```
import serial
from time import localtime
from time import sleep
from time import strftime
from time import time
from utils import file_read
from utils import file_write
from utils import lookup_var
# ++++++++++++
def ms4448():
    print('ms4448 started')
    # +++++++++
    # Variables
    s_port = '/dev/ttyUSB0'
    data = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
            0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
    inv_stat = [0x00,0x01,0x02,0x04,0x05,0x08,0x09,0x0A,0x10,0x20,0x40,0x50,0x60]
    inv_stat_value = ['Standby','','Float Charge','Absorb Charge','Bulk Charge','Battery
Saver','Charge','Off','Inverting',
                      'Standby','Inverting']
    inv_fault = [0x00,0x01,0x02,0x04,0x08,0x09,0x10,0x12,0x14,0x16,0x17,0x18,0x19,0x20,0x21,0x80,0x81,0x90,0x91,]
    inv_fault_value = ['No Faults','Stuck Relay','DC Overload','AC Overload','Dead Battery','AC Backfeed','Low
Battery Cutout',
                       'High Battery Cutout','High AC Output Volts','Bad FET Bridge','FET Over Temp','Internal
Fault #4',
                       'Stacker Mode Fault','Stacker Sync Clock Lost','Stacker Sync Clock Out of Phase','Stacker
AC Phase Fault',
                       'Over temperature Shutdown','Transfer Relay Fault','Charger Fault','Battery Temperature
High',
                       'Transformer Temperature Cutout Open','AC Breaker CB3 Tripped']
    pt100_stat = [3,2,1,243,19,241,17,20,242]
    pt100_stat_value = ['Sleeping','Sleeping','Sleeping','Absorb','Absorb',
                        'Float Charging','Float Charging','Equalizing','Bulk']
    while True:
        exit_code = 0
        byte_count = 0
        try:
            # ++++++++++++++++
            # Open serial port
            ser_port = serial.Serial(port = s_port, baudrate = 19200, bytesize = 8, timeout = None)
            # ++++++++++++++++++
            # Flush data waiting
            ser_port.flush()
            start_time = time()
            # ++++++++++++++++++++++++++++++++++++++++++
            # Find and store 58 bytes in a single string
            while byte_count != 58:
                if ser_port.inWaiting != 0:
                    byte_read = ord(ser_port.read(1))
                    end_time = time()
                    if end_time - start_time > 0.06:
                        byte_count = byte_count + 1
                        data[byte_count] = byte_read
                        start_time = end_time
                        while byte_count != 58:
                            if ser_port.inWaiting != 0:
                                byte_read = ord(ser_port.read(1))
                                end_time = time()
                                if end_time - start_time < 0.06:
                                    byte_count = byte_count + 1
                                    data[byte_count] = byte_read
                                else:
                                    byte_count = 0
                                    data[byte_count] = byte_read
                                start_time = end_time
            # ++++++++++++++++
            # Get current time
```

```
current_time = time()
solar_data = 'current_time=' + str(current_time) + '\n'
# ++++++++++++++++++++++++++++++
# Get the day of the week and date
day_var = strftime('%d',localtime())
solar_data = solar_data + 'day=' + str(day_var) + '\n'
date = strftime('%b %d %Y',localtime())
solar_data = solar_data + 'date=' + date + '\n'
# +++++++++++++++
# Inverter Status
try:
    index = inv_stat.index(data[0])
    inverter_status = inv_stat_value[index]
except:
    inverter_status = 'N/A'
solar_data = solar_data + 'inv_stat=' + inverter_status + '\n'
# ++++++++++++++
# Inverter Fault
try:
    index = inv_fault.index(data[1])
    inverter_fault = inv_fault_value[index]
except:
    inverter_fault = 'N/A'
solar_data = solar_data + 'inv_flt=' + inverter_fault + '\n'
# ++++++++++++++++++++++++
# Inverter battery voltage
inv_bat_volt = round ((data[2] * 256 + data[3])/10, 1)
solar_data = solar_data + 'inv_bat_volt=' + str(inv_bat_volt) + '\n'
# +++++++++++++++++
# Inverter current
inv_dc_amp = round (data[4] * 256 + data[5],1)
solar_data = solar_data + 'inv_dc_amp=' + str(inv_dc_amp) + '\n'
# ++++++++++++++++++++++++++++++
# Inverter AC RMS voltage output
inv_ac_rms_volt_out = data[6]
solar_data = solar_data + 'inv_ac_rms_volt_out=' + str(inv_ac_rms_volt_out) + '\n'
# ++++++++++++++++++++++++++++++
# Inverter AC Peak input voltage
inv_ac_peak_volt_in = data[7]
solar_data = solar_data + 'inv_ac_peak_volt_in=' + str(inv_ac_peak_volt_in) + '\n'
# ++++++++++++++++++++++
# Inverter Inverting LED
inv_inverting_led = 'OFF'
if data[8] == 1:
    inv_inverting_led = 'ON'
solar_data = solar_data + 'inv_inverting_led=' + inv_inverting_led + '\n'
# ++++++++++++++++++++
# Inverter Charging LED
inv_charging_led = 'OFF'
if data[9] == 1:
    inv_charging_led = 'ON'
solar_data = solar_data + 'inv_charging_led=' + inv_charging_led + '\n'
# +++++++++++++++++
# Inverter revision
inv_revision = str(data[10] / 10)
solar_data = solar_data + 'inv_revision=' + inv_revision + '\n'
# ++++++++++++++++++++++++++++
# Inverter Battery Temperature
inv_bat_temp = data[11]
solar_data = solar_data + 'inv_bat_temp=' + str(inv_bat_temp) + '\n'
# ++++++++++++++++++++++++++++++
# Inverter Transformer Temperature
inv_trans_temp = data[12]
solar_data = solar_data + 'inv_trans_temp=' + str(inv_trans_temp) + '\n'
# ++++++++++++++++++++++++
# Inverter FET Temperature
inv_fet_temp = data[13]
```

```
solar_data = solar_data + 'inv_fet_temp=' + str(inv_fet_temp) + '\n'
# ++++++++++++++
# Inverter model
inv_model = data[14]
solar_data = solar_data + 'inv_model=' + str(inv_model) + '\n'
# ++++++++++++++++++
# Inverter Stack mode
inv_stack_mode = data[15]
solar_data = solar_data + 'inv_stack_mode=' + str(inv_stack_mode) + '\n'
# ++++++++++++++++++++++++
# Inverter AC input current
inv_ac_input_amp = data[16]
solar_data = solar_data + 'inv_ac_input_amp=' + str(inv_ac_input_amp) + '\n'
# ++++++++++++++++++++
# Inverter solar current
inv_ac_output_amp = data[17]
solar_data = solar_data + 'inv_ac_output_amp=' + str(inv_ac_output_amp) + '\n'
# ++++++++++++++++++++++++++
# Inverter frequency
inv_frequency = round((data[18] * 256 + data[19]) / 10,1)
solar_data = solar_data + 'inv_frequency=' + str(inv_frequency) + '\n'
# ++++++++++++++++++++++++
# Bit 20 and 21 are not used
# ++++++++++++++++++
# Remote toggle bits
rem_tog_bit = data[22]
solar_data = solar_data + 'rem_tog_bit=' + str(rem_tog_bit) + '\n'
# ++++++++++++++++++
# Remote search watts
rem_search_watt = data[23]
solar_data = solar_data + 'rem_search_watt=' + str(rem_search_watt) + '\n'
# ++++++++++++++++++
# Remote battery size
rem_bat_size = data[24]  * 10
solar_data = solar_data + 'rem_bat_size=' + str(rem_bat_size) + '\n'
# ++++++++++++++++++
# Remote battery type
rem_bat_type = data[25]
solar_data = solar_data + 'rem_bat_type=' + str(rem_bat_type) + '\n'
# ++++++++++++++++++++++++
# Remote charging amps in %
rem_charger_percent = data[26]
solar_data = solar_data + 'rem_charger_percent=' + str(rem_charger_percent) + '\n'
# ++++++++++++++++++
# Remote AC shore amps
rem_ac_shore_amp = data[27]
solar_data = solar_data + 'rem_ac_shore_amp=' + str(rem_ac_shore_amp) + '\n'
# +++++++++++++++
# Remote revision
rem_revision = data[28] / 10
solar_data = solar_data + 'rem_revision=' + str(rem_revision) + '\n'
# ++++++++++++++++++++++++++++++++++++++++
# Remote parallel; threshold / Force charge
rem_force_charge = data[29]
solar_data = solar_data + 'rem_force_charge=' + str(rem_force_charge) + '\n'
# ++++++++++++++++++
# Remote auto Genstart
rem_genstart = data[30]
solar_data = solar_data + 'rem_genstart=' + str(rem_genstart) + '\n'
# ++++++++++++++++++++++++
# Remote low battery cut out
rem_lbco = data[31] / 10 * 2
solar_data = solar_data + 'rem_lbco=' + str(rem_lbco) + '\n'
# ++++++++++++++++++++++++
# Remote AC voltage cut out
rem_ac_cut_out_volt = round(data[32] / 1.9375)
solar_data = solar_data + 'rem_ac_cut_out_volt=' + str(rem_ac_cut_out_volt) + '\n'
```

```
# ++++++++++++++++++++
# Remote float voltage
rem_float_volt = data[33] * 4 / 10
solar_data = solar_data + 'rem_float_volt=' + str(rem_float_volt) + '\n'
# +++++++++++++++++
# Remote EQ voltage
##### Not calculated properly
rem_eq_volt = data[34]
solar_data = solar_data + 'rem_eq_volt=' + str(rem_eq_volt) + '\n'
# ++++++++++++++++++++++
# Remote absorbtion time
##### Not calculated properly
rem_absorb_time = round(data[35] / 10)
solar_data = solar_data + 'rem_absorb_time=' + str(rem_absorb_time) + '\n'
# ++++++++++++
# Remote hours
rem_hours = data[36]
solar_data = solar_data + 'rem_hours=' + str(rem_hours) + '\n'
# ++++++++++++++
# Remote minutes
rem_minutes = data[37]
solar_data = solar_data + 'rem_minutes=' + str(rem_minutes) + '\n'
# +++++++++
# Remote 38
rem_38 = data[38]
solar_data = solar_data + 'rem_38=' + str(rem_38) + '\n'
# +++++++++
# Remote 39
rem_39 = data[39]
solar_data = solar_data + 'rem_39=' + str(rem_39) + '\n'
# +++++++++
# Remote 40
rem_40 = data[40]
solar_data = solar_data + 'rem_40=' + str(rem_40) + '\n'
# +++++++++
# Remote 41
rem_41 = data[41]
solar_data = solar_data + 'rem_41=' + str(rem_41) + '\n'
# +++++++++
# Remote 42
rem_42 = data[42]
solar_data = solar_data + 'rem_42=' + str(rem_42) + '\n'
# +++++++++
# Remote 43
rem_43 = data[43]
solar_data = solar_data + 'rem_43=' + str(rem_43) + '\n'
# +++++++++
# Remote 44
rem_44 = data[44]
solar_data = solar_data + 'rem_44=' + str(rem_44) + '\n'
# -------------
# PT-100 Status
try:
    index = pt100_stat.index(data[45])
    pt100_status = pt100_stat_value[index]
except:
    pt100_status = 'N/A'
solar_data = solar_data + 'pt100_stat=' + pt100_status + '\n'
# ------------
# PT-100 Fault
if data[46] == 0:
    pt100_fault = 'No Faults'
else:
    pt100_fault = 'N/A'
solar_data = solar_data + 'pt100_flt=' + pt100_fault + '\n'
# ----------------------
# PT-100 battery voltage
```

```
pt100_bat_volt = str(round((data[47] * 256 + data[48]) / 10, 1))
solar_data = solar_data + 'pt100_bat_volt=' + pt100_bat_volt + '\n'
# --------------------
# PT-100 solar current
pt100_amp = str(round((data[49] * 256 + data[50]) / 10, 1))
solar_data = solar_data + 'pt100_amp=' + pt100_amp + '\n'
# ------------------------------
# Calculate PT-100 solar voltage
pt100_solar_volt = str(round((data[51] * 256 + data[52]) / 10, 1))
solar_data = solar_data + 'pt100_solar_volt=' + pt100_solar_volt + '\n'
# ---------------
# PT-100 data[53]
pt100_53 = str(data[53])
solar_data = solar_data + 'pt100_53=' + pt100_53 + '\n'
# ---------------
# PT-100 data[54]
pt100_54 = str(data[54])
solar_data = solar_data + 'pt100_54=' + pt100_54 + '\n'
# ---------------
# PT-100 data[55]
pt100_55 = str(data[55])
solar_data = solar_data + 'pt100_55=' + pt100_55 + '\n'
# --------------------------
# PT-100 Battery Temperature
pt100_bat_temp = str(data[56])
solar_data = solar_data + 'pt100_bat_temp=' + pt100_bat_temp + '\n'
# ----------------------
# PT-100 FET Temperature
pt100_fet_temp = str(data[57])
solar_data = solar_data + 'pt100_fet_temp=' + pt100_fet_temp + '\n'
# --------------------------
# PT-100 Inductor Temperature
pt100_inductor_temp = str(data[58])
solar_data = solar_data + 'pt100_inductor_temp=' + pt100_inductor_temp + '\n'
# ----------------------
# Read the solar tmp file
exit_code,x_data = file_read('var/temp_local/solar.var')
exit_code, last_time = lookup_var(x_data, 'current_time')
if exit_code == 0:
    lapse = round(current_time - float(last_time), 2)
else:
    lapse = 0
# ----------------------------------
# Calculate time lapse since last read
exit_code, last_time = lookup_var(x_data, 'current_time')
if exit_code == 0:
    lapse = str(round(current_time - float(last_time), 2))
else:
    lapse = '0'
solar_data = solar_data + 'lapse=' + lapse + '\n'
# ------------------------------------------------
# Calculate power generated in time lapse in Watt/hour
watt_hour = str(float(pt100_bat_volt) * float(pt100_amp) * float(lapse) / 3600)
solar_data = solar_data + 'lapse_watt_hour=' + watt_hour + '\n'
# ------------------------------------------------
# Calculate the total W/h generated
exit_code, total_watt_hour = lookup_var(x_data, 'total_watt_hour')
if exit_code == 0:
    total_watt_hour = str(float(total_watt_hour) + float(watt_hour))
else:
    exit_code, total_watt_hour = file_read('data/solar_total.dat')
    if exit_code == 0:
        total_watt_hour = str(float(total_watt_hour) + float(watt_hour))
    else:
        total_watt_hour = '0'
solar_data = solar_data + 'total_watt_hour=' + total_watt_hour + '\n'
# ------------------------------------------------
```

```
            # Test if it's a new day and store data accordingly
            exit_code, last_day = lookup_var(x_data, 'day')
            exit_code, daily_watt_hour = lookup_var(x_data, 'daily_watt')
            if exit_code != 0:
                daily_watt_hour = 0
            daily_watt_hour = float(daily_watt_hour) + float(watt_hour)
            if last_day != day_var:
                file_write('data/solar_total.dat',str(total_watt_hour))
                file_write('data/solar_daily.dat',str(daily_watt_hour))
                daily_watt_hour = 0
            solar_data = solar_data + 'daily_watt=' + str(daily_watt_hour) + '\n'
            if inv_bat_volt > 45 and inv_bat_volt < 70:
                file_write('var/temp_local/solar.var', solar_data)
            # -------------------------------------
            # Write the display file for the PT-100
            pt100_watt = str(round(float(pt100_bat_volt) * float(pt100_amp)))
            output = 'PT-100,30,' + pt100_status + ',30,' + str(pt100_bat_volt) + ' V / ' + str(pt100_watt) + ' W,-
30'
            file_write('var/display_local/PT100.dis', output)
            file_write('var/display/PT100.dis', output)

            exit_code, inv_bat_volt = lookup_var(solar_data, 'inv_bat_volt')
            exit_code, inv_ac_rms_volt_out = lookup_var(solar_data, 'inv_ac_rms_volt_out')
            exit_code, inv_ac_output_amp = lookup_var(solar_data, 'inv_ac_output_amp')
            inv_watt = round(float(inv_ac_rms_volt_out) * float(inv_ac_output_amp))
            output = 'Inverter,30,' + inverter_status + ',30,' + str(inv_bat_volt) + ' V / ' + str(inv_watt) + ' W,-
30'
            file_write('var/display_local/Inverter.dis', output)
            file_write('var/display/Inverter.dis', output)


            ser_port.close()
        except IOError:
            print(2)
        sleep(2)
# ++++++++++++++++++++
def ms4448_display_x():
    exit_code, user_data = file_read('user.dat')
    exit_code, var_dir = lookup_var(user_data, 'var_dir')
    exit_code, dis_dir = lookup_var(user_data, 'dis_dir')

    dis_dir = var_dir + '/' + dis_dir
    while True:
        exit_code, solar_data = file_read(var_dir + '/solar.var')
        if exit_code == 0:
            exit_code, pt100_bat_volt = lookup_var(solar_data, 'pt100_bat_volt')
            exit_code, pt100_amp = lookup_var(solar_data, 'pt100_amp')
            exit_code, pt100_solar_volt = lookup_var(solar_data, 'pt100_solar_volt')
            exit_code, pt100_stat = lookup_var(solar_data, 'pt100_stat')
            exit_code, pt100_flt = lookup_var(solar_data, 'pt100_flt')
            exit_code, pt100_bat_temp = lookup_var(solar_data, 'pt100_bat_temp')
            exit_code, pt100_fet_temp = lookup_var(solar_data, 'pt100_fet_temp')
            exit_code, pt100_inductor_temp = lookup_var(solar_data, 'pt100_inductor_temp')
            exit_code, inv_bat_volt = lookup_var(solar_data, 'inv_bat_volt')
            exit_code, inv_dc_amp = lookup_var(solar_data, 'inv_dc_amp')
            exit_code, inv_ac_rms_volt_out = lookup_var(solar_data, 'inv_ac_rms_volt_out')
            exit_code, inv_ac_input_amp = lookup_var(solar_data, 'inv_ac_input_amp')
            exit_code, inv_ac_output_amp = lookup_var(solar_data, 'inv_ac_output_amp')
            exit_code, inv_frequency = lookup_var(solar_data, 'inv_frequency')
            exit_code, inv_stat = lookup_var(solar_data, 'inv_stat')
            exit_code, inv_flt = lookup_var(solar_data, 'inv_flt')
            exit_code, inv_inverting_led = lookup_var(solar_data, 'inv_inverting_led')
            exit_code, inv_charging_led = lookup_var(solar_data, 'inv_charging_led')
            exit_code, inv_bat_temp = lookup_var(solar_data, 'inv_bat_temp')
            exit_code, inv_fet_temp = lookup_var(solar_data, 'inv_fet_temp')
            exit_code, inv_trans_temp = lookup_var(solar_data, 'inv_trans_temp')
            # ++++++++++++++++++++++++++
```

```
            # Get the time of the reading
            exit_code, current_time = lookup_var(solar_data, 'current_time')
            print('++ ' + strftime('%Y-%m-%d  %H:%M:%S', localtime(float(current_time))) + '++')
            # ++++++++++++++++++++
            # Print the PT-100 data
            pt100_watt = str(round(float(pt100_bat_volt) * float(pt100_amp)))
            print('PT-100 charge controller')
            print('- Voltage           ' + pt100_bat_volt + ' V ')
            print('- Amperage          ' + pt100_amp + ' A ')
            print('- Power             ' + pt100_watt + ' W')
            print('- Solar voltage     ' + pt100_solar_volt + ' V ')
            print('- Status            ' + pt100_stat)
            print('- Fault             ' + pt100_flt)
            print('- Battery temp      ' + pt100_bat_temp + ' C')
            print('- FET temp          ' + pt100_fet_temp + ' C')
            print('- Inductor temp     ' + pt100_inductor_temp + ' C')
            print('MS4448 inverter')
            print('- DC Voltage        ' + inv_bat_volt + ' V ')
            print('- DC Amperage       ' + inv_dc_amp + ' A ')
            print('- DC Wattage        ' + str(round(float(inv_dc_amp) * float(inv_bat_volt))) + ' W ')
            print('- AC Voltage        ' + inv_ac_rms_volt_out + ' V ')
            print('- AC Amperage IN    ' + inv_ac_input_amp + ' A ')
            print('- AC Amperage OUT   ' + inv_ac_output_amp + ' A ')
            print('- Frequency         ' + inv_frequency + ' Hz ')
            print('- Status            ' + inv_stat)
            print('- Fault             ' + inv_flt)
            print('- Inverting LED     ' + inv_inverting_led)
            print('- Charging LED      ' + inv_charging_led)
            print('- Battery temp      ' + inv_bat_temp + ' C')
            print('- FET temp          ' + inv_fet_temp + ' C')
            print('- Transformer temp  ' + inv_trans_temp + ' C')
            print('')
            inv_amps = float(inv_ac_input_amp)
            if float(inv_ac_output_amp) > float(inv_ac_input_amp):
                inv_amps = float(inv_ac_output_amp)
            inv_watt = round(float(inv_ac_rms_volt_out) * inv_amps)
            output = 'Inverter,30,' + inv_stat + ',30,' + str(inv_bat_volt) + ' V / ' + str(inv_watt) + ' W,30'
            file_write(dis_dir + '/Inverter.dis', output)
            file_write('var/local_dis/Inverter.dis', output)
            if float(pt100_watt) > 0:
                output = 'PT-100,30,' + pt100_stat + ',30,' + str(pt100_bat_volt) + ' V / ' + str(pt100_watt) + '
W,30'
                file_write(dis_dir + '/Solar.dis', output)
                file_write('var/local_dis/Solar.dis', output)
        file_write(var_dir + '/MS4448_display.var', str(time()))
        sleep(10)

ms4448()
```