УДК 004.454

УПРАВЛЕНИЕ ВНЕШНИМИ УСТРОЙСТВАМИ С ПОМОЩЬЮ ДРАЙВЕРОВ WINDOWS NT С ИСПОЛЬЗОВАНИЕМ ПРЕРЫВАНИЙ

Евстратов Е.К, Теплинский С.В.
Донецкий национальный технический университет
E-mail: evgeniy.evstratov@gmail.com

Аннотация

Евстратов Е.К, Теплинский С.В. Управление внешними устройствами с помощью драйверов windows nt с использованием прерываний*. Рассматриваются принципы обработки прерываний в среде драйверов Windows NT, правила привязки прерываний к виртуальным устройствам, примеры применения.*

Введение

В Windows предусмотрены методы подключения дополнительных обработчиков прерываний к драйверам устройств, чем обеспечивается возможность полноценного ручного обслуживания внешнего устройства, а также дополнительные возможности во взаимодействии с остальными устройствами.

Основные понятия

Процедуры обработки прерывания носят название ISR (Interrupt Service Routine). В Windows все задачи имеют приоритеты, в их цепочке пользовательские приложения занимают низшее место, за ними следуют программы режима ядра (драйверы и т.д.), и самые приоритетные – прерывания. Вследствие этого сами обработчики прерываний не позволяют выполняться всем остальным, что может приводить к так называемой «деградации системы», для этого были введены процедуры отложенного вызова DPC (Deferred Procedure Call), которые относятся к уровню программ ядра. Вследствие этого если обработка прерывания может занимать значительное время, то она переносится в специальную DPC процедуру, а обработчик прерывания лишь оставляет запрос на ее вызов. DPC процедуры привязываются к объектам устройств, и запросы на их вызовы заносятся именно через идентификаторы устройств. Для прерываний введены специальные объекты, через которые производится их отвязка от ISR, отвязку нужно производить вручную. Если, например, привязать к прерыванию ISR и, не отвязывая выгрузить драйвер – по следующему соответствующему прерыванию будет выполнено обращение к недействительной области памяти, и, скорей всего, перезапуск системы.

Объект прерывания

Объекты прерывания описываются структурами KINTERRUPT , которые создаются функцией подключения прерывания к обработчику – IoConnectInterrupt, в функцию нужно передать аппаратный вектор прерывания и уровень прерывания. Рассмотрим на примере. Мы находимся в процедуре DriverEntry, зарегистрировали устройство, подключаем обработчик прерывания:

```
KAFFINITY Affinity; // структура, отвечающая за привязку к логическому
// процессору
ULONG MappedVector; // аппаратный вектор прерывания
PKINTERRUPT pInterruptObject; // указатель на объект прерывания
ULONG InterruptLevel = 7; // IRQ7
KIRQL irql = InterruptLevel; // аппаратный уровень используемых прерываний
MappedVector = HalGetInterruptVector(Isa, // тип системной шины - ISA
    0, // номер шины
```

```
        InterruptLevel, // шинный уровень прерывания
        InterruptLevel, // шинный вектор прерывания (для Isa – одинаковые)
        &irql, // уровень прерывания – выходной параметр
        &Affinity); // привязка к процессору – выходной параметр.
// аппаратный вектор прерывания получен, регистрируем обработчик
NTSTATUS status; // статус выполнения привязки
status = IoConnectInterrupt(&pInterruptObject, // получим объект прерывания
        IsrRoutine, // наш обработчик прерывания
        pDeviceObject, // дополнительный параметр, передаваемый при вызове ISR
        NULL, // спин-блокировка - не используем
        MappedVector, // аппаратный вектор
        irql, // уровень прерывания
        irql,
        Latched, // тип прерывания – по фронту
        FALSE, // не общедоступный вектор
        Affinity, // привязка к процессору
        FALSE); // не сохраняем FPU, MMX регистры при вызове
if(status != STATUS_SUCCESS)
{ // ошибка
}
else
        // успех, теперь прерывание подключено к обработчику.
```

Здесь мы подключили прерывание на IRQ7 шины ISA(0), что, в основном, соответствует портам LPT1, LPT2.

По завершению работы нужно отключить прерывание:

```
IoDisconnectInterrupt( pInterruptObject ); // функция ничего не возвращает
```

Функция обработчика прерывания (ISR) должна иметь вид:

```
BOOLEAN IsrRoutine(IN PKINTERRUPT pInterrupt,IN PVOID pContext);
```

Она принимает указатель на объект самого прерывания (чем дает возможность сделать отвязку в самой обработке), и «контекст устройства» – указатель без типа, это тот самый «дополнительный параметр, передаваемый при вызове» который был указан в IoConnectInterrupt, в данном случае это будет pDeviceObject.

Процедура отложенной обработки (DPC)

DPC, как было сказано, привязывается к объекту устройства:

```
IoInitializeDpcRequest(pDeviceObject, // указатель на объект устройства
        DpcRoutine); // процедура отложенного вызова
```

Функция ничего не возвращает. Сама DPC должна иметь вид:

```
VOID DpcRoutine(IN PKDPC Dpc,IN PDEVICE_OBJECT pDeviceObject,IN PIRP pIrp,IN PVOID pContext);
```

Она получает структуру, содержащую информацию об отложенном вызове, ее объект устройства, указатель на пакет ввода/вывода (IRP), дополнительный параметр на усмотрение программиста. Теперь зарегистрировать вызов можно следующим вызовом:

```
IoRequestDpc(pDeviceObject, // указатель на объект устройства
        NULL, // указатель на IRP пакет – не используем
        NULL); // наш дополнительный параметр – не используем
```

После этого DPC поставлена в очередь на выполнение. Как видно – единственный обязательный параметр это ссылка на объект устройства, собственно без него и неизвестно что вызывать. Храня нужные данные в расширении устройства можно практически не

использовать остальные параметры.

Эксперименты

Было создано устройство, генерирующее случайные числа по запросам от компьютера. Оно подключается через порт LPT1 и использует прерывание, приведенное в примере выше, для эффективности работы системы, т.к. LPT порт работает на частоте до 100 кгц, а устройство – до 15 кгц, драйвер оставляет запрос на новое число и отдает управление системе, не ожидая впустую готовности устройства. По готовности устройство выставляет данные и генерирует прерывание, которое обрабатывается по описанной выше схеме.
Схема связей и взаимодействий приведена на рис. 1.

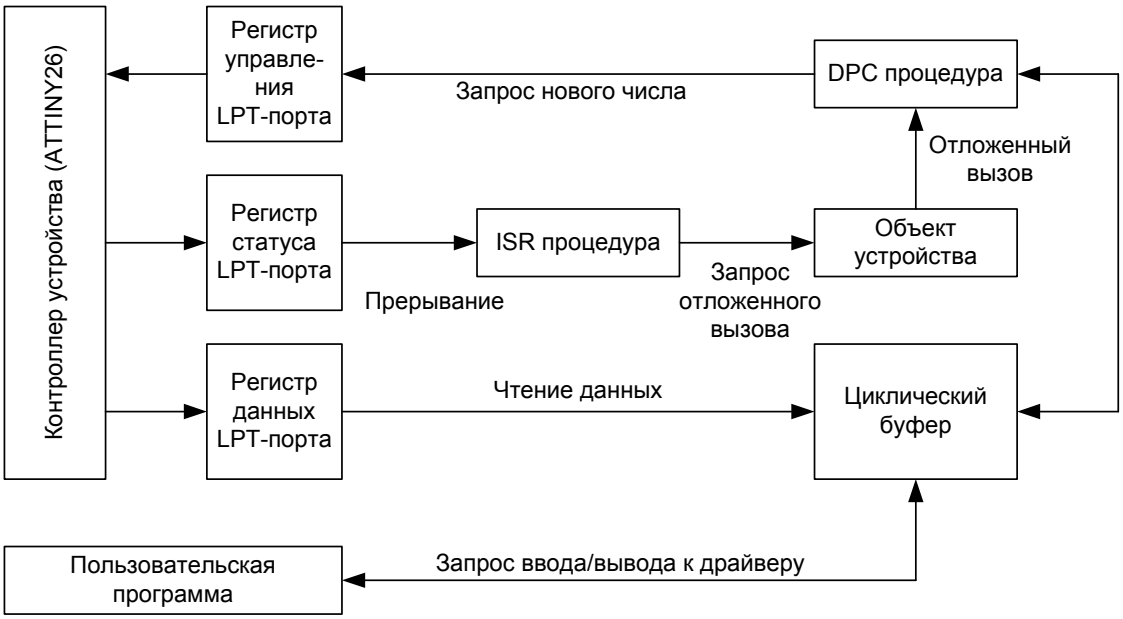

Рисунок 1.- Схема связей и взаимодействий

Циклический буфер используется для удобного одновременного чтения и записи в него, на случай, если в момент чтения программой придет прерывание. Таким образом, у буфера есть не только длина, а и его начало (хвост) – чтение происходит с хвоста, а запись с головы, которая после последнего смещения в буфере переходит в его начало (нулевое смещение) – аналогично тому, как сделан буфер клавиатуры в DOS.

Изначально у регистра данных LPT-порта предусматривалась запись со стороны компьютера и соответственно чтение со стороны внешнего устройства, поэтому здесь необходимо использовать двунаправленный режим передачи данных – устанавливается режим работы EPP в bios. Фото устройства приведены на рис. 2,3.

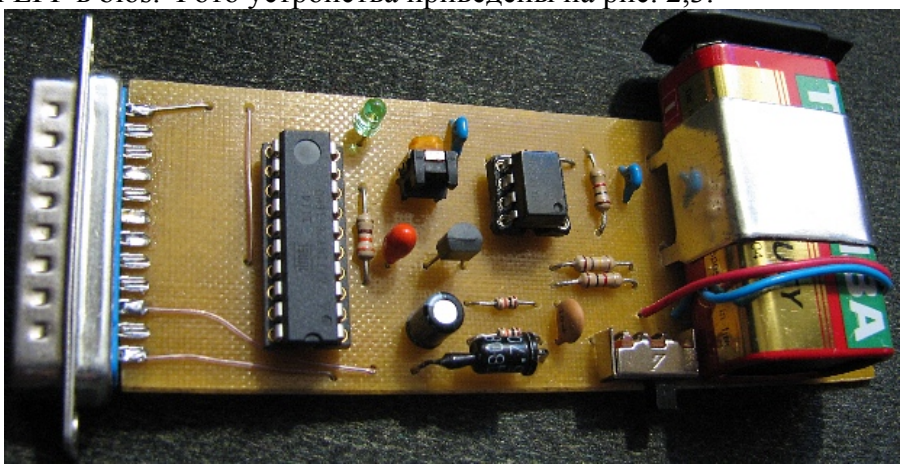

Рисунок 2. -Фото устройства (верх)

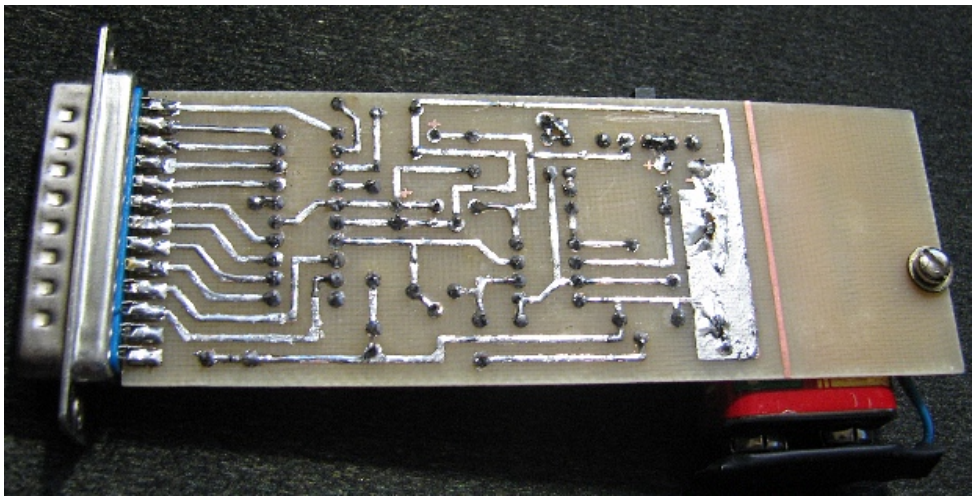

Рисунок 3 -. Фото устройства (низ)

Скриншоты программы тестирования устройства приведены на рис. 4-6.

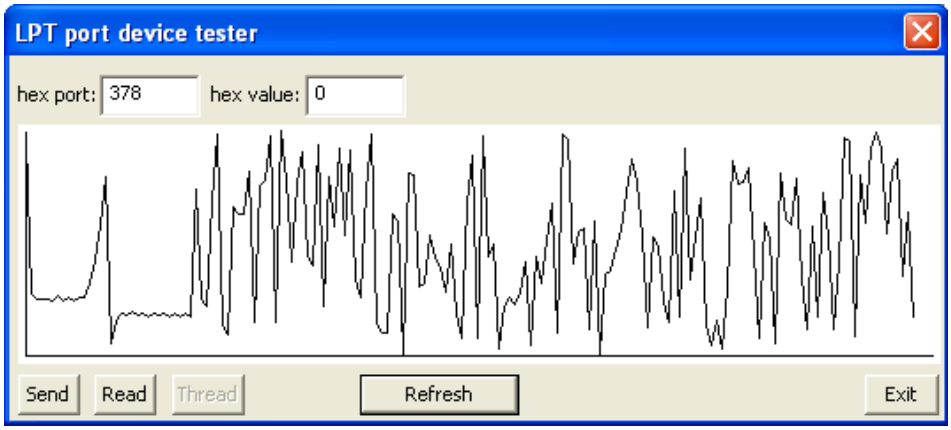

Рисунок 4 -. Тестирование устройства с включенным питанием (первые запросы)

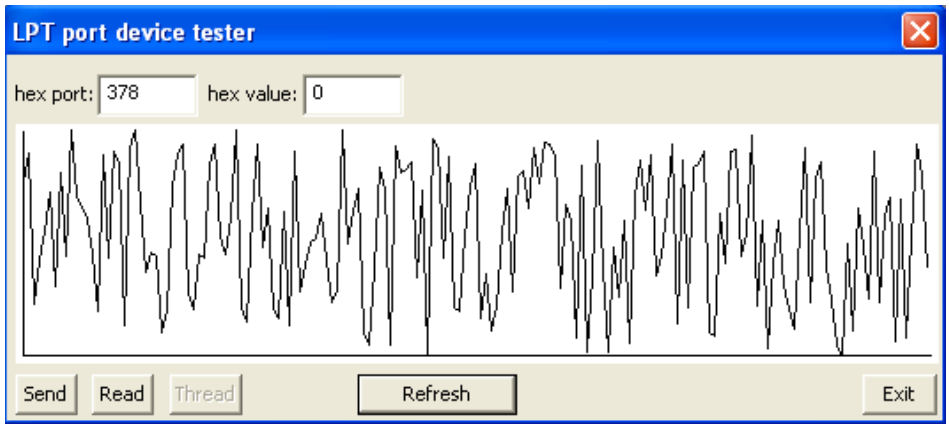

Рисунок 5 -. Тестирование устройства (последующие запросы)

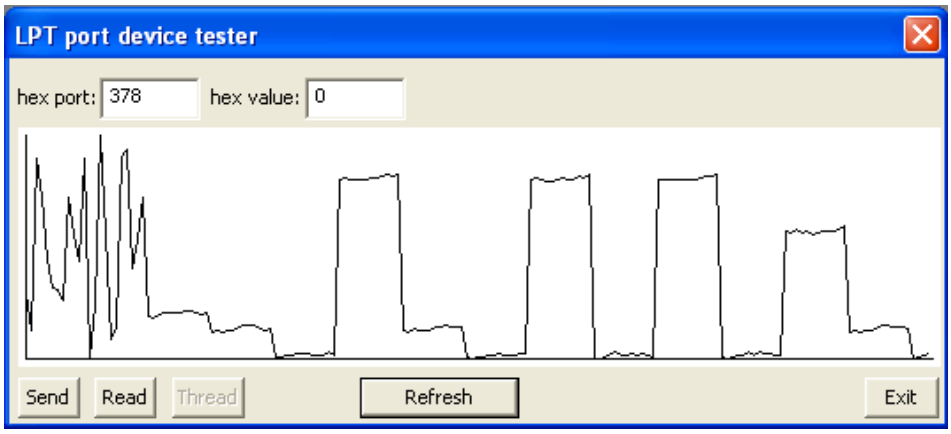

Рисунок 6 -. Тестирование устройства после выключения питания

Тестирующая программа выводит полученные случайные числа в виде графика, соединяя линиями точки, соответствующие значениям полученных чисел и отображенные с интервалом в три пикселя. При старте работы устройства всегда наблюдается некоторая инициализация (маленький разброс значений) продолжительностью около 10-15 чисел, что соответствует примерно одной миллисекунде после первого запроса к устройству. После отключения питания контроллер продолжает работать, получая +5в от LPT-порта, но остальная часть схемы уже не может работать корректно, АЦП выдает на выходе последовательность данных, похожих на прямоугольные импульсы. Судя из этого в некоторых случаях можно не подключать дополнительного питания к маломощным устройствам, в зависимости от схем. Промежуточные чтения из порта данных показали, что между запросами значение в нем не меняется, как и должно быть – буфер быстро пополняется, устройство ждет запросов от драйвера, драйвер ждет запросов от пользовательской программы.

Выводы

В данной статье были рассмотрены принципы обработки прерываний в среде драйверов Windows NT, был разработан, реализован, протестирован и описан работающий пример внешнего устройства, подключаемого через LPT-порт. Эксперименты показали, что Windows позволяет обслуживать внешние устройства с довольно высокой частотой, при этом, не перегружая себя, вследствие чего такой подход может быть использован для полноценного управления внешними объектами со своевременным реагированием на их запросы.

Список литературы

1.Литература

Рудаков П.И, Финогенов К.Г. Язык ассемблера: уроки программирования. – М.:Диалог-МИФИ, 2001. – 640 с.

2.Солдатов В.П. Программирование драйверов WINDOWS, Изд. 2-е, перераб. и доп. – М.: ООО «Бином-Пресс», 2004 г. – 480 с: ил.

3.Пишем первый драйвер. Интернет ресурс: http://www.pcports.ru/articles/ddk3.php

4.Цикл статей «Драйверы режима ядра». Интернет ресурс: http://wasm.ru/series.php?sid=9