

Raspberry Pi で RFID リーダ・ライタを制御する例 (USB 接続と UART 接続)

2024 年 2 月 9 日 第 1.3.0 版

株式会社アートファイネックス

はじめに

本書は、Raspberry Pi 3 Model B(OS:Raspberry Pi OS)を使用し、アートファイネックス社製 RFID リーダ・ライタ(組込用モジュール、USB インターフェース搭載器)を制御する例を記載したものです。

無断転載を禁じます。

本書の内容は、断りなく変更することがあります。

※ Microsoft Windows は、米国 Microsoft Corp.の登録商標です。

※ Raspberry Pi は、Raspberry Pi 財団の登録商標です。

※ FTDI は、Future Technology Devices International Limited の商標または登録商標です。

※ その他、商品名及び製品名などは一般に各社の商標または登録商標です。

1. 用意するもの

- ・開発用 PC
- ・Raspberry Pi 3 Model B(以下 ラズパイ)
 - 本体
 - 電源(micro USB から供給)
 - micro SD メモリーカードと変換アダプター(必要に応じて)
 - USB キーボード
 - USB マウス
 - モニタ(HDMI)と HDMI ケーブル
 - ネットワークケーブル(SSH 接続する場合)
- ・RFID リーダ・ライタ(モジュール、f ファミリ、CB ファミリ)とアンテナ
- ・RF タグ
- ・ラズパイ- RFID リーダ・ライタ 通信ケーブル(USB など)

2. 概要

手順の概要は以下のとおりです。(インストールや設定についての詳細は最新情報をお調べください)

1. PC で microSD カードに OS(Raspberry Pi OS)を書きます。
 - PC に Imager(SD カード作成ツール)をインストールします。
 - PC に microSD カードを接続して Imager を起動して書込みます。
2. ラズパイに microSD カードを挿して起動し、必要な設定をします。
 - 例:IP アドレス、パスワード、タムゾーン、など
 - (SSH を有効にしてラズパイを LAN 接続すると PC 上で開発できます)
 - ラズパイの UART で行うときは下の設定も必要です。
 - /boot/config.txt の末尾に下の2行を追記
 - dtoverlay=pi3-miniuart-bt
 - enable_uart=1
 - /boot/cmdline.txt に記載されている下の箇所を削除
 - console=serial0,115200
3. ラズパイと RFID リーダ・ライタを接続します。
4. python3 で RF タグの ID を読んだりするアプリを開発します。

※弊社の RFID リーダ・ライタの USB は、FTDI 社製の USB ドライバが必要ですが、Raspberry Pi OS には標準でインストールされているようです。
※必要に応じて PySerial をインストールしてください。

次章からは上の 3 と 4 について記載します。

3. 各種バージョン

確認したラズパイは下のとおりでした。

1. OS(ラズパイ)

```
pi@raspberrypi:~ $ lsb_release -a
# 以下出力
# No LSB modules are available.
# Distributor ID: Raspbian
# Description: Raspbian GNU/Linux 9.4 (stretch)
# Release: 9.4
# Codename: stretch
```

2. Kernel

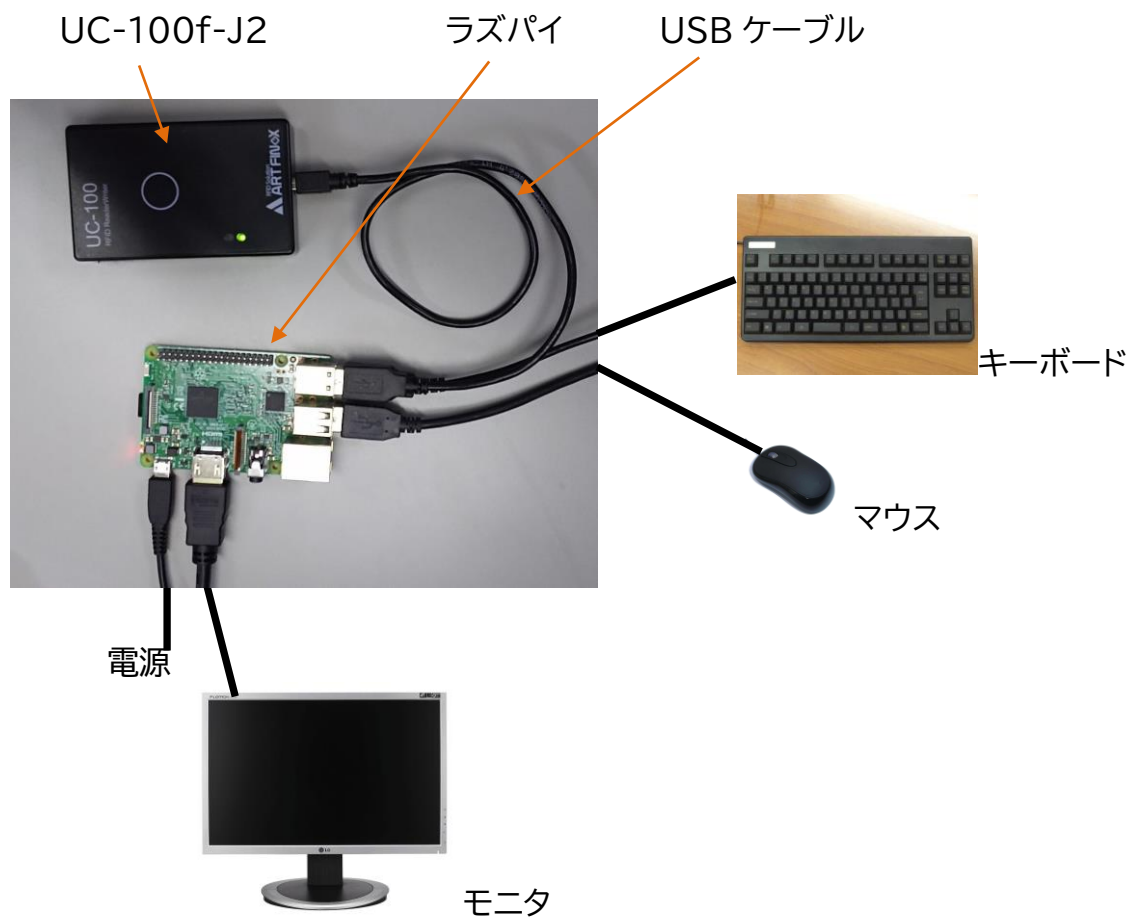
```
pi@raspberrypi:~ $ uname -a
# 以下出力
# Linux raspberrypi 4.14.34-v7+ #1110 SMP Mon Apr 16 15:18:51 BST
2018 amv71 GNU/Linux
```

3. python

```
pi@raspberrypi:~ $ python3 -V
# 以下出力
# Python 3.5.3
```

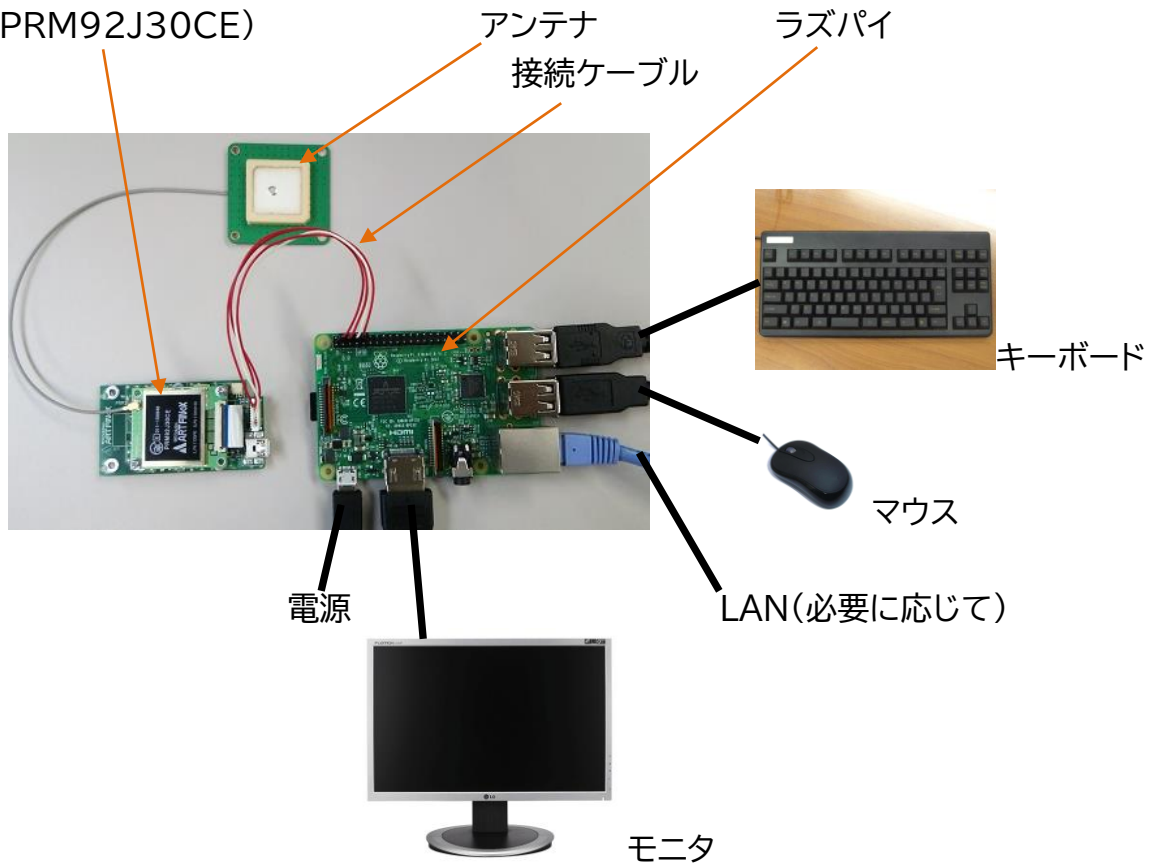
4. ラズパイと RFID リーダ・ライタとの接続例

4.1. USB での接続例



4.2. UART での接続例

RFID リーダ・ライタ + IF 変換基板
(PRM92J30CE)



接続ケーブルの配線

ラズパイ		RFID リーダ・ライタ	
ピン番号	機能	ピン番号	機能
4	5V PWR	1	5V PWR
6	GND	2	GND
8	TXD	9	RXD
10	RXD	8	TXD
※	GPIO	5	RESET

5. アプリケーションソフトウェア開発手順例

始めに、リーダ・ライタがつながっているポートを確認するために、ラズパイで下のアプリを作成します。(port.py)

```
import glob
ports = glob.glob('/dev/tty[A-Za-z]*')
for port in ports:
    print(port)
```

ラズパイにリーダ・ライタを接続して下のコマンドを実行すると

```
$python3 port.py
```

下のように表示されます。

```
/dev/ttyS0
```

```
/dev/ttyUSB0    USB で接続するとき
```

```
/dev/ttyprintk
```

```
/dev/ttyAMA0    UART で接続するとき
```

RFID リーダ・ライタには下の2種類(fファミリとCBファミリ)があります。

それぞれコマンド仕様(プロトコル仕様)が異なるため、ソフトウェアも異なります。

	f ファミリ	CB ファミリ
主な対応機種	PRM92J30CE-S UC-200f-J2 UT-200f-J2 UP-200f-J2(USB)	UP-200-J2 UP-250-J2 UP-1000-J2 UXA250 UP(2/4/8/16)-200-J2 UP(2/4/8/16)-250-J2 UP(2/4/8/16)-1000-J2

RF タグの ID を取得するアプリの例です。

◇ファミリ用

```
import serial
ser = serial.Serial("/dev/ttyUSB0")
ser.baudrate = 115200
sendData = b'\xBB\x80\x22\x00\x02\x01\xA1\x7E'
ser.write(sendData)

while (True):

    recvData = ser.read(5)
    len1 = recvData[3]*256+recvData[4]+1
    temp1 = ser.read(len1)
    if (recvData[2] == 0xFF):
        print('No tag!!')
        break
    elif (recvData[2] == 0x27):
        len2 = len(temp1)
        temp2 = temp1[2:len2-4]
        print(temp2.hex())

ser.close()
```

UART で接続するときは
ttyAMA0 にします。

ボーレートは使用する機器
に合わせてください。

コマンドを送信
\\は¥と表示されることも
あります。

最初の 5byte を読む

残りのデータ数を取得

残りのデータを読む

RF タグが無いなどのエラー

取得終了通知

RF タグデータ通知

受信データ数を取得

EPC を取得

ラズパイにリーダー・ライタを接続して下のコマンドを実行します。

```
$python3 ReadEpc_usb.py
```

RF タグがあれば、取得した RF タグの ID を表示します。(複数個取得したら複数個表示します。)

また、RF タグがかざされていないと「No tag!!」と表示されます。

「URW-SP プロトコル仕様書」内の「Read Type C Tag ID Single(メッセージ区分:0x22)」
コマンドを使用したサンプルアプリです。

※この仕様書に記載されている「メッセージフォーマット」のご理解が必要です。

◇CB ファミリ用

```
import serial
ser = serial.Serial("/dev/ttyUSB0")
ser.baudrate = 115200
sendData = b'\x53\x00\x00\x00\x20\x00\x00\x00\x20\x00\x00\x00\x00\x00\x00\x00\x93'
ser.write(sendData) # コマンドを送信
recvData = ser.read(16) # 最初の 16byte を読む
len1 = recvData[7]*256+recvData[6]+1 # 残りのデータ数を取得
tagData = ser.read(len1) # 残りのデータを読む
if (recvData[1] != 0x00): # RF タグが無いなどのエラー
    print('No tag!!')
else:
    numTag = tagData[0] # 読んだ RF タグ数を取得して表示
    print('タグ数 : ' + str(numTag))
    dataPointer = 1
    for num in range(0, numTag): # RF タグ数回繰り返す
        antNum = tagData[dataPointer] # アンテナ番号を取得
        dataPointer += 1
        idLen = tagData[dataPointer] # RF タグ情報の長さを取得
        dataPointer += 1
        # RF タグ情報は PC(2byte)+EPC(可変)+CRC(2byte)
        epc = tagData[dataPointer+2:dataPointer+idLen-2] # EPC を取得
        dataPointer += idLen
        print('Ant' + str(antNum) + ', ' + epc.hex())
ser.close()
```

ラズパイにリーダー・ライタを接続して下のコマンドを実行します。

```
$python3 ReadEpc_usb.py
```

RF タグがあれば、取得した RF タグの ID を表示します。(複数個取得したら複数個表示します。)

また、RF タグがかざされていないかったら「No tag!!」と表示されます。

「リーダー・ライタプロトコル UHF 版仕様書」内の「タグ ID 取得(メッセージ区分: 20h)」コマンドを使用したサンプルアプリです。

※この仕様書に記載されている「メッセージフォーマット」のご理解が必要です。

6. アプリケーションソフトウェアの応用例

fファミリとCBファミリ、それぞれの応用アプリを記します。

このアプリはどちらも使い方は同じです。

アプリを実行すると下のように行うコマンド(メニュー)を聞かれますので、行いたいコマンドを入力します(例 t<Enter>)。するとそのコマンドが実行されて結果が表示されます。

※コマンドによっては、いくつかのパラメータの入力を促されます。

t:タグ取得、r:リード、w:ライト、s:送信出力設定、g:送信出力取得、q:終了
>>>

◇タグ取得

かざされているタグを読み、それらのPC値とEPCを表示します。

◇リード

かざされているタグのメモリを読みます。

r の後に、メモリバンク/開始アドレス/バイト数 を入力すると、そのメモリを読んで表示します。

◇ライト

かざされているタグのメモリにデータを書きます。

w の後に、メモリバンク/開始アドレス/バイト数/書込むデータ を入力すると、メモリに書きます。

◇送信出力設定

送信出力値を設定します。

s の後に、送信出力値を入力すると、その値を設定します。

◇送信出力取得

送信出力値を取得します。

送信出力値を取得して表示します。

◇終了

アプリを終了します。

タグ取得の実行例(f ファミリ用)

6 つの RF タグを読んだ例

t:タグ取得、r:リード、w:ライト、s:送信出力設定、g:送信出力取得、q:終了

>>> t

PC : 4000, EPC : bb22222222222222222222222222222222

PC : 3000, EPC : e2040019960b014905507bce

PC : 3000, EPC : e20000001a1102181450c575

PC : 3400, EPC : e2004706e2d06026f3e10106

PC : 3000, EPC : e20042187d6060130300ba6e

PC : 3000, EPC : e20047509c1508ca31c29a08

タグ数 : 6

メモリリードの実行例(共通)

TID(12byte)を読んだ例

t:タグ取得、r:リード、w:ライト、s:送信出力設定、g:送信出力取得、q:終了

>>> r

バンク(0:Reserve, 1:EPC, 2:TID, 3:User) >> 2

先頭アドレス(ワード) >> 0

データ長さ(バイト) >> 12

data :e2003412016cff00043f6f81

◇ファミリ用

```
import serial
# ---- 関数 -----
# チェックコードを計算する関数
#  戻り値 : start から end までを XOR した値
#  array  : データが入っている配列
#  start  : 開始位置
#  end    : 終了位置
def func_check_code(array, start, end):
    code = 0
    for i in range(start, end):
        code ^= array[i]
    return code

# タグ ID を取得する関数
def func_gettag():
    # メッセージ区分 : 0x22、データ長 : 2byte、データ : 0x01
    sendData = bytearray([0xBB, 0x80, 0x22, 0x00, 0x02, 0x01, 0xA1, 0x7E])
    count = 0
    ser.write(sendData)
    while (True):
        recvData = ser.read(5)
        len1 = recvData[3] * 256 + recvData[4] + 1
        rdata = ser.read(len1)
        if (recvData[2] == 0xFF):
            if (rdata[0] == 0x15):
                print('タグなし')
            else:
                print('エラー : ' + hex(rdata[0]))
            break
        elif (recvData[2] == 0x27):
            print('タグ数 : ' + str(count))
            break
        else:
            len2 = len(rdata)
            pc = rdata[0 : 2]
            epc = rdata[2 : len2 - 4]
            print('PC : ' + pc.hex() + ', EPC : ' + epc.hex())
            count += 1

# メモリを読み取る関数
# この関数では、アクセスパスワードは全て 0x00、タグの ID は指定していません
def func_memory_read():
    # メッセージ区分 : 0x39、データ長 : 12byte、メモリバンク以降はこの後に変更
    sendData = bytearray([0xBB, 0x80, 0x39, 0x00, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                          0x01, 0x00, 0x02, 0x00, 0x0C, 0xDA, 0x7E])
    sbank = input('バンク (0:Reserve, 1:EPC, 2:TID, 3:User) >> ')
    ibank = int(sbank)
    if (ibank < 0) or (3 < ibank):
        print('エラー ')
        return
    sendData[11] = ibank
    sstart = input('先頭アドレス(ワード) >> ')
    istart = int(sstart)
```

送信出力値を設定する関数

```
def func_setpower():
    # メッセージ区分: 0xB2、データ長: 3byte、送信出力値以降はこの後に変更
    sendData = bytearray([0xBB, 0x80, 0xB2, 0x00, 0x03, 0x03, 0xE8, 0xDA, 0x7E])
    spower = input(' power (dBm) >> ') # 送信出力を入力
    ipower = int(spower) # str を int に変換
    ipower *= 100 # 100 倍
    sendData[5] = ipower // 256 # 上位バイトを変更
    sendData[6] = ipower % 256 # 下位バイトを変更
    sendData[7] = func_check_code(sendData, 1, 7) # チェックコードを計算して変更
    ser.write(sendData) # コマンドを送信
    recvdData = ser.read(5) # 最初の 5 バイトを受信
    len1 = recvdData[3] * 256 + recvdData[4] + 1 # 残りのデータ数を取得
    rdata = ser.read(len1) # 残りのデータを受信
    if (recvdData[2] == 0xB2) and (rdata[0] == 0x00): # 成功
        print(' 成功')
    else: # エラー
        print(' エラー : ' + hex(rdata[0]))
```

送信出力値を取得する関数

```
def func_getpower():
    # メッセージ区分: 0x53、データ長: 2byte、データ: 0x00
    sendData = bytearray([0xBB, 0x80, 0x53, 0x00, 0x02, 0x00, 0xD1, 0x7E])
    ser.write(sendData) # コマンドを送信
    recvdData = ser.read(5) # 最初の 5 バイトを受信
    len1 = recvdData[3] * 256 + recvdData[4] + 1 # 残りのデータ数を取得
    rdata = ser.read(len1) # 残りのデータを受信
    if (recvdData[2] == 0xFF): # エラー
        print(' エラー : ' + hex(rdata[0]))
    else: # 成功
        power = (rdata[0] * 256 + rdata[1]) // 100 # 送信出力値を取得して表示
        print(' power : ' + str(power) + ' (dBm)')
```

----- main -----

```
ser = serial.Serial("/dev/ttyUSB0", 115200) # RFID リーダ・ライタが接続されているポートを開く
```

```
while (True):
    # メニューを表示
    print(' ¥¥¥¥-----')
    print(' t:タグ取得、r:リード、w:ライト、s:送信出力設定、g:送信出力取得、q:終了')
    menu = input(' >>> ') # メニューを入力

    if (menu == 'q'): # q で終了
        break
    elif (menu == 't'): # t でタグ ID を取得
        func_gettag()
    elif (menu == 's'): # s で送信出力値を設定
        func_setpower()
    elif (menu == 'g'): # g で送信出力値を取得
        func_getpower()
    elif (menu == 'r'): # r でメモリをリード
        func_memory_read()
    elif (menu == 'w'): # w でメモリにライト
        func_memory_write()
    else: # それ以外のときはやり直し
        continue
ser.close() # ポートを閉じる
```

◇CB ファミリ用

```
import serial
# ----- 関数 -----
# BCC を計算する関数
#  戻り値 : 先頭からデータ部の最後までを加算した値の下位 1 バイト
#  array  : データが入っている配列
#  end    : 加算する数
def func_bcc(array, end):
    code = 0
    for i in range(0, end):
        code += array[i]
    return code & 0x000000FF

# タグ ID を取得する関数
def func_gettag():
    # メッセージ区分 : 0x20、データ長 : 0byte、BCC はこの後に追加
    sendData = bytearray([0x53, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, ¥
                          0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00])

    sendData.append(func_bcc(sendData, 16))    # BCC を計算して追加
    ser.write(sendData)                       # コマンドを送信
    recvData = ser.read(16)                   # 最初の 16 バイトを受信
    len1 = recvData[7] * 256 + recvData[6] + 1 # 残りのデータ数を取得
    rdata = ser.read(len1)                     # 残りのデータを受信
    if (recvData[1] != 0x00): # エラー
        if (rdata[0] == 0x20): # タグを検出しなかった
            print('タグなし')
        else: # その他のエラー
            print('エラー : ' + hex(rdata[0]))
    else: # 正常
        ptr = 0 # 受信データ内の処理する位置
        count = rdata[ptr] # 検出したタグ数を取得して表示
        print('タグ数 : ' + str(count))
        ptr += 1
        for num in range(0, count): # タグ数回繰り返す
            antNum = rdata[ptr] # アンテナ番号を取得
            ptr += 1
            idLen = rdata[ptr] # タグ ID の長さを取得
            ptr += 1
            # タグ情報は PC(2byte)+EPC(可変)+CRC(2byte)
            pc = rdata[ptr : ptr + 2] # PC を取得
            epc = rdata[ptr + 2 : ptr + idLen - 2] # EPC を取得
            ptr += idLen
            print('Ant' + str(antNum) + ', ' + 'PC : ' + pc.hex() + ', EPC : ' + epc.hex())
```

```

# メモリをリードする関数
# この関数では、アクセスパスワードは全て 0x00、タグの ID は指定していません
def func_memory_read():
    # メッセージ区分: 0x22、データ長: 11byte、メモリバンク以降はこの後に追加
    sendData = bytearray([0x53, 0x00, 0x00, 0x00, 0x22, 0x00, 0x0B, 0x00, ¥
                          0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, ¥
                          0x00, 0x00, 0x00, 0x00, 0x00, 0x00])
    sbank = input(' バンク (0:Reserve, 1:EPC, 2:TID, 3:User) >> ') # リードするメモリバンクを入力
    ibank = int(sbank) # str を int に変換
    if (ibank < 0) or (3 < ibank): # 範囲確認
        print(' エラー ')
        return
    sendData.append(ibank) # バンクを追加
    sstart = input(' 先頭アドレス (ワード) >> ') # 先頭アドレスをワード単位で入力
    istart = int(sstart) # str を int に変換
    sendData.append(istart // 256) # 上位バイトを追加
    sendData.append(istart % 256) # 下位バイトを追加
    slen = input(' データ長さ (バイト) >> ') # データ長さをバイト単位で入力
    ilen = int(slen) # str を int に変換
    sendData.append(ilen // 256) # 上位バイトを追加
    sendData.append(ilen % 256) # 下位バイトを追加
    sendData.append(func_bcc(sendData, 27)) # BCC を計算して追加
    ser.write(sendData) # コマンドを送信
    recvdData = ser.read(16) # 最初の 16 バイトを受信
    len1 = recvdData[7] * 256 + recvdData[6] + 1 # 残りのデータ数を取得
    rdata = ser.read(len1) # 残りのデータを受信
    if (recvdData[1] != 0x00): # エラー
        print(' エラー : ' + hex(rdata[0]))
    else: # 成功
        len2 = rdata[3] * 256 + rdata[2] # リードしたデータ長さを取得
        data = rdata[4 : len2 + 4] # データを保存して表示
        print(' data : ' + data.hex())

# メモリにライトする関数
# この関数では、アクセスパスワードは全て 0x00、タグの ID は指定していません
def func_memory_write():
    # メッセージ区分: 0x23、データ長: 後で変更、メモリバンク以降はこの後に追加
    sendData = bytearray([0x53, 0x00, 0x00, 0x00, 0x23, 0x00, 0x0B, 0x00, ¥
                          0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, ¥
                          0x00, 0x00, 0x00, 0x00, 0x00, 0x00])
    sbank = input(' バンク (0:Reserve, 1:EPC, 2:TID, 3:User) >> ') # ライトするメモリバンクを入力
    ibank = int(sbank) # str を int に変換
    if (ibank < 0) or (3 < ibank): # 範囲確認
        print(' エラー ')
        return
    sendData.append(ibank) # バンクを追加
    sstart = input(' 先頭アドレス (ワード) >> ') # 先頭アドレスをワード単位で入力
    istart = int(sstart) # str を int に変換
    sendData.append(istart // 256) # 上位バイトを追加
    sendData.append(istart % 256) # 下位バイトを追加
    slen = input(' データ長さ (バイト) >> ') # データ長さをバイト単位で入力
    ilen = int(slen) # str を int に変換
    sendData.append(ilen // 256) # 上位バイトを追加
    sendData.append(ilen % 256) # 下位バイトを追加
    sdata = input(' データ (hex) >> ') # ライトするデータを 16 進数で入力
    datalen = len(sdata) # 入力データ数を取得

```

```

if (datalen != ilen * 2):
    print('データ数が合っていない')
    return
for i in range(0, ilen):
    sendData.append(int(sdata[i * 2 : i * 2 + 2], 16))
ptr = 27 + ilen
sendData[6] = 11 + ilen
sendData.append(func_bcc(sendData, ptr))
ser.write(sendData)
recvData = ser.read(16)
len1 = recvData[7] * 256 + recvData[6] + 1
rdata = ser.read(len1)
if (recvData[1] != 0x00):
    print('エラー : ' + hex(rdata[0]))
else:
    print('成功')

# 送信出力値を設定する関数
def func_setpower():
    # メッセージ区分: 0x16、データ長: 2byte、送信出力値以降はこの後に追加
    sendData = bytearray([0x53, 0x00, 0x00, 0x00, 0x16, 0x00, 0x02, 0x00, 0x00,
                          0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00])

    spower = input('power (dBm) >> ')
    ipower = int(spower)
    ipower *= 100
    sendData.append(ipower // 256)
    sendData.append(ipower % 256)
    sendData.append(func_bcc(sendData, 18))
    ser.write(sendData)
    recvData = ser.read(16)
    len1 = recvData[7] * 256 + recvData[6] + 1
    rdata = ser.read(len1)
    if (recvData[1] != 0x00):
        print('エラー : ' + hex(rdata[0]))
    else:
        print('成功')

# 送信出力値を取得する関数
def func_getpower():
    # メッセージ区分: 0x1B、データ長: 0byte、BCC はこの後に追加
    sendData = bytearray([0x53, 0x00, 0x00, 0x00, 0x1B, 0x00, 0x00, 0x00, 0x00,
                          0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00])

    sendData.append(func_bcc(sendData, 16))
    ser.write(sendData)
    recvData = ser.read(16)
    len1 = recvData[7] * 256 + recvData[6] + 1
    rdata = ser.read(len1)
    if (recvData[1] != 0x00):
        print('エラー : ' + hex(rdata[0]))
    else:
        power = (rdata[0]*256 + rdata[1]) // 100
        print('power : ' + str(power) + ' (dBm)')

```

```
# ---- main -----
ser = serial.Serial("/dev/ttyUSB0", 115200) # RFID リーダ・ライタが接続されているポートを開く

while (True):
    # メニューを表示
    print('¥n¥n-----')
    print('t:タグ取得、r:リード、w:ライト、s:送信出力設定、g:送信出力取得、q:終了')
    menu = input('>>> ') # メニューを入力

    if (menu == 'q'): # q で終了
        break
    elif (menu == 't'): # t でタグ ID を取得
        func_gettag()
    elif (menu == 's'): # s で送信出力値を設定
        func_setpower()
    elif (menu == 'g'): # g で送信出力値を取得
        func_getpower()
    elif (menu == 'r'): # r でメモリをリード
        func_memory_read()
    elif (menu == 'w'): # w でメモリにライト
        func_memory_write()
    else: # それ以外のときはやり直し
        continue
ser.close() # ポートを閉じる
```

以上