

Программная среда для работы с BITBUS

*Новосибирское опытно-конструкторское бюро
геофизического приборостроения*

Оглавление

1. Введение	3
2. Комплект поставки	4
3. Установка драйвера.....	5
4. Конфигурационная утилита.....	6
5. Специализированный интерфейс BITBUS	7
6. Универсальный интерфейс	12
7. Подключение внешних интерфейсных библиотек.....	15
8. Литература	17

1. Введение

Основным назначением программной среды является централизация управления контроллерами BITBUS и предоставление пользовательским приложениям аппаратно-независимого интерфейса. Рассмотрим структуру системы более подробно (рис. 1).

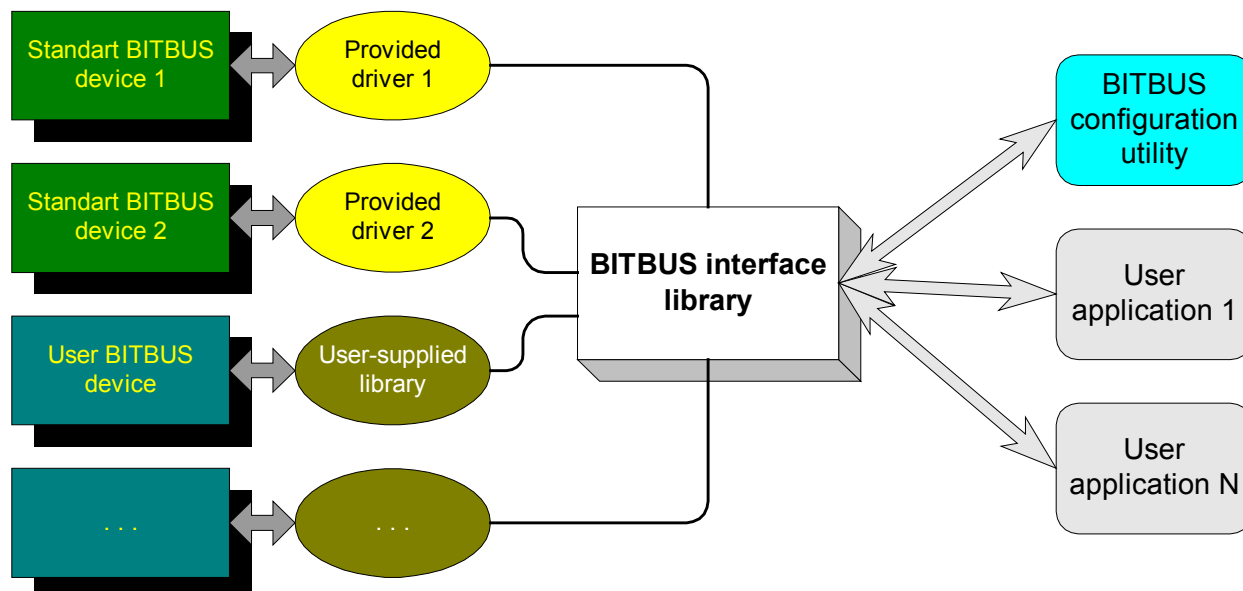


Рис. 1

Обозначения:

- **Standart BITBUS device** – устройство, поставляемое НОКБ ГП: BB_ISA, BB_PCI или контроллер фирмы Tecon - MicroTCX.
- **User BITBUS device** – любой BITBUS-контроллер.
- **Provided driver** – один из драйверов, предоставляемых НОКБ ГП: драйверы для контроллеров BB_ISA, BB_PCI или MicroTCX.
- **User-supplied library** – библиотека, экспортирующая интерфейс для работы с контроллером BITBUS.
- **BITBUS interface library** – универсальная интерфейсная библиотека для работы с контроллерами BITBUS, сглаживающая аппаратные особенности.
- **BITBUS configuration utility** – приложение для тестирования и конфигурирования контроллеров BITBUS.

Таким образом, программная среда позволяет:

- ✓ работать с контроллерами BB_ISA, BB_PCI и MicroTCX;
- ✓ легко интегрировать в систему любые контроллеры BITBUS посредством написания интерфейсных библиотек;
- ✓ централизованно конфигурировать все установленные контроллеры.

2. Комплект поставки

\doc	
BITBUS_software.pdf	Этот документ
\drivers	
bitbus.inf	INF-файл для установки драйверов
bb_isa.sys	Драйвер контроллера BB_ISA
bb_pci.sys	Драйвер контроллера BB_PCI
bb_tcx.sys	Драйвер контроллера MicroTCX
\inc	
bbtypes.h	Заголовок со стандартными типами данных
bbus_int.h	Заголовок с универсальным интерфейсом
bbus_std.h	Заголовок с устаревшим интерфейсом, предоставляемым для обратной совместимости
\lib	
bbus_int.dll	DLL содержащая универсальный интерфейс
bbus_int.lib	Библиотека для load-time линковки bbus_int.dll
bbus_std.lib	Библиотека с устаревшим интерфейсом
\tools	
bbconfig.exe	Конфигурационная утилита
uninst.exe	Очистка реестра для деинсталляции
bbus_int.dll	DLL содержащая универсальный интерфейс

3. Установка драйвера

Все поставляемые драйверы работают на платформах Windows 98/Me/2000/XP.
Для установки драйвера в Windows 2000 вы должны выполнить следующие шаги:

1. В **Control Panel** выбрать **Add/Remove Hardware**.
2. Выбрать **Add/Troubleshoot a device**.
3. В списке устройств выбрать **Add a new device**.
4. Выбрать **No, I want to select the hardware from a list**.
5. В списке типов устройств выбрать **Other devices**.
6. Нажать кнопку **Have disk** и указать путь к файлу bitbus.inf.
7. После установки ISA-адаптера система предложит вам выбрать аппаратные ресурсы, используемые платой. Следует указать значения в соответствии с аппаратной конфигурацией платы (см. соответствующую документацию).
Примечание. Ресурсы, используемые платой, не должны использоваться другими устройствами во избежание конфликтов.

Установка драйвера для VB_PCI требует выполнения только 6-го шага (устройство обнаруживается автоматически). На остальных Windows-платформах драйвер устанавливается аналогично.

4. Конфигурационная утилита

Конфигурационная утилита находится в каталоге tools и называется bbconfig.exe. Вот её основной экран (рис. 2):

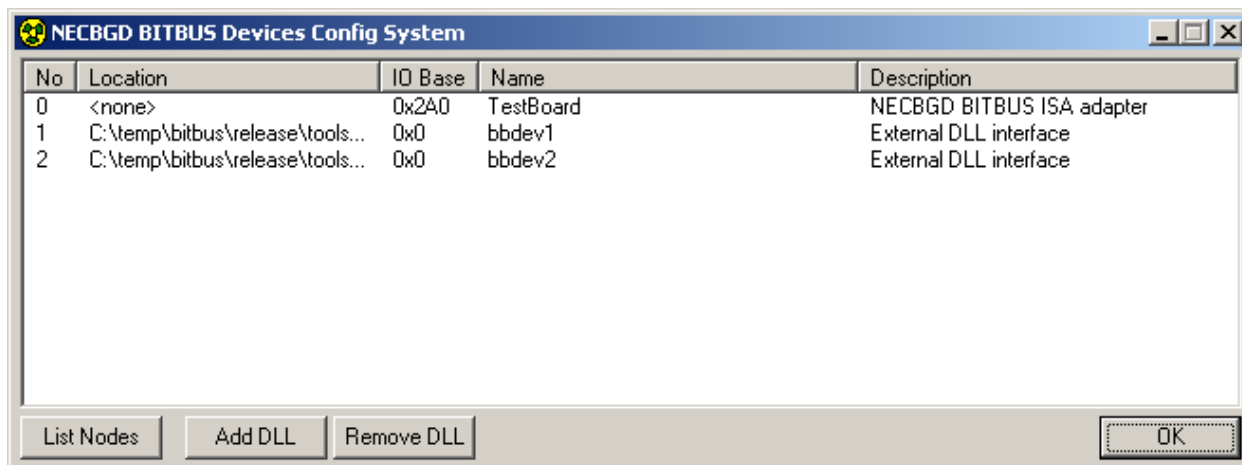


Рис.2

Поля списка имеют следующие значения:

- **No** – логический номер устройства. Может меняться в зависимости от конфигурации.
- **Location** – физическое расположение устройства для контроллера BB_PCI, <none> для BB_ISA и MicroTCX, и путь к интерфейсной библиотеке для остальных устройств.
- **IO Base** – начало диапазона ввода-вывода устройства.
- **Name** – имя устройства, по которому может быть открыт дескриптор. Имя *должно* состоять из латинских букв (регистр не важен), цифр, пробелов и знаков подчёркивания. Количество пробелов *учитывается*.
- **Description** – описание устройства.

Данная утилита позволяет переименовывать контроллеры BB_ISA, BB_PCI и MicroTCX. Для остальных устройств имена являются жёстко заданными. Если в системе присутствует несколько устройств с одинаковым именем, то при открытии дескриптора по имени будет использоваться устройство с наименьшим логическим номером.

- List Nodes** – вывести список всех нод, находящихся в состоянии online
- Add DLL** – добавить внешний интерфейс
- Remove DLL** – удалить внешний интерфейс

Примечание. Информация об именах устройств хранится в системном реестре. Для удаления всей информации об устройствах BITBUS из реестра запустите утилиту uninst.exe, находящуюся в каталоге tools.

5. Специализированный интерфейс BITBUS

Специализированный интерфейс реализован в библиотеке `bbus_int.dll`, которая располагается в директории `lib`. Для load-time линковки этой библиотеки используйте статическую библиотеку `bbus_int.lib`. Интерфейс библиотеки описан в заголовочном файле `bbus_int.h`.

Примечание. Включение заголовка `bbus_int.h` требует предварительного включения заголовка `bbtypes.h`, содержащего типы данных BITBUS.

5.1. Структуры данных

```
typedef struct _BITBUS_INFO {
    DWORD        dwDeviceNumber;
    DWORD        dwBaseAddress;
    BOOL         bExternalDll;
    TCHAR        szDeviceLocation[LOCATION_STR_LEN_MAX];
    TCHAR        szDeviceDescription[DESCRIPTION_STR_LEN_MAX];
    TCHAR        szDeviceName[BITBUS_DEVICE_NAME_MAX];
} BITBUS_INFO, *PBITBUS_INFO;
```

Параметры

- **dwDeviceNumber** – порядковый номер BITBUS-адаптера
 - **dwBaseAddress** – базовый адрес IO-пространства платы
 - **bExternalDll** – TRUE, если устройство работает через внешний интерфейс (DLL)
 - **szDeviceLocation** – строка, указывающая расположение устройства
 - **szDeviceDescription** – строка, содержащая описание устройства
 - **szDeviceName** – имя устройства
-

```
typedef struct _BITBUS_STATUS {
    DWORD        dwStatus;
    DWORD        dwMessagesSent;
    DWORD        dwMessagesReceived;
} BITBUS_STATUS, *PBITBUS_STATUS;
```

Параметры

- **dwStatus** – статусный регистр (значение определяется типом устройства)
 - **dwMessagesSent** – количество отправленных сообщений
 - **dwMessagesReceived** – количество принятых сообщений
-

```
typedef struct _BITBUS_MESSAGE {
    WORD         Link;
    BYTE         Length;
    BYTE         Flags;
    BYTE         Node;
    BYTE         SourceDest;
    BYTE         CmdStat;
    BYTE         Data[BITBUS_DATA_MAX];
} BITBUS_MESSAGE, *PBITBUS_MESSAGE;
```

Параметры

Подробное описание параметров приведено в документе «The BITBUS Interconnect Serial Control Bus Specification», Intel Corp. 1995.

5.2. Функции

```
BOOL BitbusGetDeviceInfo(DWORD dwDeviceNumber, PBITBUS_INFO pBitbusInfo);
```

Получить информацию об устройстве.

Параметры

- **dwDeviceNumber** – номер устройства (первое устройство имеет номер 0)
- **pBitbusInfo** – указатель на структуру, в которой при успешном завершении будет содержаться информация об устройстве

Возвращаемые значения

Возвращает TRUE если удалось успешно получить информацию об устройстве с номером DeviceNumber, FALSE в случае ошибки или при отсутствии устройства. Код ошибки можно получить при помощи функции GetLastError().

```
BOOL BitbusOpenHandle(  
    PBBHANDLE pDeviceHandle,  
    DWORD dwDeviceNumber,  
    DWORD dwFlags  
);
```

Открыть дескриптор устройства.

Параметры

- **pDeviceHandle** – указатель на структуру, в которой при успешном завершении будет содержаться дескриптор устройства
- **dwDeviceNumber** – номер устройства
- **dwFlags** – регистр флагов. Может содержать комбинацию значений:

Значение	Описание
BB_FULL_ACCESS	Если установлен, то дальнейшее открытие дескриптора возможно только в том случае, если этот флаг сброшен, т.е. в системе может быть только один дескриптор устройства, открытый с этим флагом.

Возвращаемые значения

TRUE в случае нормального завершения, FALSE в случае ошибки. Код ошибки можно получить при помощи функции GetLastError().

```
BOOL BitbusOpenNamedHandle(  
    PBBHANDLE pDeviceHandle,  
    LPCTSTR szDeviceName,  
    DWORD dwFlags  
);
```

Открыть дескриптор платы, установленной по умолчанию.

Параметры

- **pDeviceHandle** – указатель на структуру, в которой при успешном завершении будет содержаться дескриптор устройства
- **szDeviceName** – имя устройства. Имя *должно* состоять из латинских букв (регистр не важен), цифр, пробелов и знаков подчёркивания. Количество пробелов *учитывается*.
- **dwFlags** – регистр флагов. См. описание функции BitbusOpenHandle().

Возвращаемые значения

TRUE в случае нормального завершения, FALSE в случае ошибки. Код ошибки можно получить при помощи функции GetLastError().

```
VOID BitbusCloseHandle(PBBHANDLE pDeviceHandle);
```

Закрыть дескриптор устройства. После закрытия дескриптор не может быть использован.

Параметры

- **pDeviceHandle** – указатель на дескриптор устройства
-

```
BOOL BitbusReset(PBBHANDLE pDeviceHandle);
```

Сброс адаптера.

Параметры

- **pDeviceHandle** – указатель на дескриптор устройства

Возвращаемые значения

TRUE в случае нормального завершения, FALSE в случае ошибки. Код ошибки можно получить при помощи функции GetLastError().

```
BOOL BitbusGetStatus(PBBHANDLE pDeviceHandle, PBITBUS_STATUS pBitbusStatus);
```

Получить статус BITBUS-адаптера.

Параметры

- **pDeviceHandle** – указатель на дескриптор устройства
- **pBitbusStatus** – указатель на структуру, в которую при успешном завершении записывается статус устройства

Возвращаемые значения

TRUE в случае нормального завершения, FALSE в случае ошибки. Код ошибки можно получить при помощи функции GetLastError().

```
BOOL BitbusSendMessage(  
    PBBHANDLE pDeviceHandle,  
    PBITBUS_MESSAGE pSendMsg,  
    PBITBUS_MESSAGE pReplyMsg,  
    LPOVERLAPPED lpOverlapped  
);
```

Отправить сообщение и принять ответ с устройства.

Параметры

- **pDeviceHandle** – указатель на дескриптор устройства
- **pSendMsg** – указатель на буфер сообщения для отправки
- **pReplyMsg** – указатель на буфер, в котором при успешном завершении будет содержаться ответ на сообщение
- **lpOverlapped** – указатель на структуру OVERLAPPED. Если равен NULL, то поток блокируется до прихода ответа, иначе поток разблокируется сразу после отправки сообщения в драйвер. В этом случае для получения ответа нужно воспользоваться функцией GetOverlappedResult(). Для подробной информации о структуре OVERLAPPED см. MSDN. Для внешних устройств должен быть установлен в NULL.

Возвращаемые значения

TRUE в случае нормального завершения, FALSE в случае ошибки. Код ошибки можно получить при помощи функции GetLastError().

```
BOOL BitbusSetTimeout(PBBHANDLE pDeviceHandle, DWORD dwMilliseconds);
```

Установить тайм-аут ожидания ответа на сообщение. Отсчёт времени начинается после отправки сообщения в BITBUS-адаптер. Если за указанное время адаптер не ответит, возвращается сообщение об ошибке ERROR_SEM_TIMEOUT.

Параметры

- **pDeviceHandle** – указатель на дескриптор устройства
 - **dwMilliseconds** – время ожидания ответа на сообщение в миллисекундах. При установке в 0 принимает значение, равное 1 секунде.
-

6. Универсальный интерфейс

Универсальный интерфейс реализован в библиотеке `bbus_std.lib`. Для использования универсального интерфейса приложение должно быть слинковано с `bbus_int.lib` и иметь доступ к динамической библиотеке `bbus_int.dll`, так как все функции универсального интерфейса реализованы через специализированный интерфейс. Функции универсального интерфейса описаны в заголовочном файле `bbus_std.h`.

Внимание! Для работы с универсальным интерфейсом приложение *не должно* включать заголовочный файл `bbus_int.h`. Это означает, что для доступа к контроллеру BITBUS может использоваться только один из интерфейсов, но не оба одновременно.

6.1. Структуры данных

```
typedef struct BbMessage_s {
    USHORT    Link;
    UCHAR     Length;
    UCHAR     Flags;
    UCHAR     Node;
    UCHAR     SourceDest;
    UCHAR     CmdStat;
    UCHAR     Data[BITBUS_DATA_MAX];
} BbMessage_t;
```

Параметры

Подробное описание параметров приведено в документе «The BITBUS Interconnect Serial Control Bus Specification», Intel Corp. 1995.

```
typedef struct BitbusStat_s {
    UCHAR     StatusReg;
    ULONG     PortBase;
    ULONG     MessagesSent;
    ULONG     MessagesReceived;
} BitbusStat_t;
```

Параметры

- **StatusReg** – статусный регистр (значение определяется типом устройства)
 - **PortBase** – базовый адрес IO-пространства устройства
 - **MessagesSent** – количество отправленных сообщений
 - **MessagesReceived** – количество принятых сообщений
-

6.2. Функции

```
HANDLE BitbusOpenHandle(DWORD DeviceNumber, DWORD Flags);
```

Открыть дескриптор устройства по номеру.

Параметры

- **DeviceNumber** – логический номер устройства, нумерация начинается с нуля
- **Flags** – регистр флагов. См. описание функции BitbusOpenHandle() в главе 5.

Возвращаемые значения

Дескриптор платы в случае успешного завершения и INVALID_HANDLE_VALUE в случае ошибки. Код ошибки можно получить, используя функцию GetLastError().

```
HANDLE BitbusOpenNamedHandle(const char *Name, DWORD Flags);
```

Открыть дескриптор устройства по имени.

Параметры

- **Name** – строка, содержащая имя устройства и оканчивающаяся нулевым символом. Имя *должно* состоять из латинских букв (регистр не важен), цифр, пробелов и знаков подчёркивания. Количество пробелов *учитывается*.
- **Flags** – регистр флагов. См. описание функции BitbusOpenHandle() в главе 5.

Возвращаемые значения

Дескриптор платы в случае успешного завершения и INVALID_HANDLE_VALUE в случае ошибки. Код ошибки можно получить, используя функцию GetLastError().

```
VOID BitbusCloseHandle(HANDLE DevHandle);
```

Закрывает дескриптор устройства. Каждый открытый дескриптор должен быть закрыт перед завершением работы.

Параметры

- **DevHandle** – дескриптор устройства
-

```
BOOL BitbusReset(HANDLE DevHandle);
```

Сброс адаптера.

Параметры

- **DevHandle** – дескриптор устройства

Возвращаемые значения

TRUE в случае нормального завершения, FALSE в случае ошибки. Код ошибки можно получить при помощи функции GetLastError().

```
BOOL BitbusSendMessage (  
    HANDLE DevHandle,  
    BbMessage_t *SendMsg,  
    BbMessage_t *ReplyMsg  
);
```

Отправить сообщение и принять ответ с устройства.

Параметры

- **DevHandle** – дескриптор устройства
- **SendMsg** – указатель на буфер сообщения для отправки
- **ReplyMsg** – указатель на буфер, в котором при успешном завершении будет содержаться ответ на сообщение

Возвращаемые значения

TRUE в случае нормального завершения, FALSE в случае ошибки. Код ошибки можно получить при помощи функции GetLastError().

```
BOOL BitbusSetTimeout (HANDLE DevHandle, ULONG Timeout);
```

Установить тайм-аут ожидания ответа на сообщение. Отсчёт времени начинается после отправки сообщения в BITBUS-адаптер. Если за указанное время адаптер не ответит, возвращается сообщение об ошибке ERROR_SEM_TIMEOUT.

Параметры

- **DevHandle** – дескриптор устройства
- **Timeout** – время ожидания ответа на сообщение в миллисекундах. При установке в 0 принимает значение, равное 1 секунде.

Возвращаемые значения

TRUE в случае нормального завершения, FALSE в случае ошибки. Код ошибки можно получить при помощи функции GetLastError().

```
BOOL BitbusGetStatus (HANDLE DevHandle, BitbusStat_t *Status);
```

Получить статус BITBUS-адаптера.

Параметры

- **DevHandle** – дескриптор устройства
- **Status** – указатель на структуру, в которую при успешном завершении записывается статус устройства

Возвращаемые значения

TRUE в случае нормального завершения, FALSE в случае ошибки. Код ошибки можно получить при помощи функции GetLastError().

7. Подключение внешних интерфейсных библиотек

Для интегрирования в программную среду для работы с BITBUS других устройств (т.е. не являющихся аппаратно-совместимыми с BB_ISA, BB_PCI и MicroTCX) необходимо иметь интерфейсные динамические библиотеки (DLL), экспортирующие набор функций, перечисленных в пункте 7.1, для работы с устройством.

Чтобы сделать устройство доступным, нужно запустить конфигурационную утилиту `bbconfig.exe`, нажать кнопку «Add DLL» и указать путь к интерфейсной библиотеке устройства. В списке появятся все устройства, доступные через эту интерфейсную библиотеку.

7.1. Определение внешнего интерфейса

```
const char *BitbusGetName (DWORD dwDeviceNumber);
```

Получить имя устройства по номеру.

Параметры

- **dwDeviceNumber** – номер устройства. Каждая интерфейсная библиотека имеет свою нумерацию устройств, которые она поддерживает. Эти устройства нумеруются по порядку, начиная с нуля.

Возвращаемые значения

В случае успешного завершения - указатель на строку с именем устройства, оканчивающуюся нулевым символом; в случае ошибки или при отсутствии устройства с таким номером - NULL. При отсутствии устройства код последней ошибки, возвращаемый функцией `GetLastError()`, должен быть равен `ERROR_NO_MORE_ITEMS`.

```
HANDLE BitbusOpenNamedHandle (const char *szName);
```

Открыть дескриптор устройства по имени.

Параметры

- **szName** – строка, содержащая имя устройства и оканчивающаяся нулевым символом. Имя *должно* состоять из латинских букв (регистр не важен), цифр, пробелов и знаков подчёркивания. Количество пробелов *учитывается*.

Возвращаемые значения

Дескриптор платы в случае успешного завершения и `INVALID_HANDLE_VALUE` в случае ошибки. Код ошибки можно получить, используя функцию `GetLastError()`.

```
VOID BitbusCloseHandle (HANDLE hDevice);
```

Закрывает дескриптор устройства.

Параметры

- **hDevice** – дескриптор устройства

```
BOOL BitbusReset(HANDLE hDevice);
```

Сброс адаптера.

Параметры

- **hDevice** – дескриптор устройства

Возвращаемые значения

TRUE в случае нормального завершения, FALSE в случае ошибки. Код ошибки можно получить при помощи функции GetLastError().

```
BOOL BitbusSendMessage(  
    HANDLE hDevice,  
    PBITBUS_MESSAGE pSendMsg,  
    PBITBUS_MESSAGE pReplyMsg  
);
```

Отправить сообщение и принять ответ с устройства.

Параметры

- **hDevice** – дескриптор устройства
- **pSendMsg** – указатель на буфер сообщения для отправки
- **pReplyMsg** – указатель на буфер, в котором при успешном завершении будет содержаться ответ на сообщение

Возвращаемые значения

TRUE в случае нормального завершения, FALSE в случае ошибки. Код ошибки можно получить при помощи функции GetLastError().

```
BOOL BitbusSetTimeout(HANDLE hDevice, DWORD dwMilliseconds);
```

Установить тайм-аут ожидания ответа на сообщение. Отсчёт времени начинается после отправки сообщения в BITBUS-адаптер. Если за указанное время адаптер не ответит, возвращается сообщение об ошибке ERROR_SEM_TIMEOUT.

Параметры

- **hDevice** – дескриптор устройства
- **dwMilliseconds** – время ожидания ответа на сообщение в миллисекундах. При установке в 0 принимает значение, равное 1 секунде.

Возвращаемые значения

TRUE в случае нормального завершения, FALSE в случае ошибки. Код ошибки можно получить при помощи функции GetLastError().

8. Литература

- 1) Описание платы MicroTCX, файлы mtcx.pdf, mtcxref.pdf
- 2) Описание платы BB_ISA, файл bb_isa.pdf