Barcode.Go (WASM)

C++ WASMエンジン Go ラッパー

マニュアル

バージョン 1.0

有限会社 パオ・アット・オフィス

https://www.pao.ac/

目次

C++ バーコードエンジンを Go から手軽に利用。

ユーザーズマニュアル

バージョン 1.0 — 2026年2月

有限会社 パオ・アット・オフィス

https://www.pao.ac/

1.1 C++ WASM版とは

Barcode.Go （C++ WASMエンジン） は、C++ で書かれた高速バーコードエンジンを Node.js 経由 で Go から利用するラッパーライブラリです。

C++ バーコードエンジンは Emscripten で WebAssembly にコンパイルされており、Node.js をサブプロセスとして起動して WASM を実行します。Go 側は JSON で命令を送り、生成されたバーコード（Base64 PNG や SVG 文字列）を受け取ります。

```
Go  → JSON → node _barcode_runner.mjs → barcode.mjs → WASM → JSON → Go
```

Pure Go版と同じ18種のバーコードを生成でき、Go 標準ライブラリのみで動作します（外部クレート不要）。

```
import wasm "barcode_pao_wasm"

qr := wasm.NewQRCode(wasm.FormatPNG)
result, _ := qr.Draw("https://www.pao.ac/", 300)
// result  "..."
```

1.2 特長

特長	説明
C++ 高速エンジン	実績あるC++バーコードエンジンをそのまま利用。高品質な出力
Go 標準ライブラリのみ	外部依存ゼロ。os/exec と encoding/json のみ使用
Node.js ブリッジ	Emscripten WASM を Node.js 経由で実行。インストールは node のみ
シンプルAPI	Draw() が直接 Base64/SVG 文字列を返す。2ステップで完結
18種のバーコード	1D・2D・GS1・郵便まで、業務で必要なバーコードを網羅
PNG / JPEG / SVG	画面表示にはPNG、印刷にはSVG。用途で使い分け可能

1.3 対応バーコード一覧

1次元バーコード（15種類）

バーコード	コンストラクタ
Code39	NewCode39()
Code93	NewCode93()
Code128	NewCode128()
GS1-128	NewGS1128()
NW-7 (Codabar)	NewNW7()
ITF	NewITF()
Matrix 2of5	NewMatrix2of5()
NEC 2of5	NewNEC2of5()
JAN-8 (EAN-8)	NewJAN8()
JAN-13 (EAN-13)	NewJAN13()
UPC-A	NewUPCA()
UPC-E	NewUPCE()
GS1 DataBar 標準型	NewGS1DataBar14()
GS1 DataBar 限定型	NewGS1DataBarLimited()
GS1 DataBar 拡張型	NewGS1DataBarExpanded()

特殊バーコード（1種類）

バーコード	コンストラクタ
郵便カスタマバーコード	NewYubinCustomer()

2次元バーコード（3種類）

バーコード	コンストラクタ
QRコード	NewQRCode()
DataMatrix	NewDataMatrix()
PDF417	NewPDF417()

2.1 動作要件

項目	要件
Go	1.21 以降
Node.js	16 以降
OS	Windows / macOS / Linux

Node.js は WASM エンジンの実行に必要です。node コマンドが PATH に含まれている必要があります。

2.2 ダウンロード

https://www.pao.ac/barcode.go/#download

C++ WASM版のダウンロードパッケージには、Easy 2 Steps と All-in-One の2つのサンプルが含まれています。

2.3 ファイル構成

```
barcode_go_wasm_cpp/
├── easy2steps/
│   ├── main.go               # QR
│   ├── go.mod
│   └── templates/
│       └── index.html         # Web
├── allinone/
│   ├── main.go               # 18
│   ├── go.mod
│   └── templates/
│       └── index.html
└── barcode_pao_wasm/          # WASM
    ├── go.mod
    ├── wrapper.go             # Go → WASM
    └── wasm/
        ├── barcode.wasm       # C++ WASM
        ├── barcode.js         # EmscriptenJS
        ├── barcode.mjs        # ES
        └── _barcode_runner.mjs # Node.js
```

起動方法

```
cd barcode_go_wasm_cpp/easy2steps
go run main.go
# → http://localhost:5702
```

3.1 QRコードをPNGで生成

```
package main

import (
    "fmt"
    wasm "barcode_pao_wasm"
)

func main() {
    // Step 1: QR
    qr := wasm.NewQRCode(wasm.FormatPNG)

    // Step 2:  → Base64
    result, err := qr.Draw("https://www.pao.ac/", 300)
    if err != nil {
        fmt.Println("Error:", err)
        return
    }

    // result  "..."
    fmt.Println("Base64:", result[:50], "...")
}
```

ポイント: Pure Go版と異なり、Draw() が直接 Base64 文字列を返します。GetImageBase64() を呼ぶ必要はありません。

3.2 SVGベクター出力

```
import wasm "barcode_pao_wasm"

code := wasm.NewCode128(wasm.FormatSVG)
code.SetShowText(true)
code.SetTextEvenSpacing(true)

svg, err := code.Draw("Hello-2026", 400, 100)
// svg  "<svg xmlns="...">...</svg>"
```

3.3 REST APIサーバー

```
package main

import (
    "encoding/json"
    "log"
    "net/http"
    wasm "barcode_pao_wasm"
)

func main() {
    http.HandleFunc("/api/qr", func(w http.ResponseWriter, r *http.Request) {
        code := r.URL.Query().Get("code")
        if code == "" {
            code = "https://www.pao.ac/"
        }

        qr := wasm.NewQRCode(wasm.FormatPNG)
        b64, _ := qr.Draw(code, 300)

        w.Header().Set("Content-Type", "application/json")
        json.NewEncoder(w).Encode(map[string]string{
            "base64": b64,
        })
    })

    log.Println("→ http://localhost:5702/api/qr?code=Hello")
    log.Fatal(http.ListenAndServe(":5702", nil))
}
```

4.1 共通メソッド

すべてのバーコード型で使用できるメソッドです。

SetOutputFormat(format string)

出力形式を設定します（コンストラクタで指定済みの場合は不要）。

定数	説明
FormatPNG	PNG画像（Base64）
FormatJPEG	JPEG画像（Base64）
FormatSVG	SVGベクター

SetForegroundColor(r, g, b, a int)

前景色（バーの色）を RGBA で設定します。各値は 0〜255。

```
code.SetForegroundColor(0, 0, 128, 255)  //
```

SetBackgroundColor(r, g, b, a int)

背景色を RGBA で設定します。

```
code.SetBackgroundColor(255, 255, 240, 255)  //
code.SetBackgroundColor(0, 0, 0, 0)          //
```

4.2 1次元バーコード共通メソッド

1次元バーコード（郵便カスタマバーコードを除く）で使用できるメソッドです。

Draw(code string, width, height int) (string, error)

バーコードを生成し、Base64 または SVG 文字列を返します。

```
result, err := code.Draw("HELLO123", 400, 100)
```

SetShowText(show bool)

バーコード下部のテキスト表示を設定します。

SetTextEvenSpacing(even bool)

テキストの均等割付を設定します。

SetTextFontScale(scale float64)

テキストのフォントサイズ倍率を設定します。

SetFitWidth(fit bool)

指定幅にぴったり収めるかどうかを設定します。

SetPxAdjustBlack(adj int) / SetPxAdjustWhite(adj int)

黒バー / 白スペースの幅を微調整します。

4.3 各バーコード型

Code39

```
code := wasm.NewCode39(wasm.FormatPNG)
code.SetShowText(true)
code.SetShowStartStop(true)
result, _ := code.Draw("HELLO123", 400, 100)
```

固有メソッド: SetShowStartStop(show bool)

Code128

```
code := wasm.NewCode128(wasm.FormatPNG)
code.SetShowText(true)
code.SetCodeMode("AUTO")  // AUTO / A / B / C
result, _ := code.Draw("Hello123", 400, 100)
```

固有メソッド: SetCodeMode(mode string)

GS1-128

```
gs1 := wasm.NewGS1128(wasm.FormatPNG)
gs1.SetShowText(true)
result, _ := gs1.Draw("[01]04912345123459[10]ABC123", 500, 120)
```

NW-7 (Codabar)

```
code := wasm.NewNW7(wasm.FormatPNG)
code.SetShowStartStop(true)
result, _ := code.Draw("A1234567A", 400, 100)
```

固有メソッド: SetShowStartStop(show bool)

JAN-13 / JAN-8

```
jan := wasm.NewJAN13(wasm.FormatPNG)
jan.SetShowText(true)
jan.SetExtendedGuard(true)
jan.SetTextEvenSpacing(false)
result, _ := jan.Draw("491234567890", 300, 100)
```

固有メソッド: SetExtendedGuard(ext bool)

UPC-A / UPC-E

```
upc := wasm.NewUPCA(wasm.FormatPNG)
upc.SetExtendedGuard(true)
result, _ := upc.Draw("01234567890", 300, 100)
```

GS1 DataBar 標準型

```
db := wasm.NewGS1DataBar14(wasm.FormatPNG)
db.SetSymbolType("OMNIDIRECTIONAL")
result, _ := db.Draw("1234567890128", 200, 80)
```

固有メソッド: SetSymbolType(symbolType string) — "OMNIDIRECTIONAL", "STACKED", "STACKED_OMNIDIRECTIONAL"

GS1 DataBar 拡張型

```
db := wasm.NewGS1DataBarExpanded(wasm.FormatPNG)
db.SetSymbolType("UNSTACKED")
result, _ := db.Draw("[01]90012345678908[10]ABC123", 400, 80)
```

固有メソッド: SetSymbolType(symbolType string), SetNoOfColumns(cols int)

郵便カスタマバーコード

```
yubin := wasm.NewYubinCustomer(wasm.FormatPNG)
result, _ := yubin.Draw("27500263-29-2-401", 25)
```

高さのみ指定（幅は自動計算）。

QRコード

```
qr := wasm.NewQRCode(wasm.FormatPNG)
qr.SetErrorCorrectionLevel("M")  // L / M / Q / H
qr.SetVersion(0)                 // 0=
result, _ := qr.Draw("https://www.pao.ac/", 300)
```

固有メソッド: SetErrorCorrectionLevel(level string), SetVersion(v int), SetEncodeMode(mode string)

DataMatrix

```
dm := wasm.NewDataMatrix(wasm.FormatPNG)
dm.SetStringEncoding("utf-8")
result, _ := dm.Draw("Hello World", 200)
```

固有メソッド: SetCodeSize(size string), SetEncodeScheme(scheme string)

PDF417

```
pdf := wasm.NewPDF417(wasm.FormatPNG)
pdf.SetErrorLevel(2)
pdf.SetColumns(4)
result, _ := pdf.Draw("Hello World", 400, 100)
```

固有メソッド: SetErrorLevel(level int), SetColumns(cols int), SetRows(rows int), SetAspectRatio(ratio float64), SetYHeight(h int)

PDF417 の Draw() は幅と高さの両方を指定します。

項目	Pure Go版	C++ WASM版
エンジン	Pure Go 実装	C++ (Emscripten WASM)
依存	golang.org/x/image	Node.js
API方式	フィールド直接設定 + Draw + GetImageBase64	セッターメソッド + Draw（結果を直接返す）
出力取得	GetImageBase64(), GetSVG(), GetImageMemory	Draw() が直接返す
提供形態	Pure Go ライブラリ	WASM バイナリ
価格	22,000円	11,000円

API の主な違い

Pure Go版:

```
qr := barcode.NewQRCode(barcode.FormatPNG)
qr.Draw("Hello", 300)
b64, _ := qr.GetImageBase64()
```

C++ WASM版:

```
qr := wasm.NewQRCode(wasm.FormatPNG)
b64, _ := qr.Draw("Hello", 300)
```

設定方法の違い:

Pure Go版はフィールドに直接代入:

```
code.ShowText = true
code.TextEvenSpacing = true
```

C++ WASM版はセッターメソッドを使用:

```
code.SetShowText(true)
code.SetTextEvenSpacing(true)
```

項目	要件
Go	1.21 以降
Node.js	16 以降
OS	Windows / macOS / Linux

外部ライブラリの go get は不要です。Go 標準ライブラリ（os/exec, encoding/json）のみ使用しています。

使用許諾

Barcode.Go
の使用について、利用者様と有限会社パオ・アット・オフィス（以下「弊社」）は、以下の各項目に同意するものとします。

1. 使用許諾書

この使用許諾書は、利用者様がお使いのパソコンにおいて Barcode.Go を使用する場合に同意しなければならない契約書です。

2. 同意

利用者様が Barcode.Go を使用する時点で、本使用許諾書に同意されたものとします。

3. ライセンスの購入

製品版を使用して開発を行う場合、1 台の開発用コンピュータにつき 1 ライセンスの購入が必要です。お客様環境等、開発コンピュータでないマシンでの使用にはライセンスは不要です（ランタイムライセンスフリー）。

4. 著作権

Barcode.Go の著作権は、いかなる場合においても弊社に帰属いたします。

5. 免責

Barcode.Go
の使用によって、直接的または間接的に生じたいかなる損害に対しても、弊社は補償賠償の責任を負わないものとします。

6. 禁止事項

Barcode.Go
およびその複製物を第三者に譲渡・貸与することはできません。開発ツールとしての再販・再配布を禁止します。ただし、モジュールとして組み込みを行い再販・再配布する場合は問題ございません。

お問い合わせ

有限会社 パオ・アット・オフィス

Webサイト	https://www.pao.ac/
製品ページ	https://www.pao.ac/barcode.go/
メール	info@pao.ac

Barcode.Go (C++ WASMエンジン) ユーザーズマニュアル

バージョン 1.0 — 2026年2月