

「INtime documentary」 ~INtime 초보자 체험기

첫 INtime ~ 아날로그 입력·디지털 출력 기능을 INtime 시스템으로 실현할 때까지 ~



**처음에**

나는 입사 2년째의 신인 프로그래머입니다.

언어는 Visual Basic 으로 부터 시작하여, 최근 C 는 조금 정도 이며 아직 혼자서 시스템 설계를 맡았던 적은 없습니다.

이번 INtime 샘플 시스템을 제작하기에 즈음하여 생각한 것, 느낀 것을 documentary 로 정리해 보았습니다. 무엇인가 조금이나마 도움이 될 수 있으면 좋겠습니다.

**목차**

[개발·실행 환경](#)

[사양의 확인](#)

[불명점의 추출과 조사](#)

[태스크 상관도를 만들어라!!](#)

[태스크\(스레드\) 간 통신 인터페이스에 대해](#)

[INtime 어플리케이션 프로그래밍과 INtime 워저드](#)

[PCI 보드를 검출해라!!](#)

[PCI 디바이스에 다이브 하자!](#)

[아날로그 데이터 변환치 취득 스레드 완성!](#)

[데이터의 인터페이스](#)

[NT측 그래프 표시 어플리케이션](#)

[최종 체크](#)

>> INtime documentary Start!!!

## 10월 1일... 여러가지 사양

### 개발·실행 환경

OS	Windows2000
CPU	Celeron 400MHz
메모리	192MB
INtime 할당 메모리	4MB
AD/DA보드	CONTEC PCI 버스 대응 절연형 아날로그 입력 보드 AD12-16(PCI)
DO보드	CONTEC PCI 버스 대응 절연형 디지털 입출력 보드 PIO-32/32L(PCI)
DO출력 확인용 모니터	CONTEC 디지털 입출력용 신호 모니터 액세서리 CM-64(PC) E

### 사양의 확인

#### 과제

우선은 시방서의 확인. 시스템이 어떠한 움직임을 하는 것인가? 그것을 제작하는 것에 즈음해 어떠한 지식이 필요하게 되는지에 관하여 -

오늘부터 INtime을 사용한 샘플 시스템에 들어간다. INtime은 처음이니까 다소 불안하다.

우선은 사양의 확인. 이하가 나에게 줄 수 있었던 사양으로 이 사양을 기초로 설계·제작을 하였습니다. :

#### 사양 A : INtime 어플리케이션

센서 상태 8CH분의 계측치를 100 ms로 수중에 넣어 각 CH로 설정된 레벨 이상의 값을 검출했을 경우 DO출력 제어한다.

1	INtime 어플리케이션으로서 제작한다.
2	아날로그 입력 PCI 카드(AD/DA카드)를 검출할 수 있는 것과 검출할 수 없는 경우는 유사 아날로그 데이터의 생성(후술)을 서포트한다.
3	DIO 카드를 검출할 수 있을 것. 검출할 수 없는 경우는 DO신호를 출력하지 않는 모드로 동작한다.
4	아날로그 데이터의 혼잡은 인터럽트로 동작하는 것.
5	아날로그 데이터의 혼잡 주기는 100ms. 폴링을 사용하고 주기를 만든다.
6	계측되는 아날로그 데이터는 전8 CH 있어, $\pm 0 \sim +10$ V레인지로 한다.(디지털치 : 0~4095)
7	각 CH의 값이 +8.0 V이상을 넘고 있는 경우에 소정의 DO비트를 ON 한다. DO는 8비트를 사용하고 있어 비트 0(CH 0)~비트 7(CH 7)이다. 이 때의 DO포토·오프셋은 0.
8	100 ms주기에 DO8~DO15 비트를 점멸시키는 스프레드를 가질 것. DO포토·오프셋은 1.
9	수중에 넣은 계측 데이터는 100건 마다 1건분 디스크 로그를 출력한다(CH번호와 계측치(0.0 V~10.0V)), 디스크 기입은 스프레드로서 분리되어 있을 것.
10	의사 아날로그 데이터의 생성은 이하에 나타내는 : 8 CH분의 의사치 0~4095를 생성하는 스프레드를 하나 작성한다. 하나의 스프레드 중에서 100 ms, 200 ms, ... 800 ms의 값 생성 주기를 설정한다.
11	사양 A는 24시간 가동할 수 있는 것.

**사양 B : Windows NTX 어플리케이션**

사양 A 어플리케이션에 의해서 채취된 아날로그 8 CH 데이터를 리얼타임에 막대 그래프 표시하는 Windows 어플리케이션.

12	Visual C++로 작성된 Windows 프로그램.
13	INtime 커널 서비스가 기동하고 있지 않는 경우는 에러 메시지를 표시해 사양 B 어플리케이션을 종료한다.
14	사양 B 어플리케이션상에 준비된 「시작 버튼」 압하에 의해 묘화 NT스레드를 개시(통신 개시)
15	사양 A 어플리케이션이 개시되어 있지 않은 경우, 에러 메시지를 표시해 종료한다.
16	사양 A 어플리케이션으로부터 아날로그 데이터 수신을 대기한다.
17	수신한 아날로그 데이터를 기초로 해당하는 CH의 막대 그래프 표시를 갱신한다.
18	막대 그래프는 8 채널분 표시한다.
19	막대 그래프는 왼쪽에서 오른쪽 방향에 값을(0~4095 : 치가 큰 만큼 길다), 위로부터 아래방향에 CH번호 0에서 7까지 묘화 한다.
20	CH 마다 막대 그래프의 색을 달리한다(초록계의 색 : 배경색은 흑).
21	각 CH의 값이 +8.0 V를 넘을 때 DO출력을 하고 있는 것을 나타내는 붉은 램프가 점등하는 것.
22	사양 B 어플리케이션은 어떠한 타이밍에 종료해도 상관없는 것으로 하지만, 이 때 사양 A 어플리케이션의 동작을 정지시켜선 안 된다(사양 A 어플리케이션의 처리는 속행한다).
23	사양 B 어플리케이션은 24시간 가동할 수 있는 것.

**오늘의 감상**

「.....어딘지 모르게 동작은 알았습니다...」  
 <<...무엇인가를 알 수 있었을 것이다...? >>(마음의 소리)

어쨌든 INtime에 대해 거의 지식이 없고, 또 보드의 취급에 대해서도 모르는 상태로부터의 스타트였습니다. 우선 어디에 손을 대어도 좋은 것들...INtime이 앞인가, 보드가 앞인가... 그렇다고 할까...INtime이 뭐야! 시방서에 써져 모든 불분명한 말이 주마등과 같이 나의 머리 속을 뛰어 돌아다니기 시작했다...

## 10월 2일...모르는 것의 폭풍우

### 불분명점의 추출과 조사

시방서를 읽어 보면, 현재의 지식에서는 불명료한 부분이 추출되었습니다. 그러한 조사를 실시하기로 했습니다. 최초로 시방서 속에서 불분명한 부분을 추출했습니다. 그 결과는 이하입니다 :

#### INtime측

PCI 카드의 검출 방법  
DIO 카드의 검출 방법  
인터럽트에 의한 값취득 방법  
스레드 작성 방법  
유사 아날로그 데이터 생성 방법

#### NT측

INtime 커널 서비스가 기동 체크 방법?  
INtime측과 NT측과의 통신 방법.  
INtime 어플리케이션의 개시 체크 방법?  
막대 그래프 표시 방법?

### 현재의 심경

《...이만큼 불명한 부분이 있으면 시방서의 내용은 모두 불명하겠지! 반대로 알고 있는 부분을 올려 봐라!!》  
응...어쨌든 아무것도 모르지 이야기가 진행되지 않기 때문에 우선, [INtime2.10 퀵 스타트 가이드 페이지](#)를 읽기 샘플로서 쓰여져 있는 대로 샘플 프로젝트를 작성해 보았습니다. (INtime 위저드의 사용).

---INtime 위저드를 사용하고 프로젝트를 끝으면 , 몇개의 질문에 답하는 것만으로 필요한 코드를 생성해 주는---

#### 「과연...」

위저드를 사용해 포링스레드를 작성한다...로 했지만, 폴링 주기 시간등을 지정하는 것만으로 위저드가 자동으로 뼈대를 작성해 주는 것은 살아납니다. 이것으로 **스레드의 생성 방법**은 알았습니다.

위저드로 생성된 부분에서 스레드의 생성 부분은 :

```
// 포링스레드를 생성하는
strInit.hPoll1 = CreateRtThread(170, PollThread1, 4096, 0);
if (strInit.hPoll1 == BAD_RTHANDLE){
    strInit.hPoll1= NULL_RTHANDLE;
    Fail("Cannot create poll thread 1");
}
```

### 오늘의 감상

「아무튼 알고 보면, 그렇게 복잡한 것이 아니다...!(근심)」

PollThread1의 엔트리(void PollThread())를 찾으면 그 중에 스레드의 내용이 쓰여져 있는 것이 확인할 수 있었습니다(100 ms의 sleeve 처리). Windows에서도 CreateThread()는 있지만, 이것과 닮아 있어?  
아무튼 INtime이라고 해도 별로 변하지 않는다...Visual C++사용하고 있고, 별로 INtime을 의식하지 않아도 어떻게든 할 수 있을 것 같다...과 무리하게라도 사기를 올리려 했다.

## 10월 3일...지식 부족과 조사

### 불명점의 추출과 조사

이전에 디바이스 드라이버의 작성 방법을 들은 것으로부터 「디바이스 ID/벤더 ID를 사용해 보드를 검출하는 함수가 있을 것이다!!」라고 가늠해 보았습니다.

...하지만 AD보드의 메뉴얼에 2개의 정보가 쓰여지지 않은 것, 보드에 준비되어 있는 디바이스 소프트웨어가 있다고 하는 정보로부터 INtime의 API는 아니고 보드에 부속의 라이브러리를 사용하는 것인지는 안을까 생각해 버렸습니다...

그렇다고 하는 것으로 보드 벤더의 HP를 조사해 보드에 대한 액세스용 라이브러리를 열심히 찾고 있었습니다.

---INtime에서는 Windows로 사용하는 라이브러리를 사용하는 것은 일절 없습니다---

물론 벤더가 제공하는 라이브러리는 INtime용이 아닙니다...

나의 지식 부족으로부터 시간을 들여 조사를 했습니다만, 안 것은, 통상 벤더는 보드의 메뉴얼에 벤더 ID·디바이스 ID 등의 내부 정보는 게재하지 않는다고 하는 것입니다.

후에 벤더에 문의했는데 벤더 ID·디바이스 ID를 가르쳐 주었습니다. 지금까지 INtime과 Windows의 차이는 시스템 콜이 다소 다른 것만으로, 개발 환경이 같을 것이라고 생각하고 있던 나에게 있어서 새롭게 그 차이가 명확하게 되었습니다.

---INtime에서는 Windows로 사용하는 라이브러리를 사용하는 것은 일절 없습니다---

「.....」

이라고 하면...통상 Windows로 사용하는 표준 라이브러리는 어떨까...?

printf() 등 표준의 라이브러리를 INtime에서도 사용하고 있지 아니겠는가!?

(회답)

**표준 라이브러리 콜과 완전히 같은 형식인 것만으로, 내용이 리얼타임용으로 개량된 것입니다. 물론 리얼타임용의 표준 입출력 라이브러리가 있습니다.**

《Windows와는 완전히 다르잖아!!》(당연)

과연...그러나 이것으로 납득할 수 있다...

INtime 어플리케이션의 개발 환경은 Windows의 개발 환경과 완전히 같고 추가 라이브러리, 인클루드 파일은 완전히 다른 것이 들어가 있던 것이다!! 이것은 Visual Studio로 INtime 위저드 사용시의 컴파일·링크 설정과 통상 Windows Console 어플리케이션 작성시의 컴파일·링크 설정이 완전히 차이가 난일로 밝혀졌습니다.

나의 실패를 억지 쓴다면, INtime 어플리케이션이 Windows 어플리케이션과 같이 작성 가능하게 되기 위해서(전혀 INtime를 의식하지 않고), 외형에는 차이를 인식하는 부분이 적다!

그 때문에, INtime과 Windows라고 하는 경계선이 보였던 것이다...!(근심).

### 오늘의 감상

INtime과 Windows의 차이를 똑똑히 느끼는 나였습니다...

## 10월 4일...플로차트(flow chart)적 태스크 상관도

태스크 상관도를 만들어라!!

### 과제

태스크 상관도...태스크...일...일의 상관 관계도...

별로 편이라고 하지 않습니다만, 나름대로 해석해 작성해 보려고 합니다.

요전날의 조사에서 벤더 ID, 디바이스 ID를 입수했습니다.

INtime 어플리케이션에서는 INtime이 제공하는 라이브러리를 사용하는 것」이 말을 가슴에, INtime Help를 열어 보면... 거기에는 PciFindDevice()라고 하는 API가 있었습니다.

이것에 의해 PCI 디바이스의 검출이 가능하다고 하는 것을 알 수 있었습니다.

PCI 디바이스의 검출

클리어!!

점점 불분명점이 해소되어 가는 가운데 매니저로 부터 태스크 상관도의 작성을 명령받았습니다.

「태스크 상관도.....? 태스크는 무엇입니까?」

태스크 상관도에 대해 지식이 없었던 나는 인터넷으로 조사해, 거의 없는 정보로부터 있는 대답을 찾아내 납득했습니다.

태스크 상관도란... :

「multi-thread, 멀티-프로세스에 대해서, 복수의 태스크(스레드) 동작의 상관 관계를 나타내는 다이어그램」

즉 이번 시스템을 태스크로서 개념 붙여 그 상관 관계를 나타낸 도표를 그리면 좋다!라고 생각해, 조속히 작업을 시작했습니다.

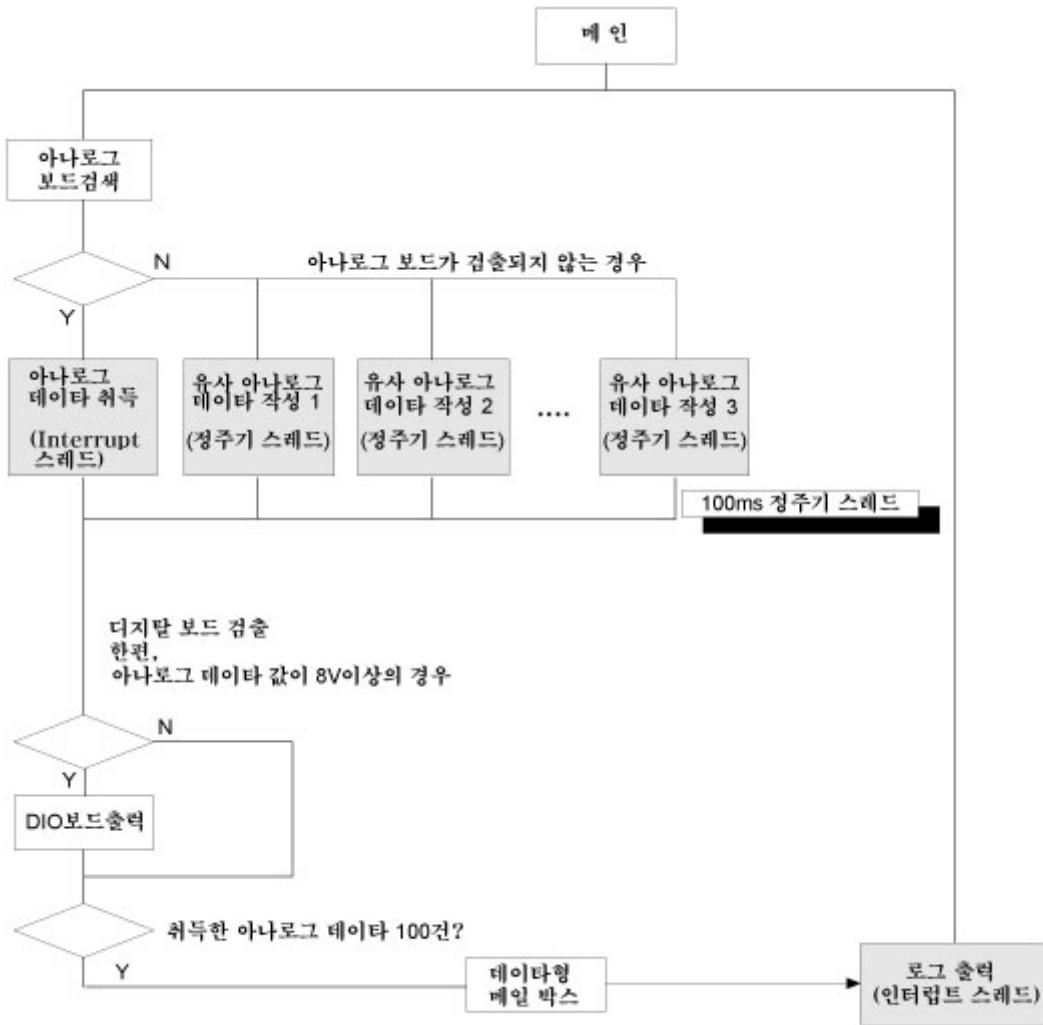
우선은 어떠한 태스크(일)가 있는지를 추출해, 그러한 태스크(일)의 상관관계를 우선 작성해 보았습니다... (불안)

### 태스크의 추출

- 초기화
- 아날로그 보드의 검출
- 초기화
- 디지털 입출력 보드의 검출
- 초기화
- 로그 출력
- 의사 아날로그 데이터 작성(8)
- 디지털 입출력 보드 출력
- 아날로그 데이터 수중에 넣어

**최초로 작성한 태스크 상관도...**

이러한 태스크(일) 상관관계라고 하는 것은 이러한 것입니까...?



처음으로 작성한 태스크 상관도

**도중 경과**

«.....조카! 이것은 플로차트(flow chart)야!!»

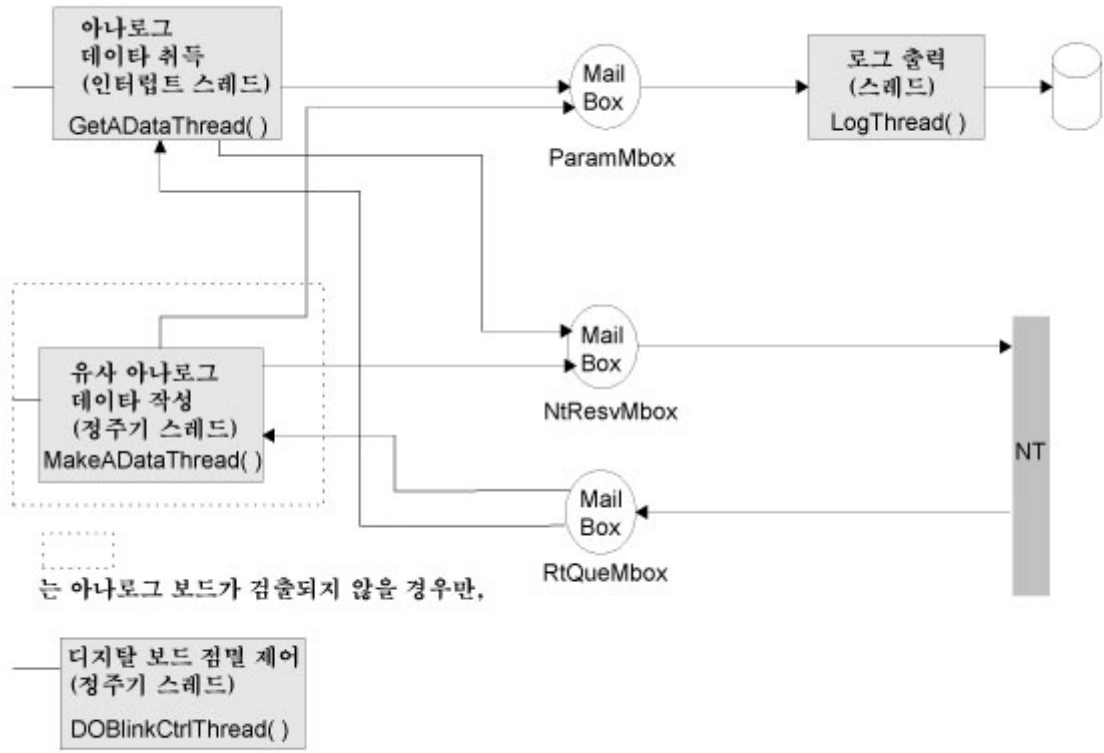
...그렇지만! 어쩔 수 없다! 우선 제출하지 않으면, 태스크 상관 도표를 그려 제출하지 않으면... 그렇다고 하는 것으로 이것을 제출했습니다...

그러자면 매니저로부터 되돌아 온 대답은... :

「.....이것은 플로차트(flow chart)예요!!」

역시!!어떤 것이 태스크 상관도라고 부를 수 있는 것인가 모르는 나는 참고가 되는 자

**최종적으로 작성한 태스크 상관도**



최종적으로 작성한 태스크 상관도

**스레드간의 데이터의 교환(메일 박스를 사용)**

**스레드(태스크) 수 : 4**

- 아날로그 데이터 혼잡 스레드      GetADDataThread
- 의사 아날로그 데이터 생성 스레드      MakeADDataThread
- 디지털 보드 점멸 제어 스레드      DOBlinkCtrlThread
- 로그 출력 스레드      LogThread

**메일 박스수 : 3**

- 로그 출력 데이터 송신용 메일 박스 :      ParamMbox
- RT/NT 인터페이스용 메일 박스 :      NtRecvMbox
- NT/RT 인터페이스용 메일 박스 :      RtQueMbox

**오늘의 감상**

이 태스크 상관도를 작성하기까지는 실제로 퇴고를 반복해, 7회 정도 수정했습니다. 태스크 상관도를 작성하는데 다소 시간이 걸렸습니지만, 이것을 작성하는 것으로써 스레드간의 데이터의 교환(메일 박스를 사용)등의 방법을 알 수 있었습니다

10월 7일...메일 박스에는 종류가 있다.

태스크(스레드) 간 통신 인터페이스에 대해

### 과제

Windows 어플리케이션에서는 메일 박스나 메시지에 의해 스레드간의 통신을 실시하는 수법이 있었습니다. INtime의 통신 인터페이스는 어떠한 것인가 상세를 조사하고 싶습니다.

### 통신 오브젝트 : 메일 박스

오브젝트형 메일 박스...오브젝트 핸들의 송수신을 실시합니다.  
데이터형 메일 박스.....128 BYTE까지의 데이터를 송수신 합니다.

### 조사 결과

메일 박스는 데이터를 받아 넘기는 것이 가능하므로 메일 박스를 사용하는 것이라고 짐작을 했습니다. 거기서 아날로그 데이터 취득 또는, 유사 데이터 작성 스레드와 로그 출력 스레드와의 통신 수단을 데이터형 메일 박스로 작성하기로 했습니다. 로그 출력 스레드에 송신하는 데이터 사이즈(WORD×8)가 작기 때문입니다. NT어플리케이션과의 통신 오브젝트는 오브젝트형 메일 박스를 사용하기로 했습니다. 이것은 INtime의 샘플 시스템으로서 2 종류의 메일 박스를 사용하는 것이 유효하다고 생각했기 때문입니다.

### 오늘의 감상

대략의 태스크(스레드)의 흐름과 그러한 통신 수단을 결정했더니, 이전보다 시스템 전체를 바라볼 수 있게 된 것 같았습니다. 상세한 부분(메일 박스의 생성 방법이나 AD보드의 액세스 방법등)에 대해서는 아직도 수수께끼를 남기고 있습니다만, 제작을 시작하기로 했습니다.

## 10월 8일...원시 코드 생성 마술사(INtime 위저드)

### INtime 어플리케이션 프로그래밍과 INtime 위저드

#### 과제

INtime 어플리케이션의 작성 수법으로 INtime 위저드를 사용하는 것으로써 베이스의 뼈대가 자동 생성됩니다. 여기로부터 제어의 종류나 방법에 의해 수정을 해 나간다고 하는 것이 일반적인 방법이라고 알았습니다.

우선은 INtime 위저드로부터 프로젝트를 시작해 보고 싶습니다.

태스크 상관도가 완성되었으므로, 프로그래밍 작업에 들어갔습니다.

우선 프로젝트 작성.

INtime 위저드를 사용해 아날로그 데이터 혼잡용 스레드, 유사 아날로그 데이터 작성 포링스레드, 로그 출력 스레드를 작성했습니다.

이하대로 :

#### 끼어들어 처리

##### 아날로그 데이터 혼잡용 스레드

Interrupt 레벨	PCI based
벤더 ID	1221
디바이스 ID	8153
Interrupt 공유	(없음)
시그널 방식	핸들러와 Interrupt 스레드
대기 시간	WAIT_FOREVER
Interrupt 큐	1

#### 규칙적 주기에 기다리는 스레드

##### 의사 아날로그 데이터 작성 스레드

웨이트 방식	SLEEP
대기 시간	100ms
스레드프라이오리티	170
스택 사이즈	4096

#### 메일 박스를 기다리는 스레드

##### 로그 데이터 출력 스레드

스레드가 기다리는 오브젝트형	데이터형 메일 박스
대기 시간	100ms
스레드프라이오리티	160
스택 사이즈	4096

#### 실행

우선 상기의 스레드를 생성하기로 했습니다.

상기속에서 불분명했던 부분은 스레드프라이어리티와 스택 사이즈의 지정이었지만 우선 신경쓰지 않고 INtime 위저드를 종료하니 소스 코드가 생성되었습니다.

작성시에는 각 각의 오브젝트명이 PollThread1나 Server1 등 디폴트명으로 결정되므로, 소스 생성 후 그러한 이름을 태스크 상관도에 준거해 변경했습니다.

```

/*****
FUNCTION:          PollThread1
* PARAMETERS:          없음
* RETURNS:          없음
* DESCRIPTION: 포링스레드 # 1.
*****/

void    PollThread1(void)
{
#ifdef _DEBUG
    printf("Poll 1 started\n");
#endif

    while (1)
    {
        RtSleep( 1000);

#ifdef _DEBUG
        printf("Poll 1 waking up\n");
#endif

        // TODO: 1000 밀리 세컨드 마다 반복하는 처리를 배치합니다

    }
}

```

포링스레드 생성시에는 디폴트로 상기와 같은 코드로 생성됩니다  
(폴링 주기 1000 ms지정시)

### 오늘의 감상

대기 시간등을 설정하면 자동으로 INtime 위저드가 환경이나 그 외 처음의 뼈대를 작성해 주므로 편했습니다.  
스레드의 기본 구조가 소스 코드로 생성되고 있으므로, 나머지 처리는 그 안에 기술하는 것만으로도 되었습니다.

## 10월 9일...인터페이스 보드에 액세스

### INtime 어플리케이션 프로그래밍과 INtime 위저드

#### 과제

드디어 좋은 징조가 생겼으므로, 처리를 프로그래밍 해 나갑니다만, 인터페이스 보드에의 액세스 부분은 다소 귀찮은 이미지가 있었습니다.

도대체 어떠한 방법으로 보드에 액세스 하는지 전혀 검토도 하지 않고 우선 조사와 작업을 진행시켜 나갑니다

아날로그 데이터를 디지털로 변환하고 그 데이터를 취득하는 것은 인터페이스 보드에 액세스 하지 않으면 안됩니다 (당연).

이라고 하는 인터페이스 보드에의 액세스는 어떠한가?

그 액세스에는 어떠한 처리가 필요한가...얼마 안되는 자료와 인터넷, 보드의 매뉴얼, [HP](#)가 아군입니다.

보드의 매뉴얼을 보면 보드의 동작 설정 순서가 게재되고 있었습니다.

우선은 보드의 초기화 처리가 필요하게 됩니다. 보드의 초기화 처리를 실시하기 위해서는 보드에 액세스 하지 않으면 안됩니다.

INtime으로 PCI 보드의 처리를 실시하는 API가 있는지, **INtime Help**로 조사하기로 했습니다.

[스타트->프로그램->INtime V2.10->INtime Help](#)

를 선택해, INtime on-line help를 열어, 키워드를 "PCI" 라고 입력했습니다.

그러자 "PCI Library Call"이 항목으로서 표시되어 [INtime에 PCI 라이브러리 콜](#)이 있는 것을 알았습니다.

#### PCI 라이브러리 콜

```
PciInitialize  
PciReadHeader  
PciFindDevice  
PciFindClass  
PciSetConfigRegister  
PciGetConfigRegister  
PciVendorName  
PciDeviceName  
PciClassName
```

현재의 심경

《.....!!! **PciFindDevice()**.....이것인가!? 이것인가!》

한층 더 자세하게 보면... :

#### 문장구조법

```
YTE PciFindDevice(  
PCIDEV *pPciDevice  
);
```

#### 파라미터

*pPciDevice* PCIDEV 구조체의 포인터 ; 상술 대로에 초기화됩니다.

#### 반환값

##### 성공

디바이스가 검출되었을 경우, TRUE 를 돌려줍니다. PCIDEV 구조체에는 PCI 디바이스 헤더의 값이 지정됩니다.

##### 실패

디바이스가 검출되지 않았던 경우, FALSE 를 돌려줍니다.

#### 오늘의 감상

##### 이것이다!!

(←감격의 눈물)

한층 더 INtime 위저드로 코드를 생성하면 util.c라고 하는 모듈이 자동적으로 생성되어 그 중에 GetPciInterruptLevel()라고 하는 함수가 있는 것을 알았습니다.

**그 함수(유틸리티) 중(안)에서 확실히, PciFindDevice()를 콜 하고 있지 않습니까!!**

이것으로 나는 확신했습니다.

**「PCI 보드는 PciFindDevice에 의해 검출할 수 있다!!」**

《기다려!!최초부터 util.c의 내용을 좀 더 잘 조사하고 있으면, 이렇게 조사할 필요도 없었기 때문에는...

최초의 그 어려운 이미지는 도대체였던 것일까...?》(결과 올 라이트)

## 10월 10일...PCI 보드를 검출할 수 없다!

### PCI 보드를 검출해라!!

어떻게 했던 말인가?! PciFindDevice() 어째서 검출할 수 없는 것인지?

왜?

보드는 제대로 삽입하고 있는데...

테스트를 위해 INtime 위저드로 자동 생성되는 유틸리티 콜 etPciInterruptLevel()로 시험해 보았지만, 결과는 마찬가지로 지었던 것입니다..

이하는 GetPciInterruptLevel()입니다 :

```
/*
 *
 * FUNCTION: GetPciInterruptLevel
 *
 * PARAMETERS: 1. PCI Vendor ID
 *              2. PCI Device ID
 *
 * RETURNS: encode 된 인터럽트 레벨, 또는 실패시 0xffff
 */
WORD GetPciInterruptLevel(
    WORD wPciVendorId,
    WORD wPciDeviceId)
{
    static const BYTE irq2level[] = {
        IRQ0_LEVEL, IRQ1_LEVEL, IRQ2_LEVEL, IRQ3_LEVEL,
        IRQ4_LEVEL, IRQ5_LEVEL, IRQ6_LEVEL, IRQ7_LEVEL,
        IRQ8_LEVEL, IRQ9_LEVEL, IRQ10_LEVEL, IRQ11_LEVEL,
        IRQ12_LEVEL, IRQ13_LEVEL, IRQ14_LEVEL, IRQ15_LEVEL };
    PCIDEV dev;

    dev.wVendorId = wPciVendorId;
    dev.wDeviceId = wPciDeviceId;
    dev.wDeviceIndex = 0; // 처음에 검출된 디바이스
    if (!PciFindDevice(&dev))
        return 0xFFFF; // 디바이스는 발견되지 않는다

    if (dev.byIntLine > 15)
        return 0xFFFF; // 이 디바이스에는 IRQ가 없는지, 인정받지 못한 IRQ치인
    return irq2level[dev.byIntLine];
}
```

이 함수의 제일 인수는 벤더 ID, 제2 인수는 디바이스 ID입니다. 나는 이 함수에 직접 벤더 ID, 디바이스 ID를 지정했던 :

```
WORD intLevel;
intLevel = GetPciInterruptLevel( 1221, 8153 );
```

### 도중 경과

결과 0xffff가 반환되었습니다. 어째서일까요?

(회답) 벤더 ID, 디바이스 ID는 올바른가요? CONTEC PCI 버스 대응비절연형 다채널 아날로그 입력 보드 AD12-16(PCI)의 벤더 ID는 0x1221, 디바이스 ID는 0x8153예요.

그러니까 벤더 ID : 1221, 디바이스 ID : 8153에...(은)는!?!

《또...또 해 버렸다!!》

0x1221(16진수) = 4641(10진수) 나의 지정한 수치 : 1221(10진수)

0x8153(16진수) = 33017(10진수) 나의 지정한 수치 : 8153(10진수)

「 아직 10진수와 16진수의 차이에 자연에 대응할 수 없는 나의 부주의로 의한 실수였습니다...

## INtime 위저드의 세치기 처리 다이얼로그

여기서 입력한 값으로 10 진수, 16 진수를 의식하지 않고, 벤더 ID(Vendor ID) 1221, 디바이스 ID(Device ID) 8153으로 입력해 버리고 있었지요.

이 후 올바르게 지정하면, 완전히 문제 없게 보드를 인식하고 있는 것을 알았습니다.  
자동 생성된 원시 코드상에서는 :

```

/*****
*
* FUNCTION:   nt1Init
*
* PARAMETERS: 없음
*
* RETURNS:   BOOL 처리의 성공을 의미합니다
*
* DESCRIPTION: 인터럽트를 취급할 수 있는 상태로 설정합니다
*               인터럽트 처리기와 끼어들어 스레드를 사용합니다
*****/

BOOL Int1Init(void)
{
    wLevel          = GetPciInterruptLevel(0x1221,0x8153);
    if(0xFFFF == wLevel)           여기서 지정해 있는
        return FALSE;

    SetRtProcessMaxPriority(NULL_RTHANDLE, 0);

    return (CreateRtThread(0, Int1Thread, 4096, 0)
            != BAD_RTHANDLE);
}

```

### 결과

그러면 「인터럽트 레벨」로 지정하는 것 "PCI based"란?

#### (회답)

PCI 버스 디바이스의 경우, 그 동작 모드에 의해 OS가 PCI 버스의 IRQ가 동적으로 설정하는 경우가 있습니다. "PCI based"에서는 PCI의 인터럽트 레벨을 취득하는 함수 GetPciInterruptLevel()를 사용해 IRQ 레벨을 취득할 필요가 있습니다.

「과연...」

**10월 11일...PCI 보드 검출의 의미는!?**

**PCI 보드를 검출해라!!**

PCI 디바이스의 검출에 대해 납득했습니다만, 끝인가? PCI 디바이스를 검출하고, 이외에 어떻게 하는지? 이것에 대하여 매니저에게 물었는데..

「PCI 디바이스를 검출한다(PciFindDevice에 의해)라고 하는 것은 그 디바이스의 I/O영역에 대해서 입출력을 할 수 있는 것이니까, 디바이스의 특정하는 레지스터 영역으로부터 메모리 상태를 읽어들이거나 혹은, 쓰거나 하는 것에 의해서, PCI 디바이스를 제어합니다...」

「.....후.....」  
 그 후, 매니저는 어쩐지 바스락바스락, PCI 버스의 상세한 것에 대하여 기술이 있는 책을 꺼내, 설명을 시작했습니다.

「이것은 PCI 디바이스의 컨피규레이션 레지스터의 구조를 알기 쉽게 테이블로 한 것이니까 참고에...」

「.....알기 쉽고.....?」      <<...어디가...?>>

31	16	15	0	
디바이스 ID		벤더 ID		00h
스태이터스		커멘드		04h
클래스·코드			리비전 ID	08h
BIST	헤더·타입	지연시간·타이머	캐쉬·라인·사이즈	0Ch
베이스·주소·레지스터				10h
				14h
				18h
				1Ch
				20h
				24h
예약				28h
예약				2Ch
확장 ROM 베이스·주소				30h
예약				34h
예약				38h
최대 지연시간	최소 그랜트	인터럽트·핀	인터럽트·라인	3Ch

**도중 경과**

「.....이 안에서 알고 있는 것이라고 말하면 「벤더 ID」와 「디바이스 ID」 정도의 것 밖에 없어요.....」  
 「그러면 이렇게 되면 알기 쉬워져? 밑그림은 이전에 사용한 PciFindDevice로 인수로서 건네주는 구조체의 상세를 나타낸 것이지만, 무엇인가 이미지가 솟지 않아?」

「무엇인가...좋은-글자...확실히 벤더 ID, 디바이스 ID...」

**와! 이것은 PCI 디바이스의 컨피규레이션 레지스터의 구조와 닮아 있군요!」**

```

typedef struct {
    WORD wBusNum; .....버스 번호 입력 필드
    WORD wDeviceNum; .....디바이스 번호 입력 필드
    WORD wFunction; .....평선 번호 입력 필드
    WORD wVendorId; .....벤더 ID          입력 필드
    WORD wDeviceId; .....디바이스 ID      입력 필드
    WORD wDeviceIndex; .....디바이스 검색 번호
    WORD wCommand; .....커맨드
    WORD wClassId; .....클래스·코드
    BYTE byInterfaceId; .....클래스·코드
    BYTE byRevId; .....리비전 ID
    BYTE byCLS; .....캐쉬 라인 사이즈
    BYTE byLatency; .....지연시간·타이머
    DWORD dwBaseAddr[6]; .....베이스 어드레스 레지스터
    DWORD dwCIS;
    WORD wSubSystem VendorId;
    WORD wSubSystemId;
    DWORD dwRomBaseAddr; .....확장 ROM 베이스 주소
    BYTE byIntLine; .....인터럽트·라인
    BYTE byIntPin; .....인터럽트·핀
    BYTE byMaxLatency; .....최대 지연시간
    BYTE byMinGrant; .....최소 그랜트
} PCIDEV, *LPPCIDEV;

```

### 오늘의 감상

「그렇습니다, PciFindDevice 는 이 구조체에 벤더 ID 와 디바이스 ID 의 정보를 격납해, PCI 디바이스를 확정하고 그 디바이스의 컨피규레이션 레지스터의 값을 이 안에 쓰고 있습니다。」

「.....」

「PCI 디바이스의 선두로부터 64바이트는 컨피규레이션 레지스터 공간 헤더라고 하고, PCI 버스의 사양으로 결정되고 있는 영역입니다.

벤더가 고유의 커스터마이징 하는 영역은 베이스 주소로 부터랍니다。」

「.....」

「즉, PCI 버스의 사양으로, 통상 벤더가 I/O공간으로서 혹은 메모리 공간으로서 구축할 수 있는 영역은 베이스 어드레스 레지스터로부터 입니다。」

「즉...?」

「즉 베이스 어드레스 레지스터가 메뉴얼로 말하는 I/O베이스 주소입니다。」

「홀름하다...!!! 명백! 그 말은 메뉴얼에 써 있는 0+8h 라든지 0+0 Ch는 베이스 주소로부터 몇 번지 떨어져 있는지(오프셋치)를 나타내고 있던 것인가!

즉, 보드 검출시에 취득해야 하는 것은...베이스 주소!

...베이스 주소라고 하면..dwBaseAddr[0];입니까!」

「그렇네요...그것과 또 하나 있습니다. 인터럽트 레벨입니다...AD보드는 인터럽트를 사용하는 거죠?」

「과연...!!」

## 10월 14일...베이스 주소가 어긋난다고!

### PCI 보드를 검출해라!!

PCI 디바이스의 검출 논리부에서, GetPciInterruptLevel()를 카피하고 수정해 인터럽트 레벨 외에 베이스 주소를 취득하도록 프로그램 한 함수입니다(FindAnalogBoard())

```
/******  
/* 함수명 : FindAnalogBord */  
/* 설명 : 아날로그 보드 검색 */  
/* 인수 : 없음 */  
/* 반환값 : 정상 : TRUE 에러시 : FALSE */  
/******  
BOOL FindAnalogBord(void)  
{  
    PCIDEV PciDevInfo; /* PCI Device Infomation*/  
    BYTE ret; /* PCI 디바이스 검색용 변수*/  
    static const BYTE irq2level[] = /* 인터럽트 레벨*/  
    {  
        IRQ0_LEVEL, IRQ1_LEVEL, IRQ2_LEVEL, IRQ3_LEVEL,  
        IRQ4_LEVEL, IRQ5_LEVEL, IRQ6_LEVEL, IRQ7_LEVEL,  
        IRQ8_LEVEL, IRQ9_LEVEL, IRQ10_LEVEL, IRQ11_LEVEL,  
        IRQ12_LEVEL, IRQ13_LEVEL, IRQ14_LEVEL, IRQ15_LEVEL  
    };  
    strInfo.analogAdd = 0;  
    /* 벤더 ID 와 디바이스 ID 를 대입*/  
    PciDevInfo.wVendorId = ANALOG_VENDORI.....(1)  
    PciDevInfo.wDeviceId = ANALOG_DEVICEI.....(2)  
    PciDevInfo.wDeviceIndex= 0;  
    /* +++ 디바이스 검색 +++*/  
    // 벤더 ID, 디바이스 ID 보다 PCI 보드의 검색을 실시하는  
    ret = PciFindDevice(&PciDevInfo.....(3)  
    /* +++ 디바이스 검색에 해당하지 않았는지 || IRQ 판정 +++*/  
    if( (ret == FALSE) || (PciDevInfo.byIntLine > 15).....(4)  
    {  
        /* 디바이스는 발견되지 않기 때문에 에러 코드를 돌려주는*/  
        return FALSE;  
    }  
    /* PCI 컨피규레이션 레지스터 기입*/  
    PciSetConfigRegister(&PciDevInfo,0x04,T_BYTE,0x01);  
    /* 주소 취득 */  
    strInfo.analogAdd = (WORD)PciDevInfo.dwBaseAddr[0.....(5)  
    printf("Analog Board Base Address=%4x\n",PciDevInfo.dwBaseAddr[0]);  
    // 인터럽트 레벨을 설정하는  
    strInfo.wIntLevel = irq2level[PciDevInfo.byIntLine.....(6)  
    printf("Interrupt Level=%4x\n",PciDevInfo.wIntLevel);  
    return TRUE;  
}
```

strInfo은 글로벌 데이터 구조체

- (1) 벤더 ID를 PCIDEV 구조체에 격납
- (2) 디바이스 ID를 PCIDEV 구조체에 격납
- (3) PciFindDevice() 콜
- (4) PCI 디바이스의 검출을 판별
- (5) 베이스 주소를 취득
- (6) 인터럽트 레벨을 취득

## 도중 경과

새롭게 김출 논리부를 이 논리로 변경해 실행해 보았는데

Analog Board Base Address=C001  
Interrupt Level=0020

로 주소가 반환되었습니다.

«이것으로 베이스 주소를 잡았고, 인터럽트 레벨도 취득할 수 있었다»

우선, 여기까지 왔으므로 한 번 매니저에게 보고하기로 했습니다. 매니저에게 현재의 상황을 설명해, 현코드를 실행해 보이면...

「저것? 베이스 주소는 이것으로 좋은 것인지?」

«이봐 이봐, 더 이상은 할 수 없을 정도로 제대로 사용할 생각이야~...!!»

조용히 PCI 버스 상세책을 꺼내,

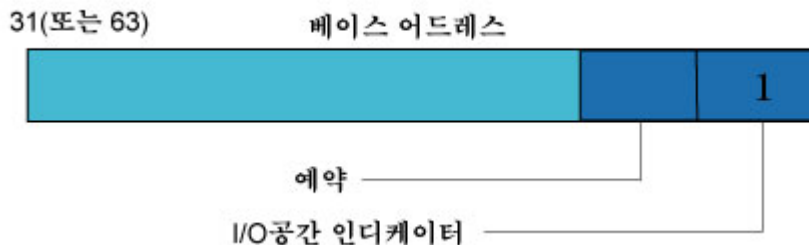
「이것에 대해 말하는 것을 잊고 있었다...!」 「.....」

«네~...무엇에 대해...?»

「베이스 레지스터에는 I/O 공간인가, 메모리 공간인가를 식별하는 플래그(I/O 공간 인디케이터(indicator))가 있습니다...」

「...」

### 【I/O공간용 베이스-어드레스 레지스터의 경우】



## 결과

「AD보드의 경우, 여기는 I/O공간이기 때문에 베이스 주소의 말미 비트에 1이 끊고 있을 것입니다。」

「...과연...그래서 C001과 1이 말미 비트에 1이 끊고 있는 것이군요...그리고 그러면 어떻게 하면 좋은 것일까요?」

「단순하게 취득한 베이스 주소의 말미 2자리수(이 경우 예약 부분도 불투명하기 때문에)를 마스크 프로세싱 할 방향으로 좋은 것이 아닙니까?」

「과연...그 말은, 이 비트를 0으로 하도록(듯이) 하려면...」

«.....마스크 : 1111111111111100 이것은.....0xFFFC.....?»

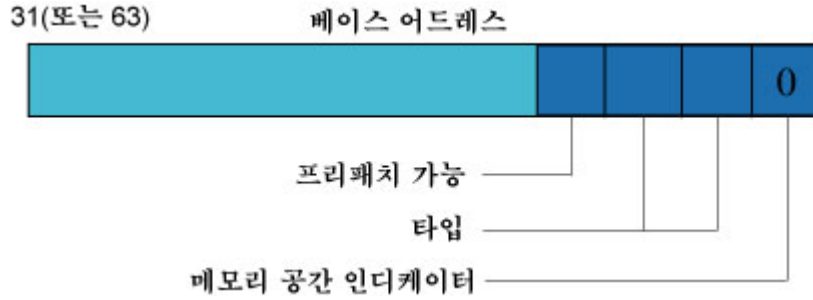
```
strInfo.analogAdd = (WORD) PciDevInfo.dwBaseAddr[0] & 0xFFFC;
```

「빙고입니다!」

「이 값을 취득한 베이스 주소와의"앤드"를 취하면 좋습니다...와 구!」

**메모리 공간용 베이스 어드레스 레지스터의 경우**

베이스 어드레스 레지스터의 경우, 메모리 공간, I/O공간인지를 식별할 필요가 있습니다. 말미 비트가 0의 경우, 메모리 공간으로서 사용되고 있습니다. 말미 비트가 1의 경우, I/O공간으로서 사용되고 있습니다.



메모리 공간용 베이스 주소의 경우는 말미 비트가 메모리 공간용 인디케이터(indicator)이며, 이 값이 0이 됩니다

## 10월 15일...PCI 디바이스의 메뉴얼의 견해

### PCI 디바이스에 다이브 하자!

#### 과제

PCI 디바이스의 검출을 할 수 있었습니다. 드디어 PCI 디바이스에 액세스 해 제어를 실시하는 것입니다만, AD보드에 대해서 실시하는 것은 아직도 완전히는 불명합니다. 우선은 메뉴얼안을 들여다 보기로 했습니다.

드디어 아날로그 데이터 취득 부분의 조사에 들어가 코어인 영역에 발을 디뎠는가! 와! 감개 깊은 것이 있습니다. 이번에 사용하는 AD/DA 보드는 CONTEC 사제 비절연형 아날로그 입력 보드 AD12-16(PCI)입니다. PC에 인스톨하는 방법, 보드의 설정 방법, Windows용 드라이버의 셋업 방법등 우선 대충 대충 훑어보고, 나의 센서에 걸리는 키워드까지는 우선 읽어 진행시켜 나갔습니다.

도중의 신호 배치나, 입력 신호의 접속 등 모르는 것은 많이 있었지만, 그것들은 메모장에 키워드로서 남겨 우선 먼저 읽어 진행하는데 전념했습니다.

메뉴얼의 중반 정도입니다. 이것은 관계가 깊다! 라는 부분을 어딘지 모르게 알았습니다.

#### 《제목에 「기능과 조작 순서」라고 쓰고 있으면 불응한 것도...》

...어쨌든 읽어 진행시켜 나가는 곳의 AD/DA 보드에는 아날로그 입력 기능의 변환 모드로서 두 개의 모드, 샘플링 채널의 지정 방법으로 두 개의 모드가 각각 있는 것이라고 하는 것을 알았습니다.

#### 변환 모드

- 소프트웨어 모드... 샘플링 개시 커맨드로 지정 채널의 샘플링을 1회 실시
  - 클락 모드..... 샘플링 클락에 동기 해 지정된 채널을 정기적으로 샘플링을 실시
- \*샘플링 클락으로서 내장 시계와 external clock를 사용할 수 있다.

#### 채널의 지정 방법

- 싱글 채널 모드.....1회의 샘플링 동작으로 지정된 1개의 채널을 변환
- 멀티 채널 모드.....1회의 샘플링 동작으로 지정된 복수의 채널을 변환

#### 도중 경과

이 모드로부터 어떤 것을 선택해야 하는 것인가?

모드의 선택 종별은 4 방법이므로 메뉴얼에 게재되고 있던 그림을 참고에 이번 사용을 대조해 보았다. 사양에서는 「센서 8 CH분의 계측치를 100 ms로 수중에 넣어...」

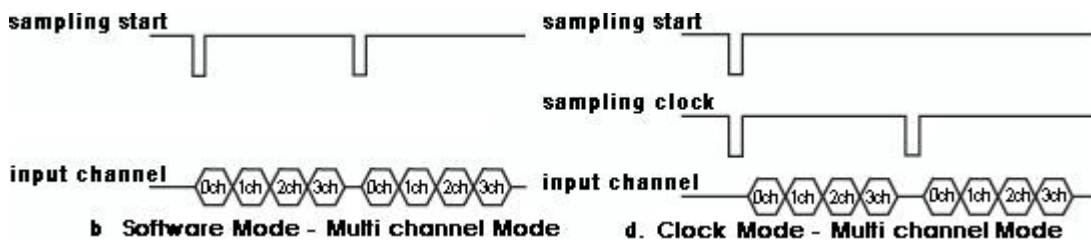
어떤의 것으로, 싱글 채널 모드는 아닐 것이라고 우선 예상을 세웠습니다.

두 개의 케이스로 좁혀집니다...

소프트웨어 모드.멀티 채널인가, 클락 모드 멀티 채널인가?

우선 이번은 샘플링 클락 되는 것을 사용하지 않는 것, 또 소프트웨어(INtime)로 샘플링 주기를 만들기 때문에...소프트웨어 모드!! 그렇다고 하는 간편한 결과로 소프트웨어.멀티 채널 모드라고 하기로 결정했습니다.

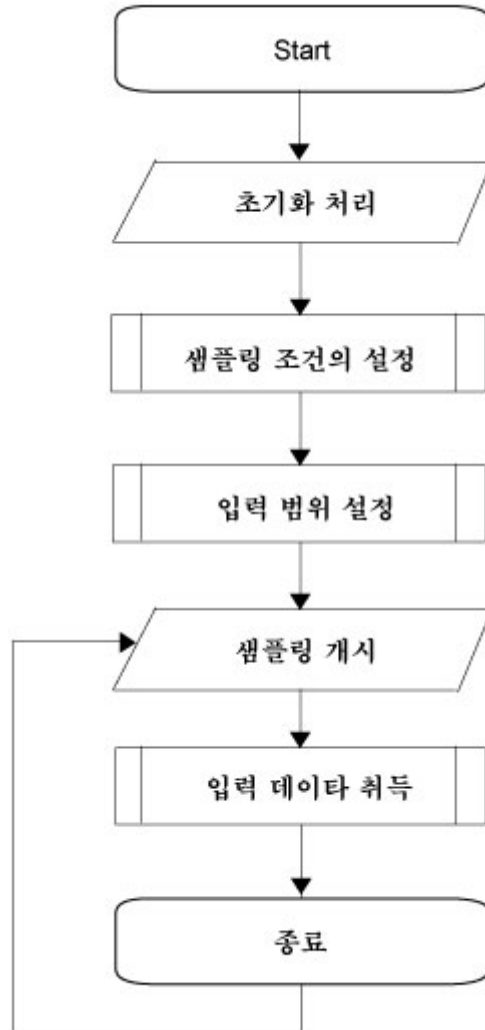
(불안했기 때문에 나중에 매니저에게 상담했습니다)



## 결과

모드의 선택 부분을 우선 결정해 먼저 읽어 진행시켜 나가면  
...있었습니다! 제일 가지고 싶었던 것이... 여기로부터 시작됩니다. 워크플로우입니다.  
아날로그 데이터 취득에 필요한 워크플로우가 그림으로 게재되어 있었습니다.  
「이 값을 취득한 베이스 주소와의 "엔드"를 취하면 좋습니다...와 구!」

## 데이터 입력 기능 : 소프트웨어 모드.멀티 채널 모드



## 오늘의 감상

「각각의 항목의 처리를 프로그래밍 해 나가면, 아날로그 데이터 취득 프로세스가 완성될 것이다!」



```
#define inhword(port)      ((unsigned short)_inpw( (unsigned short)(port) ))
```

**INtime 에 있어서의 포토 입출력 라이브러리 함수**

outbyte, outhword.....출력  
inbyte, inhword.....입력

그렇다고 하는 것으로 초기화 처리에서는....

```
outbyte(베이스 주소 + 8, 0);
```

되었습니다.

**샘플링 조건의 설정**

Output	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
+8 (+08h)	코맨드							
	0	0	0	0	0	0	0	1
+12 (+0ch)	설정데이터(조건)							
	*	*	*	*	Analog input Method	Channel Mode	Sampling Clock Source	Sampling Mode

샘플링 조건의 설정은 우선 「커맨드 레지스터(베이스 주소+ 0x08H)에 0 x01를 출력 후, 설정 데이터 0 레지스터에 샘플링 조건을 설정한다」 샘플링 조건의 설정에 대해 다소 해맨 곳이 있었던

샘플링 조건의 설정에 대해 다소 해맨 곳(점)이 있었습니다.

- D3 : Analog input Method 비트
  - 싱글 엔드 : 0
  - 차동입력 : 1
- D2 : Channel Mode 비트
  - 싱글 채널 : 0
  - 멀티 채널 : 1
- D1 : 샘플링 클럭 소스
  - 내장 시계 : 0
  - external clock : 1
- D0 : 샘플링 모드
  - 소프트웨어코만 : 0
  - 클럭 : 1

이번은"싱글 엔드","멀티 채널","내장 시계","소프트웨어"라고 하는 설정인 모아 두어 2 진수로...0100 → 0 x04를 출력하는 것이 되었습니다.

```
/* 샘플링 조건의 설정 */
outbyte(analogBaseAdd + 0x08, 0x01); //커맨드 0x01 : 샘플링 조건의 설정
/* 싱글 엔드 입력, 멀티 채널 모드, 내부 샘플링, 소프트웨어 모드 */
outbyte(analogBaseAdd + 0x0C, 0x04);
```

## 입력 레인지의 설정

<b>Output</b>	<b>D 7</b>	<b>D 6</b>	<b>D 5</b>	<b>D 4</b>	<b>D 3</b>	<b>D 2</b>	<b>D 1</b>	<b>D 0</b>
<b>+8 (+08h)</b>	코맨드							
	0	0	0	0	0	0	1	0
<b>+12 (+0ch)</b>	설정데이터0(채널)							
	<b>Channel Data 7</b>	<b>Channel Data 6</b>	<b>Channel Data 5</b>	<b>Channel Data 4</b>	<b>Channel Data 3</b>	<b>Channel Data 2</b>	<b>Channel Data 1</b>	<b>Channel Data 0</b>
<b>+13 (+0dh)</b>	설정데이터1(Range)							
	<b>Range Data 7</b>	<b>Range Data 6</b>	<b>Range Data 5</b>	<b>Range Data 4</b>	<b>Range Data 3</b>	<b>Range Data 2</b>	<b>Range Data 1</b>	<b>Range Data 0</b>

입력 레인지의 설정은 우선 「커맨드 레지스터(베이스 주소+ 0x08H)에 0x02를 출력 후, 설정 데이터 0 레지스터(베이스 주소+ 0x0cH)에 채널을 설정해, 그 후 설정 데이터 1 레지스터(베이스 주소 1 +0x0dH)로 설정 레인지를 출력합니다」

입력 레인지의 설정은 사양에 의해±0~ + 10 V라고 해 기다리고 있기 때문에

입력Range	입력Data
±10V	00H
±5V	01H
±2.5V	02H
±1.25V	03H
0~10V	04H
0~5V	05H
0~2.5V	06H
0~1.25V	07H

\*초기 상태

/\* 입력 레인지의 설정 0~10 V 와 CH 분 지정하는 \*/

```

outbyte( analogBaseAdd + 0x08, 0x02 );
for( i = 0; i < CH_CNT ; i++){
    outbyte( analogBaseAdd + 0x0c, i ); // 각 채널을 설정하는
    outbyte( analogBaseAdd + 0x0d, 0x04 ); // 0~10 V 를 설정하는
}

```

이번은 0 채널~7 채널의 전8 채널 분의 설정을 하기 위해 상기와 같이 했습니다. 내장 시계는 사용하지 않기 때문에, 설정을 하지 않습니다.

### 인터럽트 요인의 설정

인터럽트를 사용하기 때문에 인터럽트 요인의 설정이 필요하게 됩니다

<b>Output</b>	<b>D 7</b>	<b>D 6</b>	<b>D 5</b>	<b>D 4</b>	<b>D 3</b>	<b>D 2</b>	<b>D 1</b>	<b>D 0</b>
<b>+8 (+08h)</b>	코맨드							
	0	0	0	0	0	1	0	0
<b>Output</b>	<b>D 7</b>	<b>D 6</b>	<b>D 5</b>	<b>D 4</b>	<b>D 3</b>	<b>D 2</b>	<b>D 1</b>	<b>D 0</b>
<b>+12 (+0ch)</b>	설정데이터 0 (인터럽트 요인)							
	1	1	Sampling Clock Error	Sampling Clock Input	1	Data Over Write Error	Data Read Enable	1
<b>+13 (+0dh)</b>	설정데이터 1 (인터럽트 요인)							
	1	1	Timer O/R Status	Timer Status	Ext. Trigger O/R Status	Ext. Trigger Status	1	1

「사용하는 인터럽트 요인 비트에 대해서 0을 출력하는 것에 의해서 인터럽트 요인을 설정합니다.」...사용하는 인터럽트는 단순하게 말하면, 「데이터의 취득이 가능합니다」라고 하는 신호를 나타내는 요인입니다.이 인터럽트 요인의 각각의 비트는...

인터럽트 요인 비트		인터럽트 요인 종별
0x0ch	D5	샘플링 클럭 에러
	D4	샘플링 클럭 입력
	D2	데이터 덧쓰기 에러
	D1	데이터 읽기 OK
0x0dh	D5	타이머 O/R 스테이터스
	D4	타이머 스테이터스
	D3	외부 방아쇠 O/R스태이터스
	D2	외부 방아쇠 스테이터스

상기의 복수의 인터럽트 요인 중에서 사용하는 것은, 데이터 읽기 OK비트.즉 0x0ch : D1만 0을 출력해, 그 외의 비트에 마스크 출력(1)하게 됩니다.

```

outbyte( analogBaseAdd + 0x08, 0x04 );
outbyte( analogBaseAdd + 0x0c, 0xFE ); //최종 비트만 0 그 외를 마스크 프로세싱
outbyte( analogBaseAdd + 0x0d, 0xFF ); //전비트를 마스크 프로세싱
    
```

## 10월 17일...인터럽트 처리 (인터럽트 처리기·인터럽트 스레드)

### 아날로그 데이터 변환치 취득 스레드 완성!

#### 과제

아날로그 보드와 격투해, 겨우 아날로그 데이터 변환치를 취득할 수 있게 되었습니다.

아날로그의 데이터는 100 ms의 폴링 주기로 취득합니다.

스레드의 흐름으로서는,

보드 검출(아날로그 보드를 검출)→초기 처리(아날로그 보드의 초기화→아날로그 데이터 취득 준비)→샘플링 개시→데이터의 취득

이 됩니다.

사양에서는 100 ms 마다 아날로그 데이터 취득을 위해 "샘플링 개시" 커멘트를 출력해, 0~7 CH의 "데이터를 취득하는" 일로 아날로그 데이터 취득 스레드는 완성합니다.

●보드 검출 처리 ----- FindAnalogBord()

●아날로그 보드 초기 처리----- InitAnalogBoard()

아날로그 데이터 취득 스레드는 void GetDataThread(void)로, 이 스레드의 주된 태스크(일)는 100 ms 마다 아날로그 보드의 샘플링 개시 커멘트를 출력해, 8 CH분의 변환 데이터를 취득하는 것입니다. 100 ms의 폴링 주기를 만드는 것은...

아래와 같이 됩니다 :

```
while(1){  
    RtSleep(100);          //100 mssleeve  
}
```

#### 트러블 발생

변환 데이터를 취득부에서 트러블이 발생했습니다.....

「변환 데이터의 취득 인터럽트가 발생하지 않는다!

인터럽트 요인은 설정해 있는데...

인터럽트라는 것을 잘 모르는 나는, 무엇을 어떻게 조사해도 좋은 것인가 몰랐습니다.

#### 필요한 것은 보드의 IRQ와 그 보드의 인터럽트 요인의 특정

●IRQ(인터럽트 번호) : 0~15 까지의 인터럽트 번호가 있어, 각각의 하드 기기는 끼어들어 번호가 할당해지고 있습니다.

●인터럽트 요인의 특정 : 상기의 인터럽트

#### 트러블의 해결

어떤 상태가 인터럽트가 들어간 상태인가 몰랐기 때문에, 혼란해져 버렸습니다.

전혀 움직이지 않은 상태가 그 상태라고 지레 짐작해, 「움직이지 않는=인터럽트가 들어가지 않는다」라고 생각하고 있었는데, WaitForRtInterrupt()로 에러 체크를 해 보면...되돌아 온 에러의 내용은...

....인터럽트 처리기를 만들지 않았다! <<.....이래서야 인터럽트에 들어가지 않는다.....>>

「인터럽트 처리기는 특정의 인터럽트 레벨의 신호 입력을 받아 기동합니다.

인터럽트 신호 발생시, 스레드를 기동해야 할 인터럽트인지 어떤지를 판별해, 스레드 기동을 걸친다...그렇다고 하는 것이었습니다.」

그 후, 인터럽트 처리기를 만들었습니다만 수많은 실패가 있었습니다.

대표적인 것은 :

●아날로그 보드의 초기화에 실패하고 있었기 때문에, 인터럽트의 설정이 잘 되지 않았다

아날로그 보드의 액세스를 위해서, 선두 주소로부터의 오프셋치를 선언한 헤더를 작성했습니다. 이 작성한 헤더의 값이 10 진수의 값을 16진표기할 경우에 그대로 10진표기하고 있던 일에 의해 설정한 내용이 지정의 주소에 써지고 있지 않았습니다.

(BaseAddress + 12 ⇔ BaseAddress + 0x12 로 하고 있던, 이전에도 벤더 ID, 디바이스 ID지정부 이 실패를 했습니다.

●설정된 IRQ치가 차이가 났기 때문에, 인터럽트의 설정이 잘 되지 않았다

아날로그 보드의 검출 함수로 끼어들어 레벨치를 되돌리고 있던 것을, 도중에 함수의 형태를 변경해, 반환값을 TRUE/FALSE로 했습니다. 그러나, 그대로 반환값을 끼어들어 레벨과 설정해 있었기 때문에, 1(TRUE)이라고 하는 값으로 인터럽트 처리기를 설정해 있었습니다.

상기의 실패를 근거로 해 보드 검출 처리 FindAnalogBoard(아날로그 보드의 검출)의 결과, 보드 검출 OK의 경우, CreateRtThread에 의해 GetDataThread(아날로그 데이터 취득 스레드)가 생성됩니다.

GetDataThread의 개시부에서, 아날로그 보드의 초기 처리 InitAnalogBoard()를 콜 합니다, 동시에, 인터럽트 요인(Data Read Enabled)의 인터럽트시에 이벤트를 받아, 그 타이밍에 샘플링을 실시하지 않으면 안되기 때문에, 인터럽트 처리기를 설정했습니다.

인터럽트 처리기의 코드는 INtime 위저드로 인터럽트 처리기와 스레드, 를 지정하는 것으로써, 코드는 출력되기 때문에, 처리를 그 중에 기술했습니다.

이하가 인터럽트 처리기의 설정 API입니다 :  
SetRtInterruptHandler(wLevel, 1, Int1Handler))

INtime Help로부터 SetRtInterruptHandler를 조사하면, 위의 시스템 콜은 「인터럽트 레벨 wLevel의 인터럽트가 1도 입력하면 Int1Handler에 처리를 건네줍니다(Int1Handler를 콜 합니다)」 라고 읽을 수 있습니다. 벌써 위저드에 의해 코드가 되어 있기 때문에, 핸들러의 호출 부분은 좋아.

인터럽트 처리기측(Int1Handler)에서 인터럽트 요인의 특정을 실시하지 않으면 안됩니다. 인터럽트 레벨을 설정한 핸들러에는, 그 레벨의 인터럽트가 모두 들어갑니다. 인터럽트의 입력시에 있는 특정 상태지만 처리를 실시하는 경우, 인터럽트 처리기내에서 그 조건을 판별해, 조건 성립시는 SignalRtInterruptThread(wLevel)로 인터럽트 스레드 기동을 해, 그 이외는 SignalEndOfRtInterrupt(wLevel)로 억제합니다

인터럽트 처리기의 기술 부분 :

```
__INTERRUPT void Int1Handler(void)
{
    __INTERRUPT_PROLOG(); //인터럽트 처리기의 범위 개시

    // 인터럽트 레벨을 설정합니다
    EnterRtInterrupt(wLevel);

    //인터럽트 요인의 특정 :
    if ((inbyte(strInfo.analogAdd + 0x0c & 0x02) != 0x02)
        SignalEndOfRtInterrupt(wLevel); //특정의 인터럽트 조건 이외는 시그널 발신을 정지

    else
        SignalRtInterruptThread(wLevel); //특정의 인터럽트 조건 성립시는 시그널을 발하는

    __INTERRUPT_RETURN(); //인터럽트 처리기의 범위 종료
}
```

인터럽트 요인이 확정했을 경우만 출력하는(SignalRtInterruptThread)에 의해 WaitForRtInterrupt()로 끼어들어 대기 처리 이하의 처리가 실행됩니다.

실코드는 이하와 같이 되겠지요 :

## 스레드의 기술 :

```
while(1){  
    RtSleep(100);    //100 mssleeve  
  
    //변환 데이터가 설정될 때까지(인터럽트가 발생할 때까지) 기다린다 :  
    wLevel 는 아날로그 보드의 인터럽트 번호  
    if(WaitForRtInterrupt(wLevel, WAIT_FOREVER) != TRUE){  
        //블록  
    }  
  
    아날로그 변환 데이터를 취득하는  
}
```

## 오늘의 감상

인터럽트 스레드란, 인터럽트 처리기를 설정해, 핸들러에 의해 기동되는 스레드로 그 이외는 특히 통상의 스레드와 변화가 없다고 하는 것을 알았습니다(스레드의 생성 방법 등...).

**10월 18일...아날로그 데이터 변환·취득**

**아날로그 데이터 변환치 취득 스프레드 완성!**

아날로그 데이터의 취득 준비가 갖추어졌습니다.  
 아날로그 보드에 대해서 샘플링 개시 커멘드를 발행하면, 그 데이터 변환 종료의 인터럽트로서 WaitForRtInterrupt()  
 이하의 처리가 흐르는 구조입니다.  
 그러면 샘플링 개시에 대해 메뉴얼의 기술을 볼 필요가 있습니다.

**샘플링 개시**

<b>Output</b>	<b>D 7</b>	<b>D 6</b>	<b>D 5</b>	<b>D 4</b>	<b>D 3</b>	<b>D 2</b>	<b>D 1</b>	<b>D 0</b>
<b>+4 (+04h)</b>	Channel Data							
	N/A	N/A	Channel Data 5	Channel Data 4	Channel Data 3	Channel Data 2	Channel Data 1	Channel Data 0

싱글 채널 모드시 : 변환하는 채널을 지정합니다.  
 멀티 채널 모드시 : 변환하는 채널의 상한치(1이상)를 지정합니다.  
 예를 들면, 10 ch를 지정하면 0~10 ch를 샘플링 합니다.

이번 경우 멀티 채널로 취득 채널의 상한치는 7(0 CH~7 CH)이기 위해...

```
outbyte(베이스 주소 +0x04, 7);
```

되겠지요.

**변환 데이터의 입력**

메뉴얼에 의하면, 아날로그 데이터 의 디지털 데이터에의 변환은 이하와 같이 행해지는 :

$$\text{Data} = \frac{(\text{전압} + \text{Off set})}{\text{Span}} \times 2^{12}$$

입력Range	Off set	Span	입력Range	Off set	Span
-10V to +10V	10	20	0V to +10V	0	10
-5V to +5V	5	10	0V to +5V	0	5
-2.5V to +2.5V	2.5	5	0V to +2.5V	0	2.5
-1.25V to +1.25V	1.25	2.5	0V to +1.25V	0	1.25

**입력치 디지털 변환**

-10~+10의 입력 범위를 설정했을 경우, 변환 데이터는 위의 그림과 같이 됩니다. 즉 0x0000(0)~0x0FFF(4095)의 범위에서 전압 데이터가 변환됩니다.-10.000 V 시는 디지털 데이터 0,+9.995V(10V) 시는 디지털 데이터 4095가 되는 것을 나타내고 있습니다.

이번 경우 0V~10V의 범위이므로 0v(디지털 데이터 0)~10V(디지털 데이터 4095)에 변환되면 좋다고 하는 것이 됩니다.

入力電圧 (+/-10V range)	12 비트 변환 데이터
	Off set 바이너리
+9.995V	0FFF h
:	:
0.005V	0801 h
0.000V	0800 h
-0.005V	07FF h
:	:
-10.000	0000 h

Input	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
+6 (+06h)	아날로그 입력 Status							
	0	0	0	0	Sampling Clock Error	Data Over Write Error	Data Read Enable	Conversion Busy Status
+0 (+00h)	아날로그 입력 데이터 (lower)							
	Conversion Data 7	Conversion Data 6	Conversion Data 5	Conversion Data 4	Conversion Data 3	Conversion Data 2	Conversion Data 1	Conversion Data 0(LSB)
+1 (+01h)	아날로그 입력 데이터 (upper)							
	0	0	0	0	Conversion Data 11(MSB)	Conversion Data 10	Conversion Data 9	Conversion Data 8

샘플에서는 이하와 같이 기록되고 있습니다

```
while( !inp( ADR + 6 ) & 2 ){
    AiData = inpw( ADR );
}
```

이 예에서는 「+0 x06의 아날로그 입력 스테이터스의 레지스터를 폴링으로 검색해, Data Read Enable의 값이 끊고 있을 때는 아날로그 데이터를 취득한다」라고 있습니다.

이번 경우, 샘플링 개시를 출력 후, Data Read Enable의 값을 폴링으로 검색하지 않기 때문에, 이 예는 적용할 수 없습니다.그 대신에 인터럽트를 사용합니다.

**GetADataThread의 기술 부분 :**

```
void GetADataThread(void)
{
    int i;
    WORD aData[CH_CNT];

    if (!SetRtInterruptHandler(strInfo.wIntLevel, 1, Int1Handler))
        Fail("Cannot set interrupt task for Pci(0x1221,0x8153)");

    InitAnalogBord();/* 아날로그 보드의 초기화*/
    dataCnt = 0; /*취득 데이터수초기화*/

    while (1)
    {
        RtSleep(100); //샘플링 주기 10.....(1)
        /*샘플링 개시 커멘드 : BASE+0 x04-CH 수 : 7 */
        outbyte(strInfo.analogAdd + 0x04, 7);.....(2)
        //변환 데이터가 설정될 때까지(인터럽트가 발생할 때까지) 기다린다
        if(WaitForRtInterrupt(strInfo.wIntLevel, WAIT_FOREVER) != TRUE){
            ;.....(3)
        };
        /* 8 CH 분 데이터 읽어들이기 */
        // 0~7 CH의 8회 혼잡 처리를 반복한다
        for( i = 0 ; i < CH_CNT;i++){.....(4)
            aData[i] = inhword(strInfo.analogAdd) & 0x0FFF;
        }
    }
}
```

- (1) 샘플링 주기...100ms
- (2) 샘플링 개시 커멘드 출력
- (3) 인터럽트 대기
- (4) 변환 데이터 취득

이 후, 어느A 채널에 5 V가 전압을 걸어 입력치가 2047 정도인 것으로부터, 데이터 변환이 잘 되어 있는 것을 확인했습니다.

## 10월 21일...디지털 보드

### 아날로그 데이터 변환치 취득 스레드 완성!

디지털 보드 출력에 관해서는, 아날로그 보드의 경우와 같이 순서를 취했습니다.  
우선 사양을 확인하는 :

#### 디지털 보드 출력에 대한 사양 기술부

디지털 보드 출력에 대한 사양 기술부

- DO카드를 검출할 수 있는 것...할 수 없는 경우, DO출력을 하지 않는 모드로 동작
- 각 CH의 값이+8.0 V이상으로 소정 DO비트를 ON로 한다.DO는 8비트 사양 해, 비트 0이 CH0, 비트 7이 CH7와 같이 대응시킨다.DO포토의 오프셋+0
- 100 ms주기에 DO8~15비트를 점멸시키는 스레드를 가지는 것.DO포토의 오프셋+1

#### 상기의 사양으로 읽어낼 수 있는 것

디지털 보드 출력에 대한 사양 기술부

- PciFindDevice 함수를 사용해 디바이스의 검출을 행할 뿐으로 인터럽트 레벨을 이용하지 않는다 (인터럽트를 사용하지 않는다).
- 아날로그 보드 처리와 같고, 액세스 하는 주소를 설정한다
- 소정 비트 출력 DO처리는 아날로그 데이터 변환치 입력시에 실시한다.
- 100 ms주기에 DO8~15비트를 점멸시키는 스레드는 독립한 스레드(DOBlinkCtrlThread).

#### <소정 비트 출력 : 8 V 이상의 경우 DO 보드 출력한다>

·aData[i]...8 CH 분의 아날로그 데이터  
·디지털 보드 출력 그룹(BASE ADDRESS+ 0x04 ~ 0 x07)  
·wDigitalData...디지털 보드의 그룹에 출력하는 데이터

```
-----  
for( i = 0 ; i < CH_CNT;i++){  
    if( aData[i] >= VAL_8V){  
        /*혼잡 데이터가 8 V 이상의 경우 CH에 대응한 비트 ON로 한다*/  
        wDigitalData = wDigitalData | (0x01 << i);  
    }  
}  
if (디지털 보드 검출 되어 있는 경우){  
    outbyte(디지털 보드 베이스 주소 + 0x04,wDigitalData);  
}
```

#### <100 ms 주기 DO 출력 스레드>

```
void DOBlinkCtrlThread(void)  
{  
    int ctrlSW; // ON/OFF 제어 상태 플래그  
    ctrlSW = 0;  
    while (1)  
    {  
        RtSleep(100);  
        /* 100 ms 주기에 디지털 보드를 점멸시키는*/  
        if(ctrlSW == 0){  
            /* DO 비트 ON 상태로 설정 */  
            outbyte(strInfo.digitalAdd + 0x05 ,0xFF);  
            ctrlSW = 1;  
        }else{  
            /* DO 비트 OFF 상태로 설정 */  
            outbyte(strInfo.digitalAdd + 0x05 ,0x00);  
            ctrlSW = 0;  
        }  
    }  
}
```

## 10월 22일...데이터형 메일 박스

### 데이터의 인터페이스

#### 과제

아날로그 변환 데이터를 취득 후, 로그 출력 스레드, NT표시 프로세스에 그 데이터를 송신하지 않으면 안됩니다. 데이터 송신에 사용하는 오브젝트는 메일 박스입니다. 메일 박스에는 오브젝트형 메일 박스와 데이터형 메일 박스가 있어 이번은 로그 출력 스레드에 대해서 데이터형 메일 박스를 NT데이터 표시 프로세스에 오브젝트형 메일 박스를 사용하게 되었습니다.

데이터형 메일 박스는 128 BYTE까지의 데이터를 취급합니다.

데이터형 메일 박스로의 데이터 송신에 사용하는 SendRtData 함수를 사용해 로그 출력 스레드에 데이터를 송신하도록 했습니다.

데이터형 메일 박스에 대해서는 별 트러블도 없고, 순조롭게 도입할 수 있었습니다

#### 초기화시

메일 박스 생성 : 초기 스레드(main 스레드)

```
//데이터형 메일 박스
//아날로그 데이터 혼잡·의사 데이터 작성 스레드와 로그 출력 스레드와의
//데이터의 수수를 하기 위해 데이터형 메일 박스를 작성한다
strInfo.hParamMBox = CreateRtMailbox(DATA_MAILBOX | FIFO_QUEUEING);
if (strInfo.hParamMBox == BAD_RTHANDLE)
    Fail("Cannot create data mailbox ParamMBox");
if (!Catalog(NULL_RTHANDLE, strInfo.hParamMBox, "ParamMBox"))
    Fail("Cannot catalog data mailbox ParamMBox");
```

#### 데이터 송신

아날로그 변환 데이터 취득 스레드, 의사 아날로그 데이터 생성 스레드 : 데이터 100 건에 한 번 로그에 출력하는

```
/*혼잡 데이터 100 건에 1도 로그 출력을 실시하는 */
if((dataCnt % 100) == 0){
    /*100 건에 1도 로그 출력 스레드에 데이터형 메일 박스를 사용해 데이터를 송신한다 */
    SendRtData(strInfo.hParamMBox, aData, sizeof(aData));
    /* 100 건을의 제어를 실시하기 위해 카운터를 클리어 하는*/
    dataCnt = 0;
}
```

#### 데이터 수신

로그 출력 스레드 : 수신 데이터를 기초로 로그를 출력

```
while (1){
    valData = (WORD*)byMessage;
    /*아날로그 데이터 혼잡 스레드, 또는 의사 데이터 작성 스레드보다*/
    /*데이터형 메일 박스에서 데이터를 수신*/
    wActual = ReceiveRtData(strInfo.hParamMBox, valData, WAIT_FOREVER);
    if (wActual == 0)
        Fail("Receive from data mailbox ParamMBox failed");
    /* 수신한 데이터를 기초로 로그 출력을 실시하는 */
    WriteLogFile(valData); /* 로그 출력 */
}
```

## 10월 23일...오브젝트형 메일 박스

### 데이터의 인터페이스

오브젝트형 메일 박스는 INtime 오브젝트의 핸들(RTHANDLE)을 송수신 합니다.

NT측 데이터 표시 프로세스에 대해서 송신하는 데이터를 오브젝트형 메일 박스로 송신하게 되었습니다. 오브젝트형 메일 박스의 사용에는 많은 트러블이 있었습니다.

오브젝트형 메일 박스를 사용해 송수신 하는 것은 오브젝트(RTHANDLE)입니다.RTHANDLE를 수신한 시점에서, 그 RTHANDLE가 올바른 RTHANDLE인가를 판별해, 그 핸들을 맵 하는 것으로써, 데이터를 참조합니다만...

#### 실패 그 1 :

메모리오브젝트를 생성하지 않았던 송신 데이터는 데이터형 메일 박스와 같이 WORD\*8(CH분 )의 16BYTE입니다.

오브젝트형 메일 박스에 대해 전혀 모르는 나는

```
WORD *wSndData;
/* NT 측에 송신하는 데이터를 공유 메모리에 데이터를 격납하는 */
memcpy(wSndData,aData,sizeof(aData));
/* 공유 메모리핸드라를 NT 측에 송신하는 */
SendRtHandle(strInfo.hdataMbox,wSndData,NULL_RTHANDLE);
```

(와)과 같이 오브젝트 메일 박스에 대해서 데이터를 건네주거나

```
RTHANDLE hdataobj;
strInfo.sendData = (WORD*)MapRtSharedMemory(hdataobj);
/* 공유 메모리에 데이터를 격납한다 */
/* NT 측과의 통신 (위해)때문에, 공유 메모리에 데이터를 격납한다. */
memcpy(strInfo.sendData,ADatVAL,sizeof(ADatVAL));
/* 공유 메모리핸드라를 NT 측에 송신하는 */
SendRtHandle(strInfo.hNtResvMbox,strInfo.hDataObj,NULL_RTHANDLE);
```

핸들을 선언해, 무리하게 맵해 데이터를 격납해, 송신하는 등...

핸들은 어디까지나 오브젝트에 대해서 주어지는 것이어, 마음대로 데이터를 만들고, 「그것을 핸들입니다!」 라고 할 수 없다고 알았습니다. 여기서의 오브젝트와는 메모리(공유 메모리)입니다.

올바른 수속은... :

- 1.메모리를 확보한다.(AllocateRtMemory)
- 2.확보한 메모리에 대해, 오브젝트 핸들을 생성한다.(CreateRtMemoryHandle)
- 3.생성한 메모리한들을 맵 해, 데이터를 쓴다.(MapRtSharedMemory)
- 4.핸들을 송신한다.(SendRtHandle) 됩니다

```
WORD* wSendDt; 기입 데이터
WORD* wRecvDt; 읽기 데이터
```

RTHANDLE hObjHandle; 송수신오브젝트 핸들

```
●wSendDt = (WORD *)AllocateRtMemory(4096); //메모리의 할당
●hObjHandle = CreateRtMemoryHandle( wSendDt, 4096 ); //메모리한들의 생성
●wRecvDt=(WORD*)MapRtSharedMemory(hObjHandle); //메모리오브젝트의 맵
●SendRtHandle( strInfo.hNtResvMbox, hObjHandle, NULL_RTHANDLE )//오브젝트의 송신
```

**실패 그 2 :**

오브젝트 메일 박스와 데이터 메일 박스의 시스템 콜의 차이를 깨닫지 못했다.  
데이터형 메일 박스와 오브젝트형 메일 박스의 생성에 대하여는 시스템 콜에는 차이가 없고, 파라미터의 차이만입니다 :

CreateRtMailbox(DATA_MAILBOX   FIFO_QUEUEING). . . .	데이터형 메일 박스 생성
CreateRtMailbox(OBJECT_MAILBOX   FIFO_QUEUEING). .	오브젝트형 메일 박스 생성

그러나, 데이터의 송신, 오브젝트의 송신은 각각 시스템 콜이 다른 :

SendRtData(). . . . .	데이터형 메일 박스에 송신
ReceiveRtData(). . . . .	데이터형 메일 박스로부터 수신
SendRtObject(). . . . .	오브젝트형 메일 박스에 송신
ReceiveRtObject(). . . . .	오브젝트형 메일 박스로부터 수신

Windows측과 교환을 하는 경우, 공유 메모 리사이즈는 Windows의 페이지 사이즈(4Kbyte : 4096바이트)의 배수로 할 필요가 있었습디만, 이번 공유 메모 리사이즈가 작은 것으로, AllocateRtMemory( sizeof(WORD) \* 8 )와 지정해 있었습디다.

- 1.메모리를 확보한다.(AllocateRtMemory)
- 2.확보한 메모리에 대해, 오브젝트 핸들을 생성한다.(CreateRtMemoryHandle)
- 3.생성한 메모리한들을 맵 해, 데이터를 쓴다.(MapRtSharedMemory)
- 4.핸들을 송신한다.(SendRtHandle)

됩니다

```
WORD* wSendDt;   기입 데이터
WORD* wRecvDt;   읽기 데이터
RTHANDLE hObjHandle; 송수신신오브젝트 핸들
```

- wSendDt = (WORD \*)AllocateRtMemory(4096); //메모리의 할당
- hObjHandle = CreateRtMemoryHandle( wSendDt, 4096 ); //메모리한들의 생성
- wRecvDt=(WORD\*)MapRtSharedMemory(hObjHandle); //메모리오브젝트의 맵
- SendRtHandle( strInfo.hNtResvMbox, hObjHandle, NULL\_RTHANDLE )//오브젝트의 송신

**실패 그 3 :**

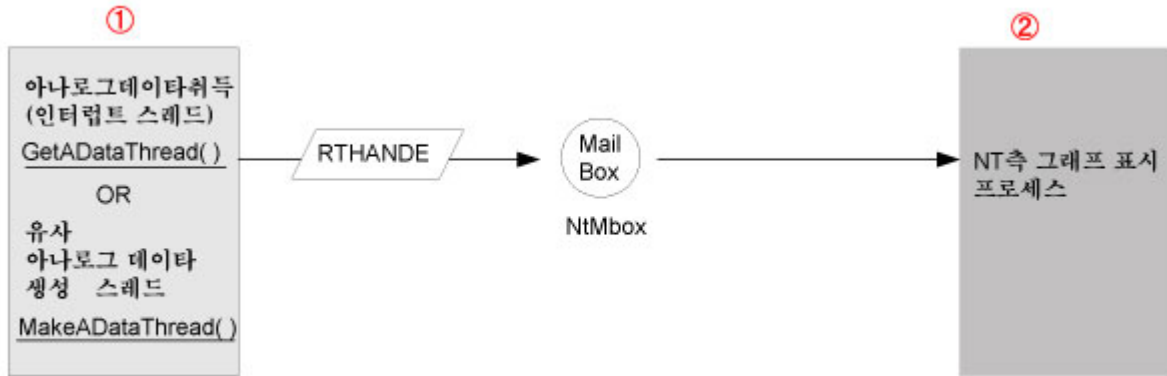
할당하는 메모 리사이즈 AllocateRtMemory로 확보하는 메모 리사이즈

Windows측과 교환을 하는 경우, 공유 메모 리사이즈는 Windows의 페이지 사이즈(4Kbyte : 4096바이트)의 배수로 할 필요가 있었습디만, 이번 공유 메모 리사이즈가 작은 것으로, AllocateRtMemory( sizeof(WORD) \* 8 )와 지정해 있었습디다.

10월 24일...오브젝트 핸들의 재검토

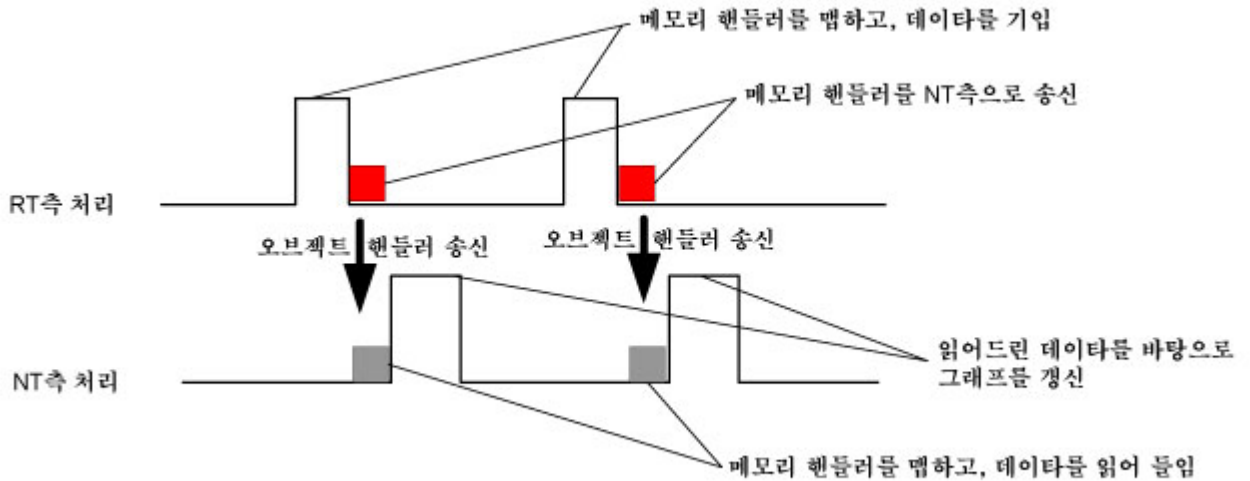
데이터의 인터페이스

RT(INtime측의 프로세스)와 Windows NT(Windows측 프로세스) 간의 통신도



RT(INtime측 프로세스)와 Windows NT(Windows측 프로세스)간의 통신도

- ①에서는 아날로그 변환치를 취득 한 후, 그 내용을 메모리 오브젝트에 쓰고, 오브젝트 핸들을 메일 박스 NtMbox 에 송신합니다.
- ②는 오브젝트를 확인 후, 그 오브젝트를 맵 해, 데이터를 취득, 그 내용을 그래프에 반영함.

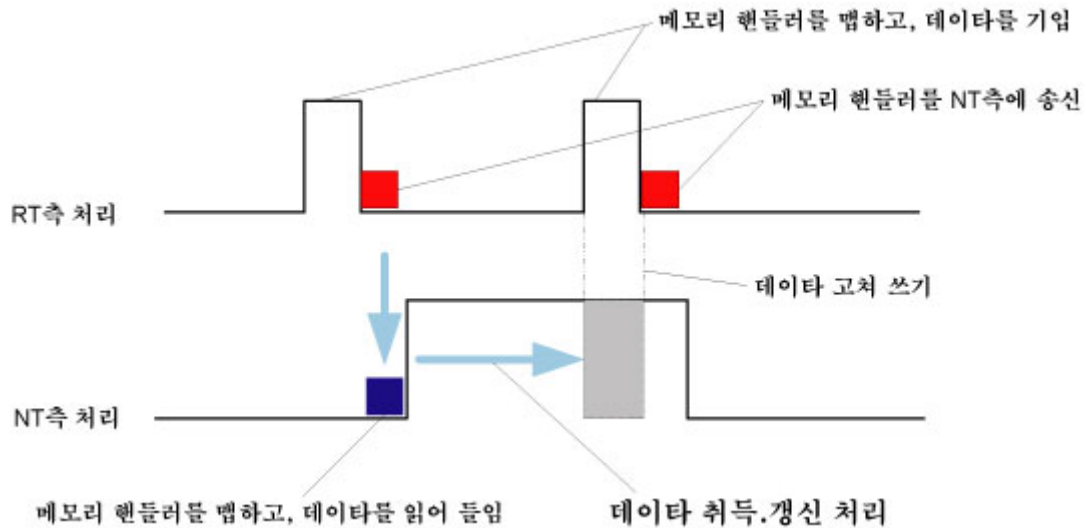


개략적 통신 인터페이스

상기와 같이 RT측은 Windows측의 처리에 관계없이, 100 ms주기로 데이터를 취득해, 오브젝트 핸들을 맵, 데이터를 격납해, 데이터를 송신하고 있습니다.

송신하는 오브젝트 핸들러는 하나로, 서로 유일한 오브젝트 핸들러를 맵, 데이터의 읽고 쓰기를 실시하는 상황에 있고, 만약, Windows측의 처리에 시간이 걸리는 처리가 발생해, 다음의 RT측의 데이터 기입 주기까지 표시 처리가 끝나지 않는 경우, RT측 데이터 취득 스레드에 의해 덧쓰기된 데이터가 NT측에서 갱신되어 버리게 됩니다.

NT측의 처리가 위법에 늦었을 경우, 이하와 같은 상황이 발생할 가능성이 있습니다.



이것은 수신측과 송신측의 스레드가 각각 동시에 맵핑, 읽고 쓰기를 실시해 버리는 케이스입니다.

이것을 해소하기 위해서, 메모리 오브젝트의 토큰을 복수 준비하는 방법이 있습니다 :

우선, 현재까지 메모리 오브젝트를 하나의 봐 생성하고 있던 부분을 복수 생성하도록 합니다.

복수 생성된 메모리 오브젝트는 모두 RT 측 메모리 오브젝트 수신용 메일 박스에 우선 송신됩니다.

물로서 RT 측 스레드는, 「RT 측 메모리 오브젝트 취득용 메일 박스로부터 오브젝트를 수신한 후에 맵 해, 데이터를 쓴다」 것으로 합니다.

데이터의 기입이 끝나면, NT 측 프로세스용 메일 박스에 송신합니다. NT 측 프로세스는 메일 박스에 닿은 오브젝트를 맵 해, 데이터를 취득합니다.

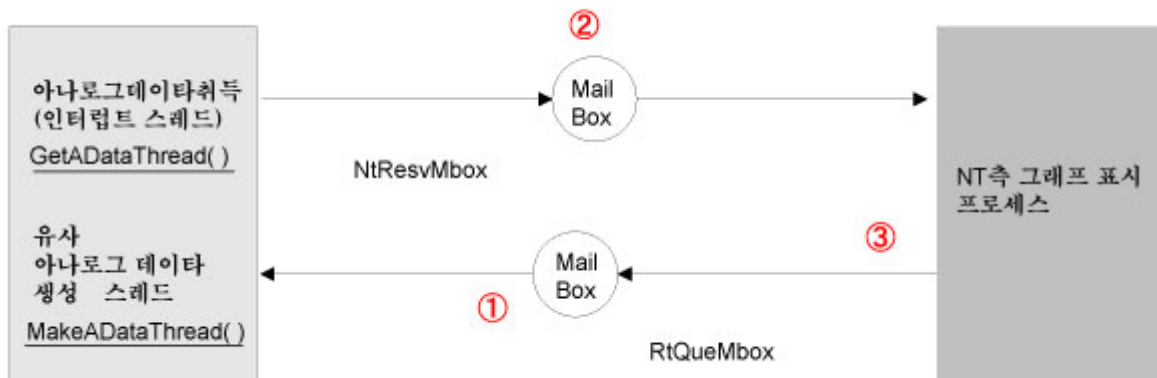
이 때, 비유 처리가 극단적으로 늦어졌을 경우에 대해서도, 메모리 오브젝트가 복수 존재하기 위해 RT 측 스레드가 NT 에 송신한 최초의 메모리를 고쳐 써 버릴 것은 없습니다.

메일 박스는 큐를 가지는 것이 가능하고(FIFO\_QUEING) 수신한 메모리 오브젝트를 차례로 참조하는 것이 가능합니다.

또 참조의 종료한 메모리토큰은 RT 측의 메모리 오브젝트 수신용의 메일 박스에 송신하는 것에 의해서, 사이클릭에 메모리 오브젝트를 사용할 수 있습니다.

오브젝트 메일 박스의 성격상, 이 처리는 메일 박스 하나로 실현 가능합니다만, 이번은 알기 쉽게 만들기 위해서 메일 박스를 두 개 준비했습니다.

① RT 측 스레드메일 토큰 취득용 메일 박스와②NT 측 프로세스 아날로그 데이터 수신용 메일 박스.



① RT 측 스레드는 취득한 데이터를 격납하기 위해서, RtQueMbox로부터 메모리토큰을 취득합니다. 오브젝트 핸들을 맵 해, 취득 데이터를 써 후, NtRecvMbox 에 데이터를 송신합니다.

② NT 측 그래프 표시 프로세스는 메일 박스에 오브젝트 핸들이 착신 후, 그 핸들을 맵해, 취득 데이터를 읽어냅니다. 예를 들어, 이 처리에 꽤 시간이 걸렸다고 해도, RT 측은 다른 메모리토큰에 데이터를 격납해 송신하기 때문에 이 메일 박스에 큐로서 저축되게 됩니다.

③ 메모리 오브젝트의 참조 종료후, NT 측 그래프 표시 프로세스는 핸들을 RtQueMbox 에 답장하겠습니다.

## 10월 25일...NTX 어플리케이션 작성 준비

### NT측 그래프 표시 어플리케이션

#### 과제

아날로그 변환 데이터 취득 스펙트, DO출력 스펙트, 로그 출력 스펙트 등, RT측의 어플리케이션을 거의 완성시켜, 다음에 NT측의 표시 어플리케이션의 작성에 착수하고 싶습니다.

이전에 Visual C++를 사용해, Windows 어플리케이션의 작성을 행했던 적이 있으므로, INtime측(RT측)의 처리와의 통신 인터페이스마져 확립되어 있으면, 그렇게 어렵지 않을 것이라고 생각하고 있었습디만, 우선 Visual C++의 환경 설정으로부터 실패했습니다.

#### 환경 설정

Windows측에서 INtime의 오브젝트를 취급하기 위한 라이브러리 파일이 있다고 하는 것처럼 듣고 있었습디만, 그 개요에 대해 별로 몰랐습디다. INtime Help보다 그 개요를 조사했습니다.

NTX.lib 파일과 NTX.h파일이 존재해, INtime의 ntx 콜을 사용하기 위해서는, 환경 설정으로, 그것들을 설정할 필요가 있었습디다.

또 같이 INtime 소프트웨어의 사용에 대해—>RT커널 시스템 콜->시스템 콜에 대해와 선택해, 표시되는 NTX 콜을 클릭해, NTX 콜의 종류를 확인했습니다

Visual C++의 메뉴로부터 프로젝트->설정과 선택해, 링크 탭의 오브젝트 라이브러리 모듈에 ntx.lib를 추가했습니다.

게다가 메뉴의 툴->옵션과 선택해, 디렉토리 탭으로부터, NTX.h의 참조 장소를 입력했습니다.

이 작업 후, 컴파일러가 무사와 있는 것 같게 되었습니다.INtime 시스템에 대해 Windows측 어플리케이션의 설정에는, NTX.lib의 추가, NTX.h의 인클루드처 참조 설정이 필요하게 되는 것을 알았습니다.

NTX.lib : C:\INtime\nt\lib\ntx.lib

NTX.h : C:\INtime\nt\include\ntx.h

#### 아이콘 설정

RT측의 아날로그 변환치 취득 데이터가 8 V이상의 데이터인 경우, 채널을 나타내는 막대 그래프의 겨드랑이에, 아이콘으로 램프 표시를 하게 됩니다.

아이콘을 자원 레벨로 설정하지 않고, 프로그램으로 동적으로 변경하는 방법에 대해 조사했습니다.

#### ●SetIcon()

다이얼로그 클래스의 초기 처리 CInTimeGraphDspDlg::OnInitDialog()로, 아이콘을 표시하는 컨트롤에의 포인터를 설정해, 각각의 포인터에 OFF 표시 아이콘을 로드합니다.자원으로 미리 IDI\_ICON\_ON(ON용 비트 맵), IDI\_ICON\_OFF(OFF용 비트 맵)를 등록되어 있습니다.

CStatic\* m\_widIcon[8]; //ON/OFF 표시 에리어 포인터 : 다이얼로그 클래스의 멤버

#### OnInitDialog() 부

//8 V 이상의 경우 붉은 램프를 점멸하기 위해서 PICT 컨트롤을 작성해 아이콘으로 ON,OFF의 변환을 실시한다

//초기 상태로서 개시시에는 모두 OFF가 되도록 OFF 아이콘을 표시시킨다

//멤버 변수 m\_widlcon에 표시 에리어(IDC\_STATIC\_ICONX(컨트롤 번호)) 컨트롤에의 포인터를 설정하는

```
m_widlcon[0] = (CStatic*)GetDlgItem(IDC_STATIC_ICON1);  
m_widlcon[1] = (CStatic*)GetDlgItem(IDC_STATIC_ICON2);  
m_widlcon[2] = (CStatic*)GetDlgItem(IDC_STATIC_ICON3);  
m_widlcon[3] = (CStatic*)GetDlgItem(IDC_STATIC_ICON4);  
m_widlcon[4] = (CStatic*)GetDlgItem(IDC_STATIC_ICON5);  
m_widlcon[5] = (CStatic*)GetDlgItem(IDC_STATIC_ICON6);  
m_widlcon[6] = (CStatic*)GetDlgItem(IDC_STATIC_ICON7);  
m_widlcon[7] = (CStatic*)GetDlgItem(IDC_STATIC_ICON8);
```

//OFF 상태로 하는(IDI\_ICON\_OFF 아이콘을 표시한다)

```
m_widlcon[0]->SetIcon( AfxGetApp()->LoadIcon(IDI_ICON_OFF));  
m_widlcon[1]->SetIcon( AfxGetApp()->LoadIcon(IDI_ICON_OFF));  
m_widlcon[2]->SetIcon( AfxGetApp()->LoadIcon(IDI_ICON_OFF));  
m_widlcon[3]->SetIcon( AfxGetApp()->LoadIcon(IDI_ICON_OFF));  
m_widlcon[4]->SetIcon( AfxGetApp()->LoadIcon(IDI_ICON_OFF));  
m_widlcon[5]->SetIcon( AfxGetApp()->LoadIcon(IDI_ICON_OFF));  
m_widlcon[6]->SetIcon( AfxGetApp()->LoadIcon(IDI_ICON_OFF));  
m_widlcon[7]->SetIcon( AfxGetApp()->LoadIcon(IDI_ICON_OFF));
```

## 그래프 표시부

```
if(m_valTbl[i] < VAL_8V){  
    m_widlcon[i]->SetIcon( AfxGetApp()->LoadIcon(IDI_ICON_OFF));  
}else{  
    m_widlcon[i]->SetIcon( AfxGetApp()->LoadIcon(IDI_ICON_ON));  
}  
8 V 이상의 데이터 m_valTbl[i]의 경우...ON의 아이콘을 로드  
8 V 이하의 데이터 m_valTbl[i]의 경우...OFF의 아이콘을 로드
```

## INtime 커널의 기동 확인부

INtime 로 제공하고 있는 샘플로부터 검색, 조사해 보았습니다.

샘플 어플리케이션의 디렉토리 위치 :

C:\INtime\Project

상기 디렉토리내의 jittermt 폴더내의 hisgraph 프로젝트로 가고 있는 처리를 참고로 했습니다.

【Bool CHistGraphDlg::OnInitDialog() 다이얼로그 초기 처리부】

```
//check to see that the RT nucleus was successfully initialized  
→RT 커널이 초기화되고 있을까 체크를 한다  
if( ntxGetRtStatus( NTX_LOCAL ) != E_OK )  
{  
    MessageBoxEx(NULL, _T( "RT Machine not present:RTPM0001" ),  
        _T( "Jitter Graph" ), MB_ICONERROR | MB_OKCANCEL, LANG_ENGLISH );  
    exit(0);  
}
```

## 10월 28일...NTX 어플리케이션 작성

### NT측 그래프 표시 어플리케이션

INtime측 어플리케이션/NT측 어플리케이션간, 혹은 INtime 어플리케이션/ INtime 어플리케이션간에 메일 박스를 통해서 통신을 실시하는 경우, 어느 쪽인지 한편에 속하는 메일 박스의 핸들을 취득할 필요가 있습니다.

### INtime-NTX간 통신의 수법

#### 메일 박스의 오브젝트 핸들의 취득 방법

CatalogRtHandle 와 LookupRtHandle

- 메일 박스를 포함한 어플리케이션은 인터페이스용의 메일 박스의 핸들러치를 있는 이름으로 카탈로그 할 필요가 있습니다. 카탈로그 하는 위치는, 루트 디렉토리나, 그 루트 디렉토리에 속하는 프로세스의 디렉토리 등, 임의로 지정할 수 있습니다.
- 메일 박스를 검색하는 측의 어플리케이션은 메일 박스 오브젝트가 카탈로그 되어 있는 디렉토리와 카탈로그 이름을 알아 둘 필요가 있습니다.

#### CatalogRtHandle()

```
BOOL CatalogRtHandle(  
    RTHANDLE hProcess, // RT 프로세스의 핸들  
    RTHANDLE hObject, // 카탈로그 하는 오브젝트의 핸들  
    LPSTR lpszName      // 카탈로그명  
);
```

#### LookupRtHandle()

```
RTHANDLE LookupRtHandle(  
    RTHANDLE hProcess, // RT 프로세스 핸들  
    LPSTR lpszName, // 오브젝트의 카탈로그명  
    DWORD dwMilliseconds // 검색하기 위한 타임 아웃치  
);
```

Windows 측(NTX 어플리케이션)에서는 ntxCatalogRtHandle()와 ntxLookupRtHandle()가 됩니다.

아날로그 데이터 변환치 취득 스펙트를 포함한 RT 프로세스에 포함되는 메일 박스를 검색한다.

#### 메일 박스 오브젝트 핸들의 취득

메일 박스 오브젝트 핸들을 취득하기 위해서는 우선 루트 Rt 프로세스 핸들의 취득  
(ntxGetRootRtProcess)

RT 측의 프로세스 오브젝트의 취득(ntxLookupNtxHandle)

사용하는 메일 박스 오브젝트 핸들의 취득(nyxLookupNtxHandle)

#### 코드상

```
//루트 프로세스 핸들러를 취득하는  
if( (m_RootProcess = ntxGetRootRtProcess(NTX_LOCAL)) == NTX_BAD_NTXHANDLE )  
{  
    AfxMessageBox( "루트 프로세스 핸들 취득 에러", MB_ICONERROR | MB_OKCANCEL, 0);  
    exit(0); } //루트 프로세스하에 있는 INtime 측의 프로세스를 검색해 핸들러를 취득한다  
if ( (m_InTimeDocProcess = ntxLookupNtxhandle(m_RootProcess, rt_process, 0xffff))  
== NTX_BAD_NTXHANDLE )  
{  
    AfxMessageBox("INtimeDoc 프로세스 핸들 취득 에러" , MB_ICONERROR | MB_OKCANCEL, 0);  
    exit(0);  
}  
//취득한 INtime 측의 핸들러를 사용해 데이터의 수수로 사용하는 메일 박스를 검색  
//핸들을 취득한다.  
If((m_RtGraphMailbox=ntxLookupNtxhandle(m_InTimeDocProcess, rt_nt_mbx, 0xffff))  
== NTX_BAD_NTXHANDLE )  
{  
    AfxMessageBox("메일 박스 핸들 취득 에러" , MB_ICONERROR | MB_OKCANCEL, 0);  
    exit(0);  
}
```

## 통신 타이밍

Ntime측에서 송신된 메일 박스를 수신하는 타이밍은 샘플에 있던 타이머(OnTimer())를 사용하면 좋은 것이라고 생각해, 타이머를 사용해 수신하는 타이밍을 작성했습니다.그러나, 이것으로는 RT측과 동기를 취하지 않고, NT측의 독자 폴링 주기(타이머 주기)에 메일 박스를 검색하는 처리가 되어 버리기 위해 스레드를 생성해 그 스레드의 선두에서 메일 박스에 오브젝트를 수신하는 처리로 변경했습니다.

이 스레드 작성에 고생했습니다.

NT측에서의 스레드를 어떻게 작성하면 좋은 것인지 몰랐던 때문, VC의 헬프에서"스레드"로 검색해 조사했습니다.

나온 함수는, \_beginthread 함수가 나왔습니다. 이 함수를 사용해 작성하는 일로 했습니다. 이 함수의 제1 인수(새로운 스레드의 실행 개시 주소)의 설정이 능숙하게 할 수 없기 때문에 체념 CreateThread 함수를 사용하는 일에 이 함수를 사용해도 같은 어려움도 안되었습니다.

다음에 CWinThread 클래스를 사용해 보았습니다. CWinThread 클래스의 상속 클래스를 우선 작성했습니다.

사용법(함수의 사용 방법)등 어떻게 하면 몰랐던 때문, 이번은 인터넷으로 조사해 보는 일로 했습니다. 거기에 조금 참고에 할 수 있을 것 같은 간단한 것을 찾아냈습니다. 작성한, 스레드를 기동하기 위해서는

afxBeginThread 함수를 콜 하면 좋은 것 같았습니다.

afxBeginThread 함수가 콜 되는 일에 의해 CWinThread 오브젝트를 신규에 작성해 CreateThread를 호출하게 되어 있는 것 같습니다.

실제, 그래프의 묘화를 행하고 있는 클래스와 이번 스레드내에서 데이터를 취득한다고 말하는 일로, 취득한 데이터의 교환을 어떻게 하는지가 문제가 되었습니다.

스레드내에 다이얼로그측의 데이터를 건네주는 방법이 전혀 모르는 상태였습니다.

클래스의 멤버 변수로서 취급하는 일로 데이터의 교환을 실시하도록 수정했습니다만, 이라고 메모리 파괴를 일으키는 상황이었습니다.

이것에는 곤란했습니다. 이 교환을 할 수 없으면 그래프의 표시를 할 수 없기 때문입니다.

최종적으로, 원인을 알 수 있었습니다. 내가, 스레드클래스포인터로서 취급하고 있던 값이 완전히 다른 것이었기 때문입니다.

AfxbeginThread() 콜로 취득한 값이, 스레드클래스의 포인터로, 그 값을 사용해 스레드클래스의 멤버 변수등에 액세스 하지 않으면 안 되는 것을, RUNTIME\_CLASS(XXXX←클래스명)로의 값으로 액세스 하려고 하고 있었습니다.

이것을 고치는 것으로, 스레드클래스의 멤버 변수 등에 액세스 할 수 있게 되어, 수중에 넣은 데이터를 다이얼로그 함수로 사용할 수 있어 그래프의 묘화를 행할 수 있게 되었습니다.

## 10월 29일...로그 출력 스레드

### NT측 그래프 표시 어플리케이션

마지막에 로그 출력 스레드를 작성했습니다. 로그스레드의 순서로서 파일명을 생성 후, 스레드 개시시에 동명 파일의 유무를 확인해, 존재하는 경우는 삭제합니다.

헤더 부분을 파일 출력해, 파일을 클로уз.그 다음은 데이터를 받을 때마다 파일을 append 모드로 열려, 데이터를 추가합니다. 시스템의 종료시, 파일 포인터가 NULL 이외의 경우, 파일을 클로уз 해, 종료합니다

### 소스 코드상 :

```
void LogThread(void)
{
    WORD          wActual;
    BYTE          byMessage[128];
    WORD          *valData;
    /* 파일명 취득 */
    GetFileName(fileName);

    /* 벌써 존재하고 있는 로그 파일을 삭제한다 */
    remove( fileName );

    /* 로그 파일 오픈 */
    fpLog = fopen(fileName,"w");
    if (fpLog == NULL){
        WriteErrorProc(1); /* 오픈 에러 처리 */
    }

    /* 로그 파일 출력 표제 1 */
    if(fprintf(fpLog, " CH0 1 2 3 4 5 6 7\n") < 0){
        WriteErrorProc(2); /* 기입 에러 처리 */
    }

    /* 로그 파일 출력 표제 2 */
    if (fprintf(fpLog, "-----\n") < 0){
        WriteErrorProc(2); /* 기입 에러 처리 */
    }

    fclose(fpLog);

    while (1) {
        /*로그 파일 append 모드로 열리는
        fopen(fileName,"a+");
        valData = (WORD*)byMessage;
        /*아날로그 데이터 혼잡 스레드, 또는 의사 데이터 작성 스레드보다 데이터
        형태 메일 박스에서 데이터를 수신 */
        wActual = ReceiveRtData(strInfo.hParamMBox, valData
        , WAIT_FOREVER);

        if (wActual == 0)
            /* 에러 체크 : 데이터 바이트수 0BYTE 시 */
            Fail("로그 데이터 취득 에러");
        /*로그 파일의 써
        if(fprintf(fpLog," %4d %4d %4d %4d %4d %4d %4d %4d\n",
        valData[0],valData[1],valData[2],valData[3],
        valData[4],valData[5],valData[6],valData[7]) < 0)
            WriteErrorProc(2); /* 기입 에러 처리 */

        fclose(fpLog);
    }
}
```

### 어플리케이션의 종료시, 파일 포인터가 NULL가 아닌 경우 :

```
if ( fpLog != NULL )
    fclose(fpLog );
fpLog = NULL;
파일을 닫아 어플리케이션을 종료한다.
```

## 10월 30일...그 외 시스템의 정리

### 최종 체크

#### 과제

시스템 전체로 실장하는 기능의 누락이 없는가를 체크했습니다.

특히, 「24시간 가동 테스트」는 테스트에 시간이 걸릴 뿐만 아니라, 만약 테스트중에 시스템이 다운했을 경우, 어느 지점에서 다운했는지를 찾아내는데 고생을 합니다.

이번은 24시간 의욕 첨부로 체크하는 것이 이길 수 없었기 때문에, 통런 테스트를 8시간에 좁혀, 그 중에 문제가 있는 부분의 수정등을 실시하도록 했습니다

마지막에 로그 출력 스크드를 작성했습니다.로그스크드의 순서로서 파일명을 생성 후, 스크드 개시시에 동명 파일의 유무를 확인해, 존재하는 경우는 삭제합니다.

헤더 부분을 파일 출력해, 파일을 클로우즈.그 다음은 데이터를 받을 때마다 파일을 append 모드로 열려, 데이터를 추가합니다.시스템의 종료시, 파일 포인터가 NULL 이외의 경우, 파일을 클로우즈 해, 종료합니다.

RT측, Windows측 모두 거의 처리를 실장할 수 있어 최종적으로 프로젝트의 정리를 실시했습니다.

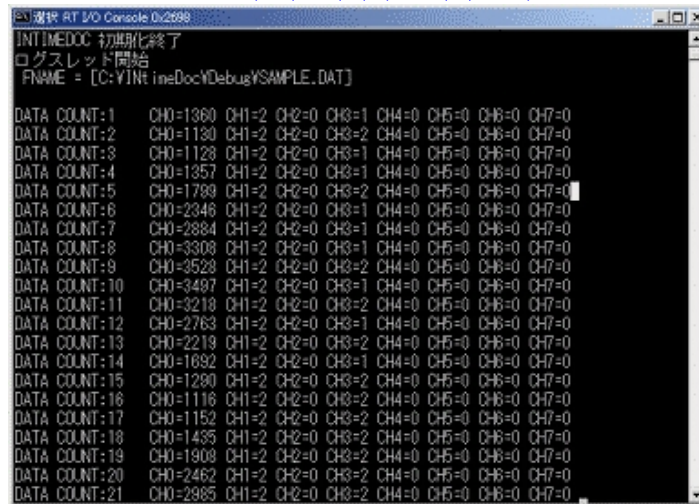
· 실수를 지정해 있던 부분을 정수로서 정의

· 코멘트를 쓴다

· 변수등의 정리

· 메모리 개방의 철저 FreeRtMemory()...맵 한 메모리의 개방을 잊고 있는 부분에서 FreeRtMemory()를 기술한다.  
(24 시간 테스트로 시스템 다운 하는 제일의 원인이라고 생각되는 부분입니다.)

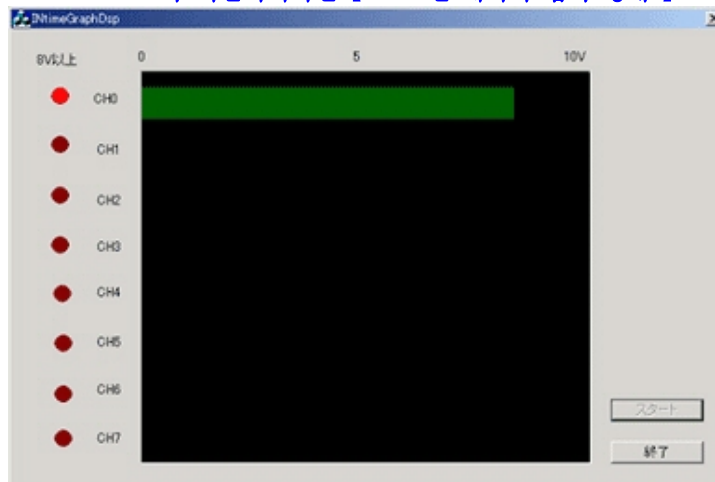
#### INtime 측 어플리케이션 디버그 화면



```
선택 RT I/O Console 0x2698
INTIMEDOC 初期化終了
ログスレッド開始
FILENAME = [C:\VInt\ineDoc\YDebug\VSAMPLE.DAT]

DATA COUNT:1   CH0=1360 CH1=2 CH2=0 CH3=1 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:2   CH0=1130 CH1=2 CH2=0 CH3=2 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:3   CH0=1128 CH1=2 CH2=0 CH3=1 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:4   CH0=1357 CH1=2 CH2=0 CH3=1 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:5   CH0=1799 CH1=2 CH2=0 CH3=2 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:6   CH0=2346 CH1=2 CH2=0 CH3=1 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:7   CH0=2884 CH1=2 CH2=0 CH3=1 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:8   CH0=3308 CH1=2 CH2=0 CH3=1 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:9   CH0=3528 CH1=2 CH2=0 CH3=2 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:10  CH0=3497 CH1=2 CH2=0 CH3=1 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:11  CH0=3218 CH1=2 CH2=0 CH3=2 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:12  CH0=2763 CH1=2 CH2=0 CH3=1 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:13  CH0=2219 CH1=2 CH2=0 CH3=2 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:14  CH0=1892 CH1=2 CH2=0 CH3=1 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:15  CH0=1290 CH1=2 CH2=0 CH3=2 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:16  CH0=1116 CH1=2 CH2=0 CH3=2 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:17  CH0=1152 CH1=2 CH2=0 CH3=2 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:18  CH0=1435 CH1=2 CH2=0 CH3=2 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:19  CH0=1908 CH1=2 CH2=0 CH3=2 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:20  CH0=2462 CH1=2 CH2=0 CH3=2 CH4=0 CH5=0 CH6=0 CH7=0
DATA COUNT:21  CH0=2985 CH1=2 CH2=0 CH3=2 CH4=0 CH5=0 CH6=0 CH7=0
```

#### Windows 측 어플리케이션 【 0 ch 만 데이터 입력 상태 】



## 정리

처음으로 INtime 어플리케이션의 작성·보드에 대한 프로그래밍을 실시했습니다만, 무사히 종료할 수 있었습니다. 문서의 내용에서는 실행한 내용이 100%반영되고 있을까 불안합니다만, 원시 코드와 함께 봐주시고, 무엇인가 참고가 되면 다행이겠습니다. 또 프로그래밍 기술 방법이나, 문서 작성 능력 등, 나 자신이 아직도 미숙하기 때문에, 폐를 끼치기도 할까 생각합니다만, 용서해 주십시오. 끝까지 정독해 주셔서 감사합니다

필자