Pythonで通信対応電源と拡張UARTで通信する方法について（Windows編）

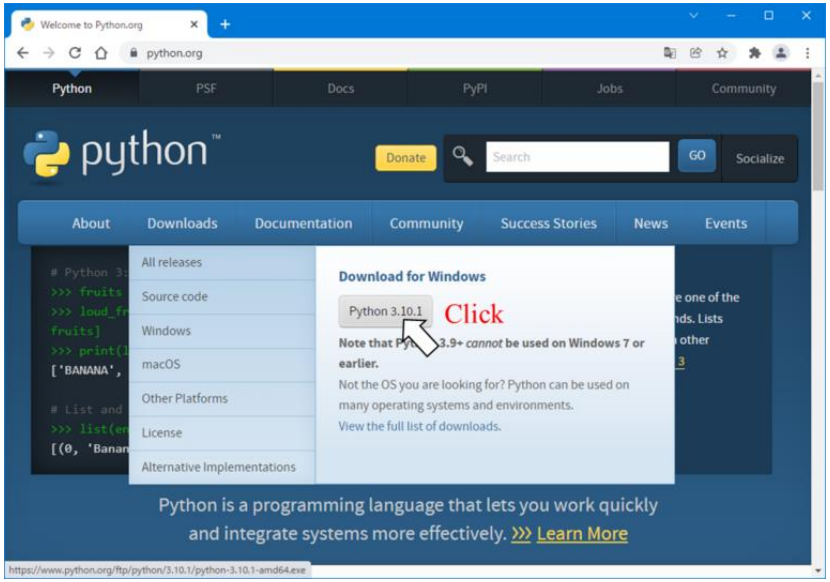
2023年12月15日　ver 1.0

目次

1．Pythonのインストール

下記のサイトからPythonをダウンロードして、PCにインストールします。
https://www.python.org/

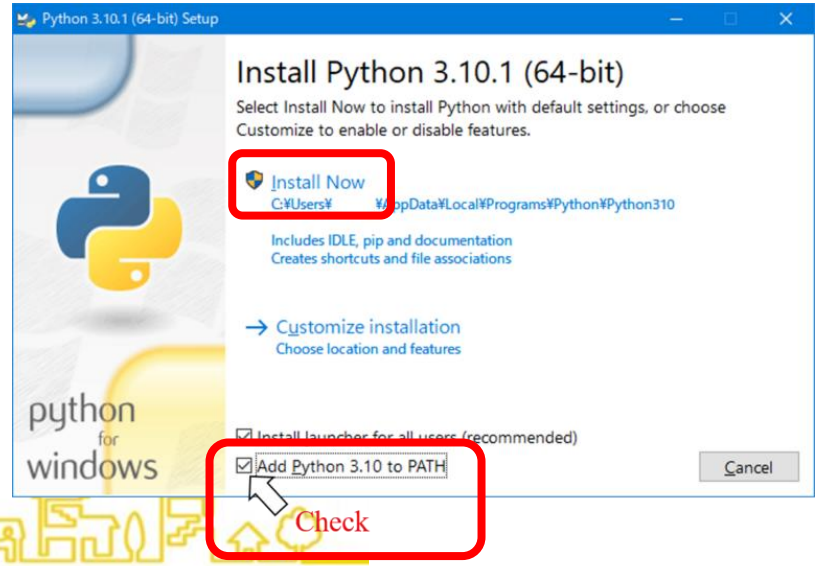
Downloads -> Python 3.xx.x からダウンロードできます。



ダウンロードしたインストーラを実行します。

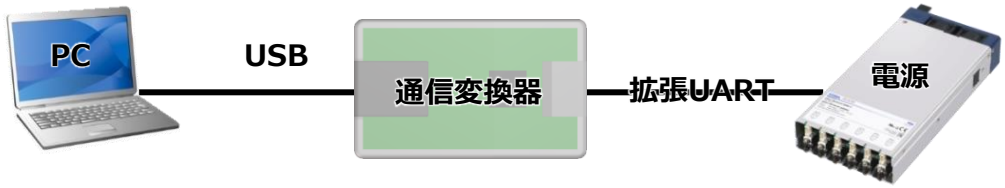
インストール時
「Add Python 3.xx to PATH」
にチェックを付けます。

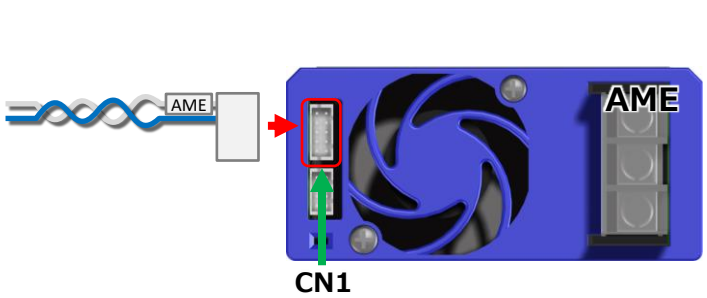
「Install Now」をクリックすると、インストールが始まります。

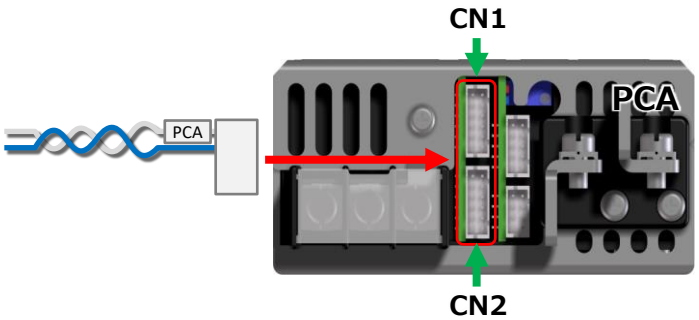


2.パソコンと電源の接続

拡張UART通信対応電源をパソコンに接続します。接続には通信変換器が必要になります。
詳細はについては、各電源の拡張UARTマニュアルをご覧下さい。



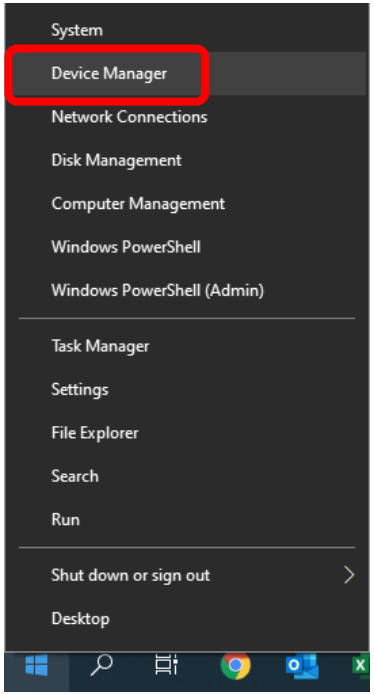


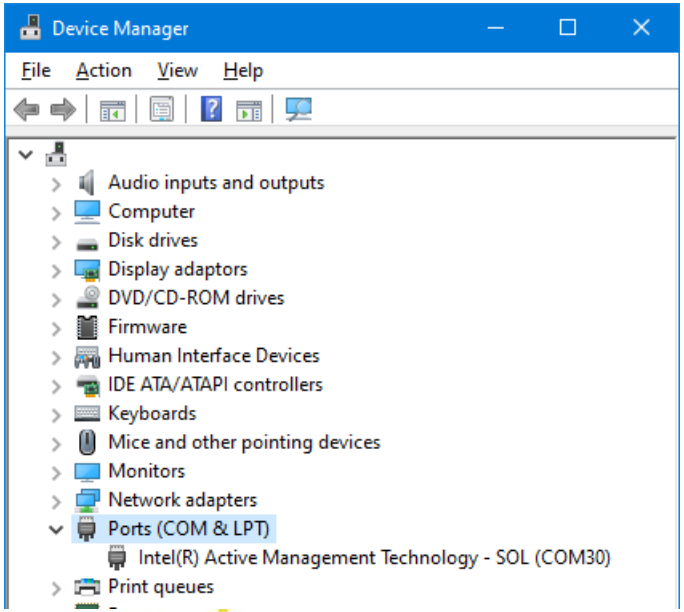


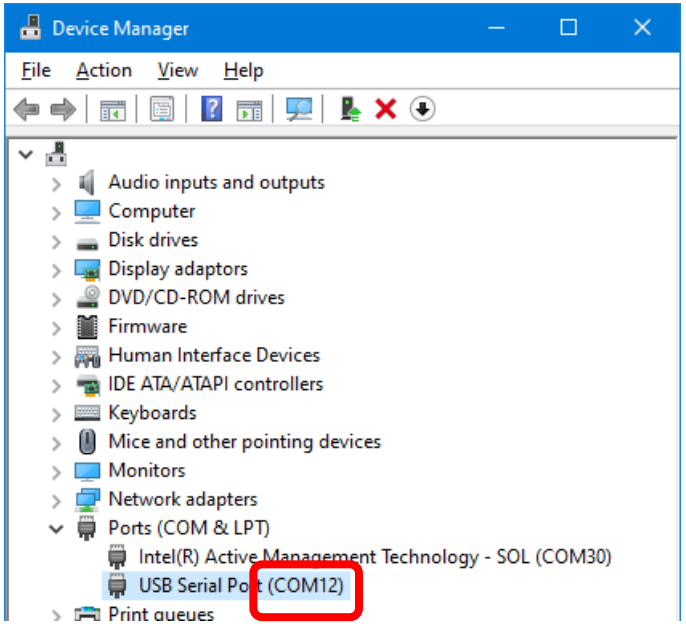
3.シリアル通信ポートの確認

シリアル通信を行うためには、電源と接続した変換器のシリアル通信ポートの名称が必要になります。

【調べ方】
・デバイスマネージャーを開きます。（Windowsアイコンを右クリックし、デバイスマネージャーをクリックします。）
・「ポート(COMとLPT)」を開きます。
・変換器をパソコンに接続します。
・新しく増えたポートの末尾の「COMxx」を記録します。(例では「COM12」)





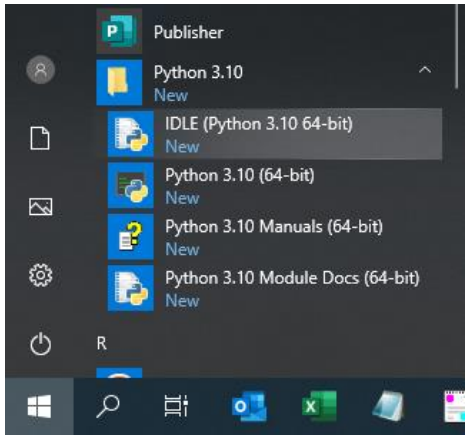


4.Pythonで通信を行う

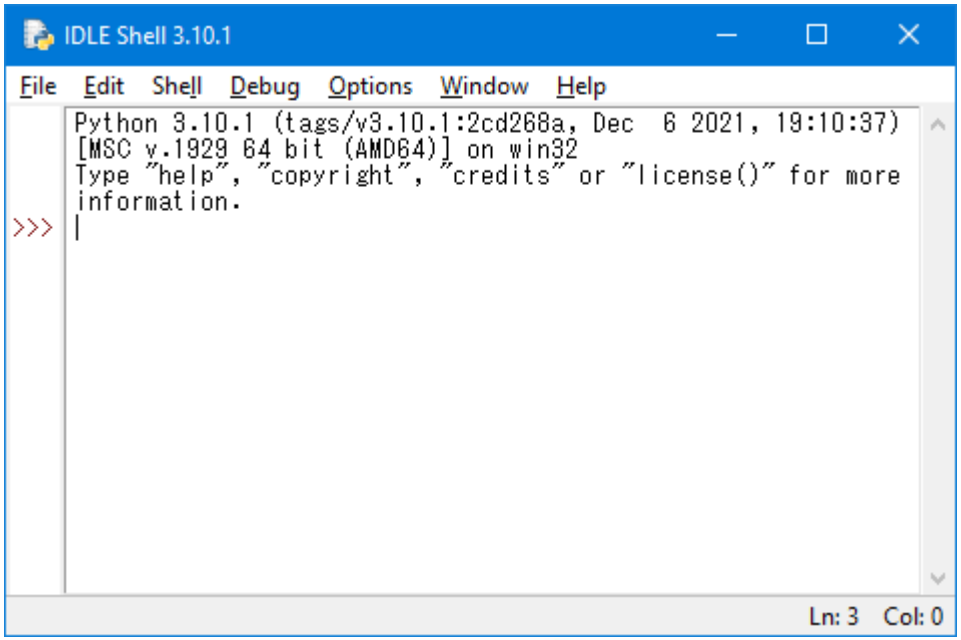
Pythonでプログラムを作成し、変換器を通して電源と通信を行っていきます。

まずは、Pythonのプログラムを作成実行するIDLEを起動します。

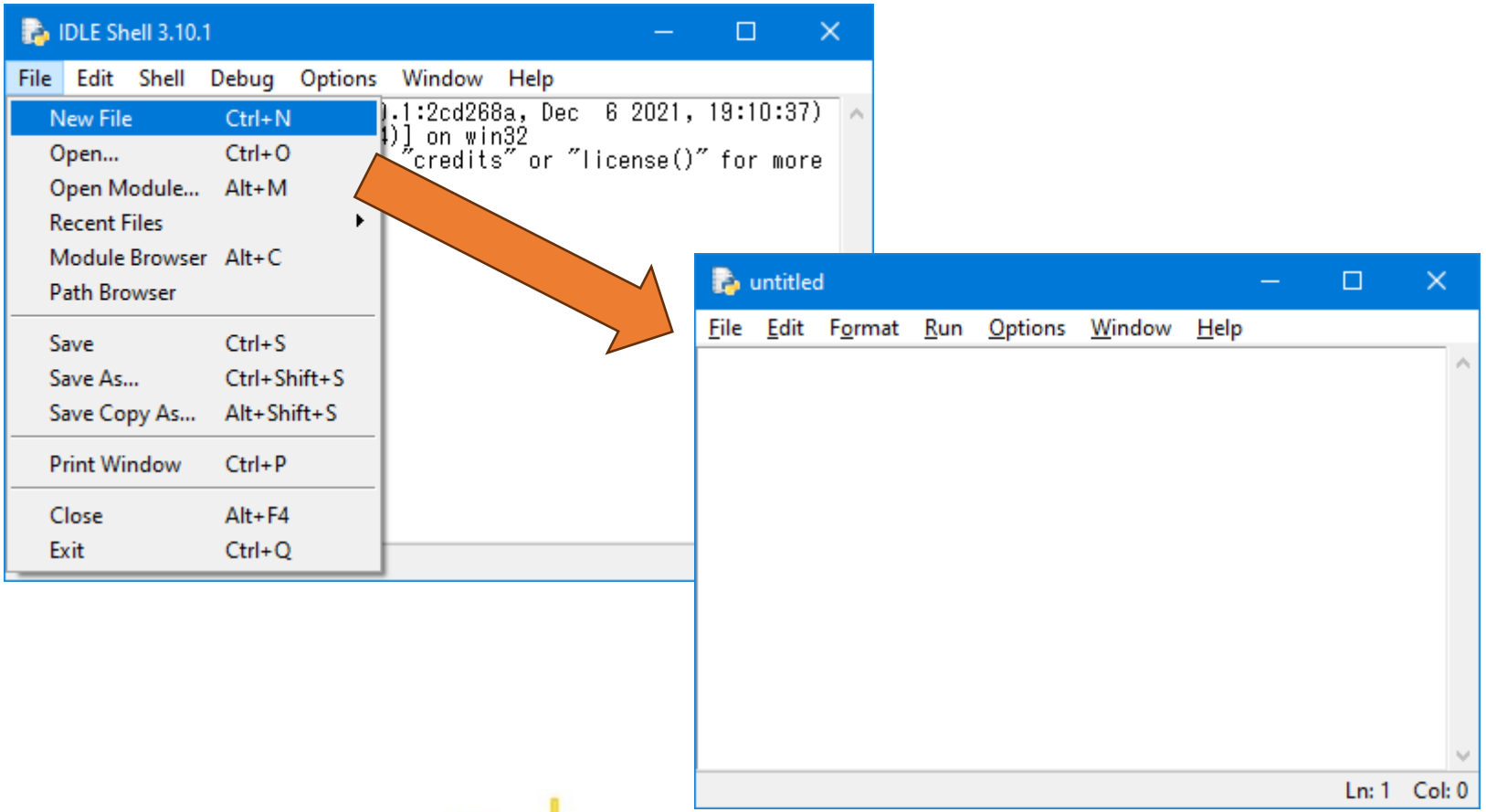
スタートメニュー -> Python 3.xx -> IDLE(Python 3. …)
で起動できます。



IDLEが起動すると下記の画面が表示されます。



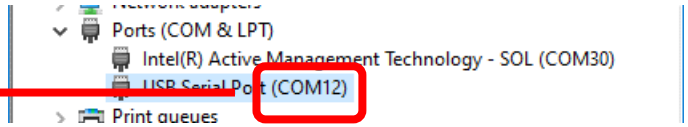
File -> New File をクリックして、プログラムを記述するエディタを用意します。



4.Pythonで通信を行う

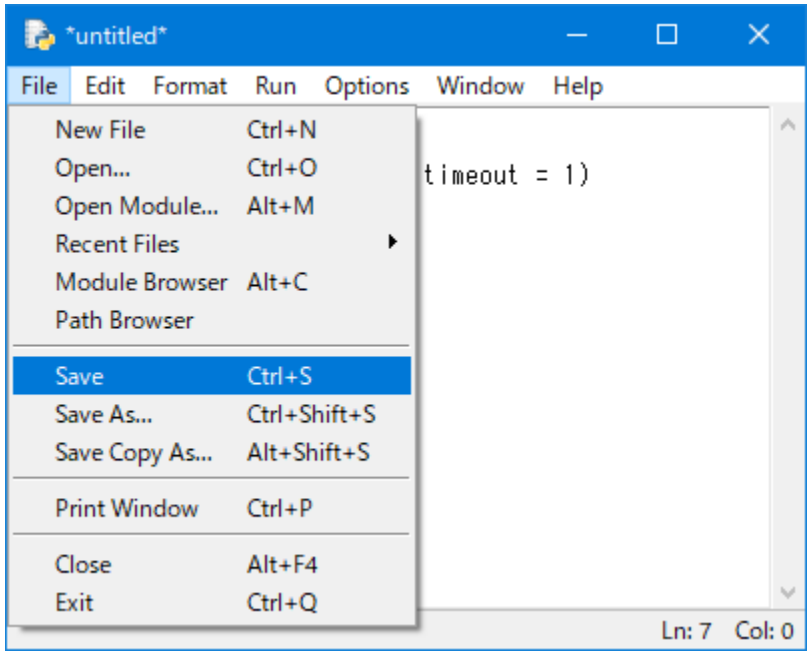
```
import serial

addr       = 7            # 電源のアドレス
send_data  = bytearray(5) # 送信データ

# 電源の出力をOFFにする
send_data[0] = 0x1E
send_data[1] = 0
send_data[2] = 0x08
send_data[3] = 0x1C
send_data[4] = 0x01

# チェックサムを計算して付加する
checksum = (sum(send_data) - send_data[1]) & 0xF
send_data[1] |= checksum << 1

# 全パケットの先頭に電源のアドレスを付加する
for i in range(len(send_data)) :
    send_data[i] |= addr << 5

# 通信する
ser = serial.Serial(
    port     = "COM12",
    baudrate = 2400,
    parity   = serial.PARITY_EVEN,
    timeout  = 0.2
  )
ser.write(send_data)
recv_data = ser.read(10)
ser.close()
print(recv_data)
```

左記のプログラムをエディタに記述します。
port　= “COM12”の行は
先ほど調べた変換器のCOMポート番号に直してください。

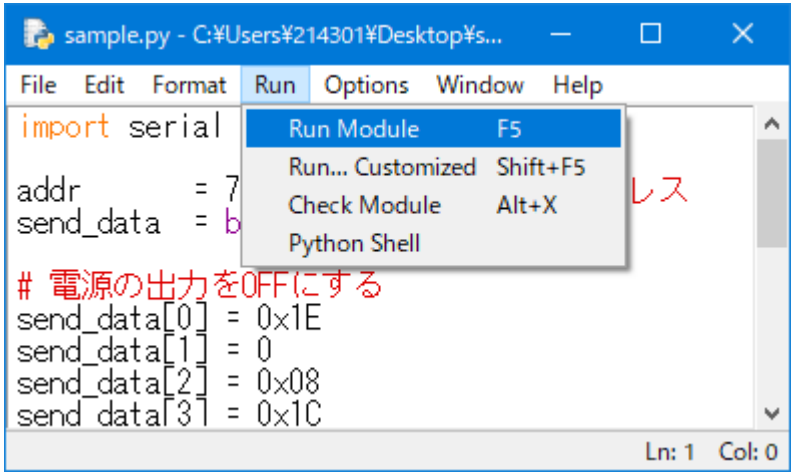


4.Pythonで通信を行う

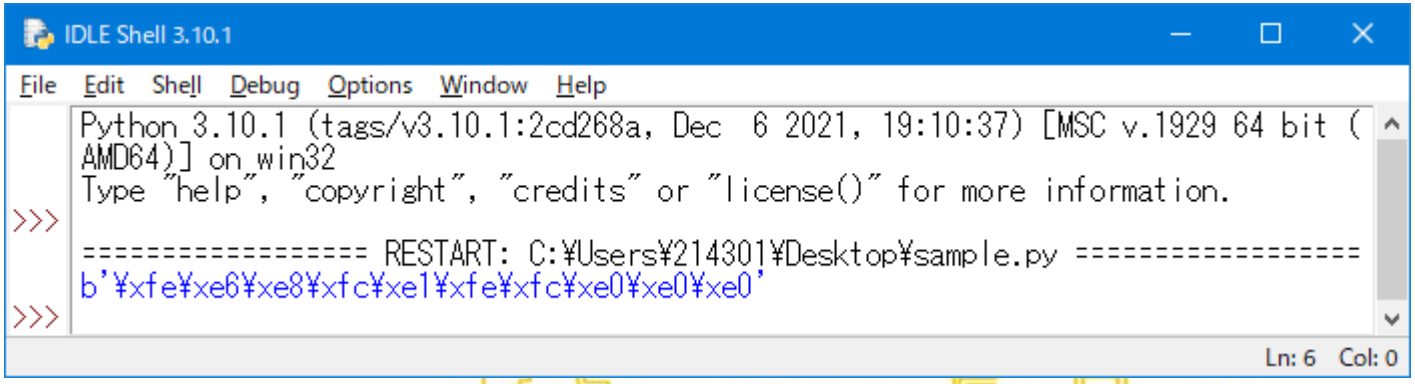
入力を行った後は、File -> Save で保存を行います。



Run -> Run Module をクリックし、プログラムを実行します。



実行すると別ウィンドウが開かれ、以下のように表示され、電源の出力がOFFになります。



5.汎用的な拡張UART通信のプログラム

4章では20ビットコマンドを送信するだけの例でしたので、プログラムを追加してモジュール化して汎用的に使えるようにします。
モジュール化したファイルを「exuart.py」として用意しました。
使う時は、importして部品として簡単に使えます。（使用例：sample.py）

exuart.py

```
import serial

class ExUartError(Exception):
    def __init__(self):
        pass
～

class ExUartClass:
    def __init__(self, port = ""):
        '''コンストラクタ'''
        self.addr       = 7             # 電源のアドレス
        self.send_data  = bytearray(5) # 送信データ
        self.recv_data  = bytes(10)     # 受信データ
        self.port       = port
        return
～


        return recv_res
```

sample.py

```
import exuart

ps = exuart.ExUartClass("COM12")

ps.set_addr(7)

print("出力電圧を変更")
ps.send_cmd_5bit(0x0A, 10000)

addr = 6
print("電源のアドレスを変更:{0}".format(addr))
ps.send_cmd_10bit(0x1A, 0x10, addr)
ps.set_addr(addr)

print("電源の出力をOFFにする")
ps.send_cmd_20bit(0x1E, 0x08, 0x1C, 0x00)

vin = ps.send_cmd_20bit(0x1E, 0x08, 0x00, 0x01)
print("入力電圧:{0}[V]".format(vin/100))

addr = 7
print("電源のアドレスを変更:{0}".format(addr))
ps.send_cmd_10bit(0x1A, 0x10, addr)
ps.set_addr(addr)
```

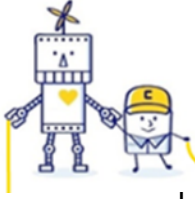
5.汎用的な拡張UART通信のプログラム

ExUartClassの説明

【コンストラクタ】
ExUartClass(port = “”)
説明
拡張UART通信を行うクラスになります。
引数
port (str) ：シリアルポート名を指定します。(例:“COM12”)

【メソッド】
・set_port(port)
説明
使用するシリアルポートを指定します。
引数
port (str) ：シリアルポート名を指定します。(例:“COM12”)
戻り値
なし

・set_addr(addr)
説明
電源のアドレスを指定します。
引数
addr (int) ：電源のアドレスを1～7で指定します。
戻り値
なし



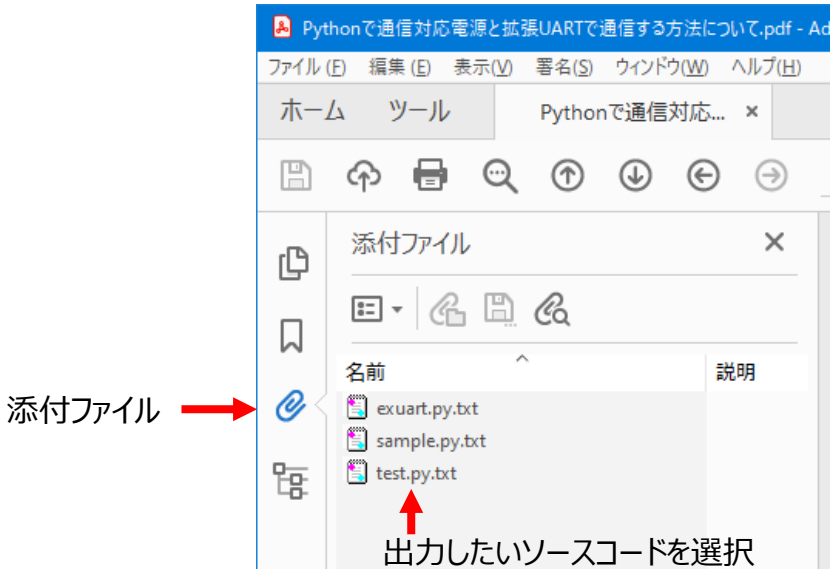
・send_cmd_5bit(cmd1, arg)
説明
　電源に5ビットコマンドを送信し戻り値を取得します。
引数
　cmd1 (int) ：拡張UARTコマンド
　arg (int) ：引数(16ビット)
戻り値
　コマンドの実行で得られた電源からの戻り値

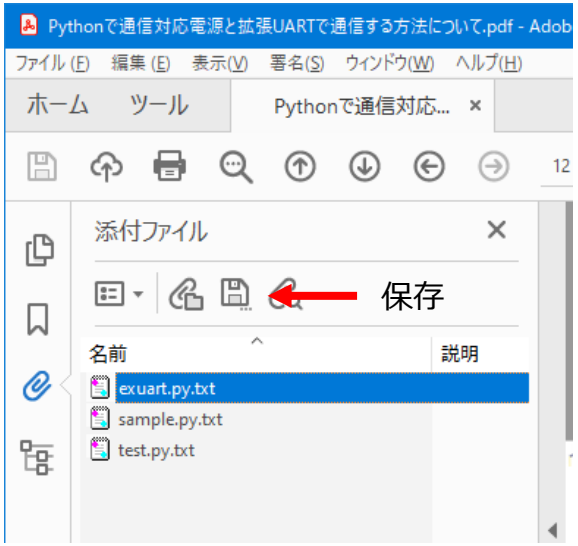
・send_cmd_10bit(cmd1, cmd2, arg)
説明
　電源に5ビットコマンドを送信し戻り値を取得します。
引数
　cmd1 (int) ：拡張UARTコマンド1番目
　cmd2 (int) ：拡張UARTコマンド2番目
　arg (int) ：引数(10ビット)
戻り値
　コマンドの実行で得られた電源からの戻り値

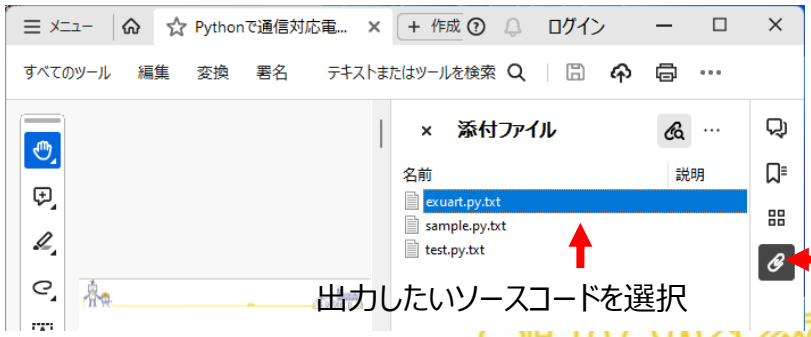
・send_cmd_20bit(cmd1, cmd2, cmd3, cmd4)
説明
　電源に5ビットコマンドを送信し戻り値を取得します。
引数
　cmd1 (int) ：拡張UARTコマンド1番目
　cmd2 (int) ：拡張UARTコマンド2番目
　cmd3 (int) ：拡張UARTコマンド3番目
　cmd4 (int) ：拡張UARTコマンド4番目
戻り値
　コマンドの実行で得られた電源からの戻り値

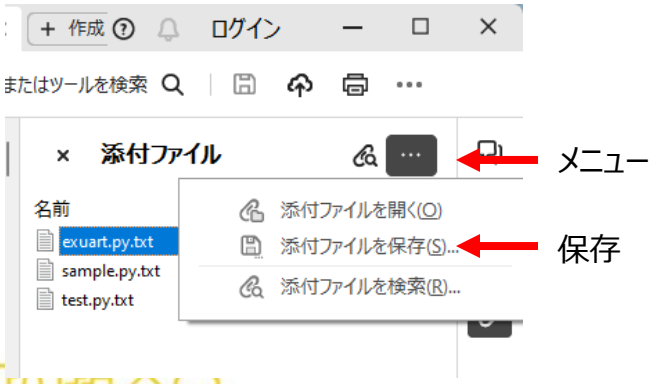
6.ソースコード

今までの説明で使用したソースコードを以降のページに収録しています。また、このPDFの添付ファイルにソースコードを添付しています。
Acrobat Readerのバージョンにより操作箇所が異なります。下記に2種類のバージョンでの例を示します。









test.py

```
import serial

addr       = 7            # 電源のアドレス
send_data  = bytearray(5) # 送信データ

# 電源の出力をOFFにする
send_data[0] = 0x1E
send_data[1] = 0
send_data[2] = 0x08
send_data[3] = 0x1C
send_data[4] = 0x01

checksum = (sum(send_data) - send_data[1]) & 0xF
send_data[1] |= checksum << 1

# 全パケットの先頭に電源のアドレスを付加する
for i in range(len(send_data)) :
    send_data[i] |= addr << 5

ser = serial.Serial(
    port     = "COM5",
    baudrate = 2400,
    parity   = serial.PARITY_EVEN,
    timeout  = 0.2
  )
ser.write(send_data)
recv_data = ser.read(10)
ser.close()
print(recv_data)

```

sample.py

```
import exuart

ps = exuart.ExUartClass("COM12")

ps.set_addr(7)

print("出力電圧を変更")
ps.send_cmd_5bit(0x0A, 10000)

addr = 6
print("電源のアドレスを変更:{0}".format(addr))
ps.send_cmd_10bit(0x1A, 0x10, addr)
ps.set_addr(addr)

print("電源の出力をOFFにする")
ps.send_cmd_20bit(0x1E, 0x08, 0x1C, 0x00)

vin = ps.send_cmd_20bit(0x1E, 0x08, 0x00, 0x01)
print("入力電圧:{0}[V]".format(vin/100))

addr = 7
print("電源のアドレスを変更:{0}".format(addr))
ps.send_cmd_10bit(0x1A, 0x10, addr)
ps.set_addr(addr)

```

exuart.py

```python
import serial

class ExUartError(Exception):
    def __init__(self):
        pass

    def __str__(self):
        return "拡張UARTでエラーが発生しました。"

class AddressError(ExUartError):
    def __str__(self):
        return "アドレスの範囲が正しくありません。"

class NumOfReceivingDataError(ExUartError):
    def __str__(self):
        return "返信パケット数が不足しています。"

class ChecksumError(ExUartError):
    def __str__(self):
        return "返信パケットのチェックサムが不一致です。"

class SoftwearError(ExUartError):
    def __init__(self, recv_id):
        self.recv_id = recv_id
        return

    def __str__(self):
        return "ソフトウェアエラーです。エラーコード：{0}".format(self.recv_id)

class ExUartClass:
    def __init__(self, port = ""):
        '''コンストラクタ'''
        self.addr       = 7            # 電源のアドレス
        self.send_data  = bytearray(5) # 送信データ
        self.recv_data  = bytes(10)    # 受信データ
        self.port       = port
        return

    def set_port(self, port : str) -> None:
        self.port = port
        return

    def set_addr(self, addr : int) -> None:
        if addr < 1 or 7 < addr :
            raise AddressError()
        self.addr = addr
        return

    def send_cmd_5bit(self, cmd1, arg) -> int:
        self.send_data[0] = cmd1
        self.send_data[1] = (arg >> 15) & 0x01
        self.send_data[2] = (arg >> 10) & 0x1F
        self.send_data[3] = (arg >>  5) & 0x1F
        self.send_data[4] = (arg      ) & 0x1F
        return self._send()

    def send_cmd_10bit(self, cmd1, cmd2, arg) -> int:
```

```
58          self.send_data[0] = cmd1
59          self.send_data[1] = 0
60          self.send_data[2] = cmd2
61          self.send_data[3] = (arg >> 5) & 0x1F
62          self.send_data[4] = (arg     ) & 0x1F
63          return self._send()
64
65      def send_cmd_20bit(self, cmd1, cmd2, cmd3, cmd4) -> int:
66          self.send_data[0] = cmd1
67          self.send_data[1] = 0
68          self.send_data[2] = cmd2
69          self.send_data[3] = cmd3
70          self.send_data[4] = cmd4
71          return self._send()
72
73      def _send(self) -> int:
74          # チェックサムを計算して付加する
75          checksum = (sum(self.send_data) - self.send_data[1]) & 0xF
76          self.send_data[1] |= checksum << 1
77
78          # 全パケットの先頭に電源のアドレスを付加する
79          for i in range(len(self.send_data)) :
80              self.send_data[i] |= self.addr << 5
81
82          # 通信
83          ser = serial.Serial(
84              port     = "COM5",
85              baudrate = 2400,
86              parity   = serial.PARITY_EVEN,
87              timeout  = 0.2
88            )
89          ser.write(self.send_data)
90          self.recv_data = ser.read(10)
91          ser.close()
92
93          # エラー確認
94          if len(self.recv_data) < 10:
95              raise NumOfReceivingDataError
96
97          checksum = (sum(self.recv_data[5:10]) - self.recv_data[6]) & 0xF
98          if checksum != ((self.recv_data[6] >> 1) & 0xF):
99              raise ChecksumError
100
101         # 識別子を取得
102         recv_id = self.recv_data[5] & 0x1F;
103
104         # 戻り値を取得
105         recv_res = self.recv_data[6] & 0x1
106         for n in self.recv_data[7:10]:
107             recv_res <<= 5
108             recv_res +=  n & 0x1F
109
110         # ソフトウェアエラーか判定
111         if 0b11111 == recv_id:
112             raise SoftwearError(recv_res)
113
114         return recv_res
115
```

注意

本資料に記載されている内容は本資料発行時点のものであり、製品の仕様変更および改良などのために予告なく変更することがあります。最新版はコーセルのホームページをご確認ください。

本資料の内容につきましては、正確さを期するために万全の注意を払っておりますが、本資料中の誤記や情報の抜け、あるいは、情報の使用に起因する間接障害を含むいかなる損害に対しても、弊社は責任を負いかねますので、あらかじめご了承ください。

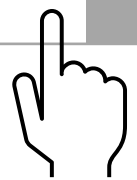
技術お問い合わせ専用ホットライン

■フリーダイヤル： 0120-52-8151

営業時間9：00～12：00／13：00～17：00（土曜・日曜・祝日・当社休日を除く）

お問い合わせは「コーセル　サポート」で検索

コーセル　サポート



COSEL

profile of COSEL CO.,LTD.

『 顧客起点のニーズを捉え、高付加価値製品とサービスの実現を図る 』