JavaScript プログラミングガイド CR1500、CR1100、CR2700、CR8x7x、および CR5200

2021年1月

The Code Corporation • 434 West Ascension Way Suite 300, Salt Lake City, UT 84123 • +1 (801) 495-2200 • FAX +1 (801) 495-0280

目次

The Code Corporation • 434 West Ascension Way Suite 300, Salt Lake City, UT 84123 • +1 (801) 495-2200 • FAX +1 (801) 495-0280

The Code Corporation • 434 West Ascension Way Suite 300, Salt Lake City, UT 84123 • +1 (801) 495-2200 • FAX +1 (801) 495-0280

The Code Corporation • 434 West Ascension Way Suite 300, Salt Lake City, UT 84123 • +1 (801) 495-2200 • FAX +1 (801) 495-0280

1 はじめに

Code Corporation (Code) は、データ収集アプリケーション用の画像ベースのリーダーおよびソフトウェアツールの設計、開発、製造を行っています。ソフトウェア開発、光学、画像処理、Bluetooth™ワイヤレス技術の専門知識を持つCodeは、オートIDおよびデータ収集業界の革新的リーダーです。

多くの Code reader は、バーコード読み取りと使いやすい JavaScript ベースのアプリケーション開発を組み合わせています。このユーザーマニュアルは、CR1100、CR1500、CR2700 シリーズリーダに適用されます。

2 文書の目的

本マニュアルでは、Code reader（CR1100、CR1500、CR2700）の JavaScript アプリケーションプログラミングインターフェース（API）について説明します。ユーザーがプログラミングの知識を持ち、JavaScript 言語に精通していることを前提としています。

- Code readerはバーコードを読み取り、選択された通信リンクを介してコードデータを送信するか、またはリーダーメモリにデータを保存するようにプログラムすることができます（利用可能な場合はバッチモード）。
- プログラミング環境は、以下のインターフェースを提供します：
  - リーダーメモリ内のデータを読み取り、操作します
  - ホストから送信されたデータにアクセスします
  - 通信リンクを介してホストコンピュータにデータを送信します
  - 通信リンクのタイプを選択します
  - リーダー構成設定の設定、変更、取得します

3 文書の読者

このドキュメントは、Code readerの動作をカスタマイズしたいCode readerユーザーや開発者向けの手引書です。説明されているコマンドは、文字の削除、ある文字セットを別の文字に変換、読み取ったバーコードのタイプに基づくさまざまなタイプの出力、文字の追加、タブや改行など、データの操作を対象としています。

4 文書とコーディング規約

本書では、読みやすさを考慮し、以下の慣例を採用しています：

- アプリケーション開発記述の一部である単語は、`Courier New`フォントを使用します。
- コード例では、**`太字のCourier New`**フォントを青で使用しています。
- プログラマーが与えなければならない変数名は`Courier New`フォントで、比較記号で囲まれています
  （例）`<variable_name>`

Code reader JavaScriptライブラリは、以下の命名規則を使用しています：

- 識別子：大文字と小文字が混在した単語の結合（いわゆるキャメルケース）
  （例）nasaSpaceShuttle, httpServer, codeXml
- 頭字語やその他の頭字語は単語と同じように大文字で表記（例）CIMCE, CPU
- 変数とプロパティ：頭文字は小文字（例）thisIsAVariable、thatIsAProperty
- クラス（つまりコンストラクター）：頭文字は大文字（例）AClassIsCapitalizedCamelCase
- 関数：変数とプロパティと同様、頭文字は小文字

The Code Corporation • 434 West Ascension Way Suite 300, Salt Lake City, UT 84123 • +1 (801) 495-2200 • FAX +1 (801) 495-0280

- 単位：名前の後ろに付加。名前とはアンダーバーで区切られ、重要な場合は大文字と小文字を正しく区別
  （例）offset_pixels、width_mm、power_mW、powerRatio_dB

5　関連文書

Code readerは、リーダーの動作を定義する大規模な構成設定によって制御されます。これらの設定は Code 文書 D027153（CR8200 CR950 CR1500 CR1100 CR2700 CRA-A271 Configuration Control Document）に記載されています。これらの構成設定の使用方法をよく理解してください。

注：Code社のウェブサイト（https://www.codecorp.com/）にアクセスし、製品のドキュメントセクションから CR8200 CR950 CR1500 CR1100 CR2700 CRA-A271 Configuration Control Document を入手してください。

6　関連ユーティリティ

USBバーチャルCOMドライバ：USBケーブルリーダー用の仮想COMポートを作成するソフトウェアドライバ。このドライバにより、USBポートに接続しながら、シリアルデバイスからの入力を必要とするコンピュータプログラムでリーダーを使用することができます。

CortexTools2：ホストとリーダー間のファイル転送を含む、すべてのCode reader機能へのホストアクセスを提供します。有効な通信モードは、USBダウンローダー、USBバーチャルCOM、およびRS232です。

これらのユーティリティは、https://www.codecorp.com/より入手可能です。

7　JavaScriptリソース

JavaScriptの専門家である必要はありませんが、この文書に基づいてRulesを使用するには、ある程度のJavaScriptの知識が必要です。このドキュメントはJavaScriptのマニュアルではありません。JavaScriptプログラミングに関する書籍は数多くありますが、以下の情報源はJavaScriptの参考書籍やオンラインドキュメントを提供しており、役に立つと思われます：

- **JavaScript：The Definitive Guide**
  デイビッド・フラナガン著
- **JavaScript, A Beginner's Guide, Third Edition (Beginner's Guide)**
  ジョン・ポロック著
- **JavaScript Demystified (Demystified)**
  ジェームス・キョウ著
- **JavaScript (TM) in 10 Simple Steps or Less**
  アルマン・ダネシュ著
- **http://www.w3schools.com/jsref/default.asp**
- **http://www.javascript.com/**

8　正規表現のリソース

正規表現の専門家である必要はありませんが、この文書に基づいてルールを使用するには、正規表現の知識が必要です。この文書は正規表現のマニュアルではありません。正規表現の使用法に関する書籍は数多くありますが、以下の情報源は正規表現の参考書籍や役に立つと思われるオンライン文書を提供しています：

- **Mastering Regular Expressions**
  ジェフリー・E・F・フリードル著

- **Beginning Regular Expressions (Programmer to Programmer)**
  アンドリュー・ワット著
- **http://en.wikipedia.org/wiki/Regular_expression#Basic_concepts**
- **http://www.regular-expressions.info/**

9 Code readerへの.codeRules.jsスクリプトのインストールと実行

Codeが提供するJavaScriptファイル.cra.jsは、リーダー上にあるCode Rulesファイルをチェックし、インクルードするために記述されます。ファイル名は次のパターンに従ってください： .codeRules<.><identifier>.js
※<.>は、<identifier>が含まれる場合は含める必要があり、<identifier>が省略されている場合は省略する必要があります。
※<identifier>は英数字と'_'を使った有効な文字列です。
Code Rulesファイルの目的は、主にデコードの試行、デコード、`onEvent`のオーバーライド関数を提供することです。

10 セキュリティ

Codeは、ライセンス保護のための暗号化ユーティリティを提供します：

- 各Code readerには固有のリーダーIDが含まれています。
- リーダーの一部の機能はライセンスにより保護されています。
- Codeは、保護された機能を有効にするライセンスファイルを提供します。
- 保護された機能を使用するためには、ライセンスファイルが各リーダーに必要です。
- サードパーティのソフトウェアも、暗号化ユーティリティを使って保護することができます。

11 デバッグ（reader.debug を参照）

12 プログラミングの概念

開発者がリーダー用の独自のアプリケーションを作成できるように、Codeは使いやすいオブジェクト指向のJavaScriptアプリケーションプログラミングインタフェース（API）を提供しています。開発者は、JavaScript APIを通じて、プロンプトやデータ入力を含む複雑なビジネスアプリケーションを作成することができます。

プログラミング・インターフェースの機能は以下の通り：

- イベントハンドラ
- シンボルデコード
- ホスト通信
- ローカル・データ・ストレージ
- Code readerの設定

このインターフェースをサポートするために、Codeはこれらの機能をサポートする3つのオブジェクトを提供しています：

- Reader
- Storage
- Comm

また、これらのオブジェクトとは直接関係のないユーティリティ機能を提供するシェル関数もある。これらの機能を使用することで、堅牢で対話的、かつ洗練されたユーザー・アプリケーションを作成することができます。

ユーザー定義のスクリプトは、デフォルトのスクリプトとして設定することも、ホストユーティリティからのコマンドやバーコードからのコマンドで起動することもできます。実行されたJavaScriptファイル（変数と関数）は、次にリセットされるまで、または別のスクリプトが実行されるまで、アクティブなままであることに注意してください。スクリプトが実行されるたびに、リーダーはJavaScriptエンジンを閉

The Code Corporation • 434 West Ascension Way Suite 300, Salt Lake City, UT 84123 • +1 (801) 495-2200 • FAX +1 (801) 495-0280

じ、リセットし、再起動します。

13 JavaScriptオブジェクトのコード

JavaScriptオブジェクトにはプロパティとメソッドがあります。Codeはプロパティを使用して、ユーザーがリーダー上のJavaScript実装をカスタマイズできるようにします。一方メソッドは、JavaScript エンジンとリーダーファームウェア間の通信を提供します。すべてのカスタムリーダーオブジェクトのプロパティとメソッドは、以下のセクションで説明します。

13.1 Readerオブジェクト

Readerオブジェクトは、選択された Code reader 機能と情報へのアプリケーションソフトウェアアクセスを提供します。

13.1.1 Callbackプロパティ

Callbackプロパティは、JavaScriptの割り当てによって提供される関数です。ほとんどのコールバック・プロパティは、.cra.jsでデフォルトの動作が定義されています。

13.1.1.1 reader.onDecodes(オブジェクト配列のデコード)

`reader.onDecodes`関数は、デコードアクションの完了時にアプリケーションプログラムに処理制御を提供します。`reader.onDecodes(decodes)`の引数 "`decodes` "はデコードオブジェクトの配列です。

```
Example:
reader.onDecodes = function(decodes)
{
    for(var i = 0; i < decodes.length ; ++i)
        comm.sendText(decodes[i].data); // send decoded data to the
host
   return true;
}
```

CR5200にこの関数を実装し、スタンドアロンで年齢検証を行いたい場合は、以下のように個々のデコードオブジェクトに対して`checkSAV`関数を呼び出す必要があります。

```
Example:
reader.onDecodes = function(decodes)
{
    for(var i = 0; i < decodes.length ; ++i)
    {
        if (checkSAV(decodes[i]) == 0)      // perform standalone age
verification
            comm.sendText(decodes[i].data); // send decoded data to the
host
   return true;
}
```


The Code Corporation • 434 West Ascension Way Suite 300, Salt Lake City, UT 84123 • +1 (801) 495-2200 • FAX +1 (801) 495-0280

13.1.1.2 reader.onEvent(イベントオブジェクト) (CR1100, CR1500)

readerオブジェクトのonEventプロパティは、JavaScriptによって処理されるキーのプレス、リリース、ホールドイベントをシミュレートし、発行して処理する方法をユーザーに提供します。onEventにはevent.typeとevent.inputの2つのプロパティを持つイベントオブジェクトが渡されます。typeの値はPRES(1)、HOLD(2)、RELS(3)で、inputの値はTRIG(0)、WAKE(1)、STND(2)、INP0(3)、INP1(4)です。

```
Example:
const TRIG = 0;
const PRES = 1;
reader.onEvent = function(event)
{
    if(event.type == PRES)
        reader.beep();
    return true;
}
```

13.1.1.3 reader.onRawData(data, len)

reader.onRawData プロパティは、ホスト・コンソールから発行された設定コマンドのように、パケット化されていないストリームでデータをリーダーが受信したときに呼び出されます。ストリーム・データは個々の文字とカウントとして提供され、必ずしも完全な文字列として提供されるわけではありません。文字列を組み立てるにはカウントを使用します。ユーザーは、キャリッジリターンが他のすべての制御文字と同様に通常の文字として渡されることに注意しなければなりません。シリアル通信チャンネルは、シリアル・モードのコマンド・チャンネルであることに注意してください。ハンドラからtrueを返す場合、システムがCFGのような通常のシリアル・コマンドを処理することを期待してはいけません。古いコマンドを検出したときにreader.configure(cmd)で新しいコマンドを送信することで、コマンドを監視して置き換えることができます（例えば、8xコマンドを新しい82xコマンドに置き換えます）。

```
Example:
reader.onRawData = function(data, len)
{
    if(len > 0)
        comm.sendText(data);
    else
        return false;
    return true;
}
```

13.1.1.4 reader.onConfigure(string config)

Code Reader の onConfigure プロパティは、リーダーが次の場合に指定された関数を呼び出します：

- 通信ポートから設定コマンドを受信する
- Code Readerによって読み取られたコードから構成コマンドをデコードする

アプリケーションはこのプロパティをイベントハンドラとして次の操作を行います：

- コマンド処理の通知を受信する

コマンドの実行を阻止する

この関数は、reader.configure呼び出しに応答して呼び出されることはありません。

コマンドを処理するようリーダーに指示する場合は false を返し、コマンドを抑制する場合は true を返します。コマンドが抑制された場合、ファームウェアはホストに応答を送信しませんが、JavaScript アプリケーションはホストに独自の応答を提供することができます。

onConfigureプロパティはブロッキング呼び出しであるため、onConfigureプロパティ内で呼び出された場合、reader.tick1Hzやreader.periodicなどの他のタスクの動作は保証されません。

```
Example:

reader.onConfigure = function(confString)
{
    if(confString.match("JSJSG") === null)
        return false;
    return true;
}
```

13.1.1.5 reader.onPowerChange(int mode)

reader.onPowerChange プロパティは、リーダーがアクティブ(0)、スタンバイ(1)、スリープ (2)、およびオフ(3)モード間の変更許可を要求するときに呼び出されます。ユーザーは、"false "を返すことでモードの変更を防ぐことができ、"true "を返すことで変更を受け入れることができます。

```
Example:

reader.onPowerChange = function(mode)
{
   if(mode == 2)
   {
       comm.sendText("No Sleep For You");
       return false;
   }
   return true;
}
```

13.1.1.6 reader.onBatteryChange(void)[i]

reader.onBatteryChangeプロパティはまだ実装されていません。

13.1.1.7 reader.onTrigger(triggerButton, buttonAction)(CR2700)

reader.onTriggerプロパティを使用すると、ユーザーはトリガが押されたことをインターセプトし、トリガイベントを処理するアクションを実行してから、トリガイベントを吸収するか（trueを返す）、JavaScriptがイベントの処理を行った後でリーダーがイベントを処理することを可能にするか（falseを返す）を選択できます。これにより、トリガーボタンが押されるたびに、様々なアクションをプログラムすることができます。

```
Example:

// constants for reader.onTrigger
const mainTrigger  = 0;
const frontTrigger = 1;
const rearTrigger  = 2;
```

```
const triggerPress = 0;
const triggerHold  = 1;   // hold is pressed for 700 mS
const triggerRelease = 2;
reader.onTrigger = function(Trigger, Type)
{
    if( Trigger == frontTrigger )
        reader.configure("CDOPG");
    if(Trigger == rearTrigger && Type == triggerHold)
        reader.configure("CFR");
    return true;
}
```

13.1.1.8 reader.onDecodeAttempt(count)

reader.onDecodeAttempt関数は、有効なデコードが行われなかった場合も含め、デコードが試みられるたびに呼び出されます。この関数は、デコード試行中に成功したデコードの数を表す整数を受け取ります。

```
Example:
reader.onDecodeAttempt = function(mode)
{
   if(count)
       comm.sendText("Successful Decode");
   else
       comm.sendText("No Decode");
}
```

13.1.1.9 reader.periodic(void)

reader.periodic関数のプロパティは、設定された周期レートで呼び出されます。この周期レートはJSPM_ITパラメータで変更可能です。デフォルト値は1Hzです。

```
Example:
reader.periodic = function()
{
    comm.sendText("Some Time Has Passed");
    return;
}
```

13.1.1.10 reader.tick1Hz(void)

reader.tick1Hz関数プロパティは、1秒に1回呼び出されます。これは設定されたレートであり、変更することはできません。

```
Example:
reader.tick1Hz = function()
{
    comm.sendText("Some Time Has Passed");
    return;
}
```

13.1.2 情報プロパティ

情報プロパティは、呼び出されたときにリーダーに関する情報を提供するプロパティです。

13.1.2.1 reader.charging

`reader.charging`プロパティはまだ実装されていません。

13.1.2.2 reader.green

`reader.green`プロパティはまだ実装されていません。

13.1.2.3 reader.amber

`reader.amber`プロパティはまだ実装されていません。

13.1.2.4 reader.red

`reader.red`プロパティはまだ実装されていません。

13.1.2.5 reader.black

`reader.black`プロパティはまだ実装されていません。

13.1.2.6 reader.softwareVersion

`reader`オブジェクトの `softwareVersion` プロパティには、Code readerで現在実行されているファームウェアのバージョン番号を含む読み取り専用の文字列が格納されています。

```
Example:

swVersion = reader.softwareVersion;
comm.sendText("Version X.X: " + swVersion);
```

13.1.2.7 reader.baseModel

`reader`オブジェクトの `baseModel` プロパティには、ロックされたフラッシュメモリのCode readerのベースモデル（"CR1500"）などを含む読み取り専用の文字列が格納されています。

```
Example:
rbmod = reader.baseModel
```

13.1.2.8 reader.model

`reader`オブジェクトの `model` プロパティには、ロックされたフラッシュメモリのCode readerモデル（"2MD0"）などを含む読み取り専用の文字列が格納されています。

```
Example:
rmod = reader.model
```

13.1.2.9 reader.readerId

readerオブジェクトのreaderIdプロパティには、ロックされたフラッシュメモリからのCode reader固有のID（"FFFFFFFFFFFFFF"）などを含む読み取り専用の文字列が格納されています。

```
Example:
rid = reader.readerId;
```

13.1.2.10 reader.hardwareVersion

readerオブジェクトのhardwareVersionプロパティには、Code readerのハードウェア・リビジョン番号（"00"）などを含む読み取り専用 の文字列が格納されています。

```
Example:
hwVersion = reader.hardwareVersion;
comm.sendText("Version XX: " + hwVersion);
```

13.1.2.11 reader.cabled

readerオブジェクトの cabled プロパティには、リーダーが有線か無線かを示す読み取り専用のブール値が格納されています。CR1100 および CR1500 は true を返し、CR2700 は false を返します。

```
Example:
if(reader.cabled == true)
    reader.indicateError();
```

13.1.2.12 reader.locked

注：reader.lockedプロパティはまだ実装されていません。

reader オブジェクトの locked プロパティには、reader がロックされているかどうかを示すブール値が格納されます。また、reader.locked プロパティに有効なロック解除文字列を代入することで、ロック解除もサポートします。

```
Example:
reader.locked = "magic unlock string";
if(reader.locked == true)
     reader.indicateError();
```

13.1.2.13 reader.debug

readerオブジェクトのdebugプロパティには、リーダーの現在のデバッグ・レベルを示す整数値が格納されています。

```
Example:
dbglvl = reader.debug;
reader.debug = 1;
```

13.1.2.14 reader.decodes

readerオブジェクトのdecodesプロパティには、onDecodesコールバックに渡された以前のdecodesオブジェクトに対応するdecodesオブジェクトが格納されています。

```
Example:
dcds = reader.decodes;
```

13.1.2.15 reader.bdAddr

reader.bdAddr プロパティはまだ実装されていません。

13.1.3 Readerメソッド

13.1.3.1 reader.runScript

runScript メソッドは、指定された JavaScript のロード、コンパイル、実行をスケジュールするよう Code reader に指示します。Code reader は、現在実行中のイベントハンドラまたはメインスクリプトが完了した直後に、スクリプトの実行をスケジュールします。runScript メソッドには、呼び出し元のスクリプトに戻るメカニズムは含まれていません。スクリプトを実行すると、現在実行中の JavaScript コンテキストとランタイムが閉じられ、新しいコンテキストとランタイムが開かれます。このとき、それまで実行されていたスクリプトの設定はすべて失われます。複数のスクリプトを一緒に動作させる必要がある場合は、メインのスクリプト内でinclude ("script"); ステートメントを使用してください。

```
Example:
var execute = reader.runScript("codeXml.js");
```

13.1.3.2 reader.configure

configure メソッドは、設定コマンドの実行を Code reader に指示します。これは、readSetting、writeSetting、defaultSettings、および saveSettings に代わるものです。具体的な設定方法については、CR8200 ICD/CCD（D026160）を参照してください。

```
Example:
reader.configure("CDOPPR2") // Sets maximum 2 barcode decodes per read
```

13.1.3.3 reader.beep

beep メソッドは、Code readerにビープ音を鳴らします。ユーザーは、以下の1つ（必須）から4つの引数を指定してコマンドを呼び出すことができます：

reader.beep(numberOfBeeps,beepOnTime,beepOffTime,delayBeforeBeep)。

reader.beepの音量はgoodReadBeepの音量によって制御されるため、読み取り成功のビープ音量を設定するとreader.beepの音量も制御されます。

```
Example:
// beep 6 times, 250mS on, 1S off each beep, no delay before beep
reader.beep(6, 250, 1000, 0); //
reader.configure("FBGRSVO0"); // Set the volume of the beep to 0
```


The Code Corporation • 434 West Ascension Way Suite 300, Salt Lake City, UT 84123 • +1 (801) 495-2200 • FAX +1 (801) 495-0280

```
// beep once with the default
valuesreader.beep(1);
```

13.1.3.4 reader.vibrate

`reader.vibrate`は引数なしで呼び出すことができ、デフォルトのバイブレーションを与え、また、振動パルスの数、オン時間、オフ時間、開始前の遅延 (ミリ秒単位) の値を入力してカスタマイズすることもできます。

```
Example:
// vibrate 3 times, 500mS on, 500mS off, 100mS delay
reader.vibrate(3, 500, 500, 100);

// vibrate once with default values
Reader.vibrate();
```

13.1.3.5 reader.indicateGoodRead

このメソッドは、有効なデコードと同じ表示を行います。デフォルトの動作は、緑のLEDを500ミリ秒点灯させ、ビープ音を1回鳴らします。

```
Example:

reader.indicateGoodRead();
```

13.1.3.6 reader.indicateError

このメソッドは赤のエラーLEDを点灯させ、エラーが発生した場合と同じ表示を行います。デフォルトの動作はビープ音を3回鳴らします。

```
Example:

reader.indicateError();
```

13.1.3.7 reader.shiftJisToUnicode

このメソッドは、有効な shiftJIS 文字列を unicode に変換し、結果の文字列を返します。

```
Example:

myUnicodeString = reader.shiftJisToUnicode(shiftjisStr);
```

13.1.3.8 reader.unicodeToShiftJis

このメソッドは、有効な Unicode 文字列を shiftJIS エンコーディングに変換し、結果の文字列を返します。

```
Example:

uniString = reader.unicodeToShiftJis(unicodeStr):
```

13.1.3.9 reader.indicateData

`reader.indicateData`プロパティはまだ実装されていません。

13.1.3.10 reader.indicateWireless

`reader.indicateWireless`プロパティはまだ実装されていません。

13.1.3.11 reader.indicateBattery

`reader.indicateBattery`プロパティはまだ実装されていません。

13.1.3.12 reader.getKeyboardStatus

このメソッドは、現在のキーボードの状態に対応するバイト値の整数表現を返します。最下位のビットはキーボードの「caps lock」を表します。次のビットはキーボードの「num lock」を表します。

```
Example:
keyStat = reader.getKeyboardStatus()；
```

13.1.3.13 reader.getLastImage

`getLastImage`は、キャプチャエンジンから最後にキャプチャされた画像を取得し、アドレスの上位ワードと下位ローワード、および画像のサイズをjavascriptに返します。これにより、javascriptは画像データの周りにカスタムパケットを作成することができます。エンコードが選択されている場合、画像アドレスはエンコードされた画像のアドレスになります。

```
例
info = reader.getLastImage();
addrH = info.imgHAddress;
addrL = info.imgLAddress;
sizeL = info.imgLSize；
sizeH = info.imgHSize；
```

13.1.4 Reader Decodes オブジェクト

`decodes`オブジェクトは`onDecodes`関数によって提供されます。引数はデコードオブジェクトの配列です（`onDecodes`を参照）。以下のプロパティが提供されます：

13.1.4.1 decodes[i].decoderType

13.1.4.2 decodes[i].valid

デコーダーは、デコードされたデータを有効か無効かマークします。このフラグは、データがすでに処理されたか無視されたかをシステムに通知するために使用することができます。

```
Example:
var validDecode = decodes[i].valid；
```

13.1.4.3 decodes[i].x

`decode.x`プロパティは読み取り専用のプロパティで、リーダーによってキャプチャされ、デコードエンジンによって解析された画像全体に対する、デコードされたバーコードの水平位置をピクセル単位で定義します。

decode.xの値は、0が画像の中心であることに基づいて、正または負の値になります。

```
Example:
var hPos = decodes[i].x;
```

13.1.4.4 decodes[i].y

decode.yプロパティは読み取り専用のプロパティで、リーダーによってキャプチャされ、デコードエンジンによって解析された画像全体に対する、デコードされたバーコードの垂直位置をピクセル単位で定義します。
decode.yの値は、0 が画像の中心であることに基づいて、正または負の値になります。

```
Example:
var vPos = decodes[i].y;
```

13.1.4.5 decodes[i].bounds

bounds プロパティは、最後にデコードされたバーコードの位置に関する情報を与える 4 つの x 座標と y 座標のセットの配列です。4つの点（x, y）は、キャプチャされた画像の左上（0, 0）の位置からバーコードの4隅の位置をピクセル単位で整数として示します。リーダーによっては、CortexToolsで表示するときに画像が90°回転していることがあります。

//画像内のバーコードの右上隅のx座標とy座標：
pos.tR = (decode.bounds[0].x, decode.bounds[0].y)

//画像内のバーコードの左上隅のx座標とy座標：
pos.tL = (decode.bounds[1].x, decode.bounds[1].y)

//画像内のバーコードの左下隅のx座標とy座標：
pos.bL = (decode.bounds[2].x, decode.bounds[2].y)

//画像内のバーコードの右下隅のx座標とy座標：
pos.bR = (decode.bounds[3].x, decode.bounds[3].y)

13.1.4.6 decodes[i].time

decode.timeプロパティは、解析されたバーコードをデコードするためにデコードエンジンが要した時間をミリ秒単位で定義する読み取り専用のプロパティです。

```
Example:
var decTime = decodes[i].time;
```

13.1.4.7 decodes[i].symbology

decode.symbology プロパティは、Code Corporation によって割り当てられたシンボロジーインデックスを含む読み取り専用のプロパティです。symbology の有効な値は、付録 5.3 で定義されています。

```
Example:
var symbology = decodes[i].symbology;
```

13.1.4.8 decodes[i].quality_percent

decode.quality_percentプロパティは、キャプチャされた画 像を解析する際に、デコードエンジンによって決定される内部定義の画質値を定義する読み出し専用のプロパティです。
これは、印刷または表示されたバーコードの品質ではなく、デコードに使用するためにキャプチャされた画像の品質です。

```
Example:

var quality_percent = decodes[i].quality_percent;
```

13.1.4.9 decodes[i].aimSymbology

プロパティは読み取り専用のプロパティで、デコードされたバーコードのデコードエンジンによって決定されたAIM（Automatic Identification and Mobility）シンボロジーの最初の文字を示します。AIM 規格の詳細については、Association for Automatic Identification & Mobility のウェブサイトを参照してください。

13.1.4.10 decodes[i].symbologyModifier

decode.symbologyModifier プロパティは、デコードエンジンによってデコードされたバーコードのシンボロジー修飾子情報を含む読み取り専用プロパティです。symbologyModifier の有効な値は、古い独自のデコーダーに基づいており、付録で定義されています。値は、UPC を除くすべてのシンボルに対する aimModifier と同じです。

```
Example:

var symbologyModifier = decodes[i].symbologyModifier;
```

13.1.4.11 decodes[i].aimModifier

decode.aimModifier プロパティは、デコードされたバーコードのデコードエンジンによって決定された AIM（自動識別とモビリティ）修飾子を与える読み取り専用のプロパティです。AIM 規格の詳細については、Association for Automatic Identification & Mobility の Web サイトを参照してください。

aimModifierの有効な値は5.3節で定義されています。

13.1.4.12 decodes[i].linkage

decode.linkage プロパティは読み取り専用のプロパティで、複合バーコードがデコード エンジンによってデコードされた場合に、複合バーコードを構成するセグメント間のリンク コードに関する情報を提供します。デコードされたバーコードが複合バーコードでない場合、このプロパティは NULL です。

```
Example:

var link = decodes[i].linkage;
if( link === null ) print("not compositen");
```


The Code Corporation • 434 West Ascension Way Suite 300, Salt Lake City, UT 84123 • +1 (801) 495-2200 • FAX +1 (801) 495-0280

13.1.4.13 decodes[i].saPosition

実用的な大きさのメッセージを1つのシンボルで扱うために、データ・メッセージを複数のQRコード・シンボルに分散させることができます。より多くのデータを伝えるために、最大16個のQRコード・シンボルを構造化形式で付加することができます。シンボルが構造化アペンドの一部である場合、バーコードテキスト内の構造化アペンドブロックによって示されます。

構造化アペンド形式のQRコードシンボルセット内のシンボルの位置インデックスは、文字列形式で1～16（境界を含む）の整数です。

```
Example:
var numSplits = decodes[i].saTotal;
var idx       decodes[i].saPosition;
print("this is " + idx + " of " + numSplits + "\n");
```

13.1.4.14 decodes[i].saTotal

構造化アペンド形式のQRコードシンボルセット内のシンボルの総量で、文字列形式で 2 から 16 (境界を含む) までの整数です。

```
Example:
var numSplits = decodes[i].saTotal;
```

13.1.4.15 decodes[i].saParity

Parity Date：元の入力データをパラメータとして、`GetParity` メソッドを使用して得られる 8 ビットのバイト値です。これは、構造化アペンド内のすべてのシンボルで同一であり、読み取られたすべてのシンボルが同一の構造化アペンドメッセージの一部であることを検証できます。

```
Example:
var parity = decodes[i].saParity;
```

13.1.4.16 decodes[i].isConfig

デコーダーは、コンフィギュレーション・コードの書式に一致するデコードデータを`isConfig` trueでマークします。その後、JavaScript はコードを処理するかしないかを決定しなければなりません。

```
Example:
var isConfigCode = decodes[i].isConfig;
```

13.1.4.17 decodes[i].decodeOutputFormat

`decode.decodeOutputFormat`プロパティはJavaScript が期待する出力のタイプを指定する読み取り専用プロパティです。

値	定義
0	生出力：デコーダーによるフォーマットなし
1	顧客フォーマットデータ：これは、顧客が提供したデータから適用されるフォーマットです。 .parseファイル。これらのファイルは、CortexToolsのDL/IDタブを使用して生成できます。

値	定義
2	JSON形式のデータ：JavaScript Object Notation（JSON）でフォーマットされたデータ。現在、デコーダーがJSONでフォーマットできるのは、DL/IDデータのみです。JSONの詳細については、JSON.orgを参照してください。

13.1.4.18 decodes[i].data（文字列型）

`decodes[i].data` プロパティは、デコードされたバーコードのペイロードを表す読み取り/書き込み値です。このデータには、ユーザーがデコードエンジンに行った設定に基づいて、データ処理、チェックサム情報、またはデータフォーマットが含まれることもあります。デコードエンジンに特別な処理設定が適用されていない場合、この情報は、バーコードが印刷または表示される前にバーコードのエンコードに使用された文字列と一致する必要があります。

```
Example:
var decodeText = decodes[i].data;
```

13.2 Storageオブジェクト

ストレージオブジェクトはアプリケーションソフトウェアにCode readerのファイルストレージへのアクセスを提供します。ファイルは`storage.write`メソッドやホストからのダウンロードによってストレージに書き込まれます。

注：ファイル名は、最大200文字の印刷可能なASCII文字を使用できます。ホスト・ファイル・システムとの互換性のため、Codeではホスト・オペレーティング・システムで予約されている文字(/、¥、:、?、*、[、]、'、" など)は使用しないことを推奨しています。ファイルは、Unicode 文字をそれぞれ 1 バイト以上でエンコードするUTF8 形式で保存されます。ホストから送られてくるマルチバイト文字を使用するファイルは、保存されたファイルがUTF-8で正しく処理されるように、すべてUTF-8でエンコードされていなければなりません。JavaScriptはファイルシステムからこれらのファイルを読み込むときにUTF-16に変換します。

13.2.1 情報プロパティ

13.2.1.1 storage.isFull

`storage.isFull`プロパティは読み取り専用のブール値で、ストレージが満杯で追加できない場合はtrue、そうでない場合はfalseとなります。

13.2.1.2 storage.fullness_percent

`storage.fullness_percent`プロパティは、使用中のストレージのパーセンテージを示す読み取り専用の整数です。

13.2.2 Storageメソッド

以下のセクションでは、Code readerストレージ・オブジェクトに定義されているメソッドについて説明します。

このセクションでは、タイムカードの記録が従業員番号ごとに整理されたファイルとして管理されていることを前提としたタイムカードアプリケーションの要素を使用します。タイムカードのレコードの命名規則は TimeCard<employee_number> です。

13.2.2.1 storage.read

`storage.read`メソッドはファイルを読み込みます。このファイルはJavaScriptで使用するためにUTF-16に変換されます。

```
Example:
var timeInfo = storage.read("TimeCard1");
```

13.2.2.2 storage.write

`storage.write`メソッドはファイルをストレージに書き込みます。ファイルが存在しない場合は、Code readerがファイルを作成します。同じ名前の既存のファイルがあった場合は、それを置き換えます。ファイルにマルチバイト文字が含まれている場合は、UTF-8でエンコードする必要があります。

```
Example:
var employee57Info;
storage.write("TimeCard57", employee57Info);
```

13.2.2.3 storage.append

`storage.append`メソッドは、ファイルの最後にデータを追加します。

```
Example:
var extraInfo = " extra info";
storage.append("timeCard57", extraInfo);
```

13.2.2.4 storage.rename

`storage.rename`メソッドはファイルの名前を変更します。

```
Example:
storage.rename("newEmployeeTime", "timeCard58");
```

13.2.2.5 storage.erase

`storage.erase`メソッドはファイルを消去します。

```
Example:
storage.erase("newEmployeeTime");
```

13.2.2.6 storage.uploadFile

`storage.uploadFile`メソッドは、正規表現に一致するファイル名を、現在アクティブなホストの通信ポート経由でホストにアップロードします。

```
Example:
storage.uploadFile("timeCard58");
```

13.2.2.7 storage.defragment

`storage.defragment`は、分割または分散されたファイルブロックを、可能な限り連続したブロックにまとめます。

Example:

```
storage.defragment();
```

13.2.2.8 storage.findFirstInternal

storage.findFirstInternalメソッドは、ファイルシステム内の最初のファイルを見つけます。

Example:

```
var name = storage.findFirstInternal();
```

13.2.2.9 storage.findNextInternal

storage.findNextInternalメソッドは、前回のstorage.findFirstInternal、storage.findFirst、storage.findNextInternal、またはstorage.findNextの呼び出しの後に、ファイルシステム内の次のファイルを見つけます。

Example:

```
var name = storage.findNextInternal();
```

13.2.2.10 storage.findFirst

storage.findFirstメソッドは、呼び出しパラメータで指定された正規表現と名前が一致する最初のファイルを見つけます。

Example:

```
var name = storage.findFirst(regex);
```

13.2.2.11 storage.findNext

storage.findNextメソッドは、前のstorage.findFirst呼び出しのexpressionパラメータで指定された正規表現と名前が一致する次のファイルを探します。一致する名前は順序付けられませんが、重複することはありません。findFirstNative-findNextNativeのシーケンスは、他のストレージメソッド呼び出しが介在しない限り、一致するすべてのファイルを返します。ファイルを配列に入れ、並べ替えが必要な場合はJavaScriptのソートメソッドを使うことができます。

Example:

```
var name = storage.findNext();
```

13.2.2.12 storage.size

storage.sizeメソッドは、ファイルのサイズをバイト単位で返します。

Example:

```
var fileSize = storage.size("timeCard57);
```

13.2.2.13 storage.getHeader

storage.getHeaderメソッドは、JavaScriptファイルの最初の複数行コメントブロックを返します。適切な開発者キーがインストールされていれば、暗号化されたファイルも含まれます。

```
Example:

var headerInfo = storage.getHeader("timeCard57");
```

13.2.2.14 storage.listFiles

`storage.listFiles` メソッドは、何も入力されなければ、ファイルシステム上のファイル名を含むテキスト文字列を返します。このメソッドにテキスト文字列が入力として与えられると、与えられた文字列と一致する部分文字列を持つすべてのファイルをリストします。

```
Example:

var allFilesonFS = storage.listFiles();
var filesThatContainFile = storage.listFiles("File");
```

13.2.3 Code readerローカルストレージのデータ

アプリケーション開発環境は、ストレージオブジェクトを通してCode readerのローカル・ストレージへのプログラム・アクセスを提供します。データはファイルと呼ばれる名前付きオブジェクトとしてストレージに保持されます。CortexTools2 はホストデータをCode reader ファイルに転送できます。Code reader アプリケーションは、データをファイルに保存することもできます。

Code reader のファイル名には、印刷可能な ASCII 文字を使用することができます。

ファイルを管理するには、ストレージオブジェクトの消去メソッドと書き込みメソッドを使用します。ファイルを見つけるには、`findFirst`メソッドと`findNext`メソッドを使用します。ファイルにアクセスするには `read` メソッドを、ホストに送信するには `upload` メソッドを使用します。

注：`storage.findFirst`と`storage.findNext`は、Codeによってライブラリファイル.jselib.jsに実装されています。スキャナは`storage.findFirstInternal`関数と`storage.findNextInternal`関数を提供します。

13.3 Commオブジェクト

Code Reader アプリケーション開発環境では、ホスト常駐アプリケーションとの通信をサポートするホスト通信 Comm オブジェクトを定義しています。例えば、CortexTools2（セクション6）は、Code reader にファイルをダウンロードするためにCode reader と通信するホスト常駐ユーティリティです。

ホストコンピュータから見ると、Code readerはシリアルポートまたはUSBポートからアクセスできるシリアルデバイスです。Code readerの設定により、アクティブなホスト通信ポートが定義されます。JavaScriptプログラムは、Code Rreader commオブジェクトによって定義されたメソッドを使用して、Code reader ホスト通信ポートに書き込むことにより、ホストにデータを転送します。Code readerのホスト通信実装は、生テキストとパケットという2つの基本的な通信スタイルをサポートしています。Code reader ホスト通信実装は、ネイティブプロトコルのセットもサポートしています。

13.3.1 情報プロパティ

13.3.1.1 comm.isConnected

`comm` オブジェクトの `isConnected` プロパティには、ホストの接続状態を示す読み取り専用のブール値が含まれます。可能な接続値は以下のとおりです：

True：リーダーはホストに接続されています。

The Code Corporation • 434 West Ascension Way Suite 300, Salt Lake City, UT 84123 • +1 (801) 495-2200 • FAX +1 (801) 495-0280

False：リーダーはホストに接続されていません。

```
Example:
if( comm.isConnected == false )
  comm.connect;
```

13.3.2 Commメソッド

13.3.2.1 comm.connect

connect メソッドは、Code reader通信ドライバに接続の確立を試みるよう指示します。

注：有線製品には適用されません

13.3.2.2 comm.disconnect

disconnectメソッドは、Code reader通信ドライバにホストからの切断を指示します。

注：有線製品には適用されません

13.3.2.3 comm.sendPacket

sendPacket メソッドは、Code reader の通信設定によって指定された通信ポートを介して、データパケットをホストに送信します。Code reader は、アクティブな Code reader のパケットプロトコル設定に従ってフォーマットされたパケットを作成します。

データパケットについては、http://www.codecorp.com からダウンロードできるコード・インターフェイス設定資料を参照のこと。

```
Example:
var myPacket = "my data packet";
comm.sendPacket(myPacket);
```

13.3.2.4 comm.sendText

sendText メソッドは、Code readerがアクティブな通信ポートを介してホストへテキスト（NULL 文字を含む場合があります）を送信するよう指示します。

```
Example:
var myPacket = "my data packet";
comm.sendText(myPacket);
```

13.3.2.5 comm.sendData

sendDataメソッドは、アドレスポインタから指定されたバイト数をcommHostに送信します。これはフィルタリングされていない生のデータとして扱われるため、文字以外のデータが含まれることがあります。

Example:

The Code Corporation • 434 West Ascension Way Suite 300, Salt Lake City, UT 84123 • +1 (801) 495-2200 • FAX +1 (801) 495-0280

```
var addressHigh = 0x5038;
var addressLow = 0x0200;
var sizeH = 0x0002；
var sizeL = 0x0192；
comm.sendData(addressHigh、addressLow、sizeH、sizeL)；
```

13.3.2.6 comm.sendMessage

protocol_sendMessage を使ってホストにメッセージを送信します。

```
Example:
var myMsg = "my data message"；
comm.sendMessage(myMsg)；
```

13.3.2.7 comm.getSendStatus

`comm.getSendStatus`プロパティはまだ実装されていません。

13.3.2.8 comm.sendUsbScanCode

`comm.sendUsbScanCode`メソッドは、リーダーがキーボードモードのときのみ利用可能で、キーボードから送信されたかのように、ユーザーがキーボード・スキャンコードをホストに送信できるように設計されています。各スキャンコードは8バイトのデータで構成され、キーボードに設定された言語に対応するか、構成設定によってユーザーによって選択されます。

```
Example:
comm.sendUsbScanCode(0,0,0,0,0,0,0,0)；which represents a key release.
```

13.3.2.9 comm.sendUsbScanCodes

`comm.sendUsbScanCodes`メソッドは、リーダーがキーボードモードのとき、ユーザーが一度に複数のスキャンコード（13.3.2.7参照）をホストに送信することを可能にします。これは、大文字、または一緒に送信される制御文字シーケンスを表します。これらのスキャンコードは、キーボードモードのリーダーによって送信されるスキャンコード用に設定されたスキャンコード遅延を使用します。8バイトの各グループは、キーボードと同じように、HIDレポートとしてホストに送信されます。ユーザーが大文字の「a」を送信したい場合、最初の8バイトはシフトキーだけを押した状態で送信し、次の8バイトはシフトキーと「a」キーを押した状態でを送信し、最後にすべての値がキーリリースを「0」である8バイトのグループを送信します。

```
Example:
comm.sendUsbScanCodes(2,0,0,0,0,0,0,0,2,0,4,0,0,0,0,0,
0,0,0,0,0,0,0,0);
```

13.3.2.10 comm.sendHidReport

`comm.sendHidReport`はまだ実装されていません。

13.4 シェル関数

13.4.1 help(*[name]*)

`help` 関数は利用可能なシェルコマンドを表示します。オプションで、`[name]` に任意のシェルコマンドを指定すると、`[name]` の詳細情報が表示されます。

```
Example:
help(include);
```

13.4.2 include(name)

`include`関数は、インクルードされたスクリプトをインラインで実行します。

```
Example:
// Adds the definitions in myScript.js to the application.
// The definitions become part of the "including" script.
include("myScript.js");
```

13.4.3 print (string) – see reader.print(string)

`print`関数は`reader.print(string)`メソッドのエイリアスです。

```
Example:
print("some random text");
```

13.4.4 gc()

`gc`関数は、割り当て済みだがランタイム環境では不要になったメモリーをクリーンアップします。この関数はプロセッサに負荷がかかるため、使用するとパフォーマンスが低下する可能性があります。

```
Example:
gc();
```

13.4.5 sleep_ms(time_ms)

`sleep`関数は、time_msのミリ秒数だけインラインでブロック遅延を行います。

```
Example:
sleep_ms(1000);
```

13.4.6 format(string format, ...)

`format`は、フォーマット文字列を引数として、フォーマットに適用する引数リストを受け取り、フォーマットされたデータを戻り値としてJavaScriptに送り返します。

```
Example:
{
    var myString = "";
    format("This is int %d and this is string %s", 274, "Two
        hundred seventy four", &myString);
    print(myString);
}
```

13.4.7 wdt_pet()

長時間の処理では、操作中にファームウェア・ウォッチドッグがタイムアウトすることがあります。プロセッサに負荷のかかる操作中にウォッチドッグがタイムアウトすると、リーダーはリブートし、エラーログにエラーが記録されます。プロセッサを集中的に使用するセクションでリーダーが再起動するのを防ぐには、ファームウェア ウォッチドッグ タイマーを調整する必要があります。`wdt_pet()`は、ウォッチドッグタイマーがタイムアウトするまでの秒数を示す引数を取ります。

```
Example:
wdt_pet(1);
```

13.4.8 logMessage(string)

`logMessage`関数は、与えられた文字列をログに書き込みます。

```
Example:
logMessage("Writing to log");
```

13.5 定義済みスクリプト

Code readerコアJavaScriptアプリケーションは、最も一般的なタスクを処理するためのシンプルな方法を提供します。この方法では、プログラミング知識がほとんどない人でも、JavaScriptのサブセットと正規表現を使って、ニーズに合わせて製品をカスタマイズすることができます。正規表現は、.matchや.replaceのようなJavaScriptの文字列メソッドと組み合わせることで、強力なデータ操作になります。

JavaScriptが有効な場合、リーダーはまずデフォルトのライブラリファイル「.jseLib.js」をロードします。JavaScriptは次に「include(.default.js);」というinclude文を含む「.startup.js」を読み込みます。デフォルトの「.default.js」ファイルは「include(.cra.js)」となります。これらのファイルは、リーダーに基本的なJavaScript機能を提供します。

13.5.1 .jseLib.js ライブラリ関数

13.5.1.1 storage.findFirst(regex)

`storage.findFirst(regex)`の実装は、すべてのファイルをスキャンして正規表現に一致するファイルを探し、最初に一致したファイルで停止します。

13.5.1.2 storage.findNext()

`storage.findNext()`の実装は、前回の`storage.findFirst()` / `storage.findNext()`から残っているすべてのファイルをスキャンし、`storage.findFirst(regex)`で確立された正規表現に一致するファイルを探し、次の一致で停止します。

13.5.1.3 reader.setInterval (func, interval)

setIntervalメソッドは、秒単位で指定された間隔で関数を呼び出すか、式を評価します。setIntervalメソッドは、clearIntervalが呼び出されるまで関数を呼び出し続けます。

setIntervalによって返されたID値は、clearIntervalメソッドのパラメータとして使用されます。

```
Example:
intervalId = reader.setInterval(function, interval_sec);
```

13.5.1.4 reader.clearInterval(intervalId)

clearIntervalメソッドは、ハンドルintervalIdを持つsetIntervalのインスタンスを削除します。

```
Example:
reader.clearInterval(intervalId);
```

13.5.1.5 reader.setTimeout (func, delay)

reader.setTimeout() メソッドは、指定された秒数の遅延後に関数を呼び出します。この関数は、コード定義済みオブジェクト・メソッドにすることはできません。

reader.setTimeout()によって返されたID値は、reader.clearTimeout() メソッドのパラメータとして使用されます。

```
Example:
timeoutId = reader.setTimeout(function, timeout_sec);
```

13.5.1.6 reader.clearTimeout(timeoutId)

clearTimeoutメソッドは、ハンドルtimeoutIdを持つsetTimeoutのインスタンスを削除します。

```
Example:
reader.clearTimeout(timeoutId);
```

13.5.1.7 reader.tick1Hz() - デフォルトの実装

13.5.1.8 reader.now() (CR5200)

ハードウェアがサポートしていれば、1970年1月1日午前0時からの現在時刻を秒単位で返します。

13.5.1.9 checkSAV(decode) (CR5200)

CR5200 のスタンドアロン年齢検証は、reader.onDecodes 関数で実行されます。この関数がユーザーによって実装（オーバーライド）されている場合、checkSAV を呼び出してスタンドアロン年齢検証を手動で実行することができます。個別のデコードオブジェクトを渡す必要があります。checkSAV関数は、デコードが運転免許証でないか、ハードウェアが機能をサポートしていない場合、値0（ゼロ）を返します。デコードが有効な運転免許証であり、ハードウェアがその機能をサポートしている場合、値1（1）が返されます。エラーがあった場合は、-1（マイナス1）が返されます。

13.5.2 .cra.js

Codeは、JavaScriptに特別な機能を実装することで、デフォルトのリーダー動作を提供します。これらの特別な機能は「.cra.js」にあります。

13.5.2.1 reader.onEvent(event)

この関数は、関数変数 "rules_onEvent "をチェックし、それが "null "でなければ、その関数を呼び出します。

rules_onEvent "がfalseを返した場合、その関数が呼び出しを処理します。

13.5.2.2 reader.onDecodeAttempt(count)

この関数は、関数変数 "rules_onDecodeAttempt "をチェックし、それが "null "でなければ、その関数を呼び出します。

13.5.2.3 reader.onDecode(decode)

この関数は、関数変数 "rules_onDecode "をチェックし、それが "null "でなければ、その関数を呼び出します。

rules_onDecode "がfalseを返さなければ、その関数がデコードを処理します。

13.5.2.4 reader.onDecodes(decodes)

この関数は、関数変数 "rules_onDecodes "をチェックし、それが "null "でなければ、その関数を呼び出します。rules_onDecodes "がfalseを返さず、"rules_onDecodeAttempt "がfalseを返さない場合、この関数は、decodes配列内の各デコードオブジェクトに対して`reader.onDecode()`を呼び出してデコードを処理します。

13.5.2.5 reader.onConfigure(data)

この関数は、関数変数 "rules_onConfigure "をチェックし、それが "null "でなければ、その関数を呼び出します。rules_onConfigure "がtrueを返した場合、この関数は直ちに戻ります。"rules_onConfigure "がfalseを返した場合、この関数はコンフィギュレーション・データの処理を試み、処理されればtrueを返し、そうでなければfalseを返します。

13.5.3 .codeRules.js

Codeでは、「.codeRules.js 」ファイルを定義する機能があり、このファイルでリストアップされたメソッドの機能を指定することができます。

13.5.3.1 rules_onDecodes

decodes配列オブジェクト（セクション13.1.4）は関数に渡され、関数によって返されることもあります。関数によって論理値 false が返された場合、リーダーはそれ以上のデコード処理を行いません。Code readerは、バーコードが見つかるとすぐに、読み取りが成功したことを最初に示します。

rules_onDecodes() 関数は、 decodes 配列オブジェクトか値 'false' のいずれかを返さなければなりません。 false' を返すと、reader.onDecodes() 関数による処理を停止します。

この例では、正規表現を使ってデコード文字列の大文字'A'を小文字'a'に置き換えています。Decodeオブジェクトのプロパティについてはセクション4で説明します。

```
Example:
rules_onDecodes = function(decodes)
{
    decodes[0].data.replace(/A/g, "a");
    return decodes;
};
```

この例では、デコードデータの特定のパターンに一致する正規表現を使用し、一致したデコードデータから最後の桁を削除するJavaScriptメソッドを使用しています。

```
Example:
rules_onDecodes = function(decodes)
{
    if(decodes[0].data.match(/^[0-9]{9}$/g)!= null)
                decodes[0].data =
                decodes[0].data.substring()
            0, decodes[0].data.length - 1);
    return decode;
};
```

13.5.3.2 rules_onDecodeAttempt

rules_onDecodeAttempt() 関数は、デフォルトのreader.onDecodeAttempt() が呼び出されたときに実行されます。rules_onDecodeAttempt() には、decodes配列の長さが渡されます（セクション13.1.1）。

13.5.3.3 rules_onDecode

rules_onDecode() 関数は、デフォルトのreader.onDecodes() が呼び出されたときに実行されます。rules_onDecode() には個々のデコードオブジェクトが渡され、decodes配列に含まれる各デコードオブジェクトに対して呼び出されます。

rules_onDecode() 関数は、デコードオブジェクトか値"false "を返します。false "を返すと、reader.onDecode() 関数はそれ以上の動作を停止します。

13.5.3.4 rules_onEvent

rules_onEvent() は、userEventが利用されるたびに呼び出されます。ユーザー・イベント番号はevent_user0からevent_user9に制限されます。最も一般的な使用方法は、ボタン押下に基づいて何らかのロジックを実装すること、または読み取り失敗通知を送信することです。ユーザーは以下のイベントのいずれかを事前に処理することができます。"false"を返すと、ファームウェアはそのイベントを処理することができます。一方、"true "を返すと、ファームウェアはそのイベントが処理され、それ以上処理する必要がないことを認識します。

以下の例では、リーダーはまずボタン押下をインターセプトするように設定されています。次にリーダーは、codeXml enter キー入力を通信ポート経由で送信します。詳細については、CR8200 CCD、コード文書番号 D026160 を参照してください。

```
Example：

var enter = "\x01Xxx1ean//nxxx04";
rules_onEvent = function(event)
{
    if (event.type == buttonPress)
    {
        comm.sendPacket(enter)；
        return true; // tell the firmware that we handled the press
    }
};
```

13.5.3.5 rules_onConfgure(data)

コンフィギュレーション・コードが読み込まれると、`rules_onConfigure()` がコンフィギュレーション・コードの内容をデータとして呼び出されます。設定データが `rules_onConfigure()` で処理される場合、値 "true" が返され、`reader.onConfigure()` によるそれ以降の動作が停止します。

以下の例では、`rules_onConfigure()`は`jsSettings.parseUserParams()`を呼び出して、JavaScript のユーザー・パラメーター・コマンドのためにコンフィギュレーション・データをスキャンし、コンフィギュレーション・データが正常に処理された場合は "true "を、そうでない場合は "false "を返します。

```
Example:
include(".settings.js")；
rules_onConfigure = function(data)
{
    if( jsSettings.parseUserParams )
        return jsSettings.parseUserParams(data)；
    return false；
};
```

13.5.3.6 Optional Global variables

writerは、あらかじめ定義された関数の外部のグローバル変数をJavaScriptルールファイルに含めることができます。これらの変数は、JavaScriptエンジンが起動し、readerが起動したときにインスタンス化されます。

13.5.4 .setttings.js

Codeは、アプリケーションに".settings.js "を含めることで、永続的なユーザー設定を保存する機能を提供します。コンフィギュレーション・バーコードを作成することができますが、以下のフォーマットと一致する必要があります：

<JS識別子> JSUP **<P | S | R | G>** **<PP>** **<Val>**,

<JS識別子>: このコマンドが JavaScript によって排他的に処理されることを示す識別子: ¥x1

The Code Corporation • 434 West Ascension Way Suite 300, Salt Lake City, UT 84123 • +1 (801) 495-2200 • FAX +1 (801) 495-0280

JS: 主要カテゴリー

UP: サブカテゴリー

<P｜S｜R｜G>： アクション：プット、セット、リセット、ゲット

<PP>: 2文字のパラメータID

<Val>: 引用（オプション）文字列

```
Example 1：
\x1bJSUPSP01 ≒ JSUPSP0"1"
Example 2：
\ x1bJSUPSP1”exampleString” ≈ \x1bJSUPSP1exampleString
```

13.5.4.1 jsSettings.def(param, val)

パラメータ *param* にデフォルト値 *val* を設定します。

これは、パラメータが初めて参照される前に行う必要があります。

つまり、`jsSettings.set()`/jsSettings.`put()`/jsSettings.`reset()`/jsSettings.`get()`を呼び出す前に、`jsSettings.def()`を呼び出す必要があります。これは、設定ファイルが空のときに、実行時の設定リストに適切なデフォルト値が含まれるようにするために重要です。

13.5.4.2 jsSettings.put(param, val)

パラメータ *param* に値 *val* を設定し、RAM に保存します。これは一時的な設定であり、再起動しても持続しません。

13.5.4.3 jsSettings.set(param, val)

パラメータparamに値*val*を設定し、RAMの設定を更新し、リーダー上の'.settings.txt'ファイルに保存します。これは永続的なセットであり、再起動後も保持されます。

13.5.4.4 jsSettings.reset(param)

*param*を`jsSettings.def()`で設定したデフォルト値に戻します。

13.5.4.5 jsSettings.get(param)

パラメータ *param* の現在値を取得します。これによりRAMに格納されている現在値を取得します。

この例では、正規表現を使ってデコード文字列の大文字'A'を小文字'a'に置き換えています。Decodeオブジェクトのプロパティについてはセクション4で説明します。

```
Example 1：
include(".settings.js")；
function triggerDecodeNormal()
{
```


The Code Corporation • 434 West Ascension Way Suite 300, Salt Lake City, UT 84123 • +1 (801) 495-2200 • FAX +1 (801) 495-0280

```
    P0_intervalDelay_sec = jsSettings.get("P0");
    if( P0_last != P0_intervalDelay_sec )
    {
        P0_last = P0_intervalDelay_sec;
        reader.clearInterval(intervalId);
        intervalId = reader.setInterval(triggerDecodeNormal,
          P0_intervalDelay_sec);
    }
};

rules_onConfigure = function(data)
{
    if( jsSettings.parseUserParams )
        return jsSettings.parseUserParams(data);

    return false;
};

function initSettings()
{
    jsSettings.def("P0", 3);
}

// Initialize any user parameters that are used for this application
// Note: This must be called before the first call to jsSettings.get()
initSettings();

// Get the current value of applicable user parameters
var P0_intervalDelay_sec = jsSettings.get("P0");

// Keep a copy of the last parameter value for detecting a parameter
change
var P0_last = P0_intervalDelay_sec;

// Trigger a decode on the interval defined by user parameter P0
var intervalId = reader.setInterval(triggerDecodeNormal,
P0_intervalDelay_sec);
```


The Code Corporation • 434 West Ascension Way Suite 300, Salt Lake City, UT 84123 • +1 (801) 495-2200 • FAX +1 (801) 495-0280

14 付録

14.1 キー入力の送信 (CodeXml)

Code reader製品は、多くの場合、キーボード入力を使ってPCに接続されます。バーコードに含まれるデータは、単にPCアプリケーションに「入力」されます。また、「enter」キーのような特定のキーをアプリケーションに送信する必要があることもよくあります。enterキーは、ASCIIのキャリッジリターン（0x13）とは異なることに注意してください。

Enter接尾辞を追加するには、以下の書式を使用することができます。/nはEnterキーを表します。/nの代わりに使用できるキーの一覧を以下に示します。

enter = "¥x01Y¥x1ean/2F/2Fn¥x04";
decode.data = decode.data + enter;

文字	キー
/a	Alt
/g	AltGr(Alt Grave；右Alt)
/c	Ctrl
/m	Menu
/s	Shift
/w	Windowsロゴ
/u	上矢印
/l	左矢印
/r	右矢印
/d	下矢印
/t	Tab
/z	Delete
/e	Esc
/n	Enter
/v	End
/b	Backspace
/i	Insert
/p	Page up
/x	Page down
/h	Home
/,	500 ms delay

文字	キー
/0 - /9	テンキー
/f1～/f12	ファンクションキー
//	/
/k	キーボード スキャンコード

上記のキーストローク表現を使ってキーボードのキーストロークを送信するだけでなく、CodeXMLには、非ASCII文字のキーボード上の正確なキーを識別するためのUSBスキャンコード(/k)を送信する機能もあります。

このような使用例として、一部の言語キーボード（例えばイタリア語）では、左の「Alt」キーを「Alt」、右の「Alt」キーを「AltGr」とラベル付けし、キー、Shift+キー、AltGr+キー、さらにはAltGr+Shift+キーだけに基づいて、キーストロークに対して異なる言語文字を入力します。CodeXMLを使用してAltGr（右Alt）のスキャンコードを識別すると、リーダーはAltGrとキーのスキャンコードを送信することにより、AltGr（Alt Grave）が押されたときにのみ使用可能な言語文字を送信することができます。

USB スキャン コードは「修飾子」を提供します。つまり、Ctrl、Shift、Alt、AltGr、および/またはMeta/GUI（たとえば「Windows」）キーが通常のキーと同時に押され、キーのキーストロークが「変更」されているかどうかを示します。例えば、スキャンコードを使用して「a」文字だけを送信するには、「a」キーのスキャンコード（0x04）を修飾子なし（0x00）で送信する必要があります。しかし、「A」文字を送信するには、「a」キーのスキャンコードに「Shift」修飾子（0x02（左Shift）または0x20（右Shift））を付けて送信する必要があります。

以下の表は、「修飾子」キーの2桁の16進数表現を示しています。

キー	修飾子
左 Ctrl	0x01
左 Shift	0x02
左 Alt	0x04
左 Meta/GUI	0x08
右 Ctrl	0x10
右 Shift	0x20
右 Alt (AltGr)	0x40
右 Meta/GUI	0x80

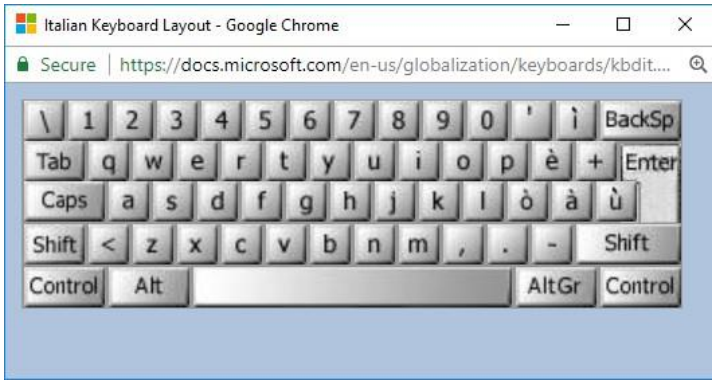
修飾子キーは、値を論理和することで組み合わせることができます。例えば、左Shift + 右Alt = 0x42です。

スキャンコードを送信するためのCodeXMLの構文は、CodeXMLヘッダーの後に"/k "を続け、その後にそれぞれ修飾子とキースキャンコードを示す2桁の16進数値を2つ続けます。

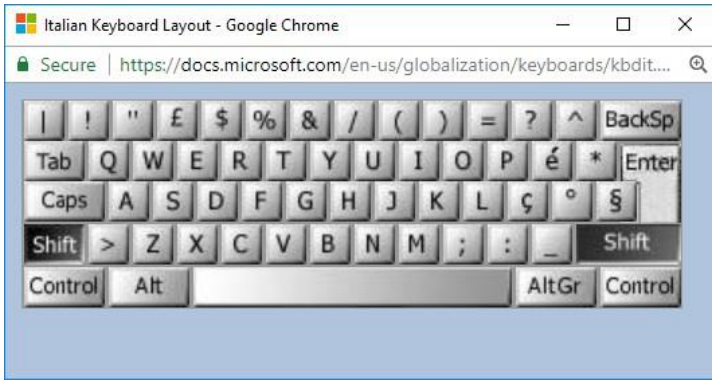
例として、イタリア語ベースのアプリケーションが、イタリア語のキーボードで使用するために、非ASCII文字であるユーロ記号「€」を必要とするとします。

以下は、押された修飾子キーに基づくイタリア語のキーボード文字レイアウトです。ユーロ記号はAltGr+5またはAltGr+eで入力できます。

修飾キーなし

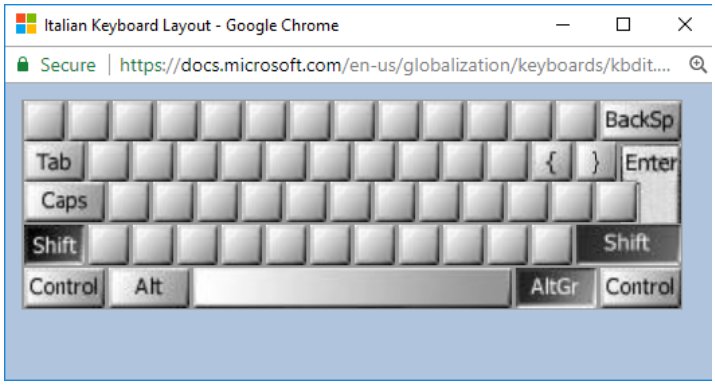


シフト修飾キー



AltGr修飾キー

AltGr+Shift修飾キー



英語キーボードとイタリア語キーボードの両方で同じキー位置にある「5」と「e」キーのUSBスキャンコードは、それぞれ0x22と0x08です。英語キーボードの右Altキーでもあるイタリア語AltGrキー位置のUSB修飾子スキャンコードは0x40です。

以下は、イタリア語キーボードでユーロ記号を入力するためのAlt+eスキャンコードのCodeXMLです。

CodeXML：

```
\x01Y\x1Ean/2F/2Fk4008\x04
```

14.2 用語集と略語

用語	定義
Code Data:	データキャプチャまたはバーコード読み取り後のデコード処理から得られるデータ
Smart Quote：	通常ワープロソフトで見られる、以前の書式の引用符
Consume：	ユーザー定義のアプリケーションまたはファームウェアによって戻り値なしで使用されます

14.3 JavaScriptファイルの暗号化

JavaScriptファイルは暗号化によって保護することができます。暗号化キーについてはサポートにお問い合わせください。
https://www.codecorp.com/

14.4 対応JavaScriptコア

これは'codeRules.js'関数でサポートされているJavaScriptのリストです：

オブジェクト、メソッド、プロパティ

```
Array
Boolean
Date
Function
Math
Number
Object
Packages
RegExp
String
sun
```

トップレベルのプロパティと関数

```
decodeURI
decodeURIComponent
encodeURI
encodeURIComponent
eval
Infinity
isFinite
isNaN
NaN
Number
parseFloat
parseInt
String
undefined
```

ステートメント

```
break
```

const

```
continue
do...
while
export
for
for...in
function
if...else
import
label
return
switch
throw
try...catch
var
while
with
```

オペレーター

代入演算子

比較演算子

算術演算子

% （剰余）
++ （増加）
-- （減少）
- （単項否定）

ビット演算子

ビットごとの論理演算子

ビットごとのシフト演算子

論理演算子

文字列演算子

特殊演算子

?: （条件演算子）
, （コンマ演算子）

```
delete
function
in
instanceof
new
typeof
void
```


The Code Corporation • 434 West Ascension Way Suite 300, Salt Lake City, UT 84123 • +1 (801) 495-2200 • FAX +1 (801) 495-0280

14.5 記号識別子の値

シンボロジー名	CSI	CSI Mod	ID Char	ID Mod
Interleaved 2 of 5（チェックサムをチェックしない）	17	48	73	48
Interleaved 2 of 5（チェックサムをチェックして送信）	17	49	73	49
Interleaved 2 of 5（チェックサムをチェックするが送信しない）	17	51	73	51
Code 39（チェックサムをチェックしない）	18	48	65	48
Code 39（チェックサムをチェックして送信）	18	49	65	49
Code 39（チェックサムをチェックするが送信しない）	18	51	65	51
Code 39フルASCII（チェックサムをチェックしない）	18	52	65	52
Code 39 フル ASCII（チェックサムをチェックして送信）	18	53	65	53
Code 39 フルASCII（チェックサムをチェックするが送信しない）	18	55	65	55
Code 128（標準）	19	48	67	48
Code 128（先頭にFNC1）	19	49	67	49
Code 128（2文字目にFNC1）（別名UCC/EAN 128）	19	50	67	50
UCC/EAN 128（別名 Code 128 (2文字目にFNC1)	19	49	67	49
Codabar（チェックサムをチェックしない）	20	48	70	48
Codabar（チェックサムをチェックして送信）	20	50	70	50
Codabar（チェックサムをチェックするが送信しない）	20	54	70	54
Code93	21	48	71	48
Australia Post	29	97	88	97
Aztec	30	48	122	48
Aztec（先頭にFNC1）	30	49	122	49
Aztec（先頭または数字のペアの後にFNC1）	30	50	122	50
Aztec（ECIプロトコル実装）	30	51	122	51
Aztec（ECIプロトコル実装、FNC1が先頭）	30	52	122	52
Aztec（ECIプロトコル実装、先頭または数字のペアの後にFNC1）	30	53	122	53
Aztec（Structured Appendヘッダーを含みます）	30	54	122	54
Aztec（Structured Append、FNC1が先頭）	30	55	122	55
Aztec（Structured Append、先頭または数字のペアの後にFNC1）	30	56	122	56
Aztec（Structured Append、ECIプロトコル実装）	30	57	122	57
Aztec（Structured Append、ECIプロトコル実装、FNC1が先頭）	30	65	122	65
Aztec（Structured Append、先頭または数字ペアの後にFNC1、ECIプロトコル実装)	30	66	122	66
Data Matrix（ECC 000 - 140、CortexDecoderではサポートされていません）	31	48	100	48
Data Matrix（ECC 200）	31	49	100	49
Data Matrix（ECC 200、FNC1が1番目または5番目）	31	50	100	50
Data Matrix（ECC 200、FNC1が2番目または6番目）	31	51	100	51
Data Matrix（ECIプロトコルをサポートするECC 200）	31	52	100	52
Data Matrix（ECC 200、FNC1を1番目または5番目に配置＋ECIプロトコル対応）	31	53	100	53
Data Matrix（ECC 200、FNC1を2番目または6番目に配置＋ECIプロトコル対応）	31	54	100	54
Straight 2 of 5 with 2-Bar Start/Stop	32	48	82	48
Straight 2 of 5 with 3-Bar Start/Stop	33	48	83	48
Japan Post	34	106	88	106

Dutch KIX Post	35	100	88	100
MSI Plessey	36	48	77	48
Maxi	37	48	85	48
PDF417（標準）	38	48	76	48
PDF417（ECI対応。92文字はすべて2倍）	38	49	76	49
PDF417 （基本チャンネル操作）	38	50	76	50
USPS PLANET	39	101	88	101
USPS POSTNET	40	116	88	116
QR（モデル1シンボル）	41	48	81	48
QR（ECIプロトコル非実装）	41	49	81	49
QR（ECIプロトコル実装）	41	50	81	50
QR（ECIプロトコル非実装、FNC1が先頭）	41	51	81	51
QR（ECIプロトコル実装、FNC1が先頭）	41	52	81	52
QR（ECIプロトコル非実装、FNC1は2番目）	41	53	81	53
QR（ECIプロトコル実装、FNC1が2番目）	41	54	81	54
Royal Mail 4 State Customer	42	114	88	114
GS1 DataBar Expanded	43	48	101	48
GS1 DataBar Expanded Stacked	44	48	101	48
GS1 DataBar Limited	45	48	101	48
GS1 DataBar Omnidirectional	46	48	101	48
GS1 DataBar Stacked / GS1 DataBar Stacked Omnidirectional	47	48	101	48
GoCode	48	71	88	71
UPC-A	49	65	69	48
UPC-E0	49	66	69	48
UPC-E1	49	67	69	48
EAN/JAN-8	49	68	69	52
EAN/JAN-13	49	69	69	48
EAN Bookland（EAN13 5桁の補足データ付き）	49	52	69	48
EAN Bookland（EAN13 5桁の補足データなし）	49	69	69	51
UPC-A（2桁の補足データ付き）	49	97	69	51
UPC-A（5桁の補足データ付き）	49	48	69	51
UPC-E0（2桁の補足データ付き）	49	98	69	51
UPC-E0（5桁の補足データ付き）	49	49	69	51
UPC-E1（2桁の補足データ付き）	49	99	69	51
UPC-E1（5桁の補足データ付き）	49	50	69	51
EAN/JAN-8（2桁の補足データ付き）	49	100	69	51
EAN/JAN-8（5桁の補足データ付き）	49	51	69	51
EAN/JAN-13（2桁の補足データ付き）	49	101	69	51
EAN/JAN-13（5桁の補足データ付き）	49	52	69	51
Codablock 256: FNC1使用なし（CDでは非サポート）	50	48	79	48
Codablock 256: FNC1が先頭（CD では非サポート）	50	49	79	49
Codablock F：FNC1使用なし	50	52	79	52
Codablock F: FNC1 が先頭	50	53	79	53
Codablock A（CDでは非サポート）	50	54	79	54
Code11 (1桁または2桁のチェック文字をチェックして送信)	51	48	72	48

The Code Corporation • 434 West Ascension Way Suite 300, Salt Lake City, UT 84123 • +1 (801) 495-2200 • FAX +1 (801) 495-0280

Code11 (チェック文字をチェックするが送信しない)	51	50	72	50
Pharmacode	52	80	88	80
Matrix 2 of 5（チェックサムをチェックしない）	53	77	88	77
Matrix 2 of 5（チェックサムをチェックして送信）	53	48	88	48
Matrix 2 of 5（チェックサムをチェックするが送信しない）	53	49	88	49
NEC 2 of 5（チェックサムをチェックしない）	54	78	88	78
NEC 2 of 5（チェックサムをチェックして送信）	54	50	88	50
NEC 2 of 5（チェックサムをチェックするが送信しない）	54	51	88	51
Telepen	56	48	66	48
Trioptic Code 39	57	84	88	84
USPS 4CB（インテリジェントメール）	58	105	88	105
BC412	59	66	88	66
Micro PDF417	60	48	76	48
Han Xin	61	72	88	72
Composite CA	62	48	101	48
Composite CB	63	48	101	48
Composite CC	64	48	101	48
Code 32（イタリア薬局方）	65	108	88	108
Plessey	66	48	80	48
Hong Kong 2 of 5	67	104	88	104
Korea Post	68	107	88	107
UPU ID Tag	69	117	88	117
Micro QR	70	49	81	49
Canada Post	71	99	88	99
Code 49（標準）	72	48	84	48
Code 49（FNC1が先頭）	72	49	84	49
Code 49（FNC1が2番目）	72	50	84	50
Code 49（FNC2が先頭）	72	52	84	52
Grid Matrix	73	103	88	103

14.6 onDecode(s) 戻り値 マトリックス

以下のマトリックスでは、様々な `onDecode(s)` 関数からの戻り値が、リーダーによるデコードデータのさらなる処理にどのように影響するかを説明します。

戻り値	reader.onDecodes	rules_onDecodes	reader.onDecode	rules_onDecode
デコードオブジェクト	データ処理を停止	データ処理を継続	データ処理を停止	データ処理を停止しますが、データをホストに送信し、適切なビープ音を鳴らします
0	データ処理を継続	データ処理を停止	データ処理を停止	データ処理を停止

注意：これらの関数の戻り値として "true "や "false "を使わないこと。

14.7 onDecode(s) 互換性マトリックス

次のマトリックスは、`onDecode(s)` 関数の2つのインスタンスを呼び出すときの互換性について説明しています。

	reader.onDecodes	rules_onDecodes	reader.onDecode	rules_onDecode
reader.onDecodes	互換性なし	互換性なし	互換性なし	不適合
rules_onDecodes	不適合	互換性なし	rules_onDecodesがデコードオブジェクトを返す場合は互換性あり	互換性
reader.onDecode	互換性なし	rules_onDecodesがデコードオブジェクトを返す場合は互換性あり	互換性なし	rules_onDecodeが0を返す場合は互換性なし
rules_onDecode	互換性なし	互換性あり	rules_onDecodeが0を返す場合は互換性なし	互換性なし

[i] バッテリー駆動のリーダーのみに適用されます。