E220-900T22S(JP) モジュール

for Raspberry Pi Pico
利用ガイド

Rev.1

CLEALINK TECHNOLOGY Co., Ltd.

2023-3-10

目次

1 概要

本書は、E220-900T22S(JP) モジュールを Raspberry Pi Pico を接続して、Raspberry Pi Pico との UART での入出力によって、モジュールのパラメータ設定および LoRa での送受信を行うためのサンプルコードについて説明したチュートリアルです。なお、E220-900T22S(JP) モジュールの詳細はデータシートを参照ください。

2 E220-900T22S(JP) モジュールと Raspberry Pi Pico の接続

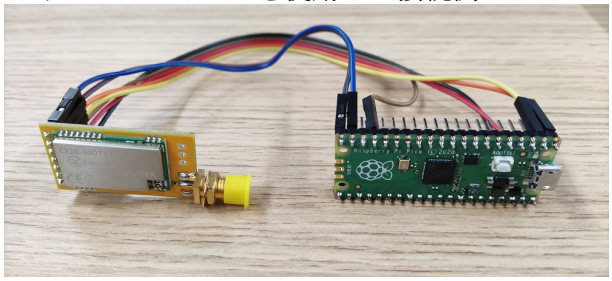
E220-900T22S(JP)-EV1 と Raspberry Pi Pico と接続は下記の配線が必要となります。

- モジュールの電源として VCC ピンと GND ピンへの接続
- UART として TXD ピンと RXD ピンへの接続
- モジュールのモード設定のため M0 ピンと M1 ピンへの接続
- モジュールからの通知信号線として AUX ピンへの接続

本チュートリアルを進めるにあたり、次表の通り接続してください。

E220-900T22S(JP)-EV1	Raspberry Pi Pico
GND	GND
VCC	5V
AUX	GPIO13
TXD	UART0RX
RXD	UART0TX
M1	GPIO15
M0	GPIO14

ジャンパーワイヤーを使用した接続例



3 サンプルコード実行環境の構築

3.1 Thonny Python IDE のインストール

下記サイトで、Raspberry Pi Pico(以下「Pico」という) も開発できる開発環境　thonny をインストールしてください。

https://thonny.org/

また、Pico 用のファームウェア（MicroPython UF2 file）を下記サイトからダウンロードしてください。rp2-pico-2023***vxxxx******.uf2 というファイル名です。

https://www.raspberrypi.com/documentation/microcontrollers/micropython.html

Thonny のインストール完了後、ＰＣと Pico を USB ケーブルで接続します。接続時に Pico の基板上の白いスイッチ「BOOTSEL」を押しながら USB ケーブルを差し込むと、ＰＣは、Pico をマスストレージ（大容量記憶装置）として認識しますので、

上記でダウンロードした (.uf2) ファイルをマスストレージへ書き込みます（drag&drop）。

Thonny を起動します。ＰＣのＣＯＭポートが正常に動いていれば、Thonny の画面右下に、MicroPython（Raspberry Pi Pico）・COM ｘと表示されますので、そこをクリックしてインタープリタ設定を必要であれば確認、変更してください。詳細は Thonny のサイトを参照してください。

これで、実行環境の構築は終了です。

4 E220-900T22S(JP) モジュールの動作説明

E220-900T22S(JP) モジュールには M0 ピンと M1 ピンがあり、M0 と M1 の High/Low 組み合わせで 4 つの動作モードを決定します。

M0	M1	mode
Low	Low	ノーマルモード (mode 0)
High	Low	WOR 送信モード (mode 1)
Low	High	WOR 受信モード (mode 2)
High	High	コンフィグモード (mode 3)

4.1 ノーマルモード (mode 0)

M0=Low, M1=Low の状態です。
UART から入力されたデータを LoRa で送信します。
LoRa で受信したデータを UART から出力します。

4.2 WOR 送信モード (mode 1)

M0=High, M1=Low の状態です。
UART から入力されたデータを LoRa で送信します。
LoRa で受信したデータを UART から出力します。

LoRa 送信時は、送信前に WOR 受信モードのデバイスをウェイクアップさせるためのプリアンブル (preamble) が自動的に追加されます。
また、LoRa 受信が可能でノーマルモードと同様の動作になります。

※ WOR (Wake on Radio): ワイヤレス電波により待機状態のモジュールを活性化する機能

4.3 WOR 受信モード (mode 2)

M0=Low, M1=High の状態です。
LoRa 受信したデータを UART から出力します。
LoRa 送信は出来ません。

LoRa 受信は、WOR 送信モードで送信されたデータのみ受信可能です。

このモードでは、低消費電力での受信待ち受けを行えます。
データ受信時、AUX ピンから通知信号が出力されるため、AUX ピンを割り込み端子として外部 MCU に接続することで、普段は外部 MCU をスリープさせておき、データ受信のときだけウェイクアップさせる使い方が出来ます。
そのため、外部 MCU と LoRa モジュールをあわせたトータルの消費電力も低く抑えることができるようになります。

4.4 コンフィグモード (mode 3)

M0=High, M1=High の状態です。
UART からコマンドを入力することでモジュールのパラメータを設定することができます。パラメータ設定時は baudrate=9600, parity=8N1 で行う必要があります。
コマンドの詳細はデータシートの *7.1* **コマンドフォーマット**を参照してください。

5 サンプルコード利用方法

ダウンロードした利用ガイドの zip を解凍すると、次のファイルが生成されます。

```
main.py         --- 初期化などの手順、送受信の方法を記述したサンプル
LoRa.py         --- E220-900T22S(JP)-EV1とPicoの通信オープンと設定を行う
ConfigTreat.py --- コンフィグ設定と参照を行う
setting.py      --- コンフィグ設定を行うスクリプトを記述したファイル
send.py         --- データ送信を行う
receive.py      --- データ受信を行う
```

thonny を起動して、.py ファイルをすべて読み込んでください。読み込み方は thonny のサイトを参照してください。（ＰＣから読み込んだ.py ファイルは、Pico で動作させるためには Pico へ保存することが必要です。）

設定ファイル (setting.py) にモジュールのパラメータ設定値を記述します。それぞれの項目についての詳細はデータシートの *7.2* **レジスタ詳細**を参照してください。

setting.py の設定例

```
own_address = '0'
baud_rate = '9600'
bw = '125'
sf = '9'
subpacket_size = '32'
rssi_ambient_noise_flag = '0'
transmitting_power = '13'
own_channel = '0'
rssi_byte_flag = '1'                # RSSIバイト有効
transmission_method_type = '1'      # 1 is transparent, 2 is Fixed
wor_cycle = '3000'
encryption_key = '0'
```

設定内容は次の通りとなっています。

- モジュールアドレスを 0
- UART の baud rate を 9600bps
- BW(帯域幅) を 125kHz
- SF(拡散率) を 9
- パケット長を 32byte
- RSSI 環境ノイズ機能を無効化
- 送信出力電力を 13dBm
- 周波数チャネルを 0
- RSSI バイトを有効化
- 送信方法を固定送信モード
- WOR サイクルを 3000ms

- 暗号化キーを 0

送受信に必要なその他の設定（setting.py）

```
payload_length = 32             # send.pyで使用する送信ペーロード長
ascii_text = True               # 送信データに”ascii”を設定
fixed_mode = False              # 変更するときは
                                # transmission_method_typeも同時に
                                # False = transparet mode : method_type 1
                                # True  = fixed mode : method type 2
target_address = 0xffff         # BroadCast の場合  0xffff
target_channel = 0x00
Argrssi = True                  #RSSIバイト有効
```

5.1 サンプルコードを実行

5.2 main() の実行

Thonny のエディット画面を main にして、実行ボタン ▶をクリックして、main.py を実行するしてください。以下の順に処理を行います。

aux チェックのための aux 設定 E220・・・・の Uart 開始コンフィグ設定値の E220 への設定 設定値の確認表示

ConfigTreat.py の configTreat.ConfigApply() の実行で、コンフィグの設定を行います。

```
# Command Request
b'\xc0\x00\x08\x00\x00p\xc1\x00\x85\x00\x00'
# Command Response
b'\xc1\x00\x08\x00\x00p\xc1\x00\x85\x00\x00\x00'
```

次にコンフィグ設定の確認、ConfigTreat.ConfigAllDisp() で現在の設定一覧を表示

```
# Command Request
b'\xc1\x00\x08'
# Command Response
b'\xc1\x00\x08\x00\x00p\xc1\x00\x85\x00\x00\x00'

現在の設定一覧
Address             : 0
UART                : 9600bps
Air Data Rate       : 1,758bps, SF: 9, BW:125kHz
Sub Packet Size     : 32bytes
RSSI Ambient noise  : Disable
Transmitting Power  : {transmit_power_setting[TRANSMIT_POWER]}
CH                  : 0
Actual frequency    : 920.6MHz
RSSI Byte           : Enable
```

```
Transmission Method : Transparent transmission mode
WOR Cycle           : 3000ms
```

5.3 send.send() で送信

```
payload: b'\x00\x01\x02\x03\x04\x05\x06\x07\x08\t\n\x0b\x0c\r\x0e\x0f
\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f'
SENT
```

5.4 receive.receive() で受信待ち

受信データ例

```
receive waiting...
RECEIVED
b'\x02\x00\x03\x00\x00\x00\x00\x00V\x00\x00\x00\x00V\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00f}\x00\x00\x92'
RSSI: -110 dBm
```