OPTO ELECTRONICS CO., LTD.

OPN-2002 SERIES/OPN-3002 SERIES

DATA collector

Android SDK ユーザーズガイド

株式会社 オプトエレクトロニクス

改版履歴

資料管理番号 ： SI12026

発 行 番 号 ： Rev.1.2

製　 品　 名 ： OPN-2002 Series / OPN-3002 Series

版	日付	変更箇所	内容
Rev.1.0	2012/08/10	－	新規作成
Rev.1.1	2012/10/22	1., 2.	段落番号追加
		1.1	対応する Android のバージョンのうち 2.2-2.3.2 を削除
		1.2	opn-spp.jar から bluetoothservice.jar に変更
		2.4	デバイスの検索を、”OPN-2002”の全部一致から先頭一致に変更
		2.6.1	読み込んだデータの編集処理追加。
		2.6.3	接続完了処理に enableAckNak()を追加
Rev.1.2	2013/02/08	表紙	OPN-3002 Series を追加
		3.1-3.4	OPN-3002 固有の設定項目を追加

目次

はじめに 1
本書の構成 1
1. 開発環境 1
2. APIの利用 1
3. コマンドリファレンス 1
1. 開発環境 2
1.1 実行環境 2
1.2 開発環境でのSDK使用手順 2
1.2.1 bluetoothservice.jarファイルを開発環境に取り込む 3
1.2.2 AndroidManifest.xmlでBluetoothの使用を許可する 4
2. APIの利用 5
2.1 Opn2002BluetoothServiceの生成と取得 5
2.2 Opn2002BluetoothServiceの状態 5
2.3 Android側がデータコレクターからの接続を待ち受ける 6
2.4 Android側からデータコレクターに接続する 7
2.5 切断する 8
2.6 データコレクターとの通信で発生したイベントを処理する 8
2.6.1 receive: データ受信 10
2.6.2 connectionLost: 切断 10
2.6.3 connected: 接続完了 11
2.6.4 connectFailed: 接続失敗 11
2.7 API標準のコマンドを実行する 12
2.8 任意のコマンドを実行する 13
3. コマンドリファレンス 14
3.2 Bluetooth設定 19

3.3 プリフィックス 24
3.4 サフィックス 26
3.5 文字列設定 27

はじめに

本書は、データコレクター「OPN-2002」「OPN-3002」を利用したAndroid向けアプリケーションの開発に利用されることを目的としています。

データコレクターの操作方法や、Android開発で必要となる一般的な技術情報につきましては、それぞれの説明書などを参照してください。

本書内の解説は「OPN-3002 固有設定」項目を除き、全て「OPN-2002」「OPN-3002」に共通となります。

本書の構成

1. **開発環境**
   本SDKを利用できる条件や手順について、解説しています。

2. **APIの利用**
   クラスライブラリの主なAPIについて、概要と利用方法をサンプルコードと合わせて解説しています。

3. **コマンドリファレンス**
   クラスライブラリのAPIを使って、データコレクターに送信できるコマンドの一覧です。

APIの詳細については、本書には記載していません。「OPN-2002 SDK APIリファレンス」を参照してください。

1. 開発環境

1.1 実行環境

本 SDK を利用して開発できる Android アプリケーションは、以下のバージョンの OS をターゲットとしたアプリケーションです。

OS	API レベル
Android 4.0.3	15
Android 4.0～4.0.2	14
Android 3.2	13
Android 3.1	12
Android 3.0	11
Android 2.3.3～2.3.7	10

1.2 開発環境でのSDK使用手順

本 SDK のクラスライブラリを利用するには、以下の 2 点を対応する必要があります。

- bluetoothservice.jar を開発環境に取り込む
- AndroidManifest.xml で Bluetooth の使用を許可する

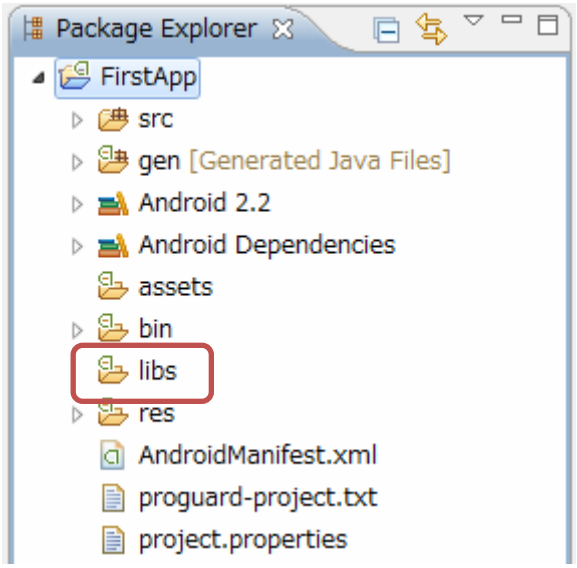
（前提）

Eclipse 上で Android アプリケーションを開発するための一般的な環境は整っていて、Android プロジェクトが既に作成されている状態を前提とします。以下の例では、Android プロジェクト「FirstApp」が作成されている状態からの手順を解説します。

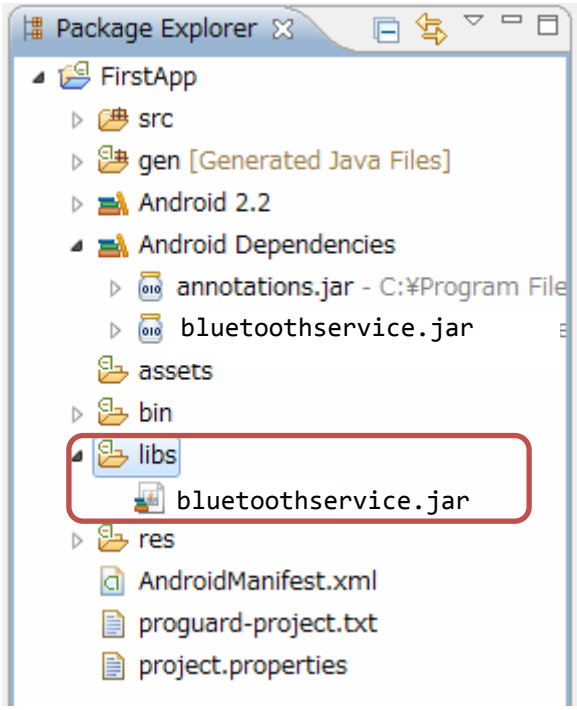
また、本書では、開発環境として Eclipse 3.7.2 を使った場合の手順を解説します。

1.2.1 bluetoothservice.jarファイルを開発環境に取り込む

① Android プロジェクト内に **libs** フォルダを作成します。



② libs フォルダ内に bluetoothservice.jar をコピーします。



→ 「Android Dependencies」以下にも、自動的に追加されます。

1.2.2 AndroidManifest.xmlでBluetoothの使用を許可する

① AndroidManifest.xml で Bluetooth を許可する設定を追加します。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ・・・・>

    ・・・・

    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.BLUETOOTH" />

    ・・・・

</manifest>
```

2. APIの利用

主な API の使用方法について、サンプルコードを用いて解説します。サンプルコードは、SDK に付属している FirstApp を使用します。

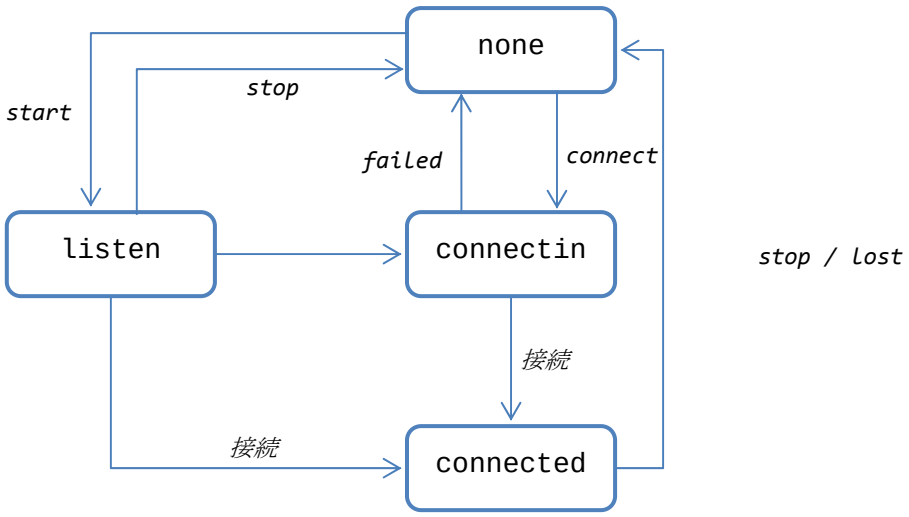
2.1 Opn2002BluetoothServiceの生成と取得

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mBTService = Opn2002BluetoothService.getInstance();
```

データコレクターとの Bluetooth 通信は、全て Opn2002BluetoothService クラスを介して行います。
本サンプルでは、onCreate の中で、Opn2002BluetoothService クラスのインスタンスを生成・取得をして、フィールドに保持しています。

2.2 Opn2002BluetoothServiceの状態



Opn2002BluetoothService には 4 つの状態があり、上記のような遷移を行います。
それぞれの特徴は次表の通りです。

none	SPP 接続が全く行われていない状態。
connecting	Android がマスターとなって、データコレクターに SPP 接続をしようとしている状態。
listen	Android がスレーブとなって、データコレクターからの接続を待ち受けている状態。
connected	SPP 接続が行われている状態。データの送受信が可能。

状態は、Opn2002BluetoothService#getState()メソッドで取得できます。本メソッドの返値は BluetoothServiceState クラス（Enum 型）のオブジェクトです。

2.3 Android側がデータコレクターからの接続を待ち受ける

```
@Override
protected void onStart() {
    super.onStart();

    if(mBTService != null){

        . . . . .

        if(mBTService.getState() == BluetoothServiceState.none){
            mBTService.start();
        }
    }
}
```

Opn2002BluetoothService#start()メソッドを実行することで、データコレクターからの接続を待ち受ける状態（BluetoothServiceState.listen）になります。
データコレクターからの接続要求があり、接続が確立した場合、connected()メソッドが実行されます（後述）。接続が確立すると、本処理（接続を待ち受ける処理）は自動的に停止します。
データコレクターがマスター設定の場合この方法を使用します。
マスター設定でもファンクションキーの長押しでの接続は次章の「Android 側からデータコレクターに接続する」の方法を使用します。

2.4 Android側からデータコレクターに接続する

```
// 接続ボタン押下
Button btnConn = (Button)findViewById(R.id.button_connect);
btnConn.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        mBTService.stop();  //Listen処理を停止
        //ペアリング済みデバイスから名前が「OPN」から始まるのデバイスを
        //1件取得して接続する
        Set<BluetoothDevice> set = mBTService.getPairedDevices();
        Iterator<BluetoothDevice> it = set.iterator();
        if(set.size() > 0) {
            while(it.hasNext()){
                BluetoothDevice device = it.next();
                if(device.getName() != null &&
                  device.getName().indexOf("OPN")>=0){
                    //接続を開始する
                    mBTService.connect(device, false);
                }
            }
            if (mBTService.getState()!=BluetoothServiceState.connected)
                mBTService.start(); //接続できなかった場合listen再開
        }
    }
});
```

Opn2002BluetoothService#connect()メソッドを実行することで、データコレクターへの接続が開始されます（BluetoothServiceState.connecting 状態）。
接続が成功した場合、connected()メソッドが実行されます（後述）。
接続が失敗した場合、connectFailed()メソッドが実行されます（後述）。
本サンプルでは、接続ボタンが押下されたときに、ペアリング済みのデバイス（名称が”OPN”から始まる）を１件取得して、そのデバイスに接続を試みています。
“OPN”から始まるデバイスが複数ある場合は、目的のデバイスを選択するように修正してください。
データコレクタがスレーブ設定の場合と、ファンクションキーの長押しでの接続の場合にこの方法を使用します。

2.5 切断する

```
// 切断ボタン押下
Button btnDisconn = (Button)findViewById(R.id.button_disconnect);
btnDisconn.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        mBTService.stop();
    }
});
```

Opn2002BluetoothService#stop()メソッドを実行することで、データコレクターとの接続が解除されます。
本サンプルでは、切断ボタンが押下されたときに、本処理を実行しています。

2.6 データコレクターとの通信で発生したイベントを処理する

```
public class FirstAppActivity extends Activity
implements IBluetoothObserver {

    . . . . .
}
```

Opn2002BluetoothService とデータコレクターが通信をする中で、「データ受信」「切断」「接続完了」「接続失敗」のイベントが発生することがあります。これらのイベントを受信して処理するためには、まず、イベントを処理するクラスで IBluetoothObserver インターフェースを実装する必要があります。
本サンプルでは、FirstAppActivity クラスが当該インターフェースを実装しています。

```
@Override
protected void onResume() {
    super.onResume();

    if(mBTService != null){
        mBTService.addObserver(this);
    }
    refreshStatusLabel();
}
```

次に、IBluetoothObserver を実装したクラスのオブジェクト（オブザーバー）を引数にして、Opn2002BluetoothService#addObserver()メソッドを実行します。これにより、イベントが発生した時点で、インターフェースを実装しているメソッドが実行されます。
本サンプルでは、onResume の中で、オブザーバーを登録しています。

```
@Override
protected void onPause(){
    super.onPause();

    if(mBTService != null){
        mBTService.removeObserver(this);
    }
}
```

Opn2002BluetoothService#addObserver()メソッド登録したオブザーバーは、同クラスの removeObserver()メソッドを実行することで、登録が解除されます。
本サンプルでは、onPause の中で、オブザーバーの登録を解除しています。

2.6.1 receive: データ受信

```
@Override
public void receive(String data) {
    if(mTextView != null && data != null){
        /* 改行コードを変換します。  */
        data = data.replace((char)0x0D, (char)0x0A);
        /* コマンドのACK/NAKが入っていた場合削除します。  */
        data = data.replaceAll(""+(char)0x6, "");
        data = data.replaceAll(""+(char)0x15, "");
        mTextView.append(data);
    }
}
```

データコレクターで、バーコードを読んだときなどに発生する「データ受信」イベントを処理します。読み込んだデータは、String 型の引数として、渡されてきます。
読み込んだデータは改行コードがデフォルトでは 0xD になってるので、0xA に変換します。また、コマンドが前後で実行された場合、ACK/NAK が連結されている場合があるので取除きます。
本サンプルでは、String 型の引数 data の内容を TextView に追加しています。

2.6.2 connectionLost: 切断

```
@Override
public void connectionLost() {
    Toast.makeText(this, "connection lost", Toast.LENGTH_LONG).show();
    refreshStatusLabel();
}
```

データコレクターとの通信が切断されたとき発生する「切断」イベントを処理します。
本サンプルでは、Toast を使ってユーザに「切断されたこと」を通知しています。また、refreshStatusLabes()メソッドで、接続状態の表示も更新しています。

2.6.3 connected: 接続完了

```
@Override
public void connected(BluetoothDevice arg0) {
    Toast.makeText(this, "connected", Toast.LENGTH_LONG).show();
    mBTService.enableAckNak();
    // 注意! Opn2002BluetoothServiceを使用する際はかならず接続時に
enableAckNak()を実行してください。
    refreshStatusLabel();
}
```

データコレクターとの SPP 接続が完了したとき発生する「接続完了」イベントを処理します。
本サンプルでは、Toast を使ってユーザに「接続が完了したこと」を通知しています。
また、refreshStatusLabes()メソッドで、接続状態の表示も更新しています。

本 SDK はコマンド応答として ACK/NAK を受信することで通信を制御しています。OPN のデフォルトはコマンド応答ですので、コネクト時に enableAckNak()を呼び出してコマンド応答機能を有効にする必要があります。

2.6.4 connectFailed: 接続失敗

```
@Override
public void connectFailed() {
    Toast.makeText(this, "connection failed ", Toast.LENGTH_LONG).show();
}
```

Android 側からデータコレクターへの接続が失敗してしまったときに発生する「接続失敗」イベントを処理します。
本サンプルでは、Toast を使ってユーザに「接続が失敗したこと」を通知しています。

2.7 API標準のコマンドを実行する

```
// バージョンボタン押下
Button btnCommand = (Button)findViewById(R.id.button_command);
btnCommand.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        mBTService.getFirmwareVersion(new GetFirmwareVersionCallback() {
            @Override
            protected void callbackExecute(String arg0) {
                receive(arg0); //画面に実行結果を表示する
            }
        });
    }
});
```

データコレクターに関するコマンドの中には、Opn2002BluetoothService クラスのメソッドとして、提供されているものがあります。これらのコマンドを実行するためのメソッドは、通常、以下のような形をしています。

```
public void メソッド名(引数1, 引数2, …, コールバック)
```

このメソッドは、非同期で実行されるため、非同期処理が完了したときに呼び出されるコールバックを指定する必要があります。コールバックは、どれも共通して IBluetoothCallback を実装したクラスのオブジェクトになっていて、非同期処理が完了したときには、callbackExceute()メソッドが実行されます。

本サンプルでは、バージョンボタンが押下されたときに、Opn2002BluetoothService クラスの getFirmwareVersion()メソッドを実行しています。コールバックには、IBluetoothCallback を実装している GetFirmwareVersionCallback クラスの匿名クラスを指定していて、処理が完了したときには、callbackExcecute()メソッドが実行されます。

2.8 任意のコマンドを実行する

```
// ブザーON
Button btnBuzzerOn = (Button)findViewById(R.id.button_buzzer_on);
btnBuzzerOn.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        mBTService.write("W8", new VoidCallback() {

            @Override
            protected void callbackExecute(Void arg0) {
                if(getIsError()){
                        showErrorMessage(); //エラーメッセージを表示
                }
            }
        });
    }
});
```

Opn2002BluetoothService クラスのメソッドとして、提供されていないメソッドを実行する場合は、Opn2002BluetoothService#write()メソッドを実行します。write()メソッドの引数に、実行したいコマンド文字列とコールバックを指定することで、メソッドが非同期で実行されます。コールバックの考え方は、前述の API 標準コマンドと同様です。
本サンプルでは、ブザーON ボタンが押下されたときに、Opn2002BluetoothService クラスの write()メソッドを実行しています。コールバックには、IBluetoothCallback を実装している VoidCallback クラスの匿名クラスを指定していて、処理が完了したときには、callbackExcecute()メソッドが実行されます。
利用できるコマンドについては、「3. コマンドリファレンス」を参照してください。

PIN(]PINS)、接続相手アドレス(]BDAS)、デバイス名([E65)、プリフィックス、サフィックス、エラーメッセージ（TH,TI）で設定する文字列の入力は「3.5 文字列設定」のコマンドを利用します。
例１：デバイス名を「OPN-2002」に設定する場合のコマンド
　　“[E650O0P0N5NQ2Q0Q0Q2[E66”
例２：UPC-A のプリフィックスを「UPCA」に設定する場合のコマンド
　　“N10U0P0C0A”

3. コマンドリファレンス

3.1 デコーダ設定設定項目	パラメータ	設定値	コマンド
UPC-A/E 読み取り	許可	1	R1
	禁止	0	[X4B
UPC-A/E Addon2 読み取り	許可	1	R2
	禁止	0	[X4C
UPC-A/E Addon5 読み取り	許可	1	R3
	禁止	0	[X4D
JAN/EAN-13/8 読み取り	許可	1	R4
	禁止	0	[X4E
JAN/EAN-13/8 Addon2 読み取り	許可	1	R5
	禁止	0	[X4F
JAN/EAN-13/8 Addon5 読み取り	許可	1	R6
	禁止	0	[X4G
Code-39 読み取り	許可	1	B2
	禁止	0	VB
Tri-optic 読み取り	許可	1	JZ
	禁止	0	[DDJ
NW-7 読み取り	許可	1	B3
	禁止	0	VC
Industrial 2 of 5 読み取り	許可	1	R7
	禁止	0	V7
Interleaved 2 of 5 読み取り	許可	1	R8
	禁止	0	V8
S-Code 読み取り	許可	1	R9
	禁止	0	[DDK
Matrix 2 of 5 読み取り	許可	1	BB
	禁止	0	[DDL
Code 93 読み取り	許可	1	B5
	禁止	0	VD
Code 128 読み取り	許可	1	B6
	禁止	0	VE
EAN-128 読み取り	許可 if Possible	2	OG

	許可　EAN-128 Only	1	JF
	禁止　Code 128 として出力	0	OF
MSI/Plessey 読み取り	許可	1	B7
	禁止	0	VF
IATA 読み取り	許可	1	B4
	禁止	0	VH
UK/Plessey 読み取り	許可	1	B1
	禁止	0	VA
Telepen 読み取り	許可	1	B9
	禁止	0	VG
GS1 DataBar(RSS-14)	許可	1	JX
	禁止	0	SJ
GS1 DataBar Limited(RSS-Limited)	許可	1	JY
	禁止	0	SK
GS1 DataBar Expanded(RSS-Expanded)	許可	1	DR
	禁止	0	SL
PDF-417 読み取り	許可	1	[BCF
	禁止	0	[BCR
MicroPDF-417 読み取り	許可	1	[BCG
	禁止	0	[BCS
Code-11 読み取り	許可	1	[BLC
	禁止	0	[BLA
Code 3 of 5 読み取り	許可	1	WH
	禁止	0	WI
UPC-A フォーマット (C/D 転送)	１３桁（先頭０とＣＤ付き）	b2=1, b4=1	E2
	１２桁（先頭０なし）	b2=1, b4=0	E3
	１２桁（ＣＤなし）	b2=0, b4=1	E4
	１１桁（先頭０とＣＤなし）	b2=0, b4=0	E5
UPC-E フォーマット	８桁（先頭０とＣＤ付き）	b2=1,	E6

(C/D 転送)		b4=1	
	７桁（先頭なし）	b2=1, b4=0	E7
	７桁（ＣＤなし）	b2=0, b4=1	E8
	６桁（先頭０とＣＤなし）	b2=0, b4=0	E9
JAN/EAN-13 フォーマット (C/D 転送)	CD 転送する	1	6K
	CD 転送しない	0	6J
JAN/EAN-8 フォーマット (C/D 転送)	CD 転送する	1	6I
	CD 転送しない	0	6H
Code 39 C/D 転送	CD 転送する	1	D9
	CD 転送しない	0	D8
NW-7 C/D 転送	CD 転送する	1	H8
	CD 転送しない	0	H9
Industrial 2 of 5/ Interleaved 2 of 5 C/D 転送	CD 転送する	1	E0
	CD 転送しない	0	E1
MSI/Plessey C/D 転送	CD 転送する CD1	1	4E
	CD 転送する CD1 and CD2	2	4F
	CD 転送しない	0	4G
IATA C/D 転送	CD 転送する	1	4L
	CD 転送しない	0	4M
GS1 DataBar family C/D 転送	CD 転送する	1	DL
	CD 転送しない	0	DM
WPC(UPC/EAN/JAN) C/D 計算	C/D 計算する	1	[XEE
	C/D 計算しない	0	[XEF
Code 39 C/D 計算	C/D 計算する	1	C0
	C/D 計算しない	0	C1
NW-7 C/D 計算	C/D 計算する Mod10/W1,2 spec1	1	[XF8
	C/D 計算する Mod16	2	H6

	C/D 計算する 7 check	3	[XFB
	C/D 計算する Mod11	4	[XFC
	C/D 計算しない	0	H7
Industrial 2 of 5/ Interleaved 2 of 5 C/D 計算	C/D 計算する	1	G1
	C/D 計算しない	0	G0
Code 93 C/D 計算	C/D 計算する	1	AC
	C/D 計算しない	0	9Q
Code 128/EAN-128 C/D 計算	C/D 計算する	1	ME
	C/D 計算しない	0	MF
MSI/Plessey C/D 計算	C/D 計算する CD1 only (Mod10)	1	4B
	C/D 計算する CD's (Mod10/Mod10)	2	4C
	C/D 計算する CD's (Mod10/Mod11)	3	4D
	C/D 計算する CD's (Mod11/Mod10)	4	4R
	C/D 計算しない	0	4A
IATA C/D 計算	C/D 計算する (CPN+FORM SERIAL)	1	4J
	C/D 計算する (FORM SERIAL)	2	4I
	C/D 計算する (ALL DATA)	3	4K
	C/D 計算しない	0	4H
Code 39 スタートストップ転送	転送する	1	D0
	転送しない	0	D1
NW-7 スタートストップ転送	転送する ABCD/TN*E	1	F1
	転送する abcd/tn*e	2	F2
	転送する ABCD	3	F3
	転送する abcd	4	F4
	転送する DC1DC2DC3DC4	5	HJ
	転送しない	0	F0
読取り出来なかった場合のエ	送信しない	0	TG

ラーメッセージ	送信する	9	TH,TI
読取り出来なかった場合に送信する文字列の設定（ラベルが見つからない）	char×4 桁		TH
読取り出来なかった場合に送信する文字列の設定（デコード失敗）	char×4 桁		TI
OPN-3002 固有設定			
Intelligent Mail 読み取り	許可	1	[D5F
	禁止	0	[D5G
Postnet 読み取り	許可	1	[D6A
	禁止	0	[D6B
Japanese postal 読み取り	許可	1	[D5P
	禁止	0	[D5Q
CodablockF 読み取り	許可	1	[D4P
	禁止	0	[D4Q
Data Matrix(ECC200) 読み取り	許可	1	[BCC
	禁止	0	[BCO
Data Matrix(ECC000-140) 読み取り	許可	1	[BG0
	禁止	0	[BG1
Aztec code 読み取り	許可	1	[BCH
	禁止	0	[BCT
Aztec runes 読み取り	許可	1	[BF2
	禁止	0	[BF3
Chinese Sensible code 読み取り	許可	1	[D4L
	禁止	0	[D4M
QR code 読み取り	許可	1	[BCD
	禁止	0	[BCP
MicroQR 読み取り	許可	1	[D2U
	禁止	0	[D2V
Maxi Code 読み取り	許可	1	[BCE
	禁止	0	[BCQ
Composite on GS1Databar 読み取り	許可	1	[BHE
	禁止	0	[BHF

Composite on UPC/EAN 読み取り	許可	1	[D1V
	禁止	0	[D1W

3.2 Bluetooth設定

読み取りモード	単発読み	1	S0
	複数読み	2	S1
	連続読み	3	S2
読み取り時間	無限	0	YM
	0 秒	-1	Y0
	1 秒	50	Y1
	2 秒	100	Y2
	3 秒	150	Y3
	4 秒	200	Y4
	5 秒	250	Y5
	6 秒	300	Y6
	7 秒	350	Y7
	8 秒	400	Y8
	9 秒	450	Y9
	読み取り時間 10 倍	-	YL
	1 回読取 0 回照合	0	X0
	2 回読取 1 回照合	1	X1
	3 回読取 2 回照合	2	X2
	4 回読取 3 回照合	3	X3
	5 回読取 4 回照合	4	BS
	6 回読取 5 回照合	5	BT
	7 回読取 6 回照合	6	BU
	8 回読取 7 回照合	7	BV
	9 回読取 8 回照合	8	BW
	10 回読取 9 回照合	9	[XBT
	11 回読取 10 回照合	10	[XBU
	12 回読取 11 回照合	11	[XBV
	13 回読取 12 回照合	12	[XBW
	14 回読取 13 回照合	13	[XBX

	15 回読取 14 回照合	14	[XBY
	16 回読取 15 回照合	15	[XBZ
二度読み防止タイマ	無限	0	AG
	50ms	3	AH
	100ms	5	AI
	200ms	10	AJ
	300ms	15	AK
	400ms	20	AL
	500ms	25	AM
	600ms	30	AN
アドオンタイマ	なし	0	XA
	250ms	13	XB
	500ms	25	XC
	750ms	38	XD
ブザーボリューム	最大	127	T0
	大	32	T1
	中	8	T2
	小	1	T3
LED 点灯時間	無効	0	T4
	200ms	10	T5
	400ms	20	T6
	600ms	30	T7
トリガモード	トリガ無効	0	S7
	トリガ有効	1	S8
トリガリピート	無効	0	/K
	有効	1	/M
ブザー音	無効	0	W0
	有効	1	W8
ブザートーン	単音	0	W1
	高低	1	W2
	低高	2	W3
	4.5KHｚ	3	[XTS
	2.2KHz-2KHz	4	[X%Q
ブザー鳴動時間	100ms	5	W4
	200ms	10	W5

	400ms	20	W6
	50ms	2	W7
ブザー鳴動タイミング	転送前ブザー	0	VY
	転送後ブザー	1	VZ
接続相手アドレス		char×12¥0	
	アドレス設定開始	-	]BDAS
	アドレス設定終了	-	]BDAE
認証	認証無し	0	]AUTD
	認証有り(毎回)	1	]AUTE
	認証有り(対応しない場合ペア)	7	]AUTO
暗号化	無効	0	]ENCD
	有効	1	]ENCE
コマンド応答	有り（ACK/NAK)	1	WC
	ACK/NAK 制御の設定に従う	0	WD
PIN コード		char×16¥0	
	PIN 設定開始	-	]PINS
	PIN 設定終了	-	]PINE
トリガ接続	無効	0	]TSCD
	有効	1	]TSCE
アドレスバーコード読み取りによる接続処理実行有無	無効	0	]DIAU
	有効	1	]ENAU
ACK/NAK 制御	無し	0	[XP5
	有り	1	P3
	有り（NoResponse）	2	P4
接続モード	SPP マスター	0	]BCMA
	SPP スレーブ	1	]BCSA
	HID	2	[C02
スレーブ接続待ち時間	30 秒	1500	]SWT0
	1 分	3000	]SWT1
	2 分	6000	]SWT2

	3 分	9000	]SWT3
	4 分	12000	]SWT4
自動再接続有効時間	無効	0	]CA00
	1 分	3000	]CA01
	2 分	6000	]CA02
	3 分	9000	]CA03
	4 分	12000	]CA04
	5 分	15000	]CA05
	6 分	18000	]CA06
	7 分	21000	]CA07
	8 分	24000	]CA08
	9 分	27000	]CA09
	10 分	30000	]CA10
	11 分	33000	]CA11
	12 分	36000	]CA12
	13 分	39000	]CA13
	14 分	42000	]CA14
	15 分	45000	]CA15
自動切断時間	無効	0	]AD00
	10 分	30000	]AD01
	20 分	60000	]AD02
	30 分	90000	]AD03
	40 分	120000	]AD04
	50 分	150000	]AD05
	60 分	180000	]AD06
	1 分	3000	]ADM1
	2 分	6000	]ADM2
	3 分	9000	]ADM3
	4 分	12000	]ADM4
	5 分	15000	]ADM5
	6 分	18000	]ADM6
	7 分	21000	]ADM7
	8 分	24000	]ADM8
	9 分	27000	]ADM9
	10 秒	500	]ADS1

	20 秒	1000	]ADS2
	30 秒	1500	]ADS3
	40 秒	2000	]ADS4
	50 秒	2500	]ADS5
トリガ接続長押し時間	トリガ接続無効	0	]PC00
	1 秒	50	]PC01
	2 秒	100	]PC02
	3 秒	150	]PC03
	4 秒	200	]PC04
	5 秒	250	]PC05
	6 秒	300	]PC06
	7 秒	350	]PC07
	8 秒	400	]PC08
	9 秒	450	]PC09
トリガ切断長押し時間	トリガ切断無効	0	]PD00
	1 秒	50	]PD01
	2 秒	100	]PD02
	3 秒	150	]PD03
	4 秒	200	]PD04
	5 秒	250	]PD05
	6 秒	300	]PD06
	7 秒	350	]PD07
	8 秒	400	]PD08
	9 秒	450	]PD09
ACK/NAK 待ち時間	100ms	5	[XI5
	500ms	25	[XI6
	1s	50	[XI7
圏外メモリー	無効	0	]DTMD
	有効	1	]DTME
データコレクト	無効	0	[BM0
	有効	1	[BM1
バーコード読取時自動接続	無効	0	]ARCD
	有効	1	]ARCE
データコレクタからの切断時ブザー	無効	0	]DSSD
	有効	1	]DSSE

接続相手からの切断時ブザー	無効	0	]DSPD
	有効	1	]DSPE
メモリーデータの出力方法	接続時即出力	0	[EBB
	ファンクションキー押下かコマンドで出力	1	[EBC
	データ出力コマンド		[EBD
USB 接続時 COM 通信	無効	0	[C10
	有効	1	[C11
ファンクション押下時出力	設定開始		[C23
	HT	0x09	[$09
	LF	0x0A	[$0A
	CR	0x0D	[$0D
	CAN	0x18	[$18
	ESC	0x1B	[$1B
	出力なし	0xFF	[$FF
	iphone ソフトキーボード表示	0xA6	[$A6
	ENTER	0xB2	[$B2
Bluetooth デバイス名（自機）	デバイス名設定開始	-	[E65
	デバイス名設定終了	-	[E66
前回のスレーブ接続時の相手アドレス(12 文字 + NUL)			
OPN-3002 固有設定			
グッドリードバイブレーター	有効	1	[EBI
	無効	0XFF	[EBH

3.3 プリフィックス

UPC-A のプリフィクス			N1
UPC-A アドオンのプリフィクス			M0
UPC-E のプリフィクス			N2
UPC-E アドオンのプリフィクス			M1
JAN/EAN-13 のプリフィクス			N3
JAN/EAN-13 アドオンのプリフィクス			M2

JAN/EAN-8 のプリフィクス			N4
JAN/EAN-8 アドオンのプリフィクス			M3
Code-39 のプリフィクス			M4
Tri-Optic のプリフィクス			MC
NW-7 のプリフィクス			M5
D.2of5 のプリフィクス			M6
I.2of5 のプリフィクス			M7
S Code のプリフィクス			MB
Matrix 2of5 のプリフィクス			GL
Code-93 のプリフィクス			M8
Code-128 のプリフィクス			M9
MSI/Plessey のプリフィクス			N0(ｾﾞﾛ)
IATA のプリフィクス			I8
UK/Plessey のプリフィクス			MA
Telepen のプリフィクス			L8
RSS のプリフィクス			OE
PDF-417 のプリフィクス			OC
Micro PDF-417 のプリフィクス			OD
Code-11 のプリフィクス			[BLD
Code 3of5 のプリフィクス			*$
EAN-128 のプリフィクス			[XMX
コモンプリフィックス設定			MZ
全コードのプリフィクス設定			RY
OPN-3002 固有設定			
Intelligent Mail のプリフィクス			[D5I
Postnet のプリフィクス			[D6D
Japanese postal のプリフィクス			[D5S
CodablockF のプリフィクス			[D4S
Data Matrix(ECC200, ECC000-140) のプリフィクス			MD
Aztec code/Aztec Runes のプリフィクス			[BF0

Chinese Sensible code のプリフィクス			[D4N
QR/MicroQR のプリフィクス			MK
Maxi code のプリフィクス			ML
Composite(on GS1Databar,UPC/EAN) のプリフィクス			RR

3.4 サフィックス

UPC-A のサフィックス			N6
UPC-A アドオンのサフィックス			O0(ｵｰｾﾞﾛ)
UPC-E のサフィックス			N7
UPC-E アドオンのサフィックス			O1
JAN/EAN-13 のサフィクス			N8
JAN/EAN-13 アドオンのサフィックス			O2
JAN/EAN-8 のサフィックス			N9
JAN/EAN-8 アドオンのサフィックス			O3
Code-39 のサフィックス			O4
Tri-Optic のサフィックス			PN
NW-7 のサフィックス			O5
D.2of5 のサフィックス			O6
I.2of5 のサフィックス			O7
S Code のサフィックス			OB
Matrix 2of5 のサフィックス			GM
Code-93 のサフィックス			O8
Code-128 のサフィックス			O9
MSI/Plessey のサフィックス			N5
IATA　のサフィックス			I9
UK/Plessey のサフィックス			OA
Telepen のサフィックス			L9
RSS　のサフィックス			PQ
PDF-417 のサフィックス			PY
Micro PDF-417 のサフィックス			PZ

Code-11 のサフィックス			[BLE
Code 3of5 のサフィックス			*%
EAN-128 のサフィックス			[XOX
コモンサフィックス			PS
全コードのサフィックス設定			RZ
OPN-3002 固有設定			
Intelligent Mail のサフィックス			[D5J
Postnet のサフィックス			[D6E
Japanese postal のサフィックス			[D5T
CodablockF のサフィックス			[D4T
Data Matrix(ECC200, ECC000-140) のサフィックス			PO
Aztec code/Aztec Runes のサフィックス			[BF1
Chinese Sensible code のサフィックス			[D4O
QR/MicroQR のサフィックス			PW
Maxi code のサフィックス			PX
Composite(on GS1Databar,UPC/EAN) のサフィックス			RS

3.5 文字列設定

設定文字	コマンド
0	Q0
1	Q1
2	Q2
3	Q3
4	Q4
5	Q5
6	Q6
7	Q7
8	Q8

9	Q9
A	0A
B	0B
C	0C
D	0D
E	0E
F	0F
G	0G
H	0H
I	0I
J	0J
K	0K
L	0L
M	0M
N	0N
O	0O
P	0P
Q	0Q
R	0R
S	0S
T	0T
U	0U
V	0V
W	0W
X	0X
Y	0Y
Z	0Z
a	$A
b	$B
c	$C
d	$D
e	$E
f	$F
g	$G
h	$H

i	$I
j	$J
k	$K
l	$L
m	$M
n	$N
o	$O
p	$P
q	$Q
r	$R
s	$S
t	$T
u	$U
v	$V
w	$W
x	$X
y	$Y
z	$Z
〈SPACE〉	5A
!	5B
″	5C
#	5D
$	5E
%	5F
&	5G
	5H
(	5I
)	5J
*	5K
+	5L
,	5M
–	5N
.	5O
/	5P
:	6A

;	6B
<	6C
=	6D
>	6E
?	6F
@	6G
[	7A
¥	7B
]	7C
^	7D
_	7E
`	7F
{	9T
|	9U
}	9V
~	9W
^@ (NULL)	9G
^A (SOH)	1A
^B (STX)	1B
^C (ETX)	1C
^D (EOT)	1D
^E (ENQ)	1E
^F (ACK)	1F
^G (BEL)	1G
^H (BS)	1H
^I (HT)	1I
^J (LF)	1J
^K (VT)	1K
^L (FF)	1L
^M (CR)	1M
^N (SO)	1N
^O (SI)	1O
^P (DLE)	1P
^Q (DC1)	1Q
^R (DC2)	1R

^S (DC3)	1S
^T (DC4)	1T
^U (NAK)	1U
^V (SYN)	1V
^W (ETB)	1W
^X (CAN)	1X
^Y (EM)	1Y
^Z (SUB)	1Z
^[(ESC)	9A
^＼(FS)	9B
^] (GS)	9C
^^ (RS)	9D
^_ (US)	9E
DEL (ASCII 127)	9F
年	[$YR
月	[$MO
日	[$DY
時	[$HR
分	[$MI
秒	[$SC
スキャンカウント	[$CT
バーコード種別	[$BT
バーコードデータ長	[$BL
電池電圧	[$BV
BD アドレス	[$AR
端末 ID	[$ID
端末名	[$NM

弊社製品名: OPN-2002 Series
OPN-3002 Series
発行番号: Rev.1.2
管理番号: SI12026

株式会社オプトエレクトロニクス

E-Mail:sales@opto.co.jp　URL:http://www.opto.co.jp