Продолжение. Начало в № 6`2006

Современные программные средства связи микроконтроллера с компьютером по интерфейсу RS232

Алексей КУЗЬМИНОВ
compmicrosys@mail.ru

3.4. Программирование интерфейса RS232 с помощью прямых команд ввода/вывода в COM-порт

3.4.1. Общие положения

Программирование интерфейса RS232 в ОС Windows с использованием прямых команд ввода/вывода в COM-порт сопряжено с некоторыми проблемами, которые, к счастью, разрешимы.

Первая проблема — каким образом вообще ввести в порт и вывести из порта информацию. Дело в том, что в Кларионе для DOS (Clarion v.3.101), на котором написана программа, приведенная в 2.3.4, есть встроенные команды ввода/вывода в порт — in (№ порта, данные) и out (№ порта, данные). Эти команды и применялись для программирования интерфейса RS232 (COM-порта). В Кларионе для Windows (Clarion v.6.0) таких встроенных команд ввода/вывода в порт нет.

Вторая проблема касается разрешения применения команд прямого ввода/вывода — in( ) и out( ) — в порт. В ОС Win'98 эта проблема не стоит, поскольку в этой ОС прямые команды ввода/вывода в порт разрешены. В ОС Win'XP команды прямого ввода/вывода в порт запрещены (по крайней мере, в официальной документации).

Третья проблема также касается только ОС Win'XP и состоит в том, что, если даже каким-либо способом обойти запрещение на применение прямых команд ввода/вывода в порт, то для того, чтобы они работали с портом (в данном случае с COM-портом), этот порт необходимо открыть. Смысл слова «открыть» состоит в том, чтобы разрешить работу самого COM-порта (в том числе и для прямого ввода/вывода туда/оттуда информации). В Win'98 этого «открытия» делать не требуется, поскольку там прямые команды ввода/вывода в порт и так разрешены, и этот вопрос не стоит.

Забегая вперед, сразу заметим, что все эти три проблемы решаемы (и каждая из них может быть решена несколькими способами).

3.4.2. Применение команд ввода/вывода в порт в Clarion v.6.0 для Windows

Задача использования отсутствующих команд ввода/вывода в порт в Clarion v.6.0 может быть решена, по крайней мере, двумя способами.

Первый (назовем его способом А) заключается в том, что существует специально написанная и бесплатно распространяемая библиотека (inpout32.dll), которая решает сразу две задачи — дает в распоряжение пользователя две внешних библиотечных функции: INP (адрес порта) и OUTP (адрес порта, данные) и разрешает их применение в Win'XP (напомним, этого разрешения в Win'98 не требуется). Применение этих функций и позволяет производить ввод/вывод в порт, а заодно и разрешает этот ввод/вывод в Win'XP. Адрес порта должен быть переменной типа USHORT, а данные типа BYTE.

Для того чтобы вывести в порт информацию, достаточно применить функцию OUTP(USHORT,BYTE). Предположим, требуется вывести в порт данных (3f8h) порта COM1 символ 'A', имеющий код ASCII — 41h. Для этого достаточно написать:

```
OUTP(3f8h,41h)
```

Чтобы ввести данные из порта, достаточно применить функцию INP (USHORT). Предположим, требуется ввести данные из порта состояния COM1 (3feh). Для этого достаточно написать:

```
B=INP(3feh),
```

где переменная B должна быть типа BYTE.

Чтобы воспользоваться в программе вышеуказанными функциями, требуется сделать две вещи. Во-первых, преобразовать библиотеку inpout32.dll в inpout32.lib с помощью программы libmaker.exe, входящую в комплект поставки Clarion v.6.0 (о том, как это сделать, написано в руководстве пользователя по языку Clarion v.6.0), и уже библиотеку inpout32.lib вставить в проект своей программы. Во-вторых, в тексте программы необходимо указать прототипы использования команд INP(USHORT) и OUTP(USHORT,BYTE). Прототипирование осуществляется стандартным образом, как это обычно делается в Кларионе:

```
!-----------------------------------------
   MODULE('INPOUT32')
   INP(USHORT),BYTE,PASCAL,NAME('Inp32')
   OUTP(USHORT,BYTE),PASCAL,NAME('Out32')
   END
!-----------------------------------------
```

После этого можно использовать две вышеуказанные функции.

Использование библиотеки inpout32.dll для прямого ввода/вывода в порт имеет свои преимущества и недостатки.

Преимущества заключаются в достаточной простоте применения функций INP( ) и OUTP( ), а также в том, что применение этих функций автоматически разрешает их использование в Win'XP (другими словами, применение этих функций автоматически снимает запрет на прямой ввод/вывод в порт в Win'XP).

Недостатки использования функций INP( ) и OUT( ) состоят, во-первых, в том, что требуется привлекать дополнительную библиотеку (inpout32.dll), что сопряжено с дополнительными «манипуляциями» в программе и необходимостью иметь эту библиотеку. Во-вторых, эти функции выполняются не очень быстро (но, конечно, на порядок быстрее, чем функции API). Правда, быстродействия их хватает, чтобы отследить самые быстротекущие процессы в интерфейсе RS232.

Второй способ (Б) применения прямых команд ввода/вывода в порт заключается в следующем.

Хотя в самом языке Clarion v.6.0 и отсутствуют команды прямого вывода в порт (in и out), эти команды присутствуют в языке Си (C++), который встроен в Clarion v.6.0. Написав на этом языке две внешних подпрограммы ввода и вывода в порт и обратившись к ним из программы на Clarion v.6.0 как к внешней функции, можно решить поставленную задачу.

Ниже приведен текст двух внешних подпрограмм In.cpp и Out.cpp, написанных автором на встроенном C++, для прямого ввода и вывода в порт.

```
//---------------------------
//Подпрограмма In.cpp
//---------------------------
#pragma save
```

```
#pragma call(inline => on, reg_param => (dx), reg_return => (ax))
static unsigned char inportb(unsigned int Port_number)=
{
      0xEC, // in al,dx
};
#pragma restore

extern «C» unsigned char INP(unsigned int Port_number)
{
      unsigned char byte;
      byte=inportb(Port_number);
      return byte;
}
//-------------------------------

//-----------------------
// Подпрограмма Out.cpp
//---------------------------
#pragma save
#pragma call(inline => on, reg_param => (dx,ax))
static void outportb(unsigned int port, unsigned char byt)=
   {
   0xEE, // out dx,al
   };
#pragma restore

extern «C» void OUTP(unsigned int Port_Number, unsigned char byte)
   {
outportb(Port_Number,byte);
   }
//--------------------------
```

В чем заключается основная идея этих двух подпрограмм?

Вначале рассмотрим подпрограмму In.cpp.

Во встроенном в Clarion v.6.0 языке C++ есть возможность использования inline процедур, которые заключаются в том, что в текст программы (на C++) можно вставлять машинные коды команд процессора. Эти inline-процедуры возможно применять только в том случае, если функция, в которой они используются, имеет атрибут static. Но атрибут static не позволяет вызвать эту функцию из программы пользователя, другими словами — из языка Кларион (но позволяет осуществить ее вызов из программы на C++ из того же модуля). С другой стороны, программа, которая может быть вызвана из языка Кларион, должна быть «внешней» по отношению к нему; в этом случае внешняя подпрограмма (или функция) должна иметь атрибут extern, то есть «внешний». Поэтому подпрограмма In.cpp состоит из двух программ.

Первая — подпрограмма, имеющая атрибут static, осуществляет непосредственное использование inline-команды процессора (in al,dx), машинный код которой 0xEC. Эта команда процессора осуществляет ввод данных (команда in), расположенных по адресу, находящемуся в регистре dx процессора, в младший байт аккумулятора (al). Для осуществления такого действия необходимо: во-первых, запомнить текущее состояние процессора (что выполняет команда #pragma save), во-вторых, включить опцию inline, передать адрес порта (RS232) в регистр dx и получить результат ввода в регистре ax (что выполняет иструкция #pragma call(inline => on, reg_param => (dx), reg_return => (ax))). Далее следует процессорная команда непосредственного ввода из порта, имеющая шестнадцатиричный код 0xEC, что означает ввод из порта, адрес которого находится в регистре dx, а результат этого ввода помещается в регистр al. Последней инструкцией подпрограммы является инструкция #pragma restore, что означает восстановление первоначального состояния процессора.

Вторая — это подпрограмма, к которой уже можно обратиться из программы на Кларионе, и в связи с этим она имеет атрибут extern. Эта подпрограмма является как бы связующим звеном между первой подпрограммой (к которой она и обращается) и программой на Кларионе, откуда и идет обращение к ней.

Действие этой подпрограммы осуществляется таким образом. Когда в Кларионе идет обращение ко второй подпрограмме командой B=INP (адрес порта ввода), где B — переменная типа BYTE, а адрес порта типа USHORT (например, 3f8h), то есть пишется:

```
B=INP(3f8h),
```

то вторая подпрограмма помещает значение адреса 3f8h в переменную Port_number, обращается к первой подпрограмме с помощью инструкции byte=inportb(Port_number) и, получив значение введенного байта в переменной byte, возвращает это значение в Кларион с помощью инструкции return byte.

Столь подробное описание подпрограммы In.cpp приводится здесь потому, что, хотя внешне эта подпрограмма выглядит достаточно простой, она является ключевым решением проблемы прямого ввода из порта, который (ввод) в Кларионе для Windows (Clarion v.6.0) отсутствует.

Аналогично работает и подпрограмма Out.cpp. Разница состоит в том, что в подпрограмме In.cpp передается один параметр (адрес порта) и возвращается значение, прочитанное из этого порта, а в подпрограмме Out.cpp передаются два параметра (адрес порта — в регистр dx и значение, которое требуется вывести в этот порт, — в регистр ax). Поэтому такое подробное описание подпрограммы In.cpp приводится еще и для того, чтобы читатель смог проследить эту аналогию в подпрограмме Out.cpp.

Но это еще не все.

Чтобы эти подпрограммы работали, необходимо, во-первых, написать в языке Кларион для них прототипы, а во-вторых, включить их в проект программы на Кларионе.

Прототипирование подпрограмм должно осуществляться в программе на Кларионе следующими инструкциями:

```
MODULE('IN.CPP')
   INP(USHORT),BYTE,NAME('_INP')
END
MODULE('OUT.CPP')
   OUTP(USHORT,BYTE),NAME('_OUTP')
END
```

Здесь также есть одна тонкость. Символ подчеркивания («_») перед названиями INP и OUTP отражает тот факт, что эти подпрограммы написаны именно на языке Си. Если посмотреть на подпрограммы In.cpp и Out.cpp, то можно заметить, что в них используется инструкция extern «C». Символ «C» и есть причина символа подчеркивания. Без символа подчеркивания в прототипе и символа «C» в подпрограмме вся конструкция работать не будет.

Для того чтобы вставить подпрограммы In.cpp и Out.cpp в проект, необходимо открыть в проекте папку External source files (внешние файлы), выбрать опцию Add (добавить) и указать подпрограмму, например, In.cpp. Забегая немного вперед, отметим, что для нижеприведенной тестовой программы Hello.clw, использующей прямые команды ввода/вывода в порты, будет приведен и текст файла-проекта (Hello.prj), где все это уже сделано.

А сейчас возвратимся ко второй проблеме использования команд прямого ввода/вывода через порты в Win'XP, а именно — к тому, как обойти запрещение применения таких команд.

3.4.3. Снятие запрета на применение команд ввода/вывода в порт для Win'XP

Как уже было упомянуто, одним из способов, разрешающих применение прямых команд ввода/вывода в порт, является использование библиотеки inpout32.dll. Но в этой библиотеке уже приведены подпрограммы ввода/вывода в порты — INP( ) и OUTP( ), поэтому надобность в подпрограммах In.cpp и Out.cpp отпадает. В связи с этим использование библиотеки inpout32.dll становится бессмысленным, если предполагается работа с подпрограммами In.cpp и Out.cpp.

«Покопавшись» в Интернете, можно обнаружить сразу несколько программ, целью которых является разрешить использование прямых команд ввода/вывода в порты в Win'XP. Перепробовав некоторые из них, автор обнаружил, что реально работают только две программы. Это программа UserPort, написанная Томасом Франзоном (Tomas Franzon), которую очень просто найти любым поисковиком, и программа Portmon, написанная Марком Руссиновичем (Mark Russinovich) [18].

Ниже будут приведены примеры применения этих программ.

Первая — программа UserPort. Эта бесплатно распространяемая программа довольно остроумно обходит запрет на использование прямых команд ввода/вывода в порты в Win'XP. Идея программы — в использовании недокументированных команд Win'XP в привилегированном режиме. Подробное описание работы программы UserPort приводится в файле UserPort.pdf. Программа UserPort работает с драйвером UserPort.sys, который входит в комплект ее поставки. Чтобы разрешить ввод/вывод в порты, драйвер UserPort.sys необходимо скопировать в папку C:\WINDOWS\SYSTEM32\DRIVERS\ и перезагрузить компьютер. После такой манипуляции уже возможно использование вышеописанных подпрограмм In.cpp и Out.cpp.

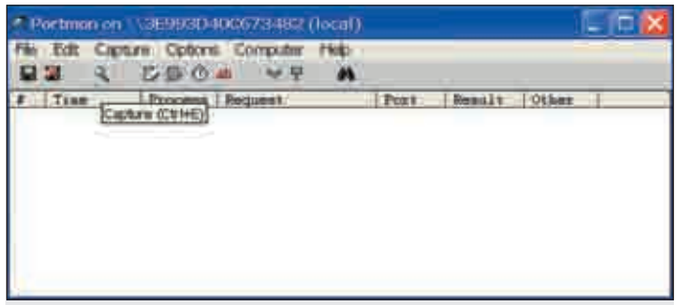

Рис. 10. Исходный вид окна программы Portmon после запуска

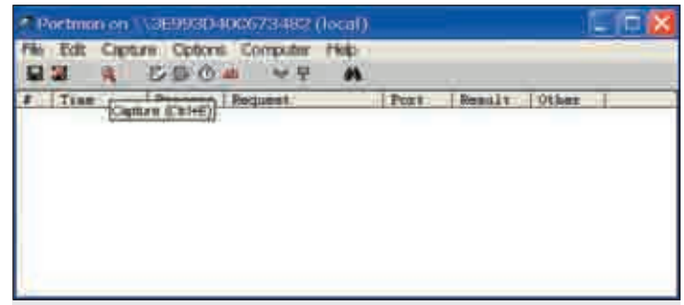

Рис. 11. Выключение опции Capture (захват)

Необходимо, однако, заметить, что использование драйвера UserPort.sys для работы подпрограмм In.cpp и Out.cpp с COM-портом компьютера является условием необходимым, но недостаточным. Как было упомянуто ранее, требуется еще «открыть» COM-порт. О способах открытия COM-порта будет рассказано несколько позже. Пока же возвратимся к программам, разрешающим использование прямых команд ввода/вывода в порты в Win'XP.

Вторая программа, которую хотелось бы представить, — Portmon («Порт-монитор»). Ее предназначение — решение более широкого круга задач, чем просто разрешение на ввод/вывод в порты. Она, например, позволяет «отслеживать» обращение к портам из той или иной программы (при выборе опции Capture — захват) и выполнять некоторые другие функции. Применительно к использованию подпрограмм In.cpp и Out.cpp для ввода/вывода в порты из языка Clarion v.6.0 программа Portmon выполняет сразу две задачи:

1) Разрешает применять прямой ввод/вывод в порты.
2) Открывает порт (в нашем случае COM-порт) для работы.

Для использования этой программы необходимо выполнить следующее.

После инсталляции и запуска программы на экран выведется окно, показанное на рис. 10. Необходимо отключить опцию Capture (захват) (рис. 11), иначе каждая попытка ввода или вывода в порт (COM-порт) будет «перехвачена» и выведена на экран, что резко «затормозит» такой ввод/вывод. Второе, что необходимо сделать, — это установить («открыть») порты, к которым будет осуществляться обращение. В нашем случае это COM1 (и, например, COM2). Для этого необходимо выбрать опцию Capture, в открывшемся подменю выбрать опцию Ports (порты) и в еще одном открывшемся подменю отметить порты COM1: Serial0 и COM2: Serial1 (рис. 12). Далее необходимо выйти из программы Portmon. После таких манипуляций можно уже производить прямой ввод/вывод в порты COM1 и COM2 не только из программы на Clarion v.6.0 (то есть использовать подпрограммы In.cpp и Out.cpp), но и вообще использовать прямые команды ввода/вывода в порты COM1 и COM2 в любой другой программе (в частности, даже в DOS-программах, например, в ТурбоБэйсике tb.exe, Кларионе для DOS — Clarion v3.101 и т. п). Можно, например, запустить программу Hello.exe, написанную на языке Кларион для DOS — Clarion v.3.101 (Hello.cla) и приведенную в п. 2.3.4.

Но, чтобы в DOS-программах правильно отражались символы русского языка на экране, в конец файла AUTOEXEC.NT необходимо вставить какой-либо DOS-драйвер монитора, например,

```
tdrvses.exe,
```

а в файл CONFIG.NT — следующие команды:

```
dos=high, umb
device=%SystemRoot%\system32\himem.sys
files=40
```

Необходимо отметить, что файлы AUTOEXEC.NT и CONFIN.NT находятся в папке C:\WINDOWS\SYSTEM32\.

Если же использование программ для DOS не предполагается, то файлы CONFIG.NT и AUTOEXEC.NT лучше не трогать.

Вернемся к Win'XP, прямым командам ввода/вывода в COM-порт и сравнению двух программ UserPort и Portmon на предмет их преимуществ и недостатков.

Вначале о программе Portmon. Эту программу рекомендуется использовать только для разработки и проверки работоспособности новых программ, поскольку после запуска Win'XP каждый раз необходимо производить манипуляции, показанные на рис. 10–12. Это легко может сделать программист, но ни в коем случае не пользователь программы, «заставлять» которого производить подобные манипуляции просто недопустимо. Пользователь должен запустить программу, написанную для его целей, и программа должна работать с портами ввода/вывода. Другими словами, работа с портами ввода/вывода должна быть осуществлена автоматически после запуска программы пользователя.

Как было упомянуто ранее, программа UserPort (а точнее — драйвер UserPort.sys, установленный в папку C:\WINDOWS\SYSTEM32\DRIVERS\) «автоматически» разрешает использование прямых команд ввода/вывода в порт после запуска Win'XP. Программа же Portmon.exe не обладает такой возможностью. Другими словами, «заставить» программу Portmon автоматически произвести манипуляции, показанные на рис. 10–12, нельзя.

Поэтому для написания пользовательской программы, в которой предполагается применение прямых команд ввода/вывода в порт, допустимо только применение драйвера UserPort.sys, который требуется установить только один раз.

Напомним, что установка драйвера UserPort.sys является необходимым условием работы с портами ввода/вывода, но недостаточным. Требуется еще «открыть» COM-порт для работы. Каким образом это сделать? Об этом будет рассказано далее.

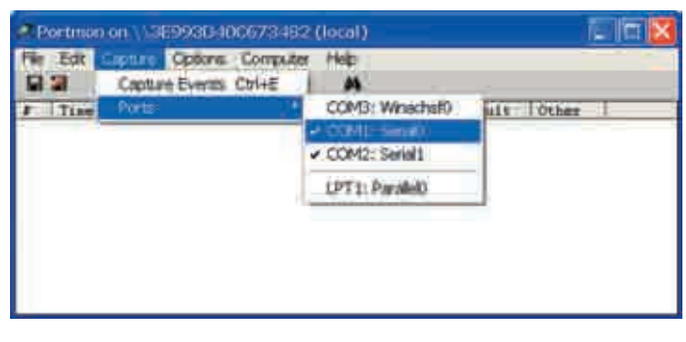

Рис. 12. Открытие портов COM1 и COM2 для работы

3.4.4. Открытие COM-порта для использования прямых команд ввода/вывода в Win'XP

Существует два способа «открытия» COM-порта. Рассмотрим их.

Первый способ — уже небезызвестные API-функции (которые, по мнению автора, только на это и годятся). Прототипы API-функций, которые используются для работы с COM-портом, были уже приведены в 3.3.1 (в рассмотренной там программе Hello.clw). Из всех API-функций нам потребуются только две: для «открытия» COM-порта и его «закрытия». Первый прототип — CreateFileA() — предназначен для открытия порта, а второй — CloseHandle() — для его закрытия. Ниже приведены эти два прототипа.

```
  MODULE('Comm Prototipes')
!-----------------------------------------
    CreateFileA(*CSTRING,ULONG,ULONG,LONG,ULONG,
                ULONG,UNSIGNED),|
    SIGNED,RAW,PASCAL,NAME('CreateFileA')
    CloseHandle(SIGNED),BOOL,PROC,PASCAL
!-----------------------------------------
  END
```

Для использования этих API-функций требуется определить несколько переменных:

```
!-----------------------------------------
!Переменные для API-COM — 1-й способ открытия порта.
!-----------------------------------------
CommPort        UNSIGNED     !(handle).
PortCString     CSTRING(10)  !'COM1/COM2'
OPEN_EXISTING   ULONG(3)
!-----------------------------------------
```

Открыть COM-порт можно следующими командами:

```
  PortCString='COM1'
  CommPort=CreateFileA(PortCString,0,0,0,OPEN_EXISTING,0,0)
```

Как можно увидеть из этих команд, переменной PortCString присваивается значение 'COM1', а API-функция CreateFileA() используется с нулевыми параметрами. Другими словами, параметры COM-порта (скорость обмена, бит паритета, количество бит данных и количество стоп-бит) не определены. Это определение можно будет сделать позже уже прямыми командами ввода/вывода в COM-порт.

Для закрытия COM-порта должна использоваться API-функция CloseHandle():

```
loop until(CloseHandle(CommPort)).    !Закрытие CommPort'a.
```

Вторым способом, которым можно осуществить открытие COM-порта, является использование бесплатно распространяемой библиотеки clacom32.dll, с помощью которой приводятся примеры применения программ работы с COM-портом в Windows, разработанных компанией Gap Development Company [17] под общим названием ClaCom (их можно приобрести за определенную плату).

Чтобы использовать библиотеку clacom32.dll для открытия COM-порта, требуется осуществить следующие действия.

Во-первых, необходимо преобразовать библиотеку clacom32.dll в clacom32.lib с помощью входящей в состав Crarion v.6.0 программы libmaker.exe и включить ее в проект программы на Clarion v.6.0.

Во-вторых, определить прототипы использования команд открытия и закрытия COM-порта:

```
MODULE('CLACOM')
SetPort(SHORT),PASCAL,NAME('_SetPort@8')
ResetPort(SHORT),PASCAL,NAME('_ResetPort@8')
END
```

и переменную для номера COM-порта:

```
Comm                    short
```

Для открытия COM-порта требуется использовать следующие команды:

```
ComNum=0 ! 0-COM1,1-COM2
SetPort(ComNum)
ResetPort(ComNum)
```

Как видно из последних команд, для того чтобы открыть порт COM1, переменной ComNum необходимо присвоить нулевое значение, для открытия COM2 — единичное.

Открытие порта осуществляется командой SetPort(ComNum), приведение его в рабочее состояние — командой ResetPort(ComNum). Эти две команды необходимо использовать в программе на Clarion v.6.0 только один раз — в начале программы.

3.4.5. Тестовая программа, использующая прямые команды ввода/вывода в COM-порт в ОС Win'98/XP

Подытожим вкратце все вышеизложенное в пп. 3.4.2, 3.4.3 и 3.4.4.

Для использования в Clarion v.6.0 прямых команд ввода/вывода в порт в ОС Win'98/XP необходимо осуществить следующие три действия.

1. Первый вариант — написать программы для прямого ввода/вывода в порт на встроенном в Clarion v.6.0 языке C++(In.cpp и Out.cpp), написать для них прототипы и включить их в файл-проект (*.prj).

   Второй вариант — не писать никаких программ ввода/вывода, а использовать готовую библиотеку inpout32.dll, преобразовать ее в библиотеку inpout32.lib с помощью программы libmaker.exe и включить ее в файл-проект (*.prj). Кроме того, написать прототипы для использования команд INP( ) и OUTP( ).
2. Разрешить использование прямых команд ввода/вывода в порт, установив драйвер UserPort.sys в папку C:\WINDOWS\SYSTEM32\DRIVERS\.

   При использовании библиотеки inpout32.dll в применении драйвера UserPort.sys нет необходимости.
3. «Открыть» COM-порт для работы.

Первый вариант — с помощью API-функций, второй вариант — с помощью библиотеки clacom32.dll.

Ниже приведен текст тестовой программы Hello.clw, в котором показаны все варианты. Неиспользуемые варианты закомментированы (перед ними установлен восклицательный знак «!»). В файл-проект, который приведен вслед за текстом программы Hello.clw, включены все варианты. В самой же программе используются подпрограммы In.cpp и Out.cpp, драйвер UserPort.sys; открытие COM-порта сделано с помощью библиотеки clacom32.dll. Отметим, что для того, чтобы получить «картинку» окна, вид «кнопок» и другие атрибуты ОС Win'XP, в файл-проект необходимо дополнительно включить файл WindowsShell.Manifest.

В программе установлены следующие параметры COM-порта: COM1:N,8,1,115200. Программа использует алгоритм аппаратной синхронизации, о котором уже не раз говорилось.

Программа Hello.clw:

```
!-------------------------------------------------------------
! Программа передачи и приема строки из 75 символов по RS232
!-------------------------------------------------------------

      PROGRAM

  INCLUDE('Equates.CLW')
  INCLUDE('TplEqu.CLW')
  INCLUDE('Keycodes.CLW')
  INCLUDE('Errors.CLW')
!---------------------------------
! Определение переменных
!---------------------------------
C                       byte
A                       byte
B                       byte
S                       string(80)
BT                      byte
M                       byte,dim(80),over(S)
S1                      string(80)
M1                      byte,dim(80),over(S1)
i                       ushort
j                       ushort
k                       ushort
CH                      byte
N1                      long
STR                     string(5)
p                       byte
!-----------------------------------------
! Переменные для QueryPerformanceCounter
!-----------------------------------------
CSTARTL                 ulong
CSTARTH                 long
CSTOPL                  ulong
CSTOPH                  long
DC                      long
DELTAC                  long
!-----------------------------------------

!-----------------------------------------------
!Переменные для API-COM — 1-й способ открытия порта.
!-----------------------------------------------
CommPort        UNSIGNED     !(handle).
PortCString     CSTRING(10)  !'COM1/COM2'
OPEN_EXISTING   ULONG(3)
!-----------------------------------------------

!-----------------------------------------------
!Переменная для CLACOM — 2-й способ открытия порта.
!-----------------------------------------------
ComNum                  short
!-----------------------------------------------

!-----------------------------------------------
! Переменные (совмещения) для частоты и счетчика
```

```
!-----------------------------------------
FREQUENCY GROUP,PRE(FRE)
FREQL             ULONG
FREQH             LONG
.
COUNTER           GROUP,PRE(CNT)
COUNTL            ULONG
COUNTH            LONG
.
!-----------

                  MAP

!######################
!-----------------------------------------
! Способ а) прямого ввода/вывода в порт
! с использованием драйвера USERPORT, но без
! использования библиотеки inpout32.dll.
!-----------------------------------------
   MODULE('IN.CPP')
      INP(ULONG),BYTE,NAME('_INP')
   END
   MODULE('OUT.CPP')
      OUTP(unsigned long,BYTE),NAME('_OUTP')
   END
!-----------------------------------------

!-----------------------------------------
! Способ б) прямого ввода/вывода в порт без
! использования дравера USERPORT, но с
! использованием библиотеки inpout32.dll.
!-----------------------------------------
!     MODULE('INPOUT32')
!     INP(USHORT),BYTE,PASCAL,NAME('Inp32')
!     OUTP(USHORT,BYTE),PASCAL,NAME('Out32')
!     END
!-----------------------------------------

!=========================================
! Прототипы API-функций — для WIN'XP.
!-----------------------------------------
   MODULE('Comm Prototipes')
!-----------------------------------------
      CreateFileA(*CSTRING,ULONG,ULONG,LONG,ULONG,
                  ULONG,UNSIGNED),|
      SIGNED,RAW,PASCAL,NAME('CreateFileA')
      CloseHandle(SIGNED),BOOL,PROC,PASCAL
!-----------------------------------------
   END
!-----------------------------------------
   MODULE('DELAY')
   QueryPerformanceCounter(*STRING),BOOL,RAW,PASCAL
   QueryPerformanceFrequency(*STRING),BOOL,RAW,PASCAL
   Sleep(ULONG),PASCAL
   END
!=========================================
   MODULE('CLACOM')
   SetPort(SHORT),PASCAL,NAME('_SetPort@8')
   ResetPort(SHORT),PASCAL,NAME('_ResetPort@8')
   END
!-----------------------------------------
!######################

                  END ! от MAP


Window WINDOW('Тест RS232 (COM1:115200,N,8,2).
                  Пр<255>мой ввод/вывод в порт.'),|
      AT(,,365,172),|
      FONT('MS Sans Serif',8,,FONT:bold,CHARSET:CYRILLIC),|
      COLOR(COLOR:BTNFACE),CENTER,SYSTEM,GRAY,MAX
      BUTTON('Выход'),AT(36,132,52,21),USE(?Cancel),|
      FONT('Tahoma',10,COLOR:Black,FONT:bold,CHARSET:
            CYRILLIC)
      BUTTON('Продолжить'),AT(244,132,73,21),USE(?Ok),|
      FONT('Tahoma',10,COLOR:Black,FONT:bold,CHARSET:
            CYRILLIC),DEFAULT
   END

!----------------------

                  CODE

!-----------------------------------------------------
!Открытие COM-порта(API) — 1-й способ открытия для WIN'XP.
!Этот способ требует закрытия COM-порта, но не требует
!дополнительной библиотеки.
!-----------------------------------------------------
! PortCString='COM1'
! CommPort=CreateFileA(PortCString,0,0,0,OPEN_EXISTING,0,0)
! Sleep(100)
!-----------------------------------------------------

!-----------------------------------------------------
! Открытие COM-порта (CLACOM) — 2-й способ открытия для
WIN'XP.
! Этот способ не требует закрытия порта, но требует
! дополнительную библиотеку clacom32.dll.
!-----------------------------------------------------
      ComNum=0 ! 0-COM1,1-COM2
      SetPort(ComNum)
      ResetPort(ComNum)
!-----------------------------------------------------


!-----------------------------------------------------
! Вычисление временной задержки в 25 мкс.
!-----------------------------------------------------
! Определение частоты работы счетчика.
!-----------------------------------
      loop until QueryPerformanceFrequency(FREQUENCY).
!-----------------------------------
!Вычисление времени счета в 20 мкс.
!-----------------------------------
      DC=int(20*FRE:FREQL/1000000)
!-----------------------------------------------------

      do INIT                    !Инициализация RS232

      k=0
      OPEN(Window)

START
      Blank

      disable(?Ok)
      disable(?Cancel)


   k=k+1

S='ABCDEFGHIJKLMNOPQRSTUVWXYZabcdifjhigklmnopqrstuvw
xyzАБВГДЕЖЗИКЛМНОПРСТУФХЦЧ'
   show(5,40,S)
   ! Ч-75-й символ

   show(5,30,'Начало передачи из PC')
   show(5,10,'k=')
   type(k)

   i=1
   C=M[1]
   M[1]=75
   do OUTBYTE !Передача мл. байта длины (75)
   M[1]=0
   do OUTBYTE !Передача ст. байта длины (0)

   M[1]=C
   loop i=1 to 75
   do OUTBYTE
   .

   show(5,50,'Конец передачи из PC')
   show(5,60,'_____________________________________________')

!-----------------------------------------------------------
! В этом месте идет переключение с вывода на ввод.
!-----------------------------------------
   B=INP(3f8h)   !Холостой ввод — ОБЯЗАТЕЛЬНО!
                 !(Для сброса бита DR-data ready).

   loop while band(INP(3fdh),1). !Ожидание сброса бита «DR»-data
                                 !ready.
   A=INP(3feh)   !Очистка 3feh для правильного ож-я
                 !старт-бита.
!-----------------------------------------------------------

!-----------------
! Sleep(5)
!-----------------

   show(5,80,'Начало передачи из микроконтроллера')

   loop i=1 to 75
   do INBYTE
   .

   show(5,90,S1)
   show(5,100,'Конец передачи из микроконтроллера')
   show(5,110,'_____________________________________________')

      p=0
      enable(?Ok)
      enable(?Cancel)

      accept
!---------------------
      case accepted()
         of ?Ok
            p=1
            break
         of ?Cancel
            p=0
            break
      .! ot case
!-------------------------
      .! от accept
      if p=0 then goto E.
            i=1
            M[1]=40h
            do OUTBYTE
      goto START

E  OUTP(3fch,00h) !Сброс микроконтроллера
   OUTP(3fbh,7h)

!-----------------------------------------------------------
! Закрытие COM-Порта (API) — 1-й способ. Для WIN'XP
!-----------------------------------------------------------
! loop until(CloseHandle(CommPort)).   !Закрытие CommPort'a.
!-----------------------------------------------------------

                  Return

!---------------------------------------
! Подпрограммы
!---------------------------------------
! П/п инициализации RS232 и микроконтроллера.
!---------------------------------------
INIT              routine
!-------------------------------
! Инициализация COM-порта.
!-------------------------------
! Установка скорости обмена.
!-----------------------
   OUTP(3fbh,80h)  ! DLAB=1 для установки делителя.
   OUTP(3f8h,1)    ! Установить мл.б. скор.:1-115200 бод,
                   ! 12-9600 бод.
   OUTP(3f9h,00h)  ! Установить ст.б. скор.=0.
!-----------------------------------------------------------
! Установка режима.
!------------------
   OUTP(3fbh,07h)  ! DLAB=0,Режим: 8 бит данных,2 стопа,
                   ! нет пар.,сброс TxD.
!-----------------------------------------------------------
! Установка запрета прерываний по COM-порту.
!-------------------------------------------
   OUTP(3f9h,00h)  ! Запрет всех прерываний по порту 3f8h.
!-----------------------------------------------------

!-----------------------------------
! Инициализация микроконтроллера.
!-----------------------------------
   OUTP(3fch,00h)  ! Сброс линии DTR (Reset микроконтроллера).
   Sleep(100)      ! Задержка 0.1 сек.
   OUTP(3fch,01h)  ! Установка линии DTR
                   ! (Запуск микроконтроллера).
   Sleep(100)      ! Задержка 0.1 сек.
   B=INP(3f8h)     ! Холостой ввод — для сброса бита 0(DR)
                   ! в 3fdh (в «0»).
   Sleep(20)

!-----------------------------------
! П/п ввода байта (байт в M1[i])
!-----------------------------------
INBYTE            routine
   OUTP(3fbh,47h)                 !Установка TxD(в +9 Вольт)
                                  !(разрешение передачи).
   loop until band(INP(3feh),2).  !Ожидание старт-бита
                                  !(изменения состояния DSR).
   do DEL20                       !Задержка 25 микросекунд.
   OUTP(3fbh,07h)                 !Сброс линии TxD(в -9 Вольт)
                                  !(запрет передачи).
   A=INP(3feh)                    !Очистка 3feh для правильного
                                  !ожидания старт-бита.
   loop until band(INP(3fdh),1).  !Ожидание прихода байта(уст-ки
                                  !бита «DR»-data ready).
   M1[i]=INP(3f8h)                !Чтение байта данных.
   A=INP(3feh)                    !Очистка 3feh для правильного
                                  !ожидания старт-бита.

!---------------------------------------
! П/п вывода байта (байт в M[i])
!---------------------------------------
OUTBYTE           routine
   loop until band(INP(3fdh),20h).  !Ожидание готовности передат-
                                    ! чика (transmitter empty).
   loop until band(INP(3feh),20h).  !Ожидание разрешения переда-
                                    ! чи (установки DSR).
   OUTP(3f8h,M[i])                  !Вывод байта.
                                    !B=INP(3feh)
   loop until band(INP(3feh),2h).   !Ожидание сброса разрешения
                                    !(изменения состояния DSR).
   loop until band(INP(3fdh),40h).  !Ожидание выхода байта из PC
                                    !(OK to send).
   B=INP(3feh)                      !Очистка 3feh для правильного
                                    !ожидания старт-бита.
!---------------------------------------

!-------------------------
DEL20 routine
   loop until QueryPerformanceCounter(COUNTER).   !Чтение
                                                  !счетчика.
   CSTARTL=CNT:COUNTL      !Занесение начального значения в
```

```
   CSTARTH=CNT:COUNTH    !две стартовых long-переменных.
   Loop                  !
     loop until QueryPerformanceCounter(COUNTER). ! Ожидание
     if CNT:COUNTH=CSTARTH                        !
       DELTAC=CNT:COUNTL-CSTARTL                  ! прошествия
     Else                                         !
       DELTAC=0ffffffffffffffffh-CSTARTL+CNT:COUNTL+1
                                                  ! времени
     .                                            !
     if DELTAC>DC then break.                     !DC
                                                  !(=25 мкс.)
   .                                              !
!-----------------------------------------------------------------------
```

Текст файла-проекта hello.prj:

```
-------------------------------------------------------------
-- Project File Title
#noedit
#system win32
#model clarion lib
#set RELEASE = on
#pragma debug(vid=>off)
#pragma optimize(cpu=>386)
#pragma define(NULL=>on)
#compile «Hello.clw»
#compile «In.cpp»
#compile «Out.cpp»
#pragma link(«hello.lib»)
#pragma link(«inpout32.lib»)
#pragma link(«clacom32.lib»)
#pragma link(«WindowsShell.Manifest»)
#link «hello.exe»
```

Сравнив подпрограммы инициализации COM-порта, инициализации микроконтроллера, ввода и вывода байта приведенной выше программы с программой Hello.cla, написанной на языке Кларион для DOS (Clarion v.3.101) и приведенной в п. 2.3.4, можно отметить практически полную их идентичность. Если же сравнить текст приведенной программы с текстом программы, использующей API-функции и приведенной в п. 3.3.1, то можно увидеть, насколько неуклюжи и громоздки API-функции. Кроме того, сравнив подпрограммы для ввода и вывода байта этих двух программ, можно заметить, что подпрограммы ввода и вывода байта в программе, использующей API-функции, выполняют не все действия, которые выполняют эти подпрограммы в программе, использующей прямой ввод/вывод в порты. Причиной этого является «неповоротливость» API-функций, не позволяющая отслеживать быстротекущие процессы в интерфейсе RS232.

Программу следует запускать совместно с подключенной к COM-порту какой-либо из макетных плат с микроконтроллером, описанных в главе по аппаратным средствам. Перед запуском программы Hello.exe кабель от макетной платы должен быть соединен с портом COM1, а питание макетной платы должно быть включено.

После запуска программы Hello.exe в ОС Win'98 на экране монитора появится окно, показанное на рис. 13. Нажатие кнопки «Продолжить» приведет к повторному запуску программы (количество нажатий на кнопку или количество запусков k = 176), нажатие кнопки «Выход» — к выходу из программы в Windows.

Общий вид окна программы при ее запуске в ОС Win'XP показан на рис. 14.

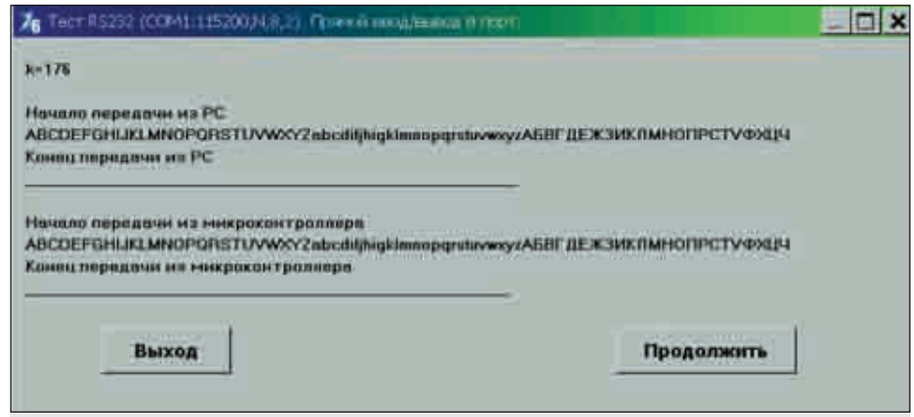

Рис. 13. Общий вид окна программы Hello.exe в ОС Win'98

Рис. 14. Общий вид окна программы Hello.exe в ОС Win'XP

4. Вместо заключения

Приведенные в статьях аппаратные и программные средства связи микроконтроллера с компьютером по интерфейсу RS232 — отнюдь не чисто научные изыскания. Эти средства имеют вполне «земное» применение — они используются автором для конструирования и производства автоматизированных систем сбора информации, поступающей с различного рода датчиков с аналоговым выходом (давления, температуры, влажности) и числоимпульсным выходом (например, количество импульсов, поступающее с различного рода — турбинных, ротационных, электромагнитных, вихревых и т. п. — счетчиков объема газа и воды).

Каков предел применения интерфейса RS232 в системах, которые используют микроконтроллеры в качестве удаленных от компьютера устройств, обменивающихся с компьютером информацией? Опыт конструирования подобного типа систем показывает, что интерфейс RS232 целесообразно использовать в таких системах сбора и обработки информации, в которых количество датчиков не превышает 20–30, а частота обновления информации, получаемой с них, не превышает 20 Гц, то есть дискретность по времени (Δt) не менее 0,05 с. На практике встречается масса задач, где требуются достаточно прецизионные измерения низкочастотных сигналов, а дискретность обновления данных намного больше, чем 0,05 с (1 с и более). В качестве примеров далее приведены технические характеристики, общий вид некоторых окон, открывающихся в программах на компьютере, и фотографии систем сбора, работающих на реальных объектах и применяющихся как автоматизированные средства поверки и градуировки счетчиков объема газа.

Целесообразность применения интерфейса RS232 в таких устройствах доказана многолетним «стажем» их работы (все подобные системы сбора, сконструированные автором и запущенные в эксплуатацию, работают до сих пор, а некоторые из них — более 10 лет). Косвенным подтверждением этой целесообразности может служить еще и тот факт, что такие ведущие фирмы-производители аналоговых микросхем — АЦП и ЦАП, как Analog Devices, Texas Instruments, Philips, Atmel, Silicon Laboratories, с недавнего времени стали выпускать микроконтроллеры «системы на кристалле», которые используют интерфейс RS232 не только в качестве обычной связи с компьютером. Эти микроконтроллеры (как уже обсуждалось ранее) имеют возможность «программирования в системе» и именно по интерфейсу RS232.

Применяя новый алгоритм обмена по RS232 (см. п. 2.1), можно получить высокие

скоростные и надежностные характеристики связи компьютера с микроконтроллером уже в Win'98/XP. Это позволяет поднять качество программ для компьютерных систем сбора и обработки информации на новый современный уровень.

В заключительной части стьтьи будут рассмотрены компьютерные системы сбора и обработки информации, поступающей с датчиков аналоговых, частотных и дискретных сигналов, на базе 51-совместимых однокристальных микроконтроллеров.

Литература

1. Баррингтон Б. Б. Как создавался Кларион // Мир ПК. 1993. № 2.
2. Кузьминов А. Ю. Интерфейс RS232. Связь между компьютером и микроконтроллером. От DOS к Windows98/XP М.: ДМК-ПРЕСС. 2006.
3. Кузьминов А. Ю. Интерфейс RS232. Связь между компьютером и микроконтроллером. М.: Радио и связь. 2004.
4. Кузьминов А. Ю. Однокристальные микроЭВМ — основа удаленных систем сбора и обработки сигналов, поступающих с датчиков // Электроника и компоненты. 1998. № 2.
5. Кузьминов А. Ю. Новые MCS51 — совместимые микроконтроллеры и их применение в системах сбора информации с датчиков // Контрольно-измерительные приборы и системы. 1997. № 6. 1998. № 7.
6. Кузьминов А. Ю. Удаленные системы сбора информации с датчиков на базе однокристальных микроЭВМ // Автоматизация и производство. 1996. № 3.
7. Кузьминов А. Ю. Универсальная система сбора и обработки данных АСИР-3 // Мир ПК. 1996. № 6.
8. Орлов А. Два звучных слова — Clarion и Delphi // Мир ПК. 1996. № 6
9. Фролов А. В., Фролов Г. В. Программирование модемов. М.: ДИАЛОГ-МИФИ. 1993.
10. **w ww.analog.c om**
11. **w ww.atmel.c om**
12. **w ww.maxim-ic.c om**
13. **w ww.semiconductor-philips.c om**
14. **ww w.silabs.c om**
15. **w ww.ti.c om**
16. **w ww.msdn.microsoft.c om/library**
17. **ww w.gapdev.c om**
18. **w ww.sysinternal.c om**