 10^6 \ \frac{Hz}{V}$$

EOM actuation slope

EOM actuates on the phase of the laser, not the frequency directly. [New Focus 4004 (https://www.newport.com/medias/sys_master/images/images/h65/hcc/8797007839262/400X-and-406X-User-Manual-Rev-J.pdf)](https://www.newport.com/medias/sys_master/images/images/h65/hcc/8797007839262/400X-and-406X-User-Manual-Rev-J.pdf) has a modulation slope of $m_s = 15mrad/V$. The rate of change of the phase acts as frequency change by the EOM:

$$\delta f(t) = \frac{1}{2\pi}\frac{d\Delta\phi}{dt} = \frac{m_s}{2\pi}\frac{dV_{EOM}}{dt}$$

Therefore, in fourier domain,

$$\delta\tilde{f}(f) = \frac{m_s}{2\pi}(-\imath 2\pi f)\tilde{V_{EOM}}(f)$$

Therefore, the actuation slope for the EOM is:

$$H_{EOM} = -\imath f m_s \ \frac{Hz}{V}$$

Plant Circuit

In Plant circuit, the frequency is stored as symbolic voltage V_S where the conversion is:

$$\eta_{HztoVs} = 1e6$$

i.e. 1 MHz is stored as 1 V_S

In [2]:
```
ff = np.logspace(-1,7, 700)
SouthZ_TI = 937     # V/A, RFPD transimpedance
NorthZ_TI = 1816    # V/A, RFPD transimpedance
Res = 0.75          # A/W, Photodiode responsivity
F =  15000          # Cavity finesse
P0 = 8e-3           # W, Incident power on cavity
beta = 0.3          # PDH modulation index
L = 1.45*25.4e-3    # m, Cavity Length
c = 3e8             # m/s Speed of light
ms = 15e-3          # rad/V EOM modulation slope
HztoVs = 1e6        # Hz/Vs, 1 MHz is stored as 1 Vs


SouthH_sensor = - SouthZ_TI*Res*8.0*P0*beta*L*F/c  # V/Hz
SouthH_sensorVs = SouthH_sensor*HztoVs             # V/Vs
NorthH_sensor = - NorthZ_TI*Res*8.0*P0*beta*L*F/c  # V/Hz
NorthH_sensorVs = NorthH_sensor*HztoVs             # V/Vs

H_PZT = 1e6                                # Hz/V
H_PZTVs = H_PZT/HztoVs                     # Vs/V

H_EOM = -1j*ff*ms                          # Hz/V
H_EOMVs = H_EOM/HztoVs                     # Vs/V
```

Mapping requried TF to plant circuit values

Gain on PZT path:

$$-\frac{R3}{R1} = H_{PZT,Vs}H_{Sensor,Vs}$$

$$-\frac{R3}{R1} = \frac{H_{PZT}}{\eta_{HztoVs}}(-Z_{TI}\mathcal{R}\frac{8P_0\beta L}{c}\mathcal{F})\eta_{HztoVs}$$

$$\frac{R3}{R1} = -H_{PZT}H_{Sensor}$$

On the EOM path:

$$(1+\frac{R3}{R1})(\frac{if}{\frac{1}{2\pi R_4C_2}+if}) = H_{EOM,Vs}H_{Sensor,Vs}$$

$$(1+\frac{R3}{R1})(\frac{if}{\frac{1}{2\pi R_4C_2}+if}) = \frac{-ifm_s}{\eta_{HzToVs}}(-Z_{TI}\mathcal{R}\frac{8P_0\beta L}{c}\mathcal{F})\eta_{HztoVs}$$

Approximately for $f << \frac{1}{2\pi R_4C_2}$:

$$(1+\frac{R3}{R1})2\pi R_4C_2 = m_sZ_{TI}\mathcal{R}\frac{8P_0\beta L}{c}\mathcal{F}$$

$$(1+\frac{R3}{R1})2\pi R_4C_2 = -m_sH_{Sensor}$$

In [3]:
```
R3_over_R1S = -H_PZT*SouthH_sensor
R4C2S = -ms*SouthH_sensor/(2*np.pi*(1+ R3_over_R1S))
R3_over_R1N = -H_PZT*NorthH_sensor
R4C2N = -ms*NorthH_sensor/(2*np.pi*(1+ R3_over_R1N))
print('South: R3/R1:', np.round(R3_over_R1S,2),
      'R4C2:', np.round(R4C2S*1e9, 2), 'ns',
      'High Pass 3-dB cutoff:',
      np.round(1/2/np.pi/R4C2S/1e6, 2), 'MHz')
print('North: R3/R1:', np.round(R3_over_R1N,2),
      'R4C2:', np.round(R4C2N*1e9, 2), 'ns',
      'High Pass 3-dB cutoff:',
      np.round(1/2/np.pi/R4C2N/1e6, 2), 'MHz')
```

```
South: R3/R1: 24.85 R4C2: 2.29 ns High Pass 3-dB cutoff: 69.35 MHz
North: R3/R1: 48.16 R4C2: 2.34 ns High Pass 3-dB cutoff: 68.05 MHz
```

In [4]:
```
R1N = Eval(100, 'E12')
R3N = Eval(R1N*R3_over_R1N, 'E12')
R4N = Eval(10, 'E12')
C2N = Eval(R4C2N/R4N, 'E12')
R1S = Eval(100, 'E12')
R3S = Eval(R1S*R3_over_R1S, 'E12')
R4S = Eval(10, 'E12')
C2S = Eval(R4C2S/R4S, 'E12')

print('South: R1:', R1S,'R3:', R3S, 'R4:', R4S, 'C2:', C2S)
print('North: R1:', R1N,'R3:', R3N, 'R4:', R4N, 'C2:', C2N)
```

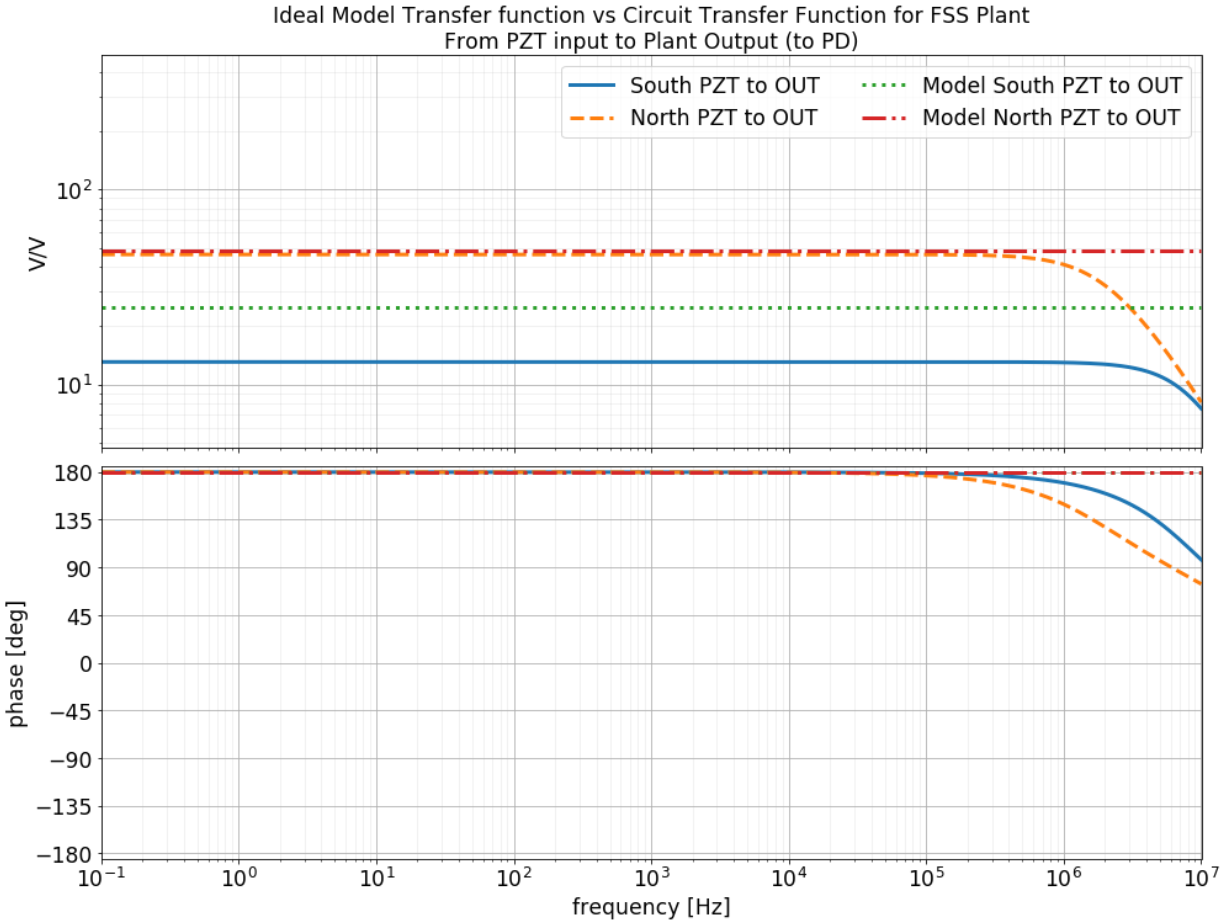
```
South: R1: 100.0 R3: 2610.0 R4: 10.0 C2: 2.2e-10
North: R1: 100.0 R3: 4640.0 R4: 10.0 C2: 2.2e-10
```

In [5]:
```
plantS = Circuit()
plantN = Circuit()
plantS.add_capacitor(name='PZTC', value='10n', node1='PZTnIN', node2='gnd')
plantS.add_capacitor(name='EOMC', value='20p', node1='EOMnIN', node2='gnd')
plantN.add_capacitor(name='PZTC', value='10n', node1='PZTnIN', node2='gnd')
plantN.add_capacitor(name='EOMC', value='20p', node1='EOMnIN', node2='gnd')

plantS.add_library_opamp(name='Buffer', model='ad829', node1='PZTnIN',
                         node2='PZTnINa', node3='PZTnINa')
plantS.add_resistor(name='R1', value=R1S, node1='PZTnINa', node2='n2')
plantS.add_resistor(name='R3', value=R3S, node1='n2', node2='nOut')
plantS.add_capacitor(name='C2', value=C2S, node1='EOMnIN', node2='n1')
plantS.add_resistor(name='R4', value=R4S, node1='n1', node2='gnd')
plantS.add_resistor(name='R4a', value=R3S, node1='n2', node2='nOut')
plantS.add_library_opamp(name='Adder', model='ad829', node1='n1',
                         node2='n2', node3='nOut')

plantN.add_library_opamp(name='Buffer', model='ad829', node1='PZTnIN',
                         node2='PZTnINa', node3='PZTnINa')
plantN.add_resistor(name='R1', value=R1N, node1='PZTnINa', node2='n2')
plantN.add_resistor(name='R3', value=R3N, node1='n2', node2='nOut')
plantN.add_capacitor(name='C2', value=C2N, node1='EOMnIN', node2='n1')
plantN.add_resistor(name='R4', value=R4N, node1='n1', node2='gnd')
plantN.add_library_opamp(name='Adder', model='ad829', node1='n1',
                         node2='n2', node3='nOut')
```

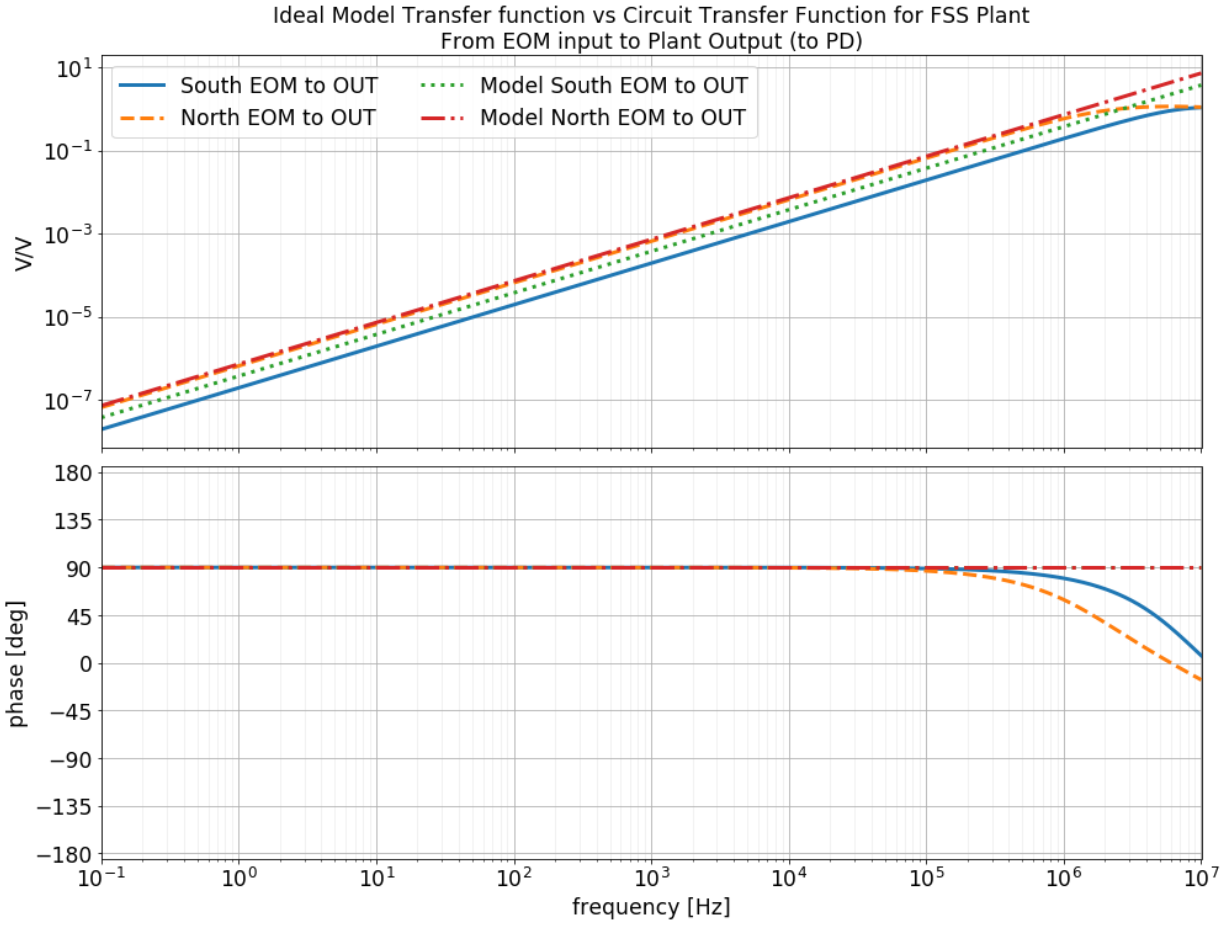
In [6]:
```
sigAnalS = AcSignalAnalysis(circuit=plantS)
TFfromPZTS = sigAnalS.calculate(frequencies=ff,
                                input_type='voltage',
                                node='PZTnIN')
TFfromEOMS = sigAnalS.calculate(frequencies=ff,
                                input_type='voltage',
                                node='EOMnIN')

sigAnalN = AcSignalAnalysis(circuit=plantN)
TFfromPZTN = sigAnalN.calculate(frequencies=ff,
                                input_type='voltage',
                                node='PZTnIN')
TFfromEOMN = sigAnalN.calculate(frequencies=ff,
                                input_type='voltage',
                                node='EOMnIN')
```

In [7]:
```
TFDict1 = {}
TFDict2 = {}
TFDict1['South PZT to OUT'] = TFfromPZTS.get_response("PZTnIn", "nOut").complex_
TFDict2['South EOM to OUT'] = TFfromEOMS.get_response("EOMnIn", "nOut").complex_
TFDict1['North PZT to OUT'] = TFfromPZTN.get_response("PZTnIn", "nOut").complex_
TFDict2['North EOM to OUT'] = TFfromEOMN.get_response("EOMnIn", "nOut").complex_
TFDict1['Model South PZT to OUT'] = H_PZTVs*SouthH_sensorVs*np.ones(len(ff))
TFDict2['Model South EOM to OUT'] = H_EOMVs*SouthH_sensorVs
TFDict1['Model North PZT to OUT'] = H_PZTVs*NorthH_sensorVs*np.ones(len(ff))
TFDict2['Model North EOM to OUT'] = H_EOMVs*NorthH_sensorVs
```

In [8]:
```
fig = plotTFs(ff, TFDict1, figsize=[16,12], lw=3)
fig.axes[0].set_ylabel('V/V')
fig.axes[0].set_title('Ideal Model Transfer function vs Circuit Transfer Functic
                      '\nFrom PZT input to Plant Output (to PD)')
figlist = [fig];
```



In [9]:
```
fig = plotTFs(ff, TFDict2, figsize=[16,12], lw=3)
fig.axes[0].set_ylabel('V/V')
fig.axes[0].set_title('Ideal Model Transfer function vs Circuit Transfer Functic
                      '\nFrom EOM input to Plant Output (to PD)')
figlist += [fig];
```



In [10]:
```
pp = PdfPages('FSS_Plant_Zero_Sim.pdf')
for fig in figlist:
    pp.savefig(fig,bbox_inches='tight')
pp.close()
```