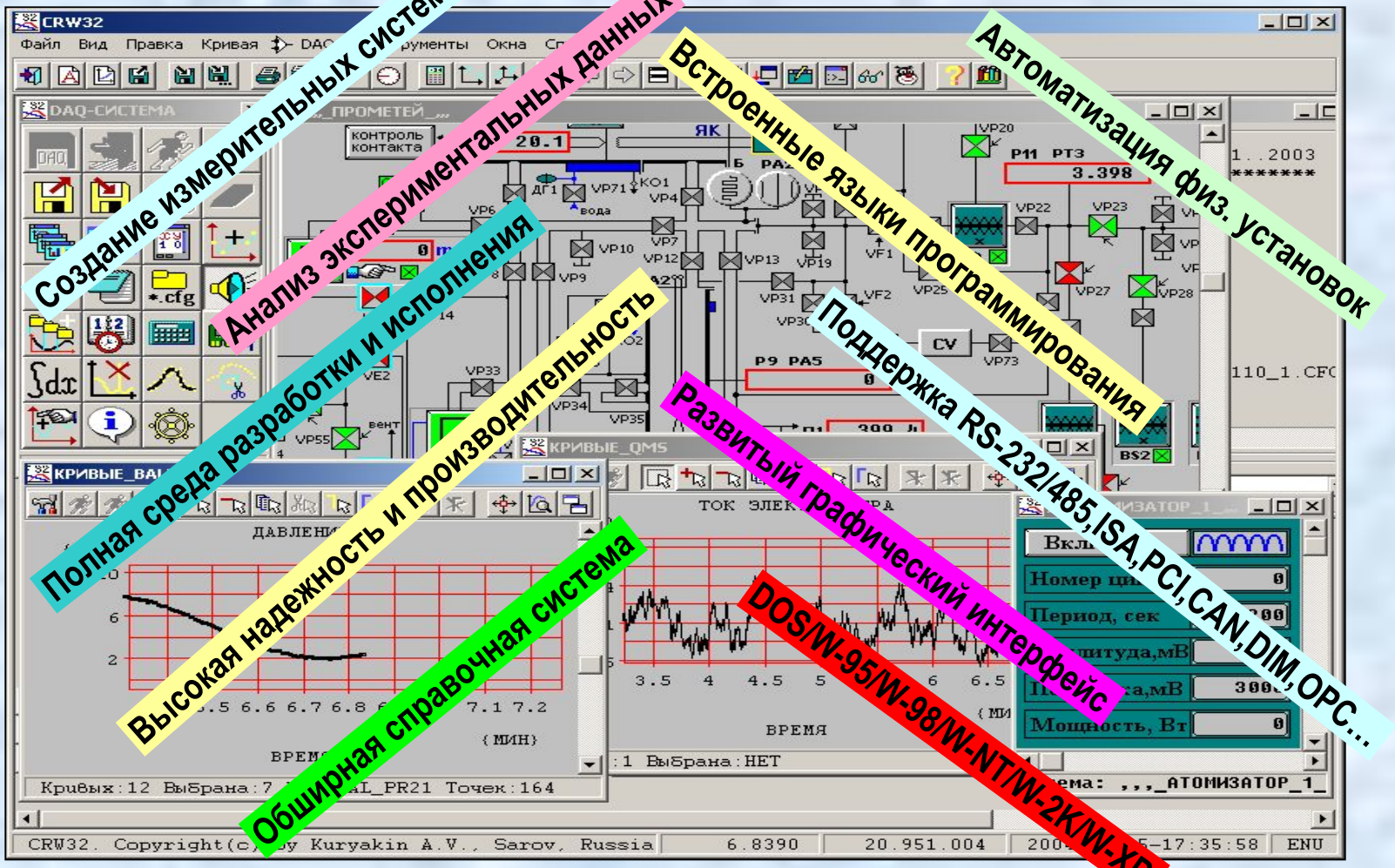
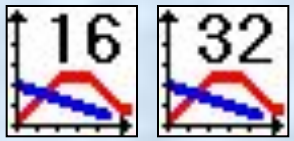
CRW-DAQ

Real Time
в пакете CRW-DAQ





CRW-DAQ

О чем эта презентация

Пакет CRW-DAQ активно используется для автоматизации исследовательских физических установок в

- Саров (ИЯРФ; 19 отд.; 33 цех; Бинар)
- Дубна (ЛЯР,ОИЯИ,ACCULINA; ЛЯП,ОИЯИ,ТРИТОН)
- Ст.Петербург(Университет, институт Бонч-Бруевича)
- Geneve, Switzerland (CERN,ALICE,PHOS)
- Москва(Университет)
- Екатеринбург
- Череповец…

Есть документация и обширный опыт эксплуатации конкретных измерительных систем

Однако есть большой недостаток документации по встроенным в CRW-DAQ методам сбора и обработки данных

Данная презентация – попытка частично восполнить этот пробел

CRW-DAQ

Назначение пакета CRW-DAQ

1. Сбор данных и управление в реальном времени

- Сбор экспериментальных данных в реальном времени
- Управление физическими установками в реальном времени
- Визуализация - отображение измеряемых данных в режиме online
- Архивация - накопление измеренных данных для последующего анализа

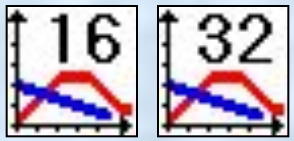
2. Предварительный анализ экспериментальных данных

- Сохранение \ загрузка накопленных экспериментальных данных
- Визуализация – отображение измеренных данных в режиме offline
- Манипуляции с кривыми: удаление фрагментов, фона, арифметические операции над кривыми, сглаживание, сортировка, интерполяция и т.д.
- Передача данных в специализированные пакеты обработки данных

CRW-DAQ содержит специальные средства для решения Real Time задач:

- поддержка приоритетного многопоточного режима работы
- поддержка быстрого опроса потоков (до 1000 Hz)
- средства быстрого и точного измерения времени
- специальные средства быстрого ввода-вывода
- службу мониторинга частоты опроса потоков
- сторожевой таймер потоков - Watchdog
- службу системного времени

Этому и посвящена данная презентация.



CRW-DAQ

Специфика Real Time

Все хотят чтобы программа работала быстро, но смысл в слово быстро вкладывают разный.

Математика – нужно обеспечить быстрые и точные расчеты, критерий качества – минимизация общего (интегрального) времени работы.

Интерактивная графика - нужно обеспечить минимальное интегральное время работы, но время реакции на действия пользователя должно быть приемлемым, < 100 мс.

Реальное время – интегральное время работы не играет никакой роли. Оно может составлять годы и определяется внешним объектом (физ. установкой). Критерий качества – предсказуемость работы и гарантированное время реакции. Гарантия означает обеспечение заданного времени реакции в худшем случае.

Т.о. гарантированное время реакции – характерная особенность Real Time систем. Дальнейшее деление происходит по тому, каково это гарантированное время:

< 10μs – только Hardware (FPGA, Signal Processors и т.д.)

10μs…10ms – Real Time OS (QNX, OS9000, Solaris, eCos, DOS etc)

> 10ms – Windows, Unix, Linux + искусство программиста

"Жесткое" реальное время – работа по аппаратным прерываниям устройств, характерное время реакции 10 микросекунд.

"Мягкое" реальное время – работа в режиме программного опроса (прерывания таймера), характерное время реакции 10 миллисекунд.

Crw32 (версия CRW-DAQ для Windows) – система "мягкого" реального времени с временем реакции порядка 10 ms.

Crw16 (версия CRW-DAQ для DOS DPMI) позволяет строить системы "жесткого" реального времени с реакцией порядка 10 μs.

CRW-DAQ

Функции измерения времени

Первое, что требует Real Time – правильно измерять время. Характеристики функций измерения времени для Real Time систем таковы:

- Квант времени
- Стоимость вызова
- Гарантия монотонности
- Гарантия непрерывности
- Единицы измерения и начало отсчета времени
  - Астрономическое (календарное) \ локальное (от старта системы) время
  - GMT (Greenwich Mean Time) \ LMT (Local Mean Time)
- Thread safe – потоковая безопасность

Win32:

1) GetSystemTimeAsFileTime(…)-время астрономическое, GMT, вызов 40 ns, квант 10 ms, монотонность и непрерывность НЕ гарантируется (смена системного времени, часовой зоны, летнее/зимнее время). В прямом виде для Real Time не пригодна.
2) GetTickCount – время локальное, вызов 20 ns, квант 10 ms, монотонность и непрерывность гарантируется (счетчик прерываний таймера), но только с учетом overflow через 49.71 суток. В прямом виде для Real Time не пригодна.
3) QueryPerformanceCounter – время локальное, вызов 1 µs, квант 1 µs, монотонность и непрерывность гарантируется. В прямом виде для Real Time не пригодна (требует дополнительно вызова QueryPerformanceFrequency).

CRW-DAQ: Использует гибрид вызовов Win32, устраняя их недостатки (_rtc.pas)

1) msecnow - время астрономическое, LMT, стоимость вызова 90 ns, квант 10 ms, монотонность и непрерывность гарантируется, thread safe.
2) mksecnow – время локальное, стоимость вызова 1 µs, квант 1 µs, монотонность и непрерывность гарантируется, thread safe.

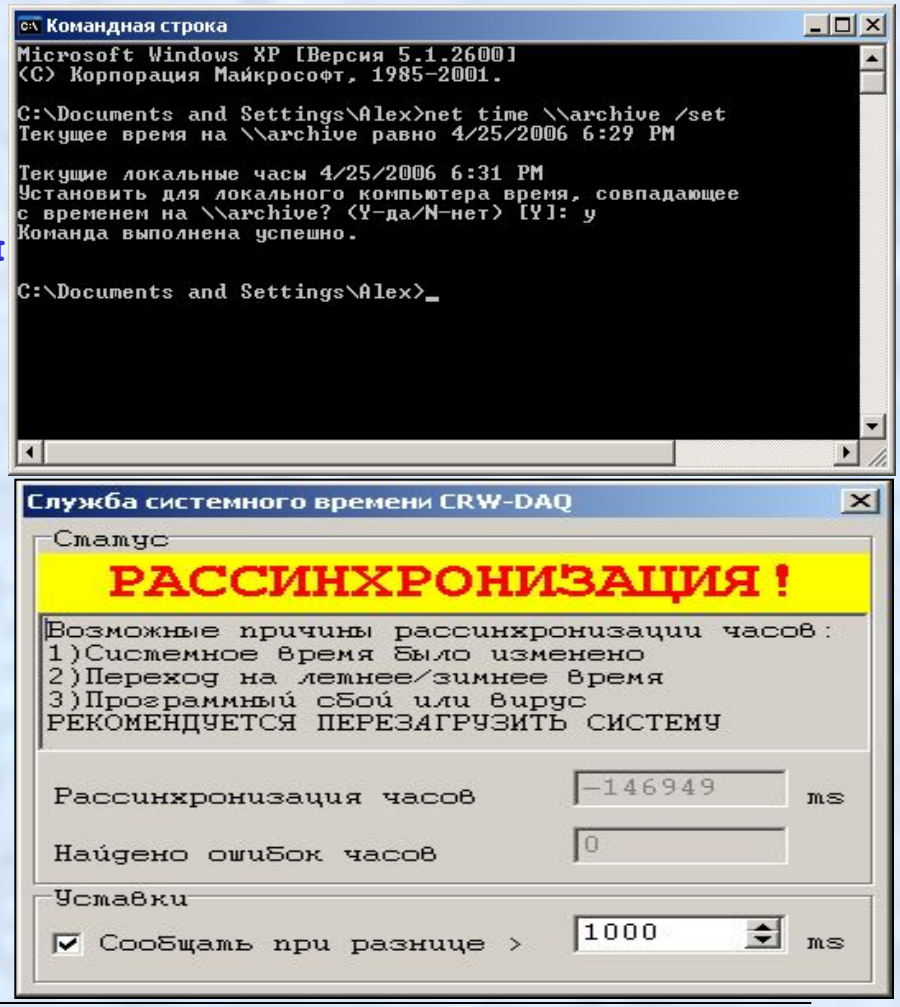
CRW-DAQ

Средства поддержки Real Time (Служба времени CRW-DAQ)

Часы CRW-DAQ монотонны и непрерывны, чтобы обеспечить корректную логику работы Real Time системы (т.е. правильные задержки, временные интервалы и т.д.).

Поскольку системное время Windows не монотонно, а системное время CRW-DAQ гарантированно монотонно, возможна разница в показаниях часов Windows и CRW-DAQ (рассинхронизация часов). Служба времени детектирует наличие рассинхронизации и выдает предупреждение.

Пример: при сетевой синхронизации часов по команде net time фиксируется рассинхронизация часов.



CRW-DAQ

Система приоритетов Windows

Windows имеет карусельную (Round Robin) дисциплину диспетчеризации потоков с учетом приоритета и кванта времени (10 ms для single, 15 ms для multi processor). Квант времени получает поток с наивысшим приоритетом (вытесняющая приоритетная многозадачность). Диспетчеризации подлежат именно потоки, а не процессы, т.к. процесс – это просто контейнер для потоков.

Класс приоритета процесса определяет диапазон приоритетов потоков.

Windows имеет 32 уровня приоритета: 0..15 - обычные приоритеты, 16..31- приоритеты процессов реального времени. Это значит, любой поток RealTime процесса вытесняет любой поток процессов Idle,Lower,Normal,Higher,High.

Класс приоритета процесса **PriorityClass**:
- 4 = Idle - для фоновых задач.
- 6 = Lower
- 8 = Normal - обычный, по умолчанию.
- 10 = Higher
- 13 = High
- 24 = RealTime - для Real Time задач.

Относительный приоритет **RelPriority** потока:
-8 = tpIdle - для фоновых потоков.
-6 = tpLowest
-4 = tpLow
-2 = tpLower
0 = tpNormal - обычный, по умолчанию.
+2 = tpHigher
+4 = tpHigh
+6 = tpHighest
+8 = tpTimeCritical - для Real Time потоков.

Процессы с классами приоритета **Idle..High** могут иметь потоки только в диапазоне обычных приоритетов 0..15. Работает правило типа:
AbsPriority=Max(1,Min(15,PriorityClass+RelPriority))

Процессы с классом приоритета **RealTime** могут иметь потоки только в диапазоне приоритетов реального времени 16..31. Работает правило:
AbsPriority=Max(16,Min(31,PriorityClass+RelPriority))

CRW-DAQ

Проблема инверсии приоритетов

Имеется поток с приоритетом tpTimeCritical(15). Программист вызывает TThread.Synchronize. Synchronize вызывает SendMessage, блокирует поток и передает управление потоку VCL с приоритетом tpNormal(8). В это время активизируется другой поток с приоритетом tpHigher(10), который занимает 100% CPU. Более высокоприоритетный поток (15) никогда (!) не сможет получить управление из-за активности более низкоприоритетного потока (10). Это – инверсия приоритета.

Мораль:

- не используйте Synchronize
- минимизируйте время блокировки общих ресурсов

К сожалению, в рамках Windows вероятность инверсии приоритетов принципиально остается всегда. Это тяжелая проблема даже для самых лучших Real Time OS.

CRW-DAQ

Потоковая модель CRW-DAQ

- Для CRW-DAQ характерна очень высокая степень параллелизма. В типичной DAQ системе работает 30-50 потоков и более. Каждая прикладная программа Daq Pascal выполняется в своем потоке. За счет асинхронного стиля программирования эффективная степень параллелизма (вытесняющий + корпоративный) еще выше.
- Класс приоритета приложения задается в конфигурационном файле.
- Потоки имеют фиксированные приоритеты и периоды опроса, задаваемые в конфигурационном файле. Каждый поток работает как простой автомат. Всю неформальную работы обеспечивает пользовательская программа (обозначенная здесь как UserPollingCallback). Например, для устройств program это вызов программы пользователя на языке Daq Pascal.
- Сторожевой монитор Watchdog следит за опросом потоков. Если поток не сбросил Watchdog за заданное время, система фиксирует факт "подвисания" потока.
- Поток может быть разбужен досрочно внешним событием:
  - сообщением от другой программы (devsend,devmsg)
  - от графического интерфейса пользователя

```
[Daq]
ProcessPriorityClass = RealTime
…
[DeviceList]
Demo = device software program
[Demo]
InquiryPeriod = 10
DevicePolling = 10, tpTimeCritical
…
```

```
// Псевдокод опроса потоков
Procedure Thread.Execute;
Var t,p,q:Double; event:Boolean;
begin
 While not Terminated do begin
  t:=msecnow;
  ResetWatchDog(t);
  event:=WaitForSingleObject(hEvent,DevicePolling)=WAIT_OBJECT0;
  if (t-q>InquiryPeriod) or event then begin
   UserPollingCallback; FillHistogramm(t-p); q:=t;
  end;
  p:=t;
 End;
End;
```

CRW-DAQ

Асинхронный стиль программирования

Для CRW-DAQ характерен асинхронный стиль программирования или стиль "по флагам состояния" а не стиль "жди ответа".

```
Program AsynchronousStyle;
Var state:integer; t0:Double;
Begin
 case state of
  0: begin
      SendRequest;
      t0:=msecnow;
      State:=1;
     end;
  1: begin
      if WaitForAnswer(0) then begin
       HandleAnswer;
       State:=0;
      end else
      if msecnow-t0>TimeOut then begin
       TimeOutDetected;
       State:=0;
      end;
     end;
 end;
 DoSomethingMore;
End;
```

```
Program SynchronousStyle;
Begin
 SendRequest;
 if WaitForAnswer(TimeOut)
 then HandleAnswer
 else TimeOutDetected;
End;
```

Синхронная программа выглядит короче, но простота ее обманчива. Вызов синхронной функции блокирует поток на время TimeOut и поток ничего больше не может делать, даже "убить" его корректно нельзя.
Зато в асинхронном случае можно DoSomethingMore с частотой 100 Hz. И поток никогда не повиснет, даже если ответа от устройства нет и не будет.
Программы CRW-DAQ обычно что-то быстро-быстро делают "по флагам" и сразу отдают управление системе, чтобы не препятствовать выполнению других потоков. За счет асинхронного стиля параллелизм (вытесняющий + корпоративный) становится еще выше (сотни и тысячи).

CRW-DAQ

Средства поддержки Real Time (мониторинг и гистограммы)

Система постоянно "отслеживает" период (частоту) опроса потоков, строит гистограммы

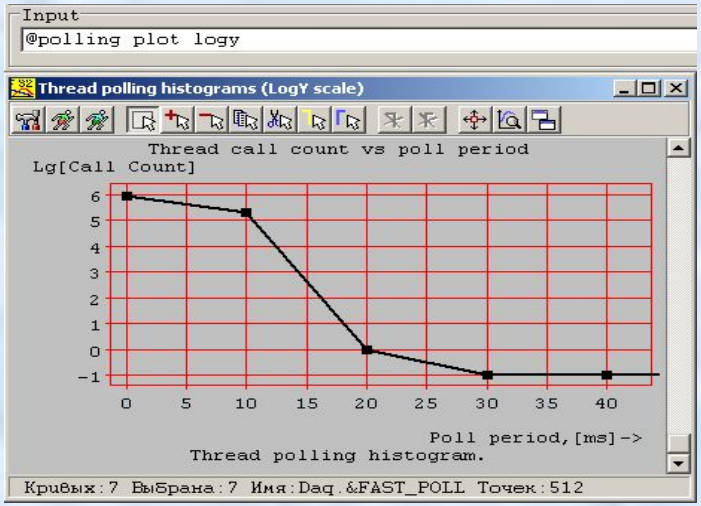
1) **Мониторинг частоты опроса потоков**

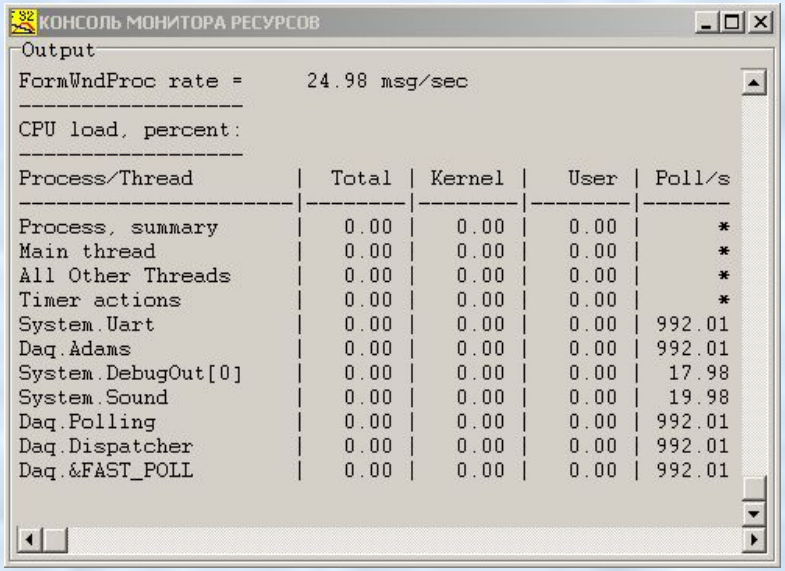
2) **Сторожевой таймер WatchDog для детектирования повисших потоков**

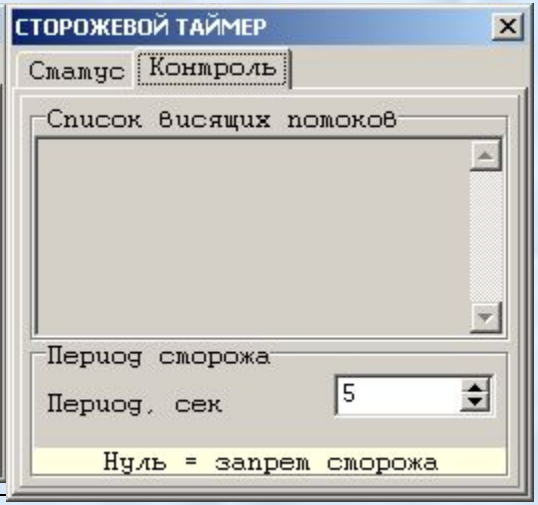
3) **Гистограммы периода опроса потоков**

Консольные команды (окно "Главная консоль")

- **@polling list - список потоков**
- **@polling plot logy - график гистограмм**
- **@polling PriorityClass RealTime - установка приоритета**







CRW-DAQ

Много-процессные системы под управлением CRW-DAQ

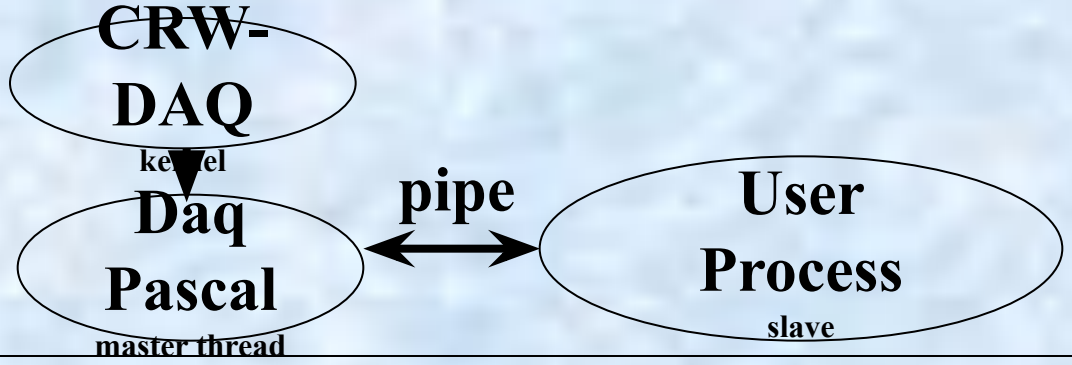
Встроенный язык Daq Pascal содержит библиотеку для организации много-процессных систем, при этом супервизором (родительским процессом) выступает CRW-DAQ. В рамках CRW-DAQ работает программа (поток) на встроенном языке Daq Pascal, управляющая (master) дочерним процессом. Возможен запуск дочернего процесса (обычно простого консольного приложения) с обменом по каналу (stdin, stdout переназначаются в анонимный канал).

Много-процессные системы позволяют повысить надежность системы:

- дочерние (slave) процессы могут "падать" без вреда для супервизора, "упавший" дочерний процесс перезапускается, если надо.
- дочерние процессы – простые и понятные консольные задачи, разработанные обычно в среде CRW-DAQ, на встроенном компиляторе Delphi.

"Сомнительные" по надежности и переносимости (содержащие COM, ActiveX, OPC…) подсистемы помещаются в дочерние процессы. Ядро CRW-DAQ остается легким и надежным.

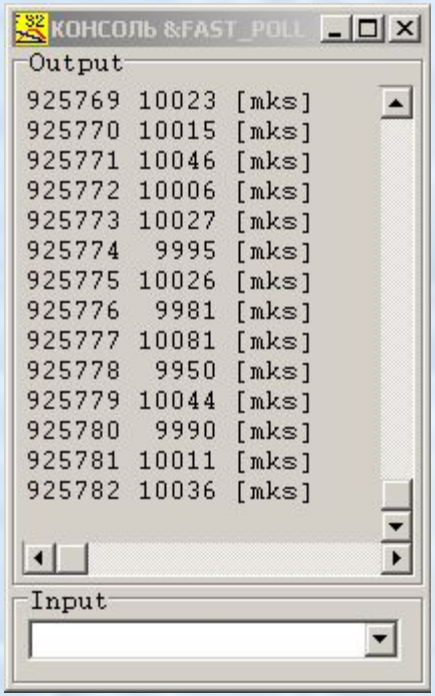
Кроме того, дочерние процессы могут иметь класс приоритета, отличный от класса приоритета CRW-DAQ, что иногда бывает необходимо (см. далее спектрометрия).

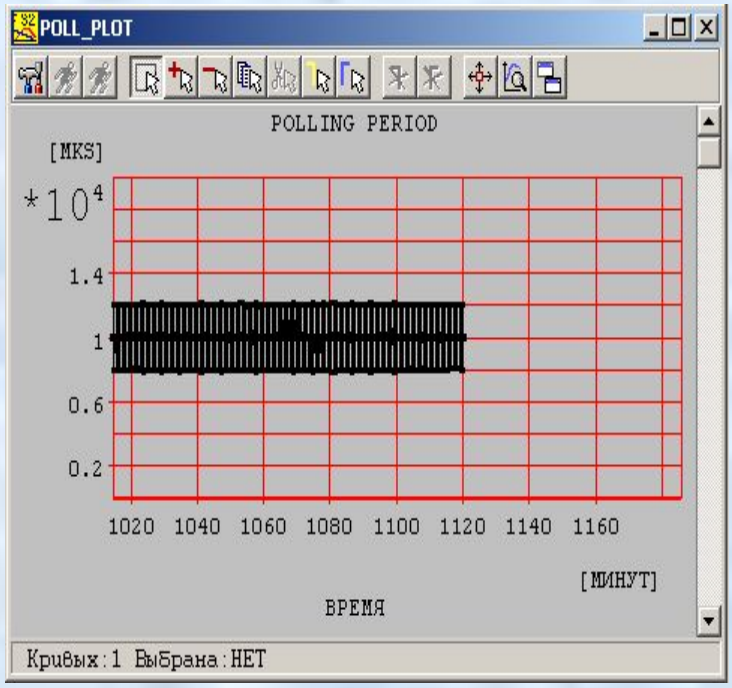


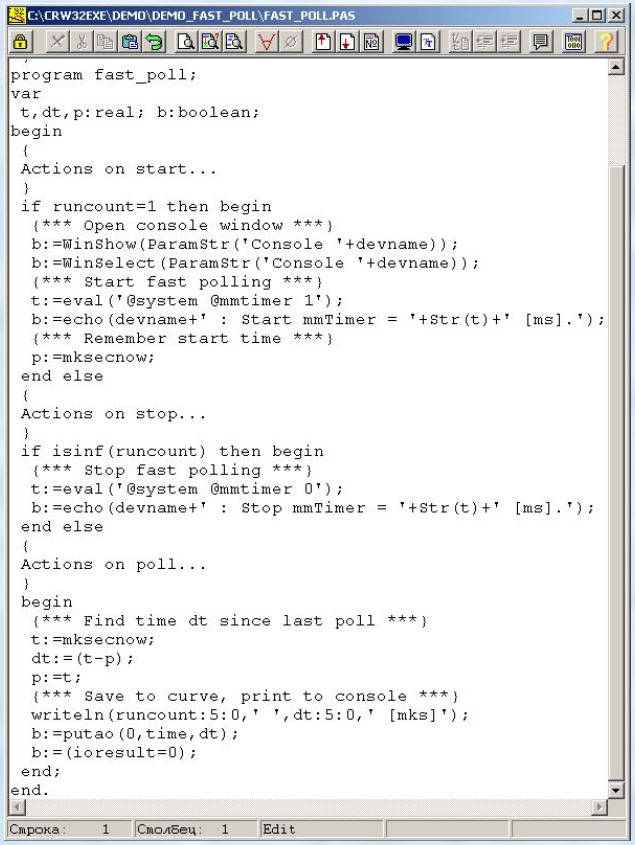
CRW-DAQ

Частота опроса потоков (стандартная модель)

Стандартная потоковая модель при правильном выборе приоритетов потоков позволяет опрашивать потоки со средним периодом 10ms (15ms для мультипроцессорных систем), однако при этом гарантировать можно порядка 20 ms.







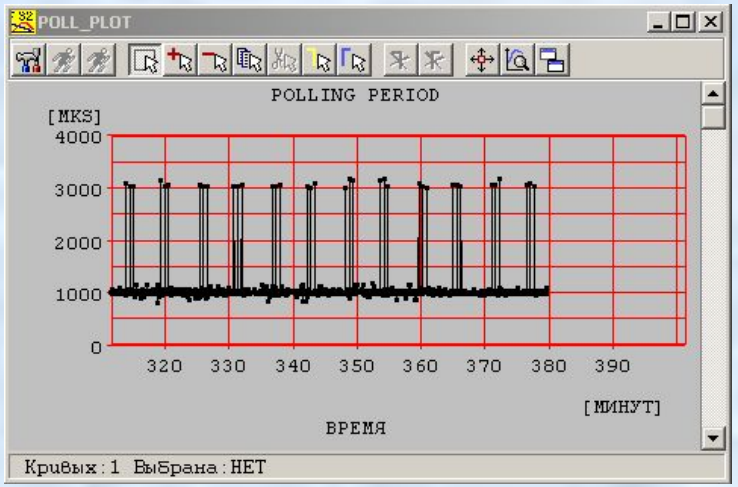
CRW-DAQ

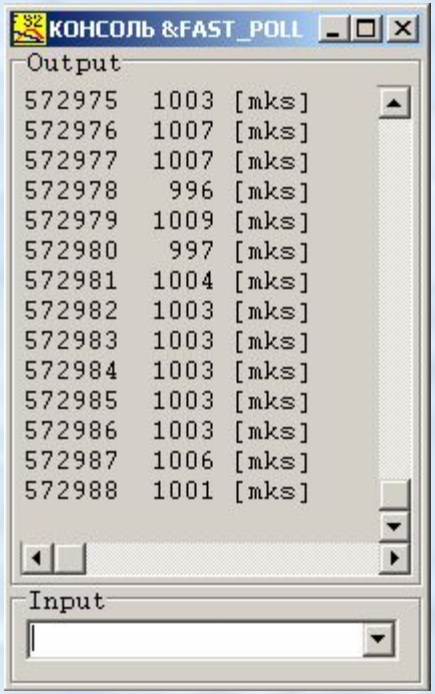
Частота опроса потоков (mmtimer)

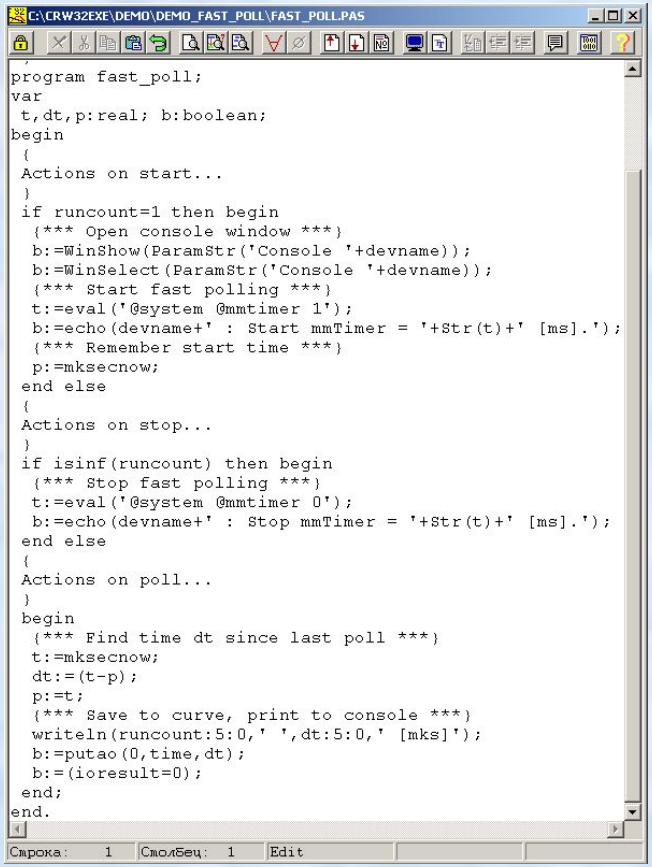
Используя mmtimer, при правильном выборе приоритетов потоков, можно опрашивать потоки с средним периодом 1 ms, однако при этом гарантировать можно опрос порядка 5 ms.

mmtimer активизируется консольной командой

@mmtimer 1





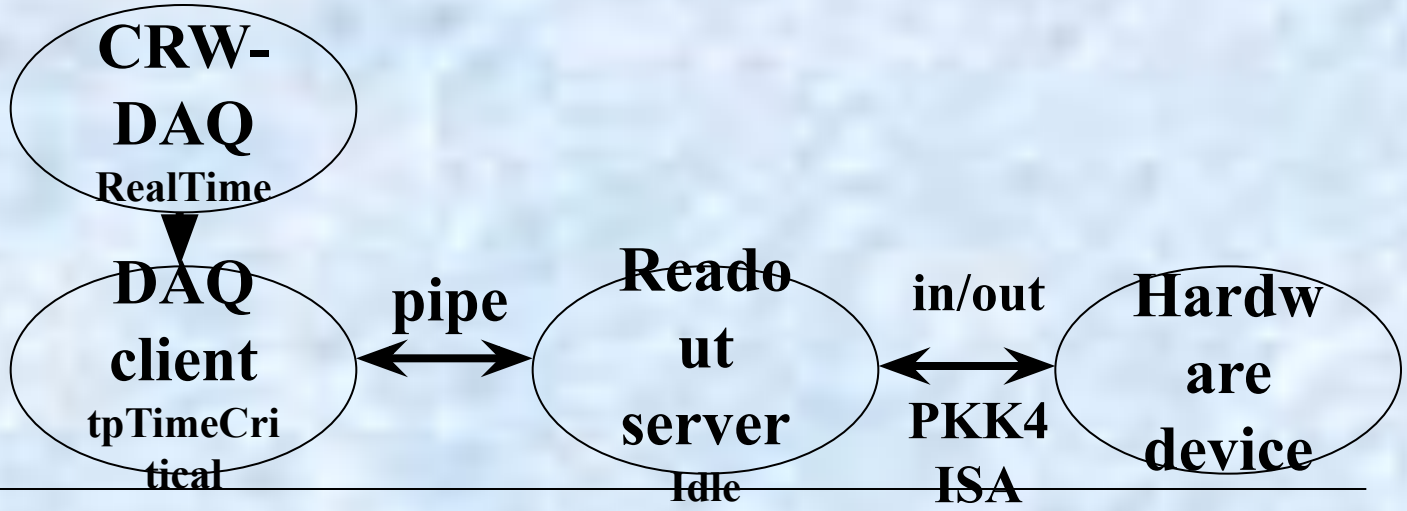


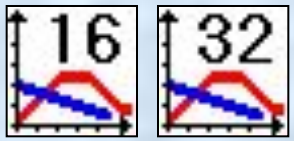
CRW-DAQ

Спектрометрия (быстрый сбор данных)

В ряде случаев (наличие аппаратного буфера, специфика задачи (спектрометрия)) требуется высокая частота опроса **в среднем**, а не гарантированное время реакции.

В этом случае применяется такая модель: CRW-DAQ работает с классом приоритета RealTime, клиент на Daq Pascal – в потоке tpTimeCritical, Windows Shell – с приоритетом Normal. Измерительная задача запускается как отдельный процесс с классом приоритета Idle, в режиме непрерывного опроса, она занимает 99% CPU, не мешая при этом всем остальным. Мертвое время измеряется аппаратными средствами (счетчик с воротами).





CRW-DAQ

Быстрые средства ввода-вывода

Прямой ввод-вывод через порты процессора (IN,OUT) применяется для программирования ISA, PCI устройств, если нет фирменных драйверов или они не устраивают по своей функциональности или скорости.

Команды IN,OUT в CPU x86 – привилегированные. Их прямое использование из User Mode под Windows NT запрещено.

Каждый процесс имеет IOPM (I/O permission map).

User Mode □ IOPM □ Kernel Mode □ Port

IOPM – битовая карта 8 kB, которая разрешает/запрещает 64 kB портов ввода-вывода для User Mode. В режиме ядра порт всегда разрешен.

К вводу/выводу из User Mode 2 подхода:

1) Классический – драйвер режима ядра, время IO > 30µs

User Mode посылает сообщение, которое выполняет драйвер режима ядра, недостаток – тратится время на переключение User□Kernel□User.

2) Модификация IOPM, т.е. снятие запрета I/O, время I/O ≈ 1µs

Драйвер режима ядра модифицирует IOPM (снимает запрет на I/O из User Mode), далее User делает I/O непосредственно.

CRW-DAQ реализует второй подход, используя свободно опубликованный драйвер GiveIO.Sys, см. файл _pio.pas в дистрибутиве CRW-DAQ.

В CRW-DAQ имеются драйверы PKK3, PKK4, LA-1.5-ISA, LA-1.5-PCI, LA-2-USB, DIO-144, PCL-812PG, PCL-818L и т.д.

CRW-DAQ

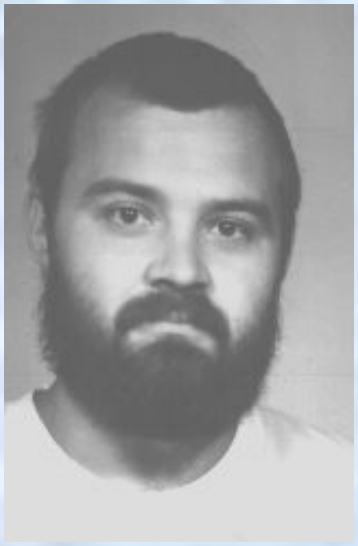
Рекомендации для Real Time систем под Win32

- Real Time – это в первую очередь процессы, потоки и приоритеты.
- Не бойтесь распараллеливания, 50 потоков – это нормально, работает быстро.
- Не используйте блокирующие вызовы: PostMessage лучше SendMessage; периодическая проверка WaitForSingleObject(h,0) лучше, чем WaitForSingleObject(h,INFINITE). Напоминаю: цель минимизировать интегральное время CPU любой ценой не стоит. Надежность и реакция – важнее.
- Не используйте VCL. Она не поддерживает многопоточный режим работы.
- Не используйте TThread.Sinchronize. Он основан на блокирующем SendMessage и нарушает принцип разделения приоритетов (блокирует высокоприоритетный поток и отдает управление низкоприоритетному – инверсия приоритетов).
- Возвращайте управление системе, как только потоку стало нечего делать. Доделаете в следующем кванте времени.
- Минимизируйте время блокировки (Lock/Unlock) общих ресурсов. Быстро делайте локальные копии данных и работайте с ними после разблокировки.
- Используйте ReadFile, WriteFile только в режиме Overlapped I/O.
- Используйте асинхронный стиль программирования “по флагам”, а не “жди ответа”.
- По возможности избегайте использования COM, ActiveX. Эти технологии a)ставят программу в зависимость от окружения, b)ухудшают временные параметры (тяжеловесные технологии), c)делают систему плохо предсказуемой d)указанные технологии носят явно выраженный коммерческий характер и удобны скорее производителям (состричь побольше денег), а не пользователям.
- Используйте по возможности только Open Source технологии. Предсказуемость – не вопрос религиозной веры в непогрешимость любимой фирмы (Microsoft etc). Нет технологий “черного ящика”, есть технологии “кота в мешке”. CRW-DAQ сознательно использует только Open Source технологии там, где это возможно.

CRW-DAQ

А это мы, DAQ группа

Ю.И.Виноградов
д.ф.м.н.,нач.лаб.
главный методист
45877



А.В.Курякин
с.н.с.,автор CRW-DAQ
ведущий программист
31959

С.В.Фильчагин
н.с.,методист
прикладное ПО
31959

О.П.Вихлянцев
инж.,методист
прикладное ПО
31959

А.Н.Вьюшин
инж.,методист
прикладное ПО
31959

Мы работаем для ВАС

уважаемые пользователи