```
        Reference:
        Evans, J.S., and C.A. Doswell, 2001: Examination of derecho environments using

        Parameters
        ----------
        prof : Profile object

        Returns
        -------
        dcp : number
            Derecho Composite Parameter (unitless)

    '''
    sfc = prof.pres[prof.sfc]
    p6km = interp.pres(prof, interp.to_msl(prof, 6000.))
    dcape_val = getattr(prof, 'dcape', dcape( prof )[0])
    mupcl = getattr(prof, 'mupcl', parcelx(prof, flag=1))
    sfc_6km_shear = getattr(prof, 'sfc_6km_shear', winds.wind_shear(prof, pbot=sfc, ptc
    mean_6km = getattr(prof, 'mean_6km', utils.comp2vec(*winds.mean_wind(prof, pbot=sfc
    mag_shear = utils.mag(sfc_6km_shear[0], sfc_6km_shear[1])
    mag_mean_wind = mean_6km[1]

    dcp = (dcape_val/980.) * (mupcl.bplus/2000.) * (mag_shear / 20. ) * (mag_mean_wind

    return dcp

def mburst1(prof):
    '''
        Microburst Windspeed Potential Index (MWPI)

        Formulated by Kenneth Pryor NOAA/NESDIS/STAR

        The Microburst Windspeed Potential Index (MWPI) is designed to quantify the mos
        in convective downburst generation in intermediate thermodynamic environments b
        CAPE, 2) the temperature lapse rate between the 670- and 850-mb levels, and 3)
        850-mb levels. The MWPI formula consists of a set of predictor variables (i.e.,
        and temperature lapse rate) that generates output of the expected microburst ri
        Scaling factors of 1000 J/kg, 5 C/km, and 5 C, respectively, are applied to the
        to yield a unitless MWPI value that expresses wind gust potential on a scale fr

        MWPI = (CAPE/1000) + LR/5 + DDD/5

        Reference:
        Pryor, K. L., 2015: Progress and Developments of Downburst Prediction Applicati

        Parameters
        ----------
        prof : Profile object

        Returns
        -------
        mburst : number
            MWPI (unitless)
    '''
```

```
    #sbpcl = getattr(prof, 'sfcpcl', parcelx(prof, flag=1))
    #sb_cape = sbpcl.bplus

    mupcl = getattr(prof, 'mupcl', parcelx(prof, flag=1))
    mu_cape = mupcl.bplus
    """
    #MWPI calculation for 500-700 mb layer
    lr75 = lapse_rate(prof, 700, 500, pres=True)
    t5 = interp.temp(prof, 500.)
    t7 = interp.temp(prof, 700.)
    td5 = interp.dwpt(prof, 500.)
    td7 = interp.dwpt(prof, 700.)
    dd5 = t5 - td5
    dd7 = t7 - td7
    ddd = dd7 - dd5
    #mburst = (sb_cape/1000) + (lr75/5) + (ddd/5)
    mburst = (mu_cape/1000) + (lr75/5) + (ddd/5)
    """
    #MWPI calculation for 650-850 mb layer
    lr86 = lapse_rate(prof, 850, 650, pres=True)
    t6 = interp.temp(prof, 650.)
    t8 = interp.temp(prof, 850.)
    td6 = interp.dwpt(prof, 650.)
    td8 = interp.dwpt(prof, 850.)
    dd6 = t6 - td6
    dd8 = t8 - td8
    ddd = dd8 - dd6
    #mburst = (sb_cape/1000) + (lr85/5) + (ddd/5)
    mburst1 = (mu_cape/1000) + (lr86/5) + (ddd/5)
    """
    #MWPI calculation for surface-based mixed layer
    lr950_750 = lapse_rate(prof, 950, 750, pres=True)
    t750 = interp.temp(prof, 750.)
    t950 = interp.temp(prof, 950.)
    td750 = interp.dwpt(prof, 750.)
    td950 = interp.dwpt(prof, 950.)
    dd750 = t750 - td750
    dd950 = t950 - td950
    ddd = dd950 - dd750
    #mburst = (sb_cape/1000) + (lr975_850/5) + (ddd/5)
    mburst = (mu_cape/1000) + (lr950_750/5) + (ddd/5)
    """
    return mburst1

def mburst2(prof):
    '''
        Microburst Windspeed Potential Index (MWPI)

        Formulated by Kenneth Pryor NOAA/NESDIS/STAR

        The Microburst Windspeed Potential Index (MWPI) is designed to quantify the mos
        in convective downburst generation in intermediate thermodynamic environments b
        CAPE, 2) the temperature lapse rate between the 670- and 850-mb levels, and 3)
        850-mb levels. The MWPI formula consists of a set of predictor variables (i.e.,
        and temperature lapse rate) that generates output of the expected microburst ri
```

```
    Scaling factors of 1000 J/kg, 5 C/km, and 5 C, respectively, are applied to the
    to yield a unitless MWPI value that expresses wind gust potential on a scale fr

    MWPI = (CAPE/1000) + LR/5 + DDD/5

    Reference:
    Pryor, K. L., 2015: Progress and Developments of Downburst Prediction Applicati

    Parameters
    ----------
    prof : Profile object

    Returns
    -------
    mburst : number
        MWPI (unitless)
'''
#sbpcl = getattr(prof, 'sfcpcl', parcelx(prof, flag=1))
#sb_cape = sbpcl.bplus

mupcl = getattr(prof, 'mupcl', parcelx(prof, flag=1))
mu_cape = mupcl.bplus
"""
#MWPI calculation for 500-700 mb layer
lr75 = lapse_rate(prof, 700, 500, pres=True)
t5 = interp.temp(prof, 500.)
t7 = interp.temp(prof, 700.)
td5 = interp.dwpt(prof, 500.)
td7 = interp.dwpt(prof, 700.)
dd5 = t5 - td5
dd7 = t7 - td7
ddd = dd7 - dd5
#mburst = (sb_cape/1000) + (lr75/5) + (ddd/5)
mburst = (mu_cape/1000) + (lr75/5) + (ddd/5)

#MWPI calculation for 650-850 mb layer
lr86 = lapse_rate(prof, 850, 650, pres=True)
t6 = interp.temp(prof, 650.)
t8 = interp.temp(prof, 850.)
td6 = interp.dwpt(prof, 650.)
td8 = interp.dwpt(prof, 850.)
dd6 = t6 - td6
dd8 = t8 - td8
ddd = dd8 - dd6
#mburst = (sb_cape/1000) + (lr85/5) + (ddd/5)
mburst = (mu_cape/1000) + (lr86/5) + (ddd/5)
"""
#MWPI calculation for surface-based mixed layer
lr950_750 = lapse_rate(prof, 950, 750, pres=True)
t750 = interp.temp(prof, 750.)
t950 = interp.temp(prof, 950.)
td750 = interp.dwpt(prof, 750.)
td950 = interp.dwpt(prof, 950.)
dd750 = t750 - td750
dd950 = t950 - td950
```

```
    ddd = dd950 - dd750
    #mburst = (sb_cape/1000) + (lr975_850/5) + (ddd/5)
    mburst2 = (mu_cape/1000) + (lr950_750/5) + (ddd/5)

    return mburst2

def ehi(prof, pcl, hbot, htop, stu=0, stv=0):
    '''
        Energy-Helicity Index

        Computes the energy helicity index (EHI) using a parcel
        object and a profile object.

        The equation is EHI = (CAPE * HELICITY) / 160000.

        Parameters
        ----------
        prof : Profile object
        pcl : Parcel object
        hbot : number
            Height of the bottom of the helicity layer [m]
        htop : number
            Height of the top of the helicity layer [m]
        stu : number
            Storm-relative wind U component [kts]
            (optional; default=0)
        stv : number
            Storm-relative wind V component [kts]
            (optional; default=0)

        Returns
        -------
        ehi : number
            Energy Helicity Index (unitless)
    '''

    helicity = winds.helicity(prof, hbot, htop, stu=stu, stv=stv)[0]
    ehi = (helicity * pcl.bplus) / 160000.

    return ehi

def sweat(prof):
    '''
        SWEAT Index

        Computes the SWEAT (Severe Weather Threat Index) using the following numbers:

        1.) 850 Dewpoint
        2.) Total Totals Index
        3.) 850 mb wind speed
        4.) 500 mb wind speed
        5.) Direction of wind at 500
        6.) Direction of wind at 850

        Parameters
```