INtime®

Real-time for Windows

近年産業用機器の制御装置にパソコンが広く使われています。パソコン技術はグローバルスタンダードなので、これを用いた製品の製造やシステム導入に伴う人・物・金・時間面でのコスト低減が可能です。さらに製品およびサービスの品質安定が期待でき、技術的進歩にも容易に対応できます。
INtimeはパソコンを産業用制御に利用するためにリアルタイム性と信頼性を提供するソフトウェア製品およびサービスです。**INtime**はWindows OSと共存するのでソフトウェア・ハードウェア両面で使いやすく、将来的にも安心して使えます。

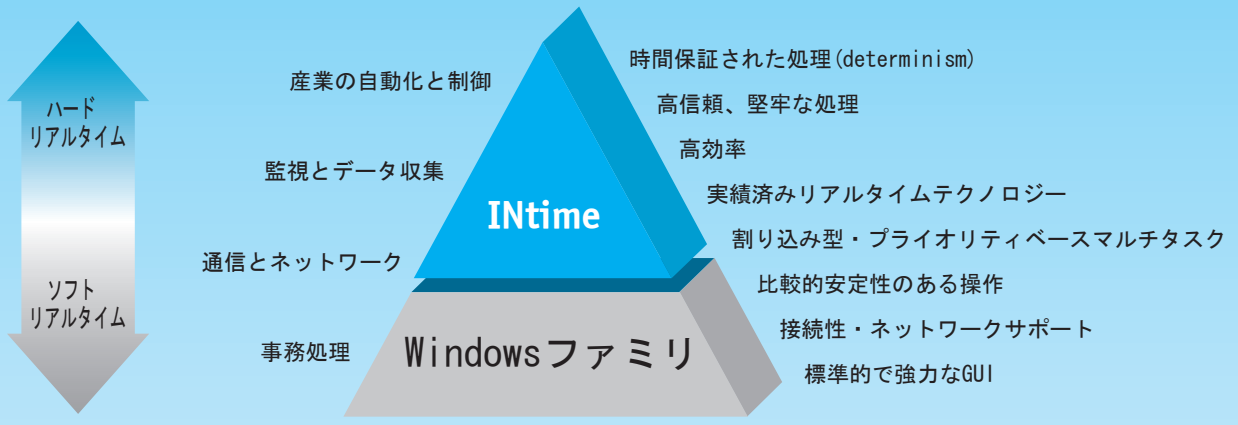


図1. **INtime**リアルタイム機能

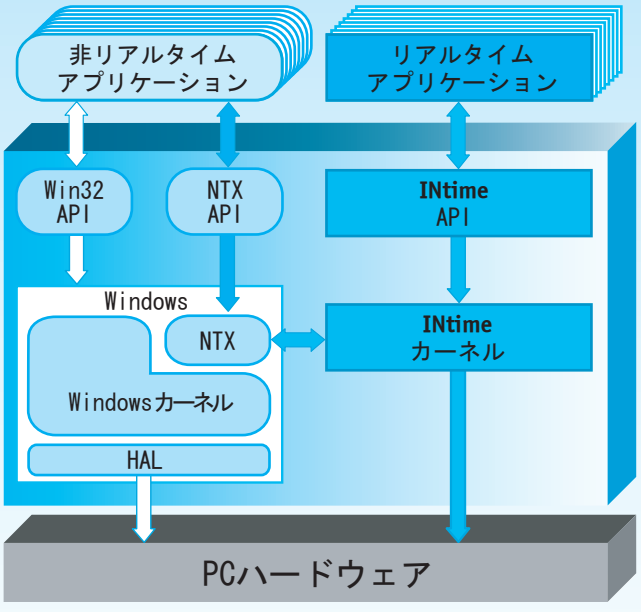


図2. **INtime**のアーキテクチャ

特徴

- Windows 2000/XP/Vistaと協調動作
- Windows XP Embedded対応(SLDファイル)
- 豊富なリアルタイムスケジューリング
  - 256プライオリティレベル
  - アプリケーションプログラムとしての割込みハンドラー(APIC,MSI対応)
  - マルチプロセス、マルチスレッド機能
  - 多彩なシステムオブジェクト(メールボックス、セマフォ、アラーム、リージョン、共有メモリ)
- マルチコアでは共有モード/専有モードの選択が可能
- Windowsなしでも動作可能
- Microsoft Visual Studioの開発環境と統合(Visual C/C++)
- iWIN32によるWindowsプログラムのリアルタイム化
- ユーザーモードによる完全メモリ保護、アドレス分離
- ソースレベルのダイナミックデバッガを標準装備
  - OSを止めずにデバッグ可能
  - Windows**と**INtime**のアプリケーションを同期デバッグ
- リアルタイムTCP/IPドライバ&ライブラリ標準装備
- 産業界で実績のあるリアルタイムテクノロジー(iRMX)

■ INtimeの仕組み

INtimeシステムでは一つのCPU上でWindowsと**INtime**リアルタイムカーネルが同時に動作します。WindowsとINtimeカーネルの切り替えはPentiumプロセッサーのハードウェアマルチタスク機能で実現していますので高速です。

INtimeでは各々のスレッドは256レベル、Windowsスレッドは32レベルの優先度で管理されています。Windows全体は254番目のプロセスとして**INtime**が管理するので、Windowsの動作の都合により**INtime**スレッドのリアルタイム性が損なわれることはありません。

INtimeアプリケーションはWindowsアプリケーションと同様にリング3のユーザーモードで動作するためメモリ保護・アドレス分離が有効に行われます。

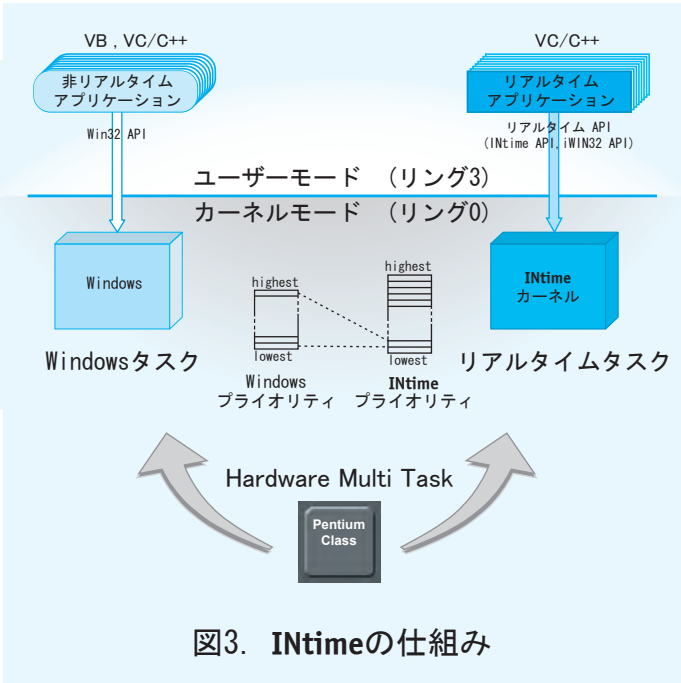


図3. **INtime**の仕組み

表2. INtime API

割込み管理
SignalEndOfRtInterrupt, SetRtInterruptHandler, ResetRtInterruptHandler, SignalRtInterruptThread, WaitForRtInterrupt, DisableRtInterrupt, EnableRtInterrupt
メールボックス管理
CreateRtMailbox *, DeleteRtMailbox *, SendRtHandle *, ReceiveRtHandle *, SendRtData *, ReceiveRtData *
メモリ管理
AllocateRtMemory, FreeRtMemory, CreateRtMemoryHandle, DeleteRtMemoryHandle, MapRtSharedMemory *, MapRtPhysicalMemory, GetRtPhysicalAddress, GetRtSize *
オブジェクト管理
CatalogRtHandle *, LookupRtHandle *, UncatalogRtHandle *, InspectRtProcessDirectory, GetRtHandleType *, GetRtHandleTypeEx
プロセス管理
CreateRtProcess *, ExitRtProcess *, RtNotifyEvent, RegisterRtDependency, UnregisterRtDependency, RegisterRtSponsor, UnregisterRtSponsor
スケジューラ管理
knRtSleep, knStartRtScheduler, knStopRtScheduler
セマフォ管理
CreateRtSemaphore *, DeleteRtSemaphore *, WaitForRtSemaphore *, ReleaseRtSemaphore *
ステータス管理
GetLastRtError *, SetLastRtError, CopyRtSystemInfo, ReportRtEvent
スレッド管理
CreateRtThread, DeleteRtThread, GetRtThreadPriority, GetRtThreadHandles, SetRtThreadPriority, SetRtProcessMaxPriority, RtSleep, GetRtThreadAccounting, SuspendRtThread, ResumeRtThread, GetRtThreadInfo
システムデータ管理
ntxGetLocationByName, ntxGetFirstLocation, ntxGetNextLocation, ntxGetNameOfLocation

(*)対応するNTX APIがあります。

■ アプリケーションインターフェース

- **メールボックスによるデータ交換**
  リアルタイムスレッド、Windowsスレッドのいずれの間でも同期的/非同期的にデータ交換できるメールボックスを設置できます。
- **共有メモリによるデータ交換**
  Windows及び**INtime**のどちらのスレッドからも読書きできる任意の大きさのメモリを確保できます。
- **リアルタイム共有ライブラリ(RSL)**
  WindowsのDLLと同様に、リアルタイムプログラミングにおいてもライブラリをダイナミックに共有できます。
- **標準デバイスドライバ**
  リアルタイムドライバとして、シリアル通信、TCP/IP、USB2.0、工業用リモートI/Oネットワークなどを準備しています。
- **外部I/Oインタフェース**
  デジタル拡張ボードなどの外部I/0へのアクセスはWindowsのような特殊なドライバは不要です。MS-DOSでのプログラミングのようにI/O空間に直接入出力できます。(表1・リスト1)
  拡張ボードとしてPCIボードを使用するためのPCI関数が標準で提供されています。
- **リアルタイムAPI**
  割込みハンドラをアプリケーションプログラムとして利用できます。**INtime** API、iWIN32 API(RTX API)を自由に使えるので非リアルタイムアプリケーションを容易にリアルタイムに移植できます。
  (表2・表3)

表1. 外部I/O関数一覧

入力	data8 = inbyte(portaddress)
	data16 = inhword(portaddress)
	data32 = inword(portaddress)
出力	outbyte(portaddress , data8)
	outhword(portaddress , data16)
	outword(portaddress , data32)
ブロック入力	blockinbyte(portaddress , size , *buffer)
	blockinhword(portaddress , size , *buffer)
	blockinword(portaddress , size , *buffer)
ブロック出力	blockoutbyte(portaddress , size , *buffer)
	blockouthword(portaddress , size , *buffer)
	blockoutword(portaddress , size , *buffer)

リスト1. 外部出力サンプル

```
// パラレルポート 50Hz パルス出力スレッド
void parallelOutThread( void )
{
    while( 1 ) {
        outbyte( parallelportaddress , 0xff ) ;
        RtSleep( 10 ) ;
        outbyte( parallelportaddress , 0x00 ) ;
        RtSleep( 10 ) ;
    }
}
```

表3. iWIN32 API

例外管理
GetExceptionCode, GetExceptionInformation, GetExitCodeProcess, GetExitCodeThread, GetLastError
オブジェクト管理
CloseHandle, CreateEvent, CreateMutex, CreateSemaphore, CreateThread, ExitProcess, ExitThread, GetCurrentProcess, GetThreadPriority, OpenEvent, OpenMutex, OpenProcess, OpenSemaphore, OpenThread, SetEvent, WaitForMultipleObjects, WaitForSingleObject
メモリ管理
HeapAlloc, HeapFree, HeapSize, InterlockedCompareExchange, InterlockedExchange, InterlockedExchangeAdd, InterlockedExchangePointer, InterlockedIncrement
割込み管理 ＊
RtAttachInterruptVector, RtDisableInterrupts, RtEnableInterrupts, RtReleaseInterruptVector
I/O操作 ＊
RtDisablePortIo, RtEnablePortIo, RtReadPort, RtSetBusDataByOffse, RtWritePort
スケジューラ管理
EnterCriticalSection, InitializeCriticalSection, LeaveCriticalSection, ResumeThread, SetThreadPriority, Sleep, SuspendThread
レジストリ管理
RegCreateKeyEx, RegEnumValue, RegLoadKey, RegOpenKeyEx, RegQueryInfoKey, RegQueryValueEx, RegSetValueEx
ファイル管理
CreateDirectory, CreateFile, DeleteFile, FreeLibrary, GetProcAddress, LoadLibrary, ReadFile, WriteFile

(*)RTX互換APIです。

■ 開発環境

INtimeシステムは非リアルタイムのWindowsアプリケーション(GUI、ディスクI/O処理など)と、リアルタイムアプリケーション(時間保証の必要な処理)とで構成されます。いずれも、アプリケーション開発プラットフォームとしてMicrosoft Visual Studioを利用します。

- Windows部分を開発する場合、統合デバッガーも含めて標準的なVisual Studio開発環境を利用します。C/C++のほか、VBなどオブジェクト型言語が使えます。
- **INtime**システムのリアルタイム部分を開発する場合、VisualCまたはC++を使用します。**INtime**の持つリアルタイムAPI、C/EC++ライブラリが利用できます(図4)。
- リアルタイムアプリケーション開発用に、VisualStudioのアドインとして**INtime** Wizardが組込まれており、リアルタイムアプリケーション開発の負担を緩和します。
- リアルタイムサンプルプログラムも多数提供されています。

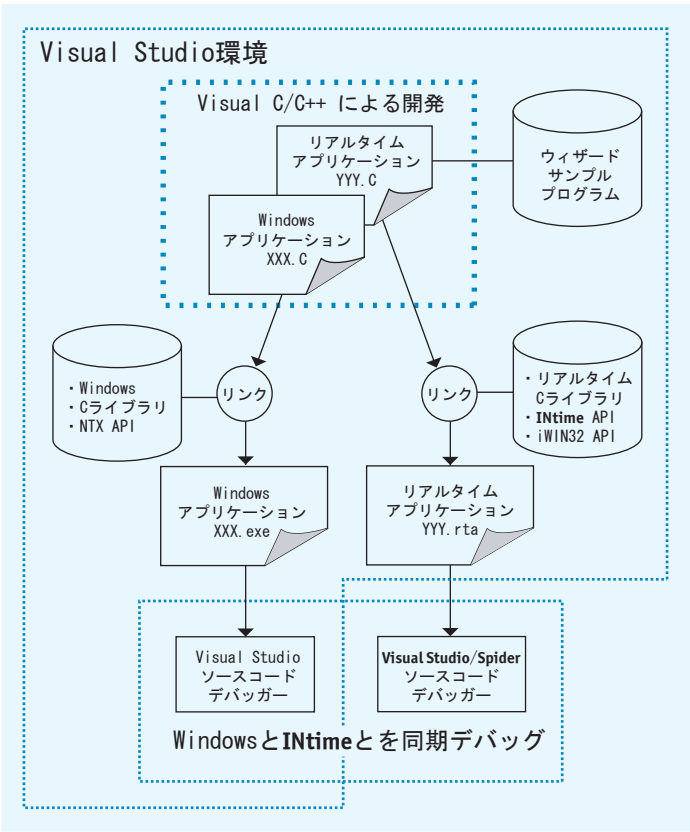


図4. **INtime**ソフトウェア開発

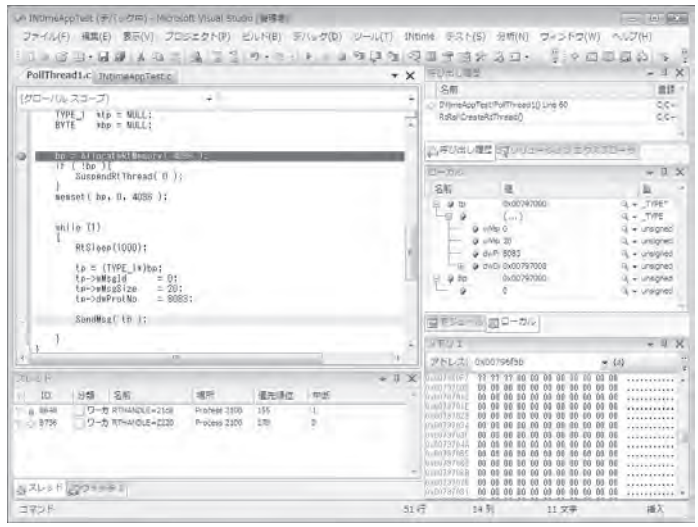


図5. ソースコードデバッガ(**Visual Studio**)

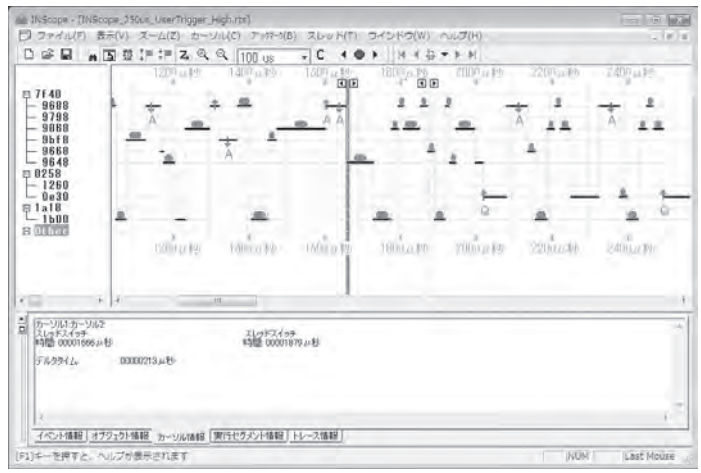


図6. リアルタイムスレッドトレーサー(**INScope**)

■ デバッグ機能

- **Visual Studioとの連携**
  非リアルタイム(Windows)アプリケーションはVisual Studioを使い、リアルタイムアプリケーションはVisual StudioまたはSpiderのソースコードデバッガを使います。これらアプリケーションを一つのディスプレイ上で同期させながら統合的にデバッグできます。
  ソースコードを見ながらブレークポイント、シングルステップトレース、スキップトレース、メモリダンプ、レジスタダンプ、変数ウォッチ、スレッドサスペンド/レジューム等の方法でデバッグできます。
- **全てのマルチスレッド操作が可能**
  スレッドウィンドウを用いてデバッグ対象のスレッドを任意に選択できます。
- **ローカル及びリモートRTノードのデバッグ**
  Visual Studio debugger/Spiderは、単一PCでWindowsと共存しているINtimeノード構成のデバッグだけでなく、シリアルやEthernetで接続されたリモートのリアルタイムノードもデバッグできます。(図5)
- **リアルタイムスレッドトレーサ(INscope)**
  リアルタイムスレッドの動作タイミングをグラフィッカルに分析できます。(図6)

■ システムの監視機能・保守機能

- **ブルースクリーンクラッシュ・ハンドリング**
  INtimeリアルタイムアプリケーションはWindowsハングアップ(ブルースクリーン)を即座にイベントとして検出できます。そして、GUIを失ったまま制御を続けることも、警報を発することも、システム再起動を行うことも可能です。
- **INtime Explorer**
  INtimeの管理する全てオブジェクト(プロセス/スレッド/メモリ/メールボックス/セマフォ他)をダイナミックに参照し、管理することができます。また、アプリケーションエラーのレポーティング機能を持っています。
- **Windowsイベントログサービスとの連携**
  リアルタイムプロセスは、必要に応じて致命的な障害発生や、警告をWindowsの管理するイベントログサービスに対し記録を依頼できます。

■ 柔軟なシステム構成

INtimeシステムは一台のPCプラットフォーム上でWindowsと**INtime**が同時に動作するスタンドアローン型システムのほか、複数のPCプラットフォーム上でネットワーク分散化されたシステムでも動作します。このとき**INtime**およびWindowsのアプリケーションソフトウェアはコード変換なしに相互に互換性があります。(図7)

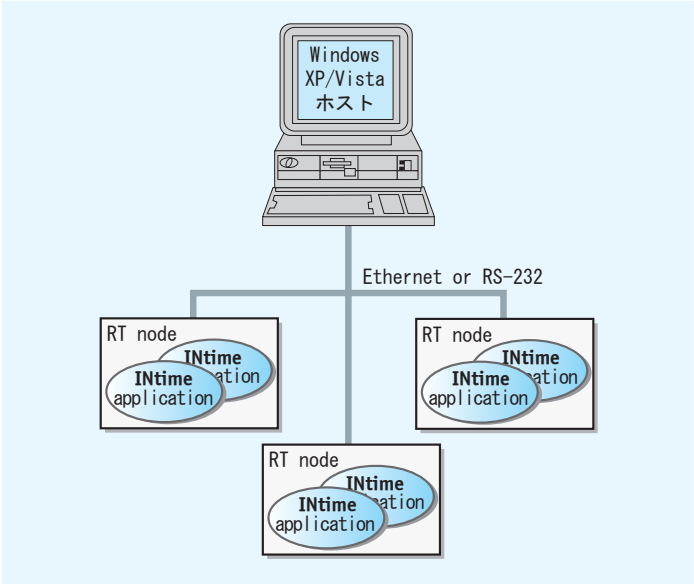


図7. ネットワーク構成

■ 日本語ドキュメント、日本語による技術サポート

INtime開発システムには日本語オンラインヘルプが用意されています。(図8)

技術的な問合わせにはスキルを積んだ日本人技術者が直接回答します。(年間契約/有償)

図8. 日本語オンラインヘルプ

INtime 仕様

プライオリティ スケジューリング	(高)0～255(低) 256段階
カーネルティック	50μs, 100μs, 200μs, 250μs, 500μs 1ms, 2ms, 5ms, 10msより選択。
ラウンドロビン スケジューリング	適用プライオリティしきい値変動可 10ms単位～100ms
メールボックス	FIFO式/優先度式
セマフォ	FIFO式/優先度式
最大セグメントサイズ	4Gバイト
割り込み	ハンドラ/スレッド構造, IRQ共有可, APICモード対応
開発言語	**Microsoft Visual Studio**
リアルタイムデバッガ	**Visual Studio**
ネットワーク (標準添付)	リアルタイムTCP/IP, UDP/IP リアルタイムUSB
最大オブジェクト数	8192個

動作適合ハードウェア

プラットホーム	PC/AT互換機
CPU	Intel Pentium互換(マルチコアを含む)
メモリ	64Mバイト以上
HDD	20Mバイト以上 FAT/FAT32/NTFSフォーマット
OS	Windows Vista Windows Embedded Standard Windows XP , Windows XP Embedded Windows 2000 Windows 7

このカタログ内の仕様は予告なしに変更することがあります。

【開発元】

http://www.tenasys.com/

tenAsys®

Windows® Embedded Gold Partner

【アジア地区総代理店】

http://www.mnc.co.jp/

株式会社 マイクロネット

〒314-0135 茨城県神栖市堀割3-8-11
Tel:0299-(90)-1733 Fax:0299-(92)-8557
E-mail: abc@mnc.co.jp

(2008/11/10K)

ご注意：INtime製品の導入に関し、
以下についてご注意ください

一部認証手続きの出来ないＣＦメディアが確認されております。適合確認済ＣＦメディアにつきましては下記URLにてご案内しておりますので導入の際は事前のご確認をお願い致します。

「 動作適合ハードウェア 」ページ
http://www.mnc.co.jp/INtime/i_platform.htm

INtime®

Real-time for Windows

Version 4.0の新機能

2010年3月

■ 新しいWindowsバージョンのサポート

Windows7、WindowsServer2008がサポートされ、最新のテクノロジーをご利用いただけます。

- Windows Xp SP2とそれ以降
- Windows 2003 Server, R2 SP2とそれ以降
- Windows Vista SP1とそれ以降
- Windows Server 2008
- Windows 7

■ マルチコア動作機能

● マルチリアルタイムカーネル

IntelマルチコアCPUあるいはマルチCPU構成のハードウェアを用いることで、１つの同じシステム上に２つ以上のリアルタイムカーネルの並列動作を実現します。

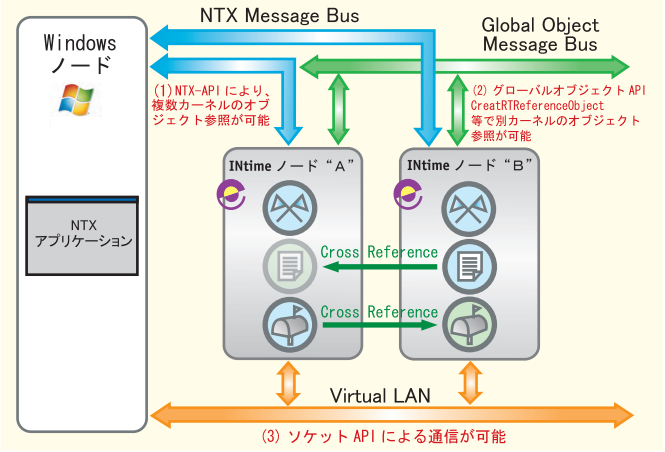
２個以上のコア、２個以上のCPUにそれぞれリアルタイムカーネルを割り当てて実行させることで、リアルタイムアプリケーションを独立した別々のコア上に非同期動作できます。Windowsやリアルタイムタスク同士の影響を受けることなく、動作の独立性を維持します。

● グローバルオブジェクトAPI

異なるリアルタイムカーネルインスタンスに存在しているオブジェクトを取り扱うための追加APIが用意されています。これにより、異なるコアに動作するリアルタイムアプリケーション同士の同期や、データ交換の手段としてセマフォ、メールボックス、共有メモリ、オブジェクトディレクトリを取り扱うことが可能です。

GetFirstRtLocation、GetNextRtLocation、 GetRtNodeLocationByName、GetRtNodeStatus、 GetRtNodeInfo、GetRemoteRootRtProcess、 CreateRtReferenceObject、DeleteRtReferenceObject、 GetRtObjectInfo、CreateGlobalRtSemaphore、 CreateGlobalRtMailbox、CreateGlobalRtMemoryObject、 CreateGlobalRtMemoryHandle

表Ａ．グローバルオブジェクトAPI



図Ｂ．新しいメッセージバスシステム

■ 装置統合・処理分散がさらに実現的となります

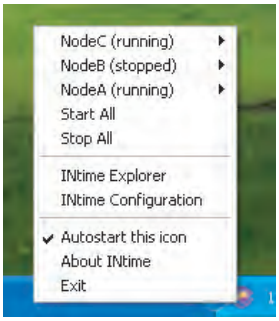
● リアルタイムカーネル制御用のNTX API

Windowsアプリケーションからローカルカーネルの起動と停止を制御できる新しいNTX APIが用意されています。この機能によってアプリケーション自らのタイミングでリアルタイムカーネルの起動・停止タイミングを制御できます。

ntxStartLocalRtNode( ) ・・・ カーネルを開始します
ntxStopLocalRtNode( ) ・・・ カーネルを停止します

● カーネルごとの設定変更が可能

複数のINtimeリアルタイムカーネルは、それぞれ自由に起動と停止を制御することが可能で、動作クロックとなるカーネルティックはそれぞれ異なる周期に設定することもできます。

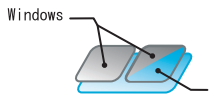
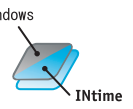
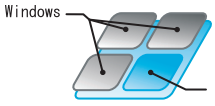
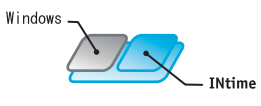
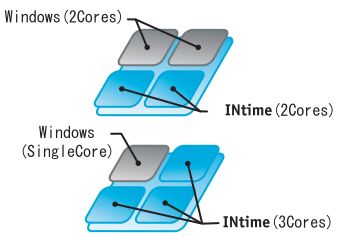


図Ｃ．リアルタイムカーネルの起動制御

● マルチコア用ランタイムライセンス

ランタイムライセンスは、シングルコア用と、マルチコア用の２つに分離されており、必要性に合わせて、最適なランタイムライセンスを選択いただくことが可能です。

INtime-RT ・・・ シングルコア用ランタイムライセンス
INtime-MCRT ・・・ マルチコア用ランタイムライセンス

	Quad Core CPU	Dual Core CPU	Single Core CPU	特徴
共有モード Shared Mode				・Windows は全てのコアを使用できます。(Windows パフォーマンスが最大) ・リアルタイムタスク動作中はすべての Windows 処理が一旦停止します。
専有モード Dedicated Mode			N/A	・１つのコアをリアルタイム専用に割り当てます。(ＯＳ切り替えがないため応答性能が向上) ・Windows とリアルタイムタスクは並列に動作できます。
マルチカーネルモード Multi Kernel Mode		N/A	N/A	・２つ以上のコアそれぞれにリアルタイムカーネルが動作します。 ・Windows と複数のリアルタイムカーネルは全て並列に動作できます。 ・コアごとに制御処理を振り分ける機能分散が実現できます。

図Ａ．マルチコアCPUとINtimeの動作モード

■ 装置統合・処理分散がさらに実現的となります

● システム性能の向上

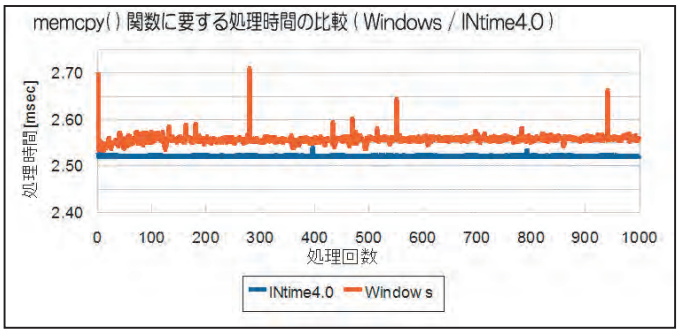
たとえば専用コントローラ、ボードコンピュータ、PLCと、HMIコンピュータで構成されている制御システムは、マルチリアルタイムカーネルによって１つのCPUの中に実現しやすくなります。
そしてこれらは、それぞれ非同期な動作が可能です。格段に早いCPUと、通信回線を必要としない、メインメモリ越しの通信によって制御システムの性能は飛躍的に向上できます。

●保守の標準化

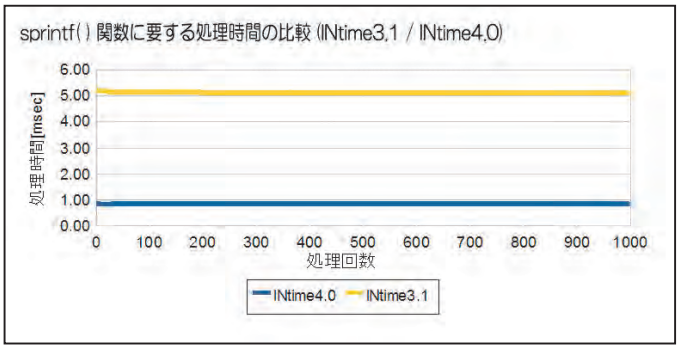
制御コントローラをＰＣハードウェアに集約することによって、PLC、ボードコンピュータなど専門知識を持ったエンジニアが専用ツールで保守を行っていたものから、誰もが使えるWindowsプラットフォームのファイル操作へ保守を標準化できます。

● 新しい規格への準拠

新しいC/C++ライブラリの実装が含まれ、ISO/IEC規格“C99”に引き上げる機能向上が行われています。さらにライブラリアーキテクチャが変更されたことによって、ヒープメモリを管理するmalloc/free関数をはじめとした性能が大幅に向上します。また、ヒープメモリのデバッグライブラリが新たに加わったことで、デバッグ効率が高まります。



図Ｄ．Ｃ関数性能はWindowsより優れる

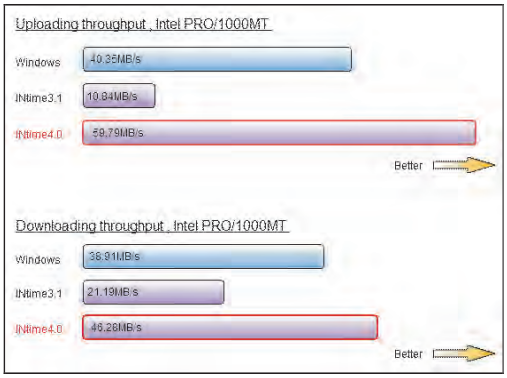


図Ｅ．旧バージョンよりもＣ関数処理性能が向上

■ 改善されたネットワークスタック

● パフォーマンスが向上したTCP/IP

INtimeの新しいTCP/IPネットワークスタックは、従来のネットワークスタックに比べ、スループットと応答性の両面でパフォーマンスが向上しています。

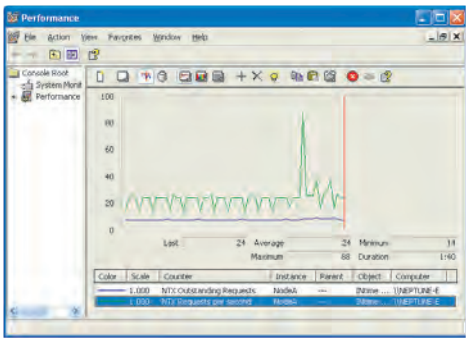


図Ｆ．TCP/IPパフォーマンスの向上(Core2Duo2.6GHz)

■ NTXパフォーマンスモニタカウンタ機能

● パフォーマンス調整ツールの追加

Windowsに新しいパフォーマンスモニタカウンタが追加されます。これまでの「RT Kernel CPU Usage」（INtimeカーネルCPU使用率）に加「NTX Outstanding Request」（NTX要求数）と「NTX Request per second」（秒間NTX要求数）が利用可能となり、アプリケーションシステム最適化の指標として、アプリケーション開発の最終段階で役立ちます。



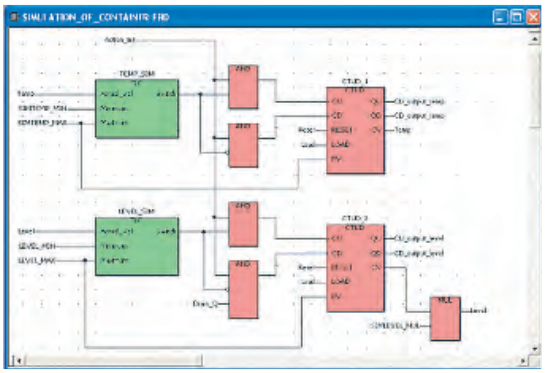
図Ｇ．Windowsパフォーマンスモニタ用の新しいインデックス

■ IEC61131-3環境のサポート （オプション）

● 独立したコアで動作できるPLCエンジン

INplc

オプション製品INplcによって、制御アプリケーションはVisualStudioによるC/C++言語以外にも、国際標準規格IEC61131-3言語によるラダー(LD)言語、ファンクションブロック(FB)言語で開発いただけるようになりました。マルチカーネル機能を利用することで、他のリアルタイムタスクやWindowsアプリケーションの影響を受けることなくPLCアプリケーションを実行できます。



図Ｈ．　IEC61131-3プログラム言語対応のINplc