

리얼 타임 OS (Real time Operating System)

철강 현장에서의 장치 제어나 플랜트 제어, 계측 제어의 현장을 경험하고 30 년간 산업용 시스템 업계에 있던 경험을 근거로 해서 INtime 과 그것에 관련되는 토픽을 생각나는 대로 적어 보았습니다.
현재는 반드시 기술자라고는 할 수 없는 발언이므로 기술적으로는 정확함과 부족한 일이 있을지도 모릅니다.

무엇인가 문의사항이나, 다른 의견 등이 있으시면 사양말고 충고를 부탁드립니다.

시간에는 먼저 타이틀만을 비망록으로서 기재하고 있는 경우도 있습니다. 용서해 주십시오.

TPI KOREA
주식회사 마이크로 넷

리얼타임 OS 란 무엇인가?

리얼타임 OS 는 결코 어플리케이션을 고속으로 동작시키기 위한 OS 가 아니고 CPU 가속기도 아닙니다. 어플리케이션을 고속으로 처리 시키려면 고속 클럭의 CPU 를 사용하면 실현됩니다.

리얼타임 OS 의 역할은 어플리케이션 시스템의 개발자나 프로그래머가 실행 타이밍이나 실행 시간을 계산한 대로 실행시킬 수 있는 것입니다.

다른 말로 하면 동작할 때마다 그러한 타이밍이나 시간이 흠어지지 않게 할 수 있는 OS 라고도 할 수 있습니다.

영어 표현에서는 Deterministic(확정론적인) 태스크 스케줄러라고 합니다.

산업용 시스템에서는 이것이 지극히 소중한 일로, 이 시간이 흠어지면 정도의 높은 제어를 실시할 수 없거나, 가끔 장치나 기계를 망가뜨려 버리기도 합니다.

이것을 실현하기 위해서 리얼타임 OS 에서는 각각의 응용 프로그램 (스레드라든지 태스크라든지라고 하는 말을 사용합니다)에 우선 순위(priority)를 붙이는 것으로, 프로그램의 실행 순위를 결정합니다.

복수의 프로그램이 동시에 동작 가능 상태가 되었을 때에는 리얼타임 OS 는 우선도의 높은 프로그램을 먼저 실행시킵니다.

아무리 고속의 Pentium4 프로세서를 사용해도 Windows 에서는 진짜 리얼타임(하드 리얼타임)을 실현할 수 없는 것은 자주 체험하고 있을 것입니다.

한편, CPU 로서는 저속의 8 비트 마이크로컴퓨터를 사용해도 리얼타임 시스템을 만들 수 있습니다.

고속 CPU 와 리얼타임 OS 의 편성이 고성능의 리얼타임 시스템입니다.

또 하나의 리얼타임 OS 로서의 요건은, 인터럽트 처리가 응용 프로그램 레벨로 프로그래밍 할 수 있다는 것입니다.

리얼타임 시스템에서는 복수의 프로그램을 효율적으로 동작시키기 위해서, 어느 A 프로그램이 이벤트라든지 사실상 기다리는 사이는 CPU 능력을 다른 프로그램을 실행시키기 위해서 할당합니다.

그리고 다시 처리를 해야 할 상황이 되면 우선 순위의 판단을 해 방금 전의 프로그램을 재개하는 것입니다.

이 때 사실적 상태를 체크하기 위해서 정기적으로 프로그램이 스캔 하도록 하면 그 만큼 CPU 파워를 쓸데 없이 소비하게 됩니다.

리얼타임 OS 에서는 이런 경우에 인터럽트의 기능을 사용해 최적의 프로그램을 합니다만, 그러기 위해서는 어플리케이션으로서 인터럽트 프로그램을 기술할 수 없으면 안됩니다.

Windows 시스템의 경우는 인터럽트 프로그램은 일반적으로 커널 모드의 드라이버 프로그램으로서 기술해, 일반적으로 고도의 프로그래밍 기술을 필요로 합니다.

INtime 에서는 이 인터럽트 프로그램을 유저 모드로 실행시키도록 프로그래밍 할 수 있습니다.

이것은 위에도 말한 것처럼 드라이버 소프트웨어를 용이하게 작성할 수 있게 됩니다.

Intime 이란 무엇인가?

INtime 은 한마디로 말하면, Windows PC 에 리얼타임성을 추가하는 확장 소프트웨어입니다.

Intime 을 인스톨 한 PC 에서는 INtim 커널과 INtime 관련 소프트웨어는 Windows 의 서비스로서 동작하고 있습니다만, INtime 커널 자신은 단독으로도 동작 가능한 32 비트 리얼타임 커널입니다.

INtime 커널은 인텔사의 리얼타임 OS 「iRMX」와 같은 커널을 사용하고 있어 산업계에서는 25 년 이상의 실적이 있는 커널로 산업용 시스템으로서 24 시간 365 일의 연속 운전에도 견딜 수 있는 신뢰성의 높은 소프트웨어입니다.

Intime 을 인스톨 한 Windows PC 를 산업용 시스템의 제어장치로서 사용하는 경우에는, 통상 제어나

연산, 데이터 수집, 통신 등의 리얼타임성을 필요로 하는 응용 프로그램은 INtime 관리하의 프로세스/스레드로서 또, 맨 머신 인터페이스(HMI,GUI) 기능이나 데이터 베이스 기능, 고도의 네트워크 기능 등은 Windows 관리하의 프로그램으로서 구성합니다.
 각 각의 응용 프로그램은 마이크로소프트의 VisulaStudio 를 사용하고, 주로 VC/C++언어로 기술합니다.

Windows 와 INtime (리얼타임 확장)



(INtime 의 Realtime 기능)

INtime 의 특징

INtime 카탈로그나 INtime 웹페이지로 INtime 의 특징을 말하고 있습니다만, 여기에서는 조금 다른 관점으로부터 INtime 의 특징이나, 다른 리얼타임 OS 와의 차이에 대해 말해 보겠습니다.

1) INtime 은 인텔사가 개발한 iRMX 커널을 사용하고 있는 것도 있어 인텔의 x86/CPU 에 특화된 소프트웨어입니다.

다른 리얼타임 OS 에는 다양한 CPU 로 동작하도록 만들어져 있는 것도 있습니다만, 이 점은 단점인 것과 동시에 x86/CPU 의 특성을 살리고 있다고 하는 의미에서는 장점이기도 합니다.

2) INtime 을 인스톨 한 Windows 시스템은 Windows 와 Intime, 2 개의 OS(커널)가 동작하는 멀티 OS(커널)의 시스템이 됩니다.

리얼타임 OS 로 이러한 구조를 도입하고 있는 것은 적습니다만 INtime 의 경우에는 이 2 개의 OS 를 CPU 가 가지고 있는 하드웨어 멀티 태스킹의 기능으로 하드적으로 가고 있습니다.

다른 말로 하면 1 개의 CPU 하드웨어를 2 개의 가상 CPU 로서 사용하고 있게 됩니다.

따라서 Windows 와 INtime 의 변환은 하드웨어적으로 바뀐다만 그 때문에 소프트웨어적으로 바꾸는 경우에 비해 변환 시간이 약간 늦어지고 있습니다.

그렇지만 그 시간은 수 μ 초에 대해서 +1~+2 μ 초 정도이기 때문에 거의 문제가 되지 않습니다.

3) 리얼타임 OS 를 사용한 시스템에서는 응용 프로그램으로부터 I/O 공간에 자유롭게 액세스 할 수 있는 것을 얻을 수 있습니다.

Windows 의 프로그램에서는 자유롭게 I/O 공간에 액세스 하지 못하고, 커널 모드로 동작하는 디바이스 드라이버가 경유 되어, 쓰거나 신뢰성에 불만이 남습니다.

INtime 에서는 응용 프로그램은 Windows 와 같이 보호된 유저 모드 (링 3) 로 동작합니다만, 응용 프로그램으로부터 직접 I/O 공간에 I/O 할 수 있습니다.

이것은 INtime 이 Windows 와는 다른 가상 CPU 를 사용하는 하드웨어 멀티 태스킹을 사용하고 있기 때문에 실현될 수 있는 것 입니다.

4) 사실 이것은 시스템의 신뢰성이나 디버거 기능과도 큰 관계가 있습니다.

다른 리얼타임 OS 의 상당수는 I/O 공간에 직접 입출력을 실시할 수 있듯이 하기 위해, x86/CPU 로 동작하는 경우에 Windows 의 드라이버와 같이 커널 모드 (링 0) 로 어플리케이션이 동작하게 되어 있습니다.

Windows 시스템의 크래시가 디바이스 드라이버의 불편으로 발생하듯이, 커널 모드로 달리는 응용 프로그램에 불편이 있으면 OS 나 시스템 영역을 파괴해 시스템 전체에 데미지를 받는 일이 있습니다.

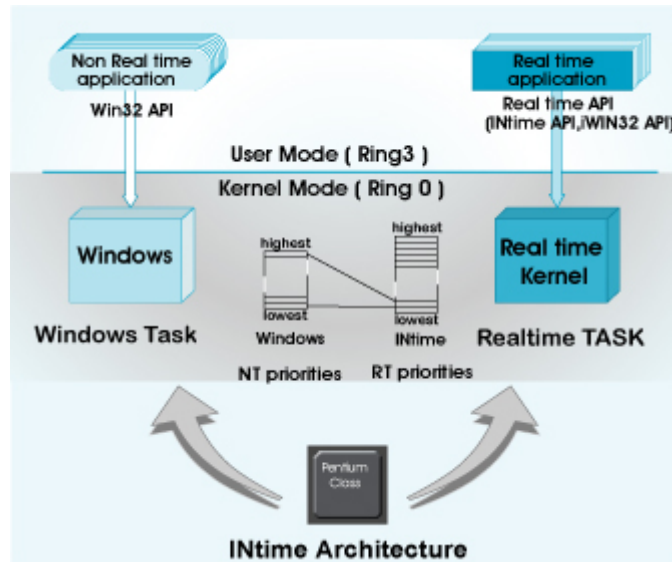
이러한 어플리케이션의 버그가 완전하게 다 취할 수 있지 않은 상태로 테스트를 해도 시스템이 크래쉬해 버리면 디버거조차 동작할 수 없습니다.

그에 대한 INtime 의 경우는 유저 모드(보호 모드)로 동작하기 때문에, 응용 프로그램이 불완전한 상태로 폭주하는 일이 있어도 OS 나 다른 어플리케이션은 보호되고 있기 때문에 시스템이 정지하거나 디버거가 정지하는 일은 없습니다.

INtime에서는 Windows 의 GUI 기능을 사용한 리얼타임 디버거를 이용해 어플리케이션의 고도의 디버그를 셸프로 실시할 수 있습니다.

5) 또, 같은 이유로 INtime 시스템에서는 리얼타임용의 드라이버 소프트웨어를 유저 모드(링 3)로 일반의 어플리케이션 소프트웨어를 작성하는 것과 같은 감각으로 개발할 수 있습니다.

아날로그 입출력이나 디지털 입출력과 같은 비교적 단순한 I/O 는 어플리케이션으로부터 직접 입출력 할 수 있기 때문에 드라이버는 불필요합니다. C 언어의 표준 함수를 사용해 입출력 합니다.



(Intime Architecture)

다른 리얼타임 OS 라는 비교

한마디로 리얼타임 OS 라고 말해도 휴대 전화나 디지털 가전 등의 비교적 소규모의 시스템에 사용되는 iTRON 와 같은, 리얼타임 OS 라고 하는 것보다도 리얼타임 모니터와 같은 것으로부터, 공작기계나 제조 장치, 플랜트 제어등과 같은 비교적 대규모 시스템에 사용되는 Intime 이나 VxWorks 와 같은 본격적 리얼타임 OS 까지 다양합니다.

현재 일반의 유저에게 잘 사용되고 있는 리얼타임 OS 와 그 특징을 표에 정리해 보았습니다.

	INtime	RTX	VxWorks	QNX	RT-Linux	WindowsCE	iTRON
대상 CPU	x86	x86	다수	x86	x86	다수	다수
비트수	32	32	8,16,32	32	32	16,32	8,16,32
Disk	有	有	有,無	有	有	有	無
GUI	WindowsXP	WindowsXP	독자	특수	Linux	WinCE	無

자주 잡지등에서 다른 리얼타임 OS 도 포함해 소개하고 있는 경우가 있습니다만, 그러한 대부분은 특정의 기업이나 어플리케이션으로 사용되는 것은 있어도, 일반의 유저에게는 극단적으로 정보가

적거나 서포트가 한정되어 있거나 해 실제로 도입하는 것은 그만큼 용이하지는 않습니다. 예를 들면 휴대 전화로 잘 사용되고 있는 신비안 등은 라이선스량은 많습디만, 주로 휴대 전화기 메이커가 채용하고 있어 라이선스수가 수백, 수천 개가지고는 상대도 해 주지 않는다고 생각합니다. 또, 그 기술은 휴대 전화기 메이커 내에서 달고 있어 공개되고 있지 않으므로 범용의 리얼타임 OS 라고는 말하지 못하고, 일반의 유저가 선정의 대상이 되는 리얼타임 OS 가 아닙니다. 이러한 「특수」 리얼타임 OS 에 종사하는 엔지니어도 많이 있습니다만, 대부분은 파견형의 소프트웨어 회사로부터 파견된 엔지니어가 단지 「특수한」 어플리케이션의 프로그래머로서 개발에 종사하고 있는 것 만으로, 다른 시스템에의 응용할 수 있는 상황이 되지 못하고 않습니다.

리얼타임 OS 를 비교하는데 있어서 자주 태스크의 변환 시간이나 인터럽트의 응답성 등의 리얼타임 성능에 관심을 갖고 계십니다만, Windows CE 를 제외하고 그러한 성능차이는 거의 없습니다. 원래 성능 비교를 하기 위해서는 CPU 나 주파수 등의 조건을 갖춘 논의가 필요합니다만, 만일 그러한 하드웨어의 조건을 같게 하면 거의 같은 성능을 얻을 수 있는 것입니다. 그 이유는 어느 리얼타임 OS 도 대략적으로 말하면, 그러한 성능을 실현하는 구조는 거의 같기 때문에 입니다. 따라서 최근에는 잡지나 논문등에서도 그러한 성능 비교를 하는 것은 거의 없어졌습니다. 오히려, 개발툴이나, GUI 기능(HMI 시스템), 이용할 수 있는 주변기기나 드라이버의 종류, 거기에 디버그 툴의 기능의 좋음과 좋지 않음이 관심적으로 되어 있습니다.

【INtime에서 본 다른 리얼타임 OS의 평가】

…오해와 비판을 두려워하지 말고…

1. iTRON

iTRON 는 비교적 소규모의 원 보드 마이크로컴퓨터 등에 짜 넣어져서 사용되는 리얼타임 커널로, 통상 라이선스 프리입니다.

표준에서는 Disk I/O 시스템이나 GUI 라이브러리, TCP/IP 는 제공되지 않기 때문에 그러한 사용법에는 향하지 않습니다. 단 TCP/IP 에 대해서는 제공하고 있는 벤더가 있는 것 같습니다. 일반적으로 iTRON 를 달리게 하는 표준의 하드웨어는 없기 때문에 통상은 어플리케이션 소프트웨어의 개발에 앞서 CPU 보드의 개발을 실시하는지, iTRON 에 대응한 보드를 선정할 필요가 있습니다.

잡지나 기술서에서는 자주 이 OS(모니터)가 다루어집디만 INtime 이 대상으로 하고 있는 규모의 시스템에서는 검토의 대상으로는 되지 않습니다.

2. WindowsCE

당초보다 리얼타임 OS 로 이름을 붙여 등장한 Windows CE 는 마이크로소프트의 의도와는 별도로(혹은 혹시 의도대로) PDA 등의 소형 정보 단말에 사용되는 것은 있어도 리얼타임 OS 로서의 기능을 살려 쓰여지는 방법은 되어 오지 않았다고 말할 수 있습니다. 그 이유는 몇 개인가 있습니다만, 다른 리얼타임 OS 정도의 리얼타임 성능이 나오지 않는 것도 이유의 하나입니다. 1 ms 보다 빠른 리얼타임성은 실현될 수 없는 것 같습니다. 이것은 최근 마이크로소프트가 Windows CE 와 iTRON 를 조합한 더블 OS 를 제안하고 있는 것 부터로도 알 수 있습니다.

또, Windows CE 는 다른 Windows 제품과 같이 시중에서 패키지로써 판매되고 있지 않습니다. 그것은 Windows CE 가 OEM 제품이며, 하드웨어를 특정하지 않으면 Windows CE 소프트웨어의 구성이 정해지지 않기 때문입니다. 즉 PDA 의 카시오페아로 사용되고 있는 Windows CE 는 카시오페아 전용의 Windows CE 이며, 다른 Windows CE 기로 사용할 수 없습니다. Windows CE 를 짜넣어 용도의 OS 로서 사용하기 위해서는 iTRON 등과 같이 하드웨어의 개발로부터 실시하지 않으면 안됩니다.

3. QNX

QNX 는 복미를 중심으로 예상 이상으로 보급되어진 리얼타임 OS 입니다. 일본에서도 강력한 판매 체제의 탓도 있어 많이 보급되어 있습니다. 비교적 규모가 큰 범용의 리얼타임 OS 로써 붙이는 누적 라이선스에서는 VxWorks 에 뒤잇는 2 번째지요. QNX 의 훌륭함은 풍부한

라이브러리와 다양한 드라이버군입니다. 특히 GUI 라이브러리는 Windows 의 화면 표시를 견딜지도 모릅니다.

QNX 가 동작하는 하드웨어 플랫폼은 x86CPU 의 PC 아키텍처입니다만, QNX 에서는 GUI 에 QNX 독자적인 방식을 채용했기 때문에 어느 PC 에서도 동작한다고 말하는 것은 아닙니다. PC 로 사용되고 있는 그래픽 칩 혹은 그래픽 보드는 CPU 이상으로 기술 혁신의 격렬한 분야에서 신제품의 PC 가 시장에 나올 때마다 하드웨어가 변경이 되어 있다고 해도 과언이 아닙니다. 통상 이러한 그래픽 하드웨어의 드라이버 소프트는 하드웨어 메이커가 준비를 해 제공합니다만, 그것은 주로 Windows 용의 드라이버입니다.

QNX 용의 그래픽 드라이버는 QNX 벤더가 개발할까? 유저 자신이 준비할 수 밖에 없습니다. 그렇지만 이 그래픽 드라이버의 개발에는 대단한 노력과 비용이 들므로인해, 실제로는 그래픽 하드웨어를 고정해 사용하게 되어, 결과적으로 어느 PC 에서도 동작하는 것이 불가능한 것입니다.

4. VxWorks

제일 많이 팔리고 있는 리얼타임 OS 입니다. 주요한 CPU 칩에도 대응하고 있고, iTRON 와 같은 비교적 소규모의 편입 보드 컴퓨터용의 리얼타임 모니터 레벨로부터, 예를 들면 InTime 과 같은 하드 디스크나 그래픽 기능을 가지는 비교적 규모의 큰 시스템까지 대응하고 있습니다.

하드웨어 플랫폼으로서 PC 이외에 VME 보드 컴퓨터라든지 유저 고유의 보드 컴퓨터로 사용할 수 있도록 구성할 수 있습니다. (어느 쪽인가 하면 PC 아키텍처로 사용하는 케이스보다 VME 보드 컴퓨터로 사용되는 케이스가 많은 듯 합니다)

VxWorks 는 GUI 기능으로서 X-Windows 와 같은 Unix 호환기능을 사용할 수 있습니다만, 이 경우도 그래픽 드라이버는 VxWorks 의 벤더 혹은 유저 자신으로 준비할 필요가 있습니다. 그래픽 칩을 변경하면 새로운 칩의 드라이버 소프트를 준비하지 않으면 시스템이 동작하지 않게 됩니다. 그 때문에 장기 안정공급을 필요로 하는 산업용 시스템에서는, 그래픽 칩이나 보드를 시작해 하드웨어의 재고 관리를 하거나 매점을 하지 않으면 안됩니다. 기술 혁신이 격렬한 컴퓨터 부품은 극히 짧은 주기에 신제품에 옮겨지고 있기 때문에, 시장으로부터 장기간 안정에 부품을 입수하는 것이 곤란하기 때문입니다.

5. RTX

RTX 는 InTime 에 잘 닮은 Windows 와 협조 동작을 하는 리얼타임 커널입니다. RTX 도 InTime 도 GUI 하부조직으로서 Windows(2000/XP)가 동작하기 때문에 VxWorks 나 QNX 와 같은 그래픽 칩의 드라이버의 문제가 없습니다. 세세한 부분을 제외하고 RTX 와 InTime 는 사양적으로 잘 닮은 OS 입니다만, 다음과 같은 차이가 있습니다.

6. I.. InTime 에서는 Windows 와 InTime 커널의 변환은 x86CPU 의 하드웨어 기능을 사용해서 가고 있습니다만, RTX 에서는 HAL 레벨로 소프트웨어적으로 가고 있습니다.

II.. InTime 커널과 InTime 어플리케이션은 Windows 와는 완전하게 독립한 환경에서 동작하고 있습니다. 그 때문에 InTime 어플리케이션은 동작이 보호된 유저 모드(링 3)로 동작하기 때문에 어플리케이션의 버그 등에 대해 터프합니다.

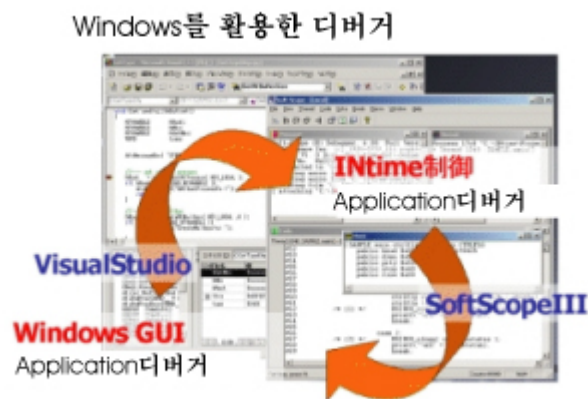
한편, RTX 의 경우에는 RTX 커널과 RTX 어플리케이션은 Windows 커널과 같은 환경의 커널 모드(링 0)로 Windows 의 드라이버로서 동작하고 있습니다. 이것은 리얼타임 어플리케이션이

OS 와 같은 특권 모드로 동작하는 것을 의미해, 어플리케이션의 버그등에서 시스템 전체를 크래쉬 시키는 위험성이 있습니다.

III. 상기는 디버거에 차이가 나옵니다. 개발 도중의 아직 버그를 포함한 상태로 어플리케이션을 디버그 하는 경우, INtime 에서는 어플리케이션의 버그에 의해서 시스템이 크래쉬 할 것은 없기 때문에 하나의 어플리케이션으로서의 디버거를 사용할 수 있습니다. 따라서 Windows 도 정지할 것은 없습니다. INtime 에는 표준으로 다이내믹 소스 코드 디버거가 부속되어 있습니다.

RTX 의 경우는 어플리케이션의 버그에 의해서 시스템이 정지해 버릴 가능성이 있기 때문에, 고도의 디버거에서도 그것 자신이 정지해 버릴 가능성이 있습니다. RTX 에서는 이것을 피하기 위해서 다른 PC 를 타겟으로 접속해 리모트로 디버그를 실시하든지, Windows 및 RTX 커널을 정지시켜 정적인 디버그로 대응합니다.

최신의 INtime2.23 이후에서는 리얼타임 어플리케이션은 종래의 INtime-API 이외에, iWIN32API 나 RTX-API 도 서포트하고 있습니다. Windows 만으로 소프트 리얼타임 시스템을 구축하고 있는 고객이나, RTX 로 시스템이 실현되고 있는 고객도 용이하게 Intime 으로 이행할 수 있습니다.



(Windows를 활용한 디버거)

동작 적합 하드웨어

■세계 공통의 하드웨어 플랫폼

Windows 가 동작할 수 있는 PC 가 플랫폼이므로 하드웨어·코스트는 최소한으로 억제할 수 있고 다양한 메이커의 다양한 하드웨어나, 소프트웨어를 활용할 수 있습니다. 세계 규모로 존재하는 PC/AT 플랫폼은 향후 몇 년간에 걸쳐 하드웨어 입수를 보증할 수 있다고 해도 좋을 것입니다.

수많은 메이커로부터, 방진·방적가공 모델, 터치 패널 모델, 소형·경량 모델 등 운용 환경에 맞추어 자유로운 선택사항을 가질 수 있는 일도 PC 플랫폼의 큰 어드밴티지입니다.

플랫폼	PC/AT호환 PC			
CPU	Intel™ Pentium®호환 *1			
메모리	64 M바이트 이상			
HDD	20 M바이트 이상 FAT/FAT32/NTFS 포맷			
OS	Windows® NT4.0(SP3) 이후 Windows® XP Embedded	Windows® NT Embedded	Windows® 2000	Windows® XP

*1 싱글 프로세서만 대응

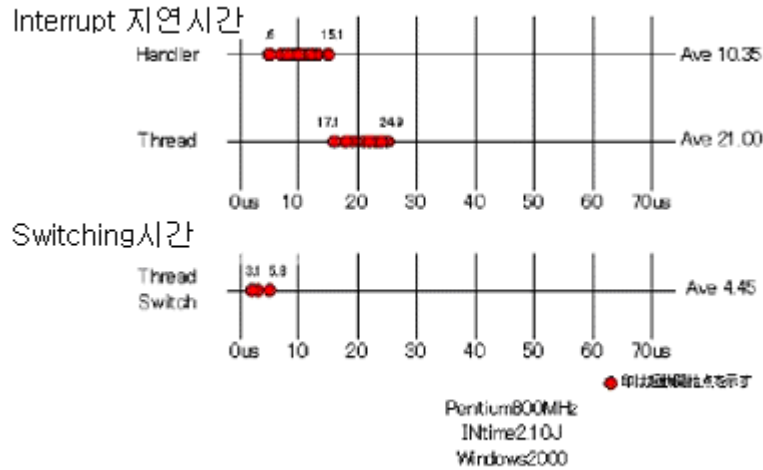
■Intel 호환 CPU 대응

INtime은 Intel 486을 시작으로 한 영가의 CPU로부터, 최신의 초고속 CPU에 이르기까지, 모든 Intel CPU에 대해 동작이 가능합니다. Intel CPU가 가지는 하드웨어 태스크 변환 메카니즘을 남김 없이 이용하고, 지금까지 없었던 획기적 어프로치로 리얼타임 요구에 대응했습니다. Intel100%호환 CPU라면 INtime은 모두 동작 가능합니다.

■사양

우선도 레벨	(고) 0~255(저) 256 단계
스레드 스테이트	Run, Ready, Asleep, Suspend, Asleep-Suspend
커널틱	100us, 200us, 250us, 500us, 1ms, 2ms, 5ms, 10 ms로부터 선택
라운드 로빈스 케쥬링	적용 priority 해 귀의치 변경가능, 10 ms단위~100 ms까지
최대 오브젝트수	8192개
메일 박스	FIFO식/우선도식
semaphore	FIFO식/우선도식
세그먼트(segment)	1바이트~4 G바이트
물리 메모리 공간 액세스	가능 0 x00000000~0 xFFFFFFF(4096바이트 구획)
I/O공간 액세스	가능 0 x0000~0 xFFFF
아프리케이션메모리모델	32 bit 플랫폼
PCI 디바이스	이용 가능, PCI 액세스 API 장비
ISA 디바이스	이용 가능
Interupt	핸들러/스레드 구조, IRQ 공유 가능
개발 언어	Microsoft VisualC/C++
디버거	SoffScopeIII(표준 첨부)
퍼포먼스 analyzer	INScope(ver2.2001후)
네트워크	리얼타임 TCP,UDP,IP(표준 첨부), 소켓 API 장비

■ 리얼타임 성능



(Pentium800MHz에 있어서의 성능 평가 결과)

■ IN time Kernel

소프트웨어만으로 Windows의 결점을 보충합니다

INtime은 Windows에 도입 가능한 소프트웨어로서 제공되고 있어 특별한 하드웨어의 추가는 일절 필요로 하지 않습니다. INtime의 도입 후, Windows 전체는 INtime 리얼타임 커널상에서 동작하는 1개의 저우선도 리얼타임스레드로서 자리매김됩니다. 리얼타임 퍼포먼스를 개선하고 싶은 제어 처리를 INtime 리얼타임 커널상에서 동작하는 리얼타임스레드로서 독립시키는 것으로, 지금까지 실현이 어려웠던 스펙에도 견딜 만하는 WindowsPC 시스템으로 할 수 있습니다.

소프트웨어의 리얼타임 정도가 높아지는 것에 의해서, 소프트웨어로 실현될 수 있는 폭은 크게 퍼집니다. 예를 들어 전용 하드웨어를 삭감할 수 있는 등 코스트면에도 기대를 가질 수 있습니다.

256 단계의 처리 우선도 레벨

INtime 에 장비된 리얼타임 커널은, 256 계층의 priority 레벨을 서포트하고 있으므로, 소규모의 시스템으로부터 대규모 시스템까지 유연하게 설계를 할 수 있습니다. 이에 더해 다양한 프로세스간 통신 메카니즘, 동기화 메카니즘을 시스템 콜로서 제공하고 있어, 어플리케이션 개발에도 유연도가 높습니다.

신뢰 있는 리얼타임 커널을 계승

INtime은, Intel사에 의해서 만들어진 리얼타임 OS 「iRMX」의 리얼타임 커널을 계승하고 있습니다. 이 리얼타임 커널은 과거 20년 이상에 걸친 역사와 수많은 채용예/실적을 가지고 있습니다. IntelCPU를 위한 코드 최적화가, Intel사 자신의 손에 의해서 행해지고 있으므로 최고의 신뢰성과 최고의 퍼포먼스가 만들어집니다.



컴팩트한 리얼타임 커널

INtime 리얼타임 커널은 약 1 MB의 컴팩트 사이즈입니다. 쓸데 없이 시스템메모리를 압박할 것은 없고, 높은 완성도를 자랑합니다.

Intime 이 채용되고 있는 분야

<u>공장 자동화(FA)</u>	생산 라인 제어·대형 공작기계·품질관리·재고 관리
<u>산업용 기기</u>	자동 판매기·자동 개찰·도로 교통·방송 기기·발전소
<u>민생용 기계</u>	전자제품·자동차·선박·항공기
<u>군사 기기 시스템</u>	
<u>의료기기 시스템</u>	인체 검사기·집중 치료 설비
<u>경비 시스템</u>	
<u>연구기관</u>	제품 내구 시험·계측·교육 시스템

■ OSEM (OS Encapsulation Mechanism)

OS 캡슐화 메카니즘

특히 「OSEM」(OS 캡슐화 기구)은, 1 개의 CPU 상에서 Windows 와 Intime 의 동시 구동을 실현합니다. OS(Windows) 전체를 1 개로 캡슐화해 Intime 상의 저우선도 스레드로서 수중에 넣는 기술입니다. 1 개에 캡슐화된 전 Windows 프로세스/스레드는, Intime 커널에 의한 스레드 스케줄링에 의해서 실행권이 관리됩니다. 이 메카니즘에 의해서 Intime 은 Windows 가 어떠한 처리를 하고 있는 경우에서도, 순간적으로 Windows 의 움직임을 멈추어 보다 우선도의 높은 리얼 타임 처리를 확실히 선제 시킬 수 있습니다.

Intime 의 부트 업

Intime 를 도입하는 것에 의해서, Intime 커널 서비스가 추가됩니다. Intime 커널 서비스가 기동되면 **Windows 와 Intime 의 입장은 역전되고** Windows 전체는 Intime 커널에 의해서 스케줄링 되게 되어 주어진 시간(Intime 의 아이돌 시간)에 구동하는 것이 허락됩니다.



보호 기구

Windows 어플리케이션은 Windows 에 의해서 시스템 전체를 보호하는 목적으로 강고한 제약 아래 설치되고 있습니다. 구체적으로는 부정확한 메모리 조작이나, 부정확한 하드웨어 액세스가 금지되고 있습니다(링 3). 그 안전성의 반면, 하드웨어 액세스를 실시하기 위해서는 커널 모드(링 0)로 동작하는 특수한 드라이버 어플리케이션의 개발이 필요하게 되어 왔습니다.

Intime 리얼타임 어플리케이션은 링 3 으로 구동합니다만, 하드웨어 액세스나 물리 메모리에의 액세스 수단마저도 제공하고 있습니다. (치명적 영역에의 액세스는 보호됩니다). 즉, 어플리케이션의 충분한 메모리 보호를 실시하면서도 개발 효율을 높이는 것에 성공하였습니다.

Windows 크래쉬 세이프티

Windows NT 가 만일 크래쉬 했을 경우에도, Intime 환경상에서 움직이고 있는 스레드에는 계속한 리얼 타임 처리의 제공이 보증됩니다. Intime 커널은 Windows 시스템에 이상이 일어난 것을 이벤트로 확인해, 모든 Intime 어플리케이션에 사상 보고하는 것과 동시에, 캡슐화된 Windows 시스템을 스레드 스케줄링 후보로부터 제외해 대응합니다. 이것은 OSEM 에 의한 OS 분리·혼잡 구조이기 때문에 더욱 실현될 수 있는 확실한 리얼타임 보증 능력입니다.

SYSTEM CALL

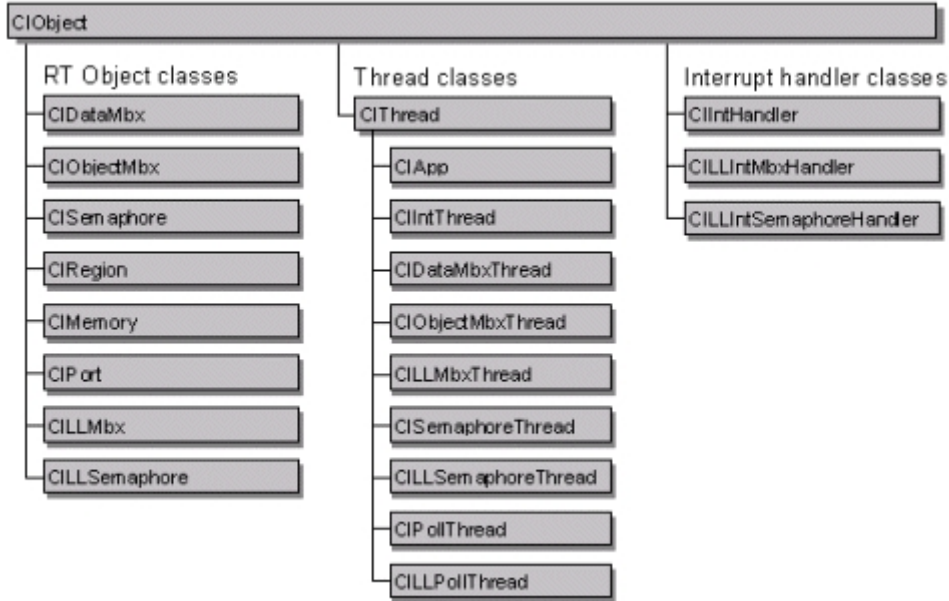
Windows 종래의 WIN32API를 사용한 Windows 어플리케이션은 GUI 어플리케이션으로서 이용합니다. INtime을 도입하면, NTX-API가 더해 이용 가능하게 됩니다. 제어·계측 처리는 INtime 커널상에서 동작하는 INtime 어플리케이션으로서 개발을 실시하고 GUI 어플리케이션은 NTX-API에 의해서 데이터 교환·이벤트 교환을 요구할 수 있습니다.

INtime API

리얼타임 성능을 100% 발휘하기 위해서, INtime 어플리케이션은 INtime 시스템 콜을 언제라도 콜 할 수 있습니다. 또, 리얼타임성을 고려 후 재설계되고 있는 부속 ANSI-C 라이브러리를 콜 할 수 있습니다. API(시스템 콜) 자세한 것은 <http://www.tpikorea.com/micronet-intime-systemcall.htm> 을 참조해 주세요.

INtime API | EC++ 클래스 라이브러리

INtime 어플리케이션의 개발은, C++언어에서도 가능합니다. 부속되어 있는 EC++ (EmbeddedC++) 클래스 라이브러리는, INtime 오브젝트 마다 클래스화 되어 있으므로, 직감적으로 알기 쉬운 프로그래밍을 가능하게 합니다.

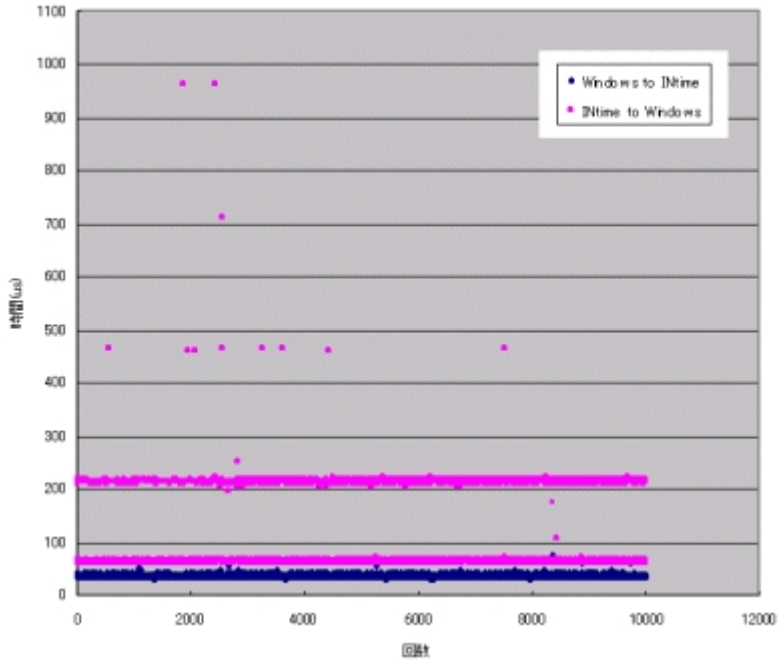


NTX API

Windows 어플리케이션에는 WIN32 시스템 콜에 가세하고, NTX 시스템 콜이 실장됩니다. NTX 시스템 콜에 의해서 Windows 어플리케이션은 INtime 시스템 콜의 일부를 자유롭게 콜 할 수 있게 됩니다. NTX 시스템 콜을 이용하는 것으로, INtime 어플리케이션과 Windows 어플리케이션의 사이에 데이터 교환이나, 이벤트 교환, 공유 메모리아크세스등을 실현될 수 있습니다. 그 후 자유로운 묘화 처리를 개발할 수 있습니다.

안정된 시스템 콜 퍼포먼스

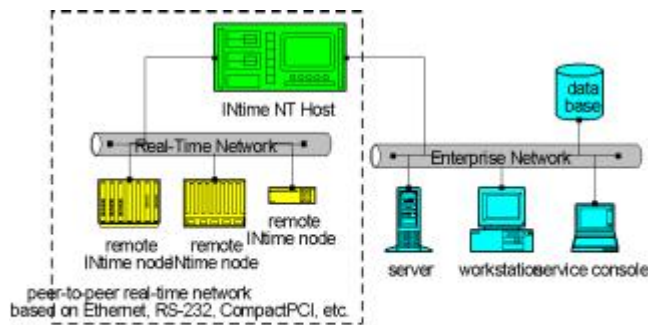
Windows 어플리케이션과 INtime 어플리케이션을 NTX 에 의해서 접속했을 때, 데이터 교환/이벤트 교환에 필요로 하는 시간을 확정적으로 할 수 있습니다. 밑그림은 Windows 어플리케이션으로부터 INtime 어플리케이션에 대한 이벤트 송신~수신에 필요로 한 시간(청색)과 INtime 어플리케이션으로부터 Windows 어플리케이션에 대한 이벤트 송신~수신에 필요로 한 시간(복숭아색)을 나타내고 있습니다. 스렛드 우선도의 관계로 청색(일반적으로 제어 처리에의 요구)은 확정적(100 마이크로 세컨드 미만), 복숭아색(일반적으로 묘화 요구)은 변동할(1 ms 미만) 가능성이 있는 것을 나타내고 있습니다.



Real Time TCP/IP Driver

INtime 에 의한 TCP/IP 통신

INtime 은 표준 제공의 리얼타임 TCP/IP 드라이버 & 스택으로, 보다 질 높은 TCP/IP 통신을 실현할 수 있는 것은 물론, 어플리케이션을 분산 제어 가능한 시스템으로 확대시킬 수 있습니다. 물론 지금까지의 네트워크 자원을 낭비하는 일은 없습니다.

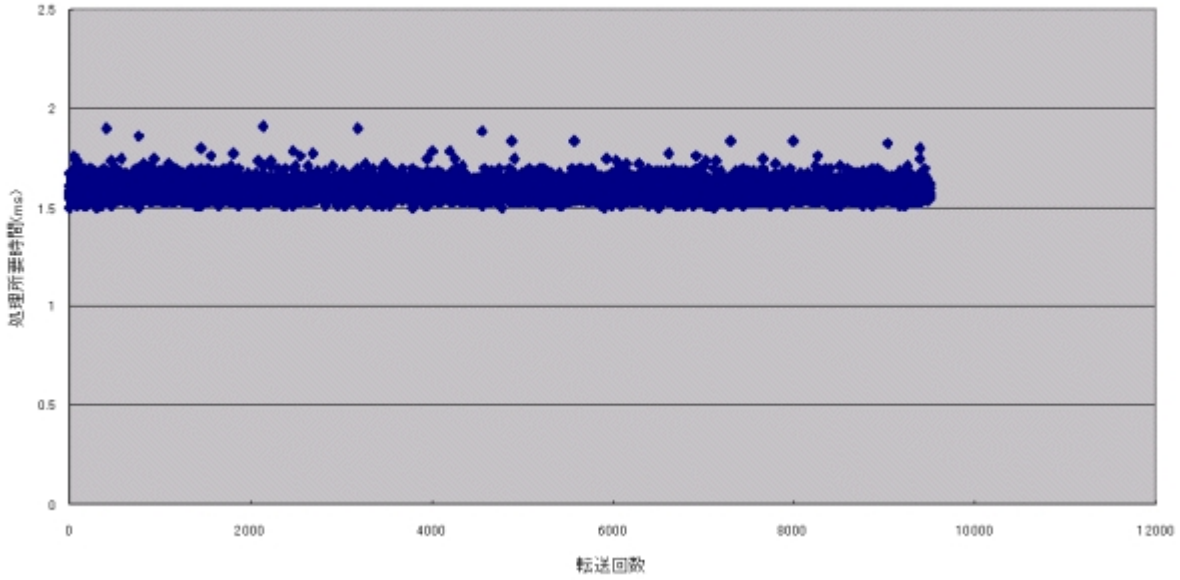


통신 퍼포먼스의 개선

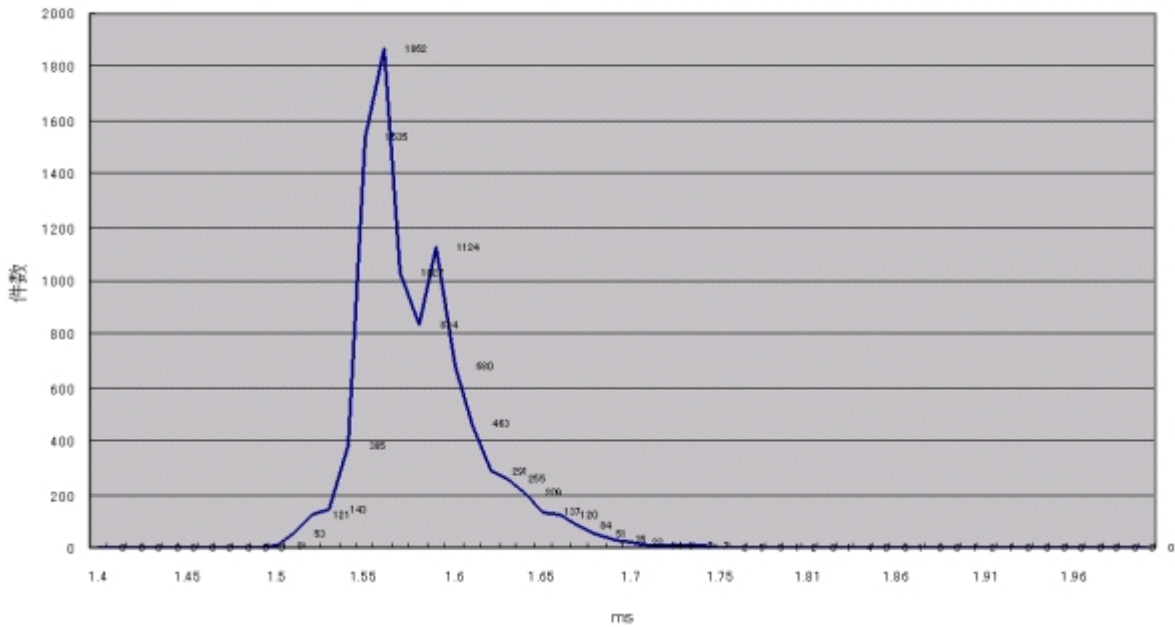
INtime 의 TCP/IP 드라이버는 리얼타임스레드로서 가동하기 때문에 Windows 의 TCP/IP(WinSock) 통신을 이용했을 경우에 비하면 성능의 안정화가 가능하며 신뢰성면에서도 향상을 전망할 수 있습니다.

2 대의 INtime 머신의 사이에 INtime TCP/IP 를 이용해 10ms 주기에 500byte 의 데이터를 교환했을 때의 송신~도달시간 격차

RT TCP/IP 送受信処理所要時間分布



RT TCP/IP 送受信処理所要時間統計



소켓 API 군

어플리케이션은 이하의 소켓 API 군이 이용 가능합니다.

```
FD_SET(n, p)
FD_CLR(n, p)
FD_ISSET(n, p)
FD_COPY(f, t)
FD_ZERO(p)
int accept (int,struct sockaddr *,int *);
int bind (int,struct sockaddr *,int);
int connect (int,struct sockaddr *,int);
int getpeername (int,struct sockaddr *,int *);
int getsockname (int,struct sockaddr *,int *);
int getsockopt (int,int,int,char *,int *);
int setsockopt (int,int,int,char *,int);
int setsockname (int,int,int,char *,int);
int listen (int,int);
int recv (int,char *,int,int);
int recvfrom (int,char *,int,int,struct sockaddr *,int *);
int recvmsg (int,struct msghdr *,int);
int select (int,fd_set *,fd_set *,fd_set *,unsigned short);
int send (int,char *,int,int);
int sendto (int,char *,int,int,struct sockaddr *,int);
int sendmsg (int,struct msghdr *,int);
int shutdown (int,int);
int socket (int,int,int);
int socketpair (int,int,int,int *);
int tcp_connect (char *,char *);
int w_setopt (int,int,int,char *,int *);
int w_getopt (int,int,int,char *,int *,int);
int socktout (int,unsigned int);
```