

# 5分でわかる MOS BC ご紹介資料

---

# MOS BC の概要

MOSBCは、WindowsPC上でリアルタイムなFA制御プログラムを実行可能な、マシンコントローラです。  
リアルタイムFA制御とIT技術(IIoT、AI、画像処理など)を1台のPC上に実装することにより、  
シンプルかつ高スループットのシステムを実現できます。

**FA制御 × IT技術で**  
「高スループット・低コスト」を実現!

**MOS BC**  
PCベースコントローラ

ベースPC 4Uラックマウントタイプ

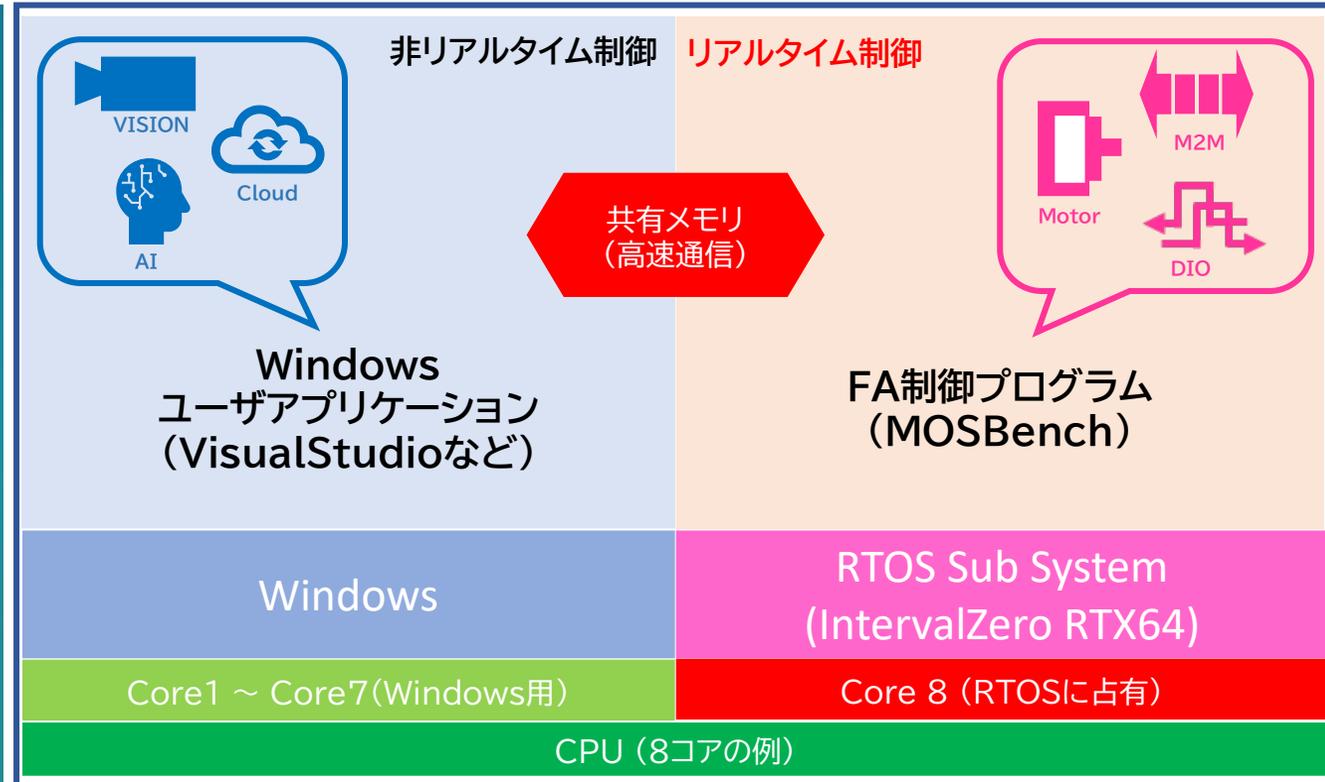
MOS BC に含まれる機能と拡張性

- 高機能モーションを使いやすい提供  
ソフトウェアモーションライブラリ
- Windows上でリアルタイム制御を実現  
ソフトウェアPLC
- PCI/PCIeカード  
モーション/  
DIO制御

MOTIWARE® MOS Bench®

※必要な機能に応じてカスタム可能  
RTEX (Realtime Express) は、パナソニック(株)が開発した  
高速サーボ通信ネットワークです。

システム概要



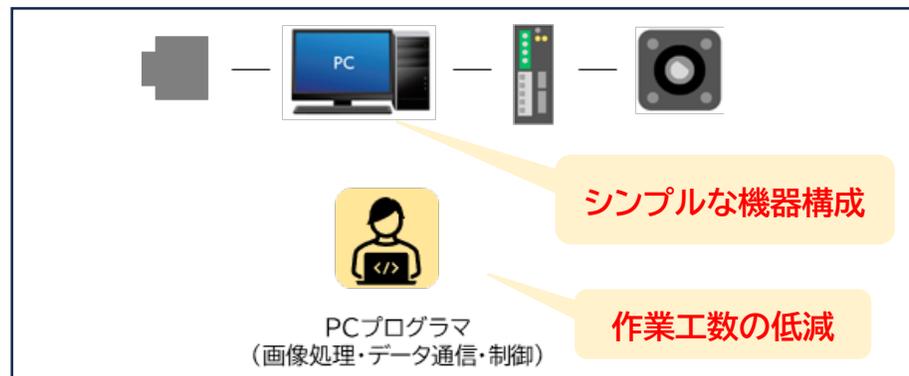
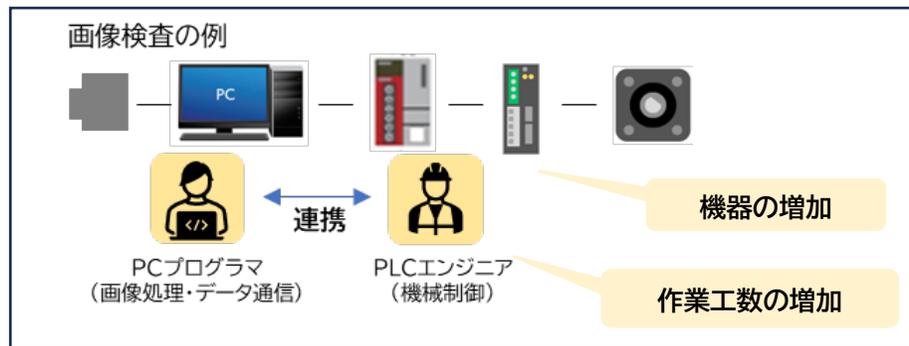
# MOSBCの特長

- WindowsPC上で1msのリアルタイム制御を実現可能。  
⇒ 使い慣れたWindowsPCをFA制御コントローラとして使用できます。
- PC1台でFA制御とIT技術を実装できるため、シンプルなシステムを実現可能。  
⇒ PCとPLC両方を準備する必要が無くシステムコストを低減できます。
- リアルタイム制御とユーザーアプリは共有メモリ経由で高速通信可能。  
⇒ PC-PLC間の通信オーバーヘッドが低減されスループットが向上します。
- 最大40軸(RTEX32軸/パルス列8軸)のモーション制御が可能。  
⇒ 多様なモーション制御を簡単に利用できます。
- FA制御プログラムはC言語ライクな「MOS言語」でプログラム可能。  
⇒ Windowsソフトプログラマでも制御プログラムが実装できます。

# MOS BC による課題解決

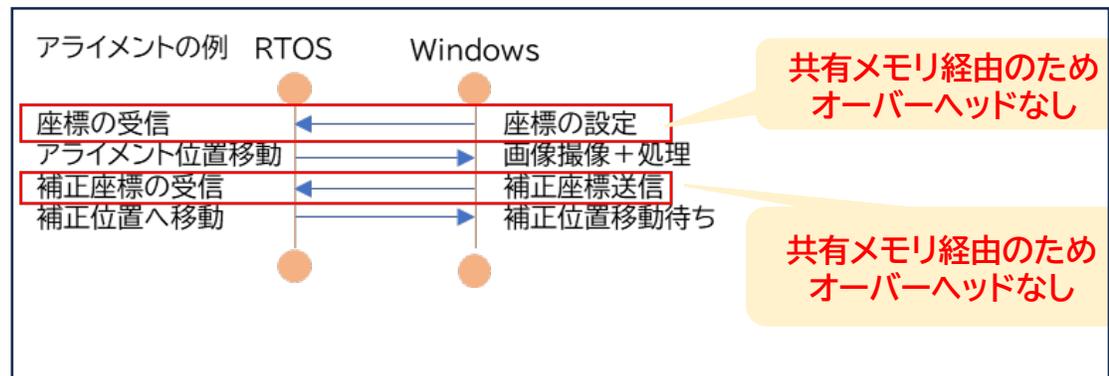
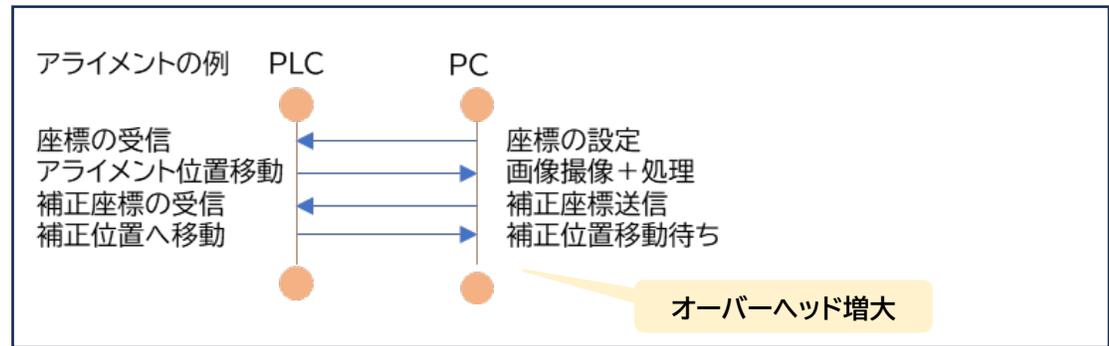
## システムの単純化によるコストダウン

設備にIT技術を導入するためにPCとPLCを組み合わせたシステムの構成例



## スループットの向上

PCとPLC間の通信が多発するシステムでは、通信のオーバーヘッドがマシナクツに大きく影響します。



# 機械制御 × 外観検査

- 全体制御
- ステージ制御
- ロボット制御
- コンベア制御
- カメラ位置決め
- . . .

- 画像撮像
- 画像処理
- データ保存
- . . .



PCベースコントローラのMOSBCは、高速で複雑なモーション制御を含む機械制御と、精密緻密な画像撮像および画像処理を含む外観検査が組み合わさったFAシステムの構築に最適です。

# 製品ラインナップ

ベースPC



4Uラックマウントタイプ



ミニタワータイプ

製品仕様

型番	PM00238	PM00244
シャーシ	4Uラックマウント	ミニタワー
CPU	Gen10 Intel Core i9	Gen10 Intel Core i3
メモリ	DDR4 16GB (8GB × 2)	
ストレージ	HDD2TB/SSD256GB(ともにRAIDオプションあり)	
インターフェース	LAN GbE × 2, USB 3.0 × 2, USB2.0 × 2, COM × 2	
電源	ニプロン 500W (100V-200V)	
PCI	4スロット	
PCIe (Gen3)	7スロット (x8 × 1, x4 × 2, x1 × 4)	2スロット (x8 × 1, x4 × 1)
OS	Windows10 IoT Enterprise LTSC2021	

モーションボード/DIOボード



JOY-RT40PR



JOY-RT8DR

製品仕様

ボード型式	I/F	概要
JOY-RT8DR	PCI	8軸(パルス列)位置指令マスターボード
JOY-RT8DS	PCI	8軸(パルス列)位置指令スレーブボード (拡張用)
JOY-RT40PR	PCI	40軸(パルス列8軸/RTEX32軸)位置指令ボード
JOY-RT40PRE	PCIe (Gen1 × 1)	40軸(パルス列8軸/RTEX32軸)位置指令ボード
JOY-IO6464E	PCIe	デジタルIO In64/Out64

他社対応製品 (Contec 社製品) PIO-16/16L(PCI)H, PIO-32/32L(PCI)H, PIO-32/32F(PCI)H, PIO-32/32T(PCI)H, PI-128L(P-CI)H, PO-128L(PCI)H, AD16-16U(PCI)EV, DA12-4(PCI), DA12-8(PCI), DA12-16(PCI), COM-2(PCI)H, COM-4(PCI)H, COM-8(PCI)H, DIO-3232F-PE, DIO-3232T-PE, DIO-6464L-PE, DI-128L-PE, DO-128L-PE, AIO-161601UE3-PE, AO-1604L-LPE, AO-1608L-LPE, AO-1616L-LPE, COM-2C-PE, COM-4C-PE, COM-8C-PE 他

軸制御

パルス列

RTEX

※同時制御可能

対応可能インタフェース

DIO

AD/DA

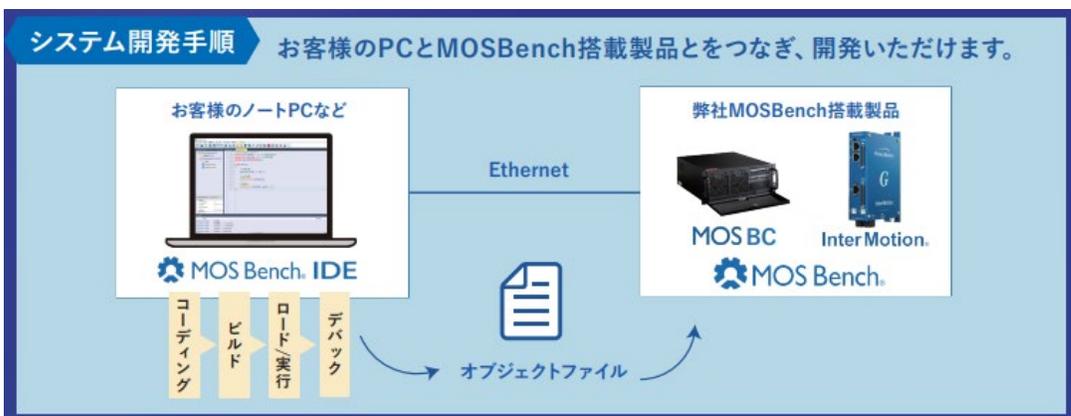
RS232

RS485

AnyWire

# 制御プログラムの実装方法

MOSBCは、弊社製ソフトウェアPLC「MOSBench」がインストールされており、制御プログラムはMOSBenchIDEを使用して実装します。



## 開発手順

- ① お客様のPCにMOSBenchに付属する開発環境「MOSBenchIDE」をインストールします。
- ② MOSBenchIDEを使用して制御プログラムを実装します。  
開発言語はC言語ライクなMOS言語を使用します。
- ③ 実装したプログラムをビルドしてオブジェクトファイルを生成します。
- ④ MOSBCにEthernet経由で接続し、オブジェクトファイルをロードします。
- ⑤ MOSBenchIDEから動作確認・デバックを行います。

**開発環境**

- ✓ シンプルで使いやすい開発環境
- ✓ C言語ライクなので、新たに言語を習得する必要がなくてすぐに開発を始められる
- ✓ ロボット動作を1行で実行可能  
その他、各種制御ライブラリをご用意  
ロボット制御関数/IO制御関数/通信関数 ほか

# ユーザーアプリケーションとの連携方法

MOSBenchでは制御プログラムとユーザーアプリケーションは共有メモリを經由して情報を共有することが可能です。  
また、制御プログラムからイベント発行も可能です。

## 画像認識の例

### 制御プログラム

```

00017 double offset[3]; //ワーク移動オフセット
00018 double time1, time2;
00019
00020 //ロボットの初期設定
00021 InitRobot();
00022
00023 while(1)
00024 {
00025     //時間計測 (タイマ開始)
00026     StartTimer(TimerIndex1);
00027
00028     //Windowsアプリにイベント発行(撮像+画像処理)
00029     FireEvent(EventIndex);
00030
00031     //共有変数から結果を取得
00032     ReadSharedVarArray( SMemAddr, 3, offset, 0, 0 );
00033
00034     //開始から画像処理完了までの時間を計測
00035     time1 = GetTimerValue(TimerIndex1);
00036
00037     //ワーク位置に移動動作
00038     Goal[0] = offset[0];
00039     Goal[1] = offset[1];
00040     Goal[2] = offset[2];
00041     RobPtpMove( RobotNumber, Goal, 1);
00042
00043     //ワーク位置移動までの時間を計測
00044     time2 = GetTimerValue(TimerIndex1);
00045
00046     //出力
00047     Printf2("time_ImageProc=%.3f time_Motion=%.3f\n", time
00048
00049     Sleep(1);
    
```

### Windowsプログラム

```

ProcNo, int EventID)
if (ProcNo == 1)
{
    //画像取得
    vCap.Read(dispNet1);

    OpenCvSharp.Size rs = new OpenCvSharp.Size();
    rs.Width = pbWidth; rs.Height = pbHeight;
    Cv2.Resize(dispNet1, dispNet2, rs);

    cx = (double)rs.Width / 2.0;
    cy = (double)rs.Height / 2.0;

    //画像処理
    double x, y, theta;
    int r = ImageProc(dispNet2, out x, out y, out theta);

    //共有変数に結果を代入
    if (r == 0)
    {
        object[] pValue = new object[3];
        pValue[0] = (double)(x / 100.0);
        pValue[1] = (double)(y / 100.0);
        pValue[2] = (double)(theta);

        sysCtrl.WriteSharedVarArray(0, 3, pValue);
    }

    //画面に表示
    this.BeginInvoke((Action)(() =>
    {
        Cv2.ImShow("test", dispNet2);
        Cv2.WaitKey(1);
    }));
}
    
```

画像取得+処理

取得した位置情報を共有変数に書き込み

イベントを利用することで制御プログラムとWindowsプログラムを簡単かつ高速に連携することが可能です。

# 導入費用およびカスタム対応について

## 導入費用

MOSBenchの開発環境を使用するためには、ライセンスが必要です。  
ライセンス価格 ¥500,000- (標準価格) ※USB dongleキータイプ  
ランタイムには特に費用は必要ございません。

## カスタム対応

お客様のご希望に応じて以下の対応が可能です(有償対応)。

MOSBenchで使用するライブラリ関数(特殊モーション機能や制御機能)の作成・提供  
Windowsアプリケーション(画像処理・上位通信・GUI作成など)の作成・提供



制御に関するお困りごとがございましたら、お気軽にお問合せください。