「IoT-Pi」 インターフェース仕様書

ご注意：
通信料金及び、本製品をご利用される場合の障害につきましては、弊社の責任範囲外とさせていただきますので、ご了承願います。

CONFIDENTIAL

Version 1.0.6
2019/11/1

株式会社ＷＤＳ

改訂履歴

作成年月日	変更箇所、理由など		
2019/03/01	初版		
2019/09/01	3.1 項修正 インストーラファイル名変更（python3 のバージョンによる）		
2019/09/05	4.1.1 項修正 APN の設定は添付 SIM 以外のものを使用する場合に訂正		
2019/09/09	4.1.1 項修正 サーバの URL 修正、ローマ字読み追記		
	install_iot-pi-xxx.bin にファイル名を変更		
2019/10/24	2.13 モジュール電源再投入　を追加		
2019/10./31	2.13 モジュール電源再投入でパワーセーブモード検出からの具体例を追加		
2019/11/1	5. eDRX と PSM の説明を追加		

目次

1. 概要

本書は、RaspberryPI で LTE-M 通信をするためのパッケージ「IoT-Pi」内のインタフェースボード（以後ターゲットボードと称します）のインターフェース（以後、本インターフェースと称します）について述べています。
本インターフェースは Python3 上で動作し以下の機能を持ちます。

番号	機能	備考
1	APN 設定	モジュールの初期化を行います
2	接続	FQDN 又は IP アドレスとポート番号を指定して TCP 接続を開始します
3	切断	TCP 接続を切断します
4	送信	データを送信します
5	受信	データの受信時指定された関数をコールします
6	電波状態取得	アンテナ本数、サービスエリア情報他
7	位置情報取得開始	位置情報取得状態を開始させます
8	位置情報取得	位置情報取得時指定された関数をコールします
9	位置情報取得終了	位置情報取得状態を終了させます
10	リセット	モジュールをリセットします
11	シャットダウン	モジュールをシャットダウンします
12	初期化	モジュールを初期化します
13	省電力設定	省電力のパラメータを設定します
14	省電力設定取得	省電力設定のパラメータを取得します
15	EDRX 設定	EDRX のパラメータを設定します
16	EDRX 設定取得	EDRX のパラメータを取得します
17	電源再投入	モジュールの電源の再投入を行います

2. インターフェース

モジュールファイル： LTEm.so

2.1. APN 設定

LTEm.setupAPN(pdpproto=proto,apn=apn)

LTE モジュールの初期化(APN 設定)を行います。工場出荷状態の KYM001 モジュールを使用できるようにします。
通常、1 度実行したら再度実行する必要はありません。

	KEY	備考
1	pdpproto	'IP','IPV6','IPV4IPV6'のいずれかの文字列が設定できます
2	apn	アクセスポイントの URL の文字列

2.2. 接続

LTEm.connect(addr=None,port=80,rx=None,timeout=None,disc=None)

addr,port で示されるサーバへ TCP/IP 接続します。
addr には FQDN あるいはドットで区切った IPv4アドレス(文字列)を指定します。
rx には受信されたときにコールされる呼び出し可能オブジェクトを指定します。
timeout で送信完了までの待ち時間を指定します。単位は秒で型は Float です。
None を指定するとタイムアウト値は無制限になります。
disc は切断されたときにコールされる呼び出し可能オブジェクトを指定します。

rx=receive と指定すると、受信があったとき、受信データを rxbuffer に入れて
receive(rxbuffer)
として呼ばれます。
txbuffer の型 は bytes です。

既に TCP/IP 接続している場合はエラーとなり例外(AlreadyConnected)がスローされます。
また、位置情報取得中の場合もエラーとなり例外(Busy)がスローされます。

接続に関しては以下のエラーコードが返されます。

DisconnectedNormal	サーバに接続できない
PDNError	PDN 接続エラー
CannotResolve	名前解決できない
DiscFromServer	サーバより切断された

2.3. 送信

LTEm.send(txbuf,timeout=None)

txbuf の内容を送信します。txbuf の型は **bytes** です。

timeout で送信完了までの待ち時間を指定します。単位は秒で型は **Float** です。

None の指定で無制限に待ちます。

送信したバイト数を返します。

未接続の場合はエラーとなり例外(NotConnected)がスローされます。

2.4. 受信

receive(rxbuffer)

LTEm.connect の **rx** パラメータに指定した関数が受信時にコールされます。

rxbuffer が受信したデータになります。**rxbuffer** の型は **bytes** です。

2.5. 切断コールバック

disconnect(reason)

LTEm.connect の **disc** パラメータに指定した関数が切断時にコールされます。

reason には以下の値がセットされます。

DisconnectedNormal	正常切断
DiscFromServer	サーバより切断された

2.6. 切断

LTEm.disconnect ()

確立しているサーバとの TCP/IP 接続を切断します。

2.7. 電波状態取得

LTEm.status()

電波状態を namedtuple で返します。

(ant=<アンテナ本数>,area=<圏外、圏内>,attach=<0,1>)

項目	詳細
ant	0: 圏外 1: アンテナ0本 2: アンテナ1本 3: アンテナ2本 4: アンテナ3本 5: アンテナ4本
area	0: 圏外 1: 圏内
attach	0: detach 状態 1: attach 状態

※ TCP/IP 接続中に読みだすとエラーとなり例外(Busy)がスローされます。

2.8. 位置情報取得開始

LTEm.startloc(notify=None)

測位を開始します。位置情報の取得の都度、notify で指定した関数がコールされます。
notify には受信したときにコールされる呼び出し可能オブジェクトを指定します。

※ TCP/IP 接続とは排他的になります。TCP/IP 接続中にコールするとエラーとなり例外(Busy)がスローされます。

2.9. 位置情報取得

locreceive(locdata)

LTEm.startloc の notify パラメータに指定した関数が位置情報受信時にコールされます。
locdata が受信したデータになります。rxbuffer の型は namedtuple で以下の様になります。

項目	型	詳細
altitude	float	緯度
longitude	float	経度
time	float	時刻 (unixtime)

※実際の位置情報を取得するには、本製品が GPS の電波を受信できる屋外や窓際でご利用ください。
また、実際の位置情報が取得できるまでには本製品に電源が投入されてから 5 分から 10 分以上時間がかかることがございます。それまでは、緯度経度ともに0が表示されます。

2.10. 位置情報取得終了

LTEm.stoploc()

測位を終了します。

2.11. リセット

LTEm.reset()

モジュールをリセットします。

2.12. シャットダウン

LTEm.shutdown()

モジュールをシャットダウンします。

2.13. モジュール電源再投入

LTEm.module_power()

通信モジュールの電源の再投入を行います。
パワーセーブモードから復旧タイマーが切れる前での復旧やシャットダウンからの復旧ができます。

パワーセーブモードあるいはシャットダウン状態になっているかは 「2.7 電波状態の取得」を発行してタイムアウトエラーになることで確認できます。
パワーセーブモードあるいはシャットダウンをご利用頂いている場合は以下の様なコードを実行してから通信を行ってください。

```
while True:
        error = 0
        # 電場状態の取得
        try:
                status = LTEm.status()
        except LTEm.LTEmErr as errcode:
                error = int(str(errcode))
        if error == 0:
                # 正常に実行できたので次へ
                break
        elif error != 10:
                # タイムアウト以外のエラーなので実行中止
                print('LTEm error %d' % error)
                exit(1)
        # 電源再投入
        LTEm.module_power()
        # 起動するまでの時間待ち
        time.sleep(10)
```

(5.項 参照)

2.14. 初期化

LTEm.init()

モジュールの初期化を行います。

2.15. 省電力設定の読み出し

LTEm.get_psm()

タプル (Periodic_Update_Time,Active_Time)

が返されます。

時間の単位は秒になります。

(5.項 参照)

2.16. 省電力の設定

LTEm.set_psm(Periodic_Update_Time,Active_Time)

Periodic_Update_Time は 2 秒から 9920 時間までの値が設定可能です。
ただしその値そのものではなく近い値が設定されます。

Active_Time は2秒から186分までの値が設定可能です。
ただしその値そのものではなく近い値が設定されます。

(5.項 参照)

2.17. EDRX の読み出し

LTEm.get_edrx()

タプル(Cycle_Timer,Paging_Time_Window)が返されます。

それぞれの単位は秒になります。

(5.項 参照)

2.18. EDRX の設定

LTEm.set_edrx(Cycle_Timer,Paging_Time_Window)

Cycle_Timer は 5.12 秒から 2621.44 秒までの値が設定できます。

Paging_Time_Window は 1.28 秒から 20.48 秒までの値が設定できます。

(5.項 参照)

2.19. エラーについて

エラーが起こった場合、例外クラス LTEmErr のメンバ error にエラーコードがセットしてスローします。
セットされるエラーコードは以下の様になります。

```
Generic              = 1
InvalidArguments     = 2
AlreadyConnected     = 3
NotConnected         = 4
DisconnectedNormal   = 5
PDNError             = 6
CannotResolve        = 7
DiscFromServer       = 8
Busy                 = 9
Timeout              = 10
ModuleNotReady       = 11
```

※ エラーコードは追加変更される可能性があります。

3. インストールについて

3.1. インストーラの実行とファイル構成

RaspverryPi の WEB ブラウザーで「http://www.wd-s.com/iot-pi_download/」からインストーラファイル「**install_iot-pi-xxx.bin**」 (xxx: stretch / bustor)をダウンロードします。

RaspberryPi にディレクトリ(フォルダー)を１つファイル名は任意でよいので作成し、その中に「**install_iot-pi-xxx.bin**」をコピーします。
※注意:-pxxx は WEB サイトからダウンロードしたファイル名にしてください。

仮に「iot-pi」というディレクトリーを作成して「**install_iot-pi-pxxx.bin**」をコピーした場合、以下のようにターミナルからコマンドを実行します。

```
pi@raspberrypi:~/iot-pi$ chmod +x install_iot-pi-pxxx.bin
pi@raspberrypi:~/iot-pi$ ./install_iot-pi-pxxx.bin
```

以下に実行例を示します。

```
pi@raspberrypi:~/iot-pi$ chmod +x install_iot-pi-p373.bin
pi@raspberrypi:~/iot-pi$ ./install_iot-pi-p373.bin
87+1 レコード入力
87+1 レコード出力
89893 bytes (90 kB, 88 KiB) copied, 0.00301826 s, 29.8 MB/s
Created symlink /etc/systemd/system/multi-user.target.wants/iot-pi-start.service → /etc/systemd/system/iot-pi-start.service.
Created symlink /etc/systemd/system/reboot.target.wants/iot-pi-poweroff.service → /etc/systemd/system/iot-pi-poweroff.service.
Created symlink /etc/systemd/system/halt.target.wants/iot-pi-poweroff.service → /etc/systemd/system/iot-pi-poweroff.service.
Created symlink /etc/systemd/system/poweroff.target.wants/iot-pi-poweroff.service → /etc/systemd/system/iot-pi-poweroff.service.
パッケージリストを読み込んでいます... 完了
依存関係ツリーを作成しています
状態情報を読み取っています... 完了
fonts-ipafont はすでに最新バージョン (00303-16) です。
python3-cairo はすでに最新バージョン (1.10.0+dfsg-5) です。
```

```
python3-click はすでに最新バージョン (6.6-1) です。
python3-gi はすでに最新バージョン (3.22.0-2) です。
python3-gi-cairo はすでに最新バージョン (3.22.0-2) です。
gir1.2-gtk-3.0 はすでに最新バージョン (3.22.11-1+rpi3) です。
アップグレード: 0 個、新規インストール: 0 個、削除: 0 個、保留: 23 個。
pi@raspberrypi:~/iot-pi$
```

以下の4つのファイルができます。

	ファイル名	備考
1	LTEm.glade	サンプルプログラムの画面デザインファイル
2	LTEm_sample.py	サンプルプログラム
3	wds.png	サンプルプログラムで表示するロゴのイメージファイル
4	alarm.py	RTC アラーム設定サンプルプログラム

3.2. RaspberryPI3 の設定

「メニュー」→「ラズパイの設定」→「インターフェース」で I2C を有効にするか
以下の手順で RaspberryPI3 の I2C インターフェースの有効化の設定を行います。

① ターミナルを開き
　　$ pi@raspberrypi:~$ sudo raspi-config
と入力します。
② 「**5 Interfacing Options**」を選択します。
③ 「**P5 I2C**」を選択します。
④ 「**Would you like the ARM I2C interface to be enabled?**」と表示されるので<**はい**>を選択します。
⑤ <**了解**>を押します。
⑥ <**finish**>を選択して raspi-config を終了します。
⑦ RaspberryPI3 を再起動します。

4. サンプルプログラム

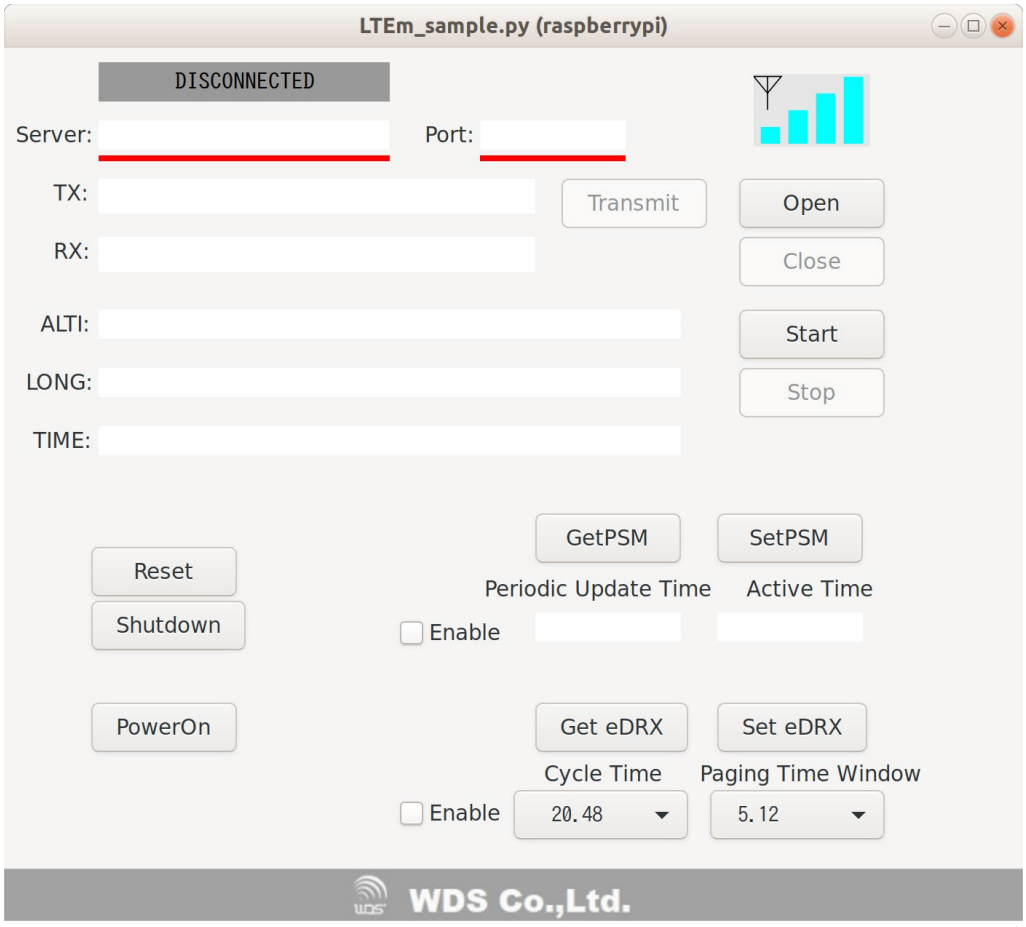
4.1. サンプルプログラムの実行

インストールディレクトリに移動し LTEm_sample.py を実行します。

iot-pi をディレクトリ~/ INSTALL にインストールした場合

```
$ cd ~/ INSTALL
$ ./LTEm_sample.py
```

数秒で以下の画面が表示されます。



	ボタン	機能
1	Open	「Server」「Port」に接続するサーバのアドレスとポートを指定してこのボタンを押すとサーバに接続されます。
2	Transmit	「TX」に送信する文字列を入力しこのボタンを押すことでその文字列がサーバに送信されます。
3	Close	サーバとの接続を切断します。
4	Start	測位を開始します。位置情報と時刻は 「ALTI」：緯度 「LONG」：経度 「TIME」：時刻 に表示されます。 ※実際の位置情報を取得するには、本製品が GPS の電波を受信できる屋外や窓際でご利用ください。 　また、実際の位置情報が取得できるまでには本製品に電源が投入されてから 5 分から 10 分以上時間が 　かかることがございます。それまでは、緯度経度ともに0が表示されます。
5	Reset	KYW01 をリセットします。
6	Shutdown	KYW01 をシャットダウンします。
7	Init	KYW01 を初期化します。
8	GetPSM	PSM 設定を読みだします。結果は「Enable」「Periodic Update Time」「Active Time」に設定されます。
9	SetPSM	「Enable」「Periodic Update Time」「Active Time」にある値で PSM を設定します。
10	Get eDRX	eDRX 設定を読みだします。結果は「Enable」「Cycle Time」「PagingTimeWindow」に設定されます。
11	Set eDRX	「Enable」「Periodic Update Time」「Active Time」にある値で eDRX を設定します。
12	PowerOn	通信モジュールの電源の再投入を行います。

4.1.1. 動作確認

① IoT-Pi に付属されていない SIM(KDDI 社から直接購入された LPWA 専用 SIM)をご使用になられる場合は、お客様専用サイト<http://www.wd-s.com/iot-pi_download/>から「APN 設定アプリケーション」をダウンロードしてモジュールの APN 設定を行ってください。設定方法については同じサイトの「APN 設定用アプリケーション説明書」をご覧ください。

② LTEm_sample.py を起動します。iot-pi をディレクトリ~/ INSTALL にインストールした場合

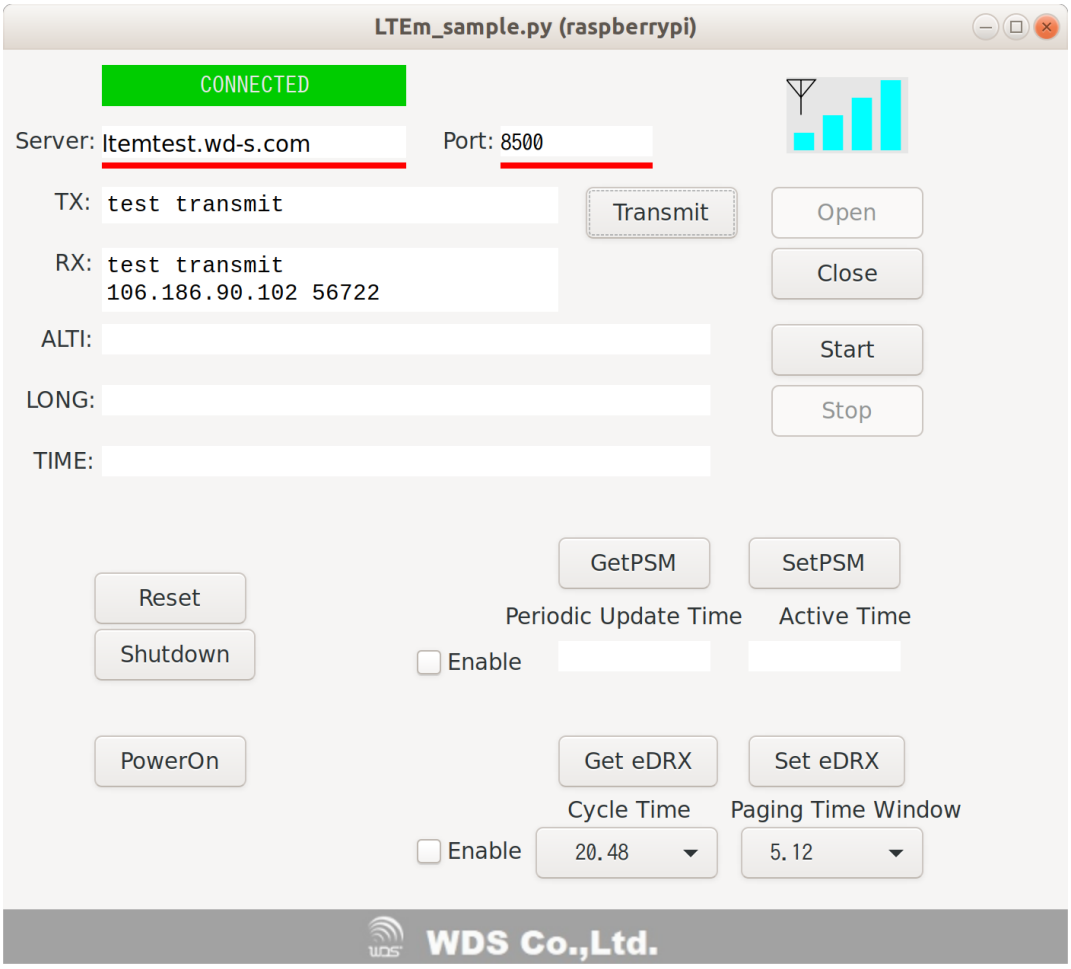
```
$ cd ~/ INSTALL
$ ./LTEm_sample.py
```

と入力します。

③ Server に「**ltemtest.wd-s.com(エル、テー、イー、エム、テー、イー、エス、テー、ドット、ダブリュー、デー、ハイフン、エス、ドット、シー、オー、エム)**」 Port に「**8500**」を指定して Open を押します。

④ TX に任意の文字列を入れて Tramsmit を押します。サーバ「**ltemtest.wd-s.com**」は入力した文字列とモジュールの IP アドレス、ポートを2行でエコーバックしますので

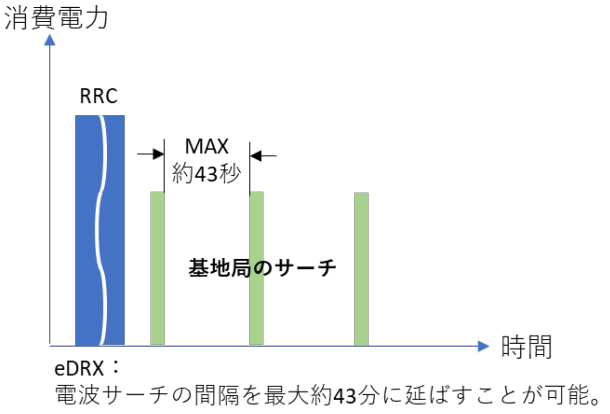
⑤ RX に②で入力した文字列と IP アドレス、ポートが表示されれば OK です。



5. PSM と eDRX について

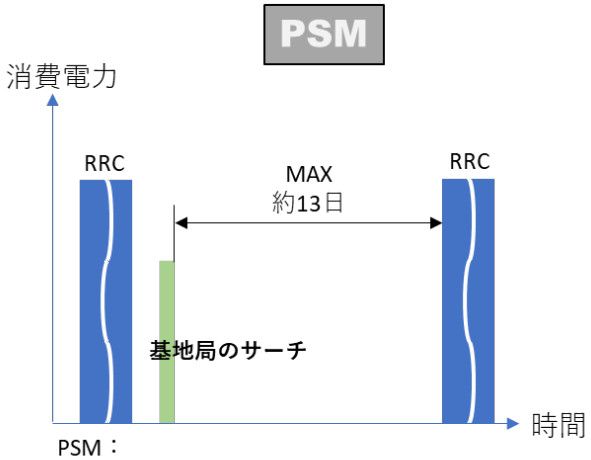
5.1. 待ち受け

■従来の LTE 端末

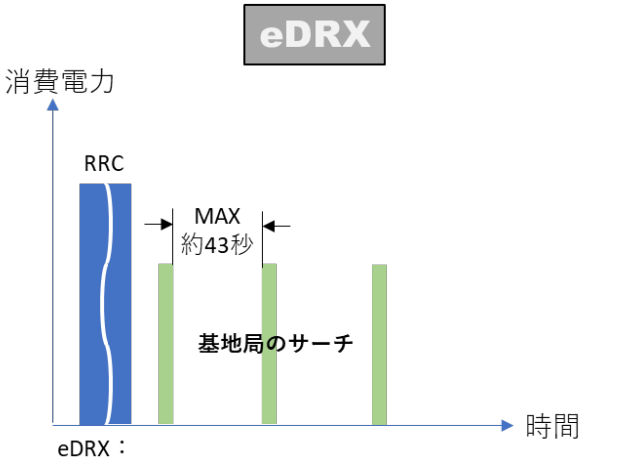


eDRX：
電波サーチの間隔を最大約43分に延ばすことが可能。

■LTE-M 端末



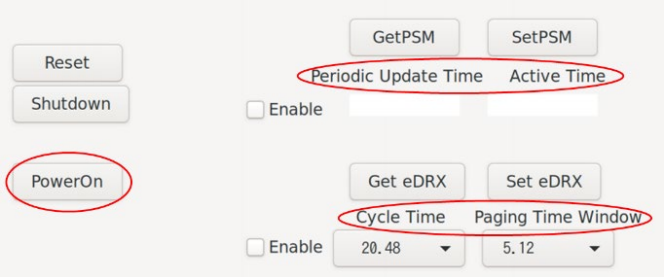
PSM：
電波サーチを最大約13日停止することが可能。

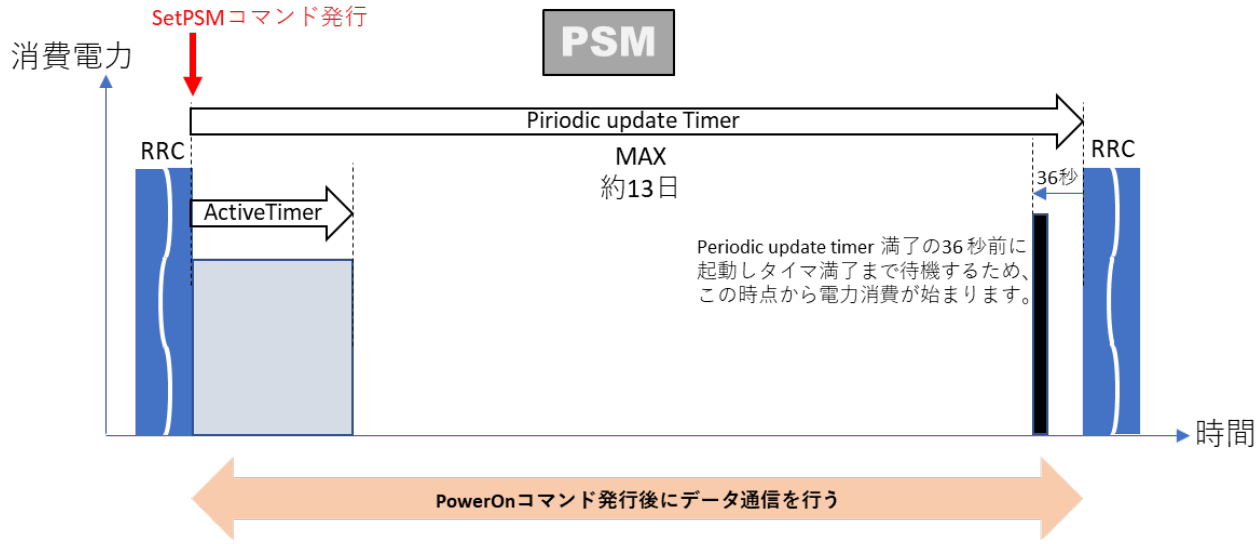


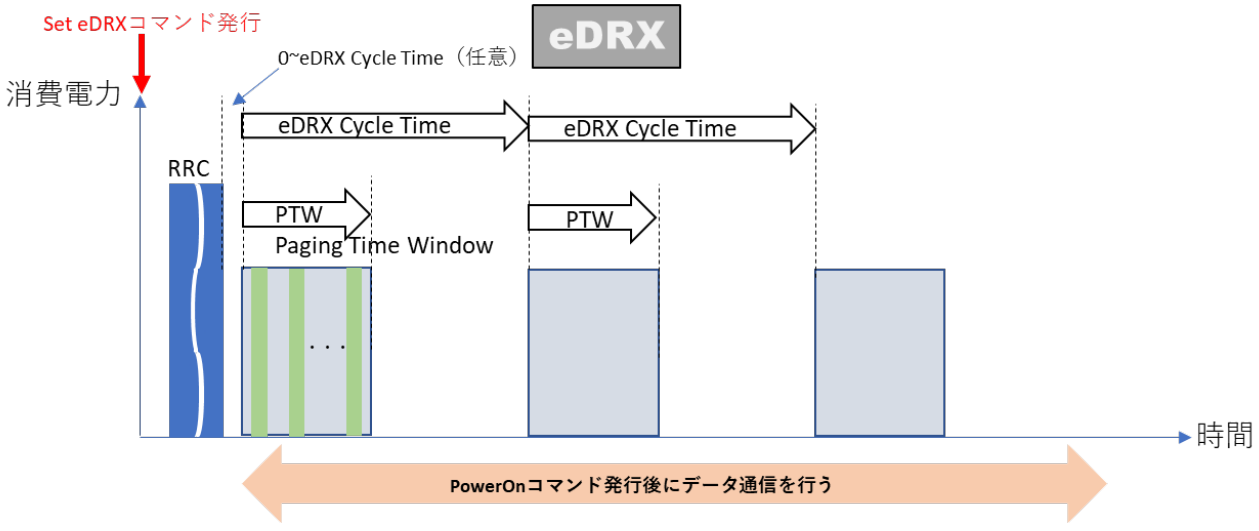
eDRX：
電波サーチの間隔を最大約43分に延ばすことが可能。

5.2. PSM 及び eDRX の Timer

サンプル画面







※データ送信を行われた後は、全ての PSM、eDRX とも Timer 値がリセットされる。