HOPE COMPUTER COMPANY LTD.

Paradox 数据库 3.0～3.5 技术丛书之五

程序员参考手册

李怡南　　编译

北京希望电脑公司

5

Paradox 数据库 3.0～3.5 技术丛书之五

程序员参考手册

李怡南　　编译

5

北京希望电脑公司

一九九一年五月

编者的话

著名的 Borland 公司已成功地向世界推出众多优秀软件，然而作为其优秀代表之一的数据库软件——Paradox 还不为广大用户所熟知，现在北京希望电脑公司组织编译了关于这个软件包的系列丛书。我们相信当用户了解该软件，将义无反顾地选择——结构精巧，功能强大的 Paradox，Paradox 支持 Turbo 系列语言的编程，用户将不再为数据库不支持自己所熟悉的 Turbo C；Turbo C++； Pascal 语言而烦恼，更为重要的是，Paradox 的库程序可以为 MS Window 程序员所利用，可以说，Paradox 的可移植性是数据库软件而无可匹敌的。

这本书的编译得到秦人华老师鼎力支持和帮助，在这里致此衷心的感谢。谭珍梅同志也为该书做了大量工作，在此一并感谢。

编者　李怡南

一九九一年五月

第一部分　C语言参考手册

目录

第一部分　C 语言参考手册

第一章　应用程序样板……3

第二章　Engine 函数……67

附录A　Engine 的错误代码……190

第二部分　Pascal 语言参考手册

第一章　样板程序……197

第二章　Engine 函数……261

附录　Engine 错误代码……369

前　言

本书是 Paradox Engine 的C语言技术参考指南，其主要内容是按照字母顺序列出的全部 Engine 函数。有经验的、熟悉 Paradox 概念及术语的C语言程序员可以从本书中找到他们需要的一切资料。如果想了解 Engine 及 Paradox 的全貌,请参阅有关的《用户指南》。

如果你尚未安装 Engine，可参阅《用户指南》的前言中有关 Paradox Engine 安装方法的说明。

本书的内容提要

除了本章“前言”之外，书中还包括下列章节：

第一章：“应用程序样板”。其中包含两个应用程序：一个是 IMPORT.C（该程序利用 Engine 把一个 ASCII 流式文本文件转换成 Paradox 的格式），另一个是 FONEDEX.C（该程序利用 Engine 建立一个电子卡片式的数据库）。

第二章：“Engine 函数”。按字母顺序列出了全部 Engine 函数。

附录A：“Engine 的错误代码”。以名称及数值为序列出了 Engine 的错误代码。

编译及使用 Engine

本节提供了如何用C语言编译和使用 Engine 的必要资料。学习使用 Engine 的最佳途径是利用样板应用程序。请阅本书第一章中关于使用样板应用程序的参考指南。

Turbo C++和 Borland C++

应确保 Turbo C++或 Borland C++知道 Engine 的蕴含文件和库文件在什么地方（安装 Turbo C++或 Borland C++时在 TURBOC.CFG 中指出）。然后就可以按自己的须要去编译应用程序。在命令行中应该用 -ml 参数选择大模式。命令行中的最后一个参数应该指定 Engine 库：PXENGTCL.LIB。命令行的格式举例如下：

```
tcc -ml <cfile.c> PXENGTCL.LIB
```

这样就可以成功地编译 Turbo C++应用程序并将其与 Engine 链接起来（应该用你的源文件名取代命令行中的<cfile.c>）。

用 Borland C++编译 Engine 应用程序时,命令行格式为：

```
bcc -ml <cfile.c> PXENGTCL.LIB
```

（应该用你的源文件名取代命令行中的<cfile.c>）。

Turbo C 2.0

应确保 Turbo C2.0 知道 Engine 的蕴含文件和库文件在什么地方（安装 Turbo C2.0 时在 TURBOC.CFG 中指出）。然后就可以按自己的须要去编译应用程序。在命令行中应该用 -ml 参数选择大模式。命令行的最后一个参数应该是 Engine 库：PXENGTCL.LIB 和 PXENGTC2.LIB。命令行的格式举例如下：

```
tcc -ml <cfile.c> PXENGTCL.LIB PXENGTC2.LIB
```

这样就可以成功地编译你的应用程序并将其与 Engine 链接起来（用你的源文件名取代命令行中的<cfile.c>）。

Microsoft C

Microsoft C 5.1 及 6.0 根据 DOS 环境变量 INCLUDE 和 LIB 的值来确定蕴含文件和库文件的位置。应确保这两个变量的值中包含着安装了 Engine 蕴含文件和库文件的子目录名。如果这些文件被安装在 C:\PXENGINE 中（缺省的安装位置），应该执行下列 DOS 命令：

```
SET INCLUDE = C:\PXENGING; %INCLUDE%
SET LIB = C:\PXENGING; %LIB%
```

然后就可以用 Microsoft C 和 Engine 来编译你的应用程序。

编译和链接

为了编译 Microsoft C 程序并将其与 Engine 链接起来，可以用 CL 命令来进行编译和链接。为了用一条 CL 命令完成一个或多个源模块的编译和链接，应键入如下命令：

```
CL /AL /F D00 [options] <source1> <sourceN> [-link opts] PXENGMSL.LIB
```

命令行中各参数的含义如下：

/AL	指定Microsoft C的大内存模式。目前 Paradox Engine只支持这一种模式。
/F D00	增加缺省的栈空间。
[option]	其它供C编译器使用的可选参数。（更详细的资料请阅读Microsoft C的有关文档）
<source1>	提供给编译器的第一个源模块的名称。
<sourceN>	提供给编译器的一个或多个其它源模块的名称。
[-link opts]	供链接器使用的可选参数（更详细的资料请阅 Microsoft C的文档）。
PXENGMSL.LIB	Paradox Engine库的名称。

如果源程序没有错误，这条命令就能够完成 Microsoft C 程序的编译并将其与 Paradox Engine 链接起来。

第一章　应用程序样板

应用程序样板

发行盘上提供了两个 Engine 样板应用程序的源代码。本章将详细讲解这两个应用程序，逐段讨论源代码并给出完整的源程序。

阅读了这些例子，就能了解如何建立 Engine 应用程序，也可以把其中一部分代码移植到你自己的应用程序中去。

编译 Engine 应用程序时切记必须使用大模式（在命令行上使用-ml 参数）。

警告：对于大型应用程序，必须重新设置 Engine 栈空间的大小，因为 Engine 的缺省栈空间不一定能满足你的程序的要求。

提供给用户的两个样板程序是：

■IMPORT.C：用于说明怎样把外部数据转换成 Paradox 的格式。

■FONEDEX.C：用于说明怎样在网络上建立一个简单的、由菜单驱动的数据库。

Engine 发行盘上还有两个简单的示例程序：SORT.C 用于说明如何将 Engine 用于排序，SEARCH.C 用于说明怎样利用 Enging 进行搜索。

应用程序之一：转换 ASCII 数据

第一个应用程序是 IMPORT.C，该程序利用 Engine 从一个 ASCII 文件中读取数据并建立一个有索引的 Paradox 文件。用户可以仿照这个例子中的方法去开发从流式文件中读取数据的应用程序。这个流式文件也可以是一个与主机或条形码扫描器连接的串行通讯口。

这个应用程序从指定的 ASCII 文件中读取数据（由用户在命令行上指定该文件的名称），该文件中包括下列字段：

■第一个字段是顾客编号，是一个长度为 8 个字节的字符型字段。

■第二个字段是部件编号，也是一个长度为 8 个字节的字符型字段。

■第三个字段是部件数量，是一个数字型字段。

■第四个字段是格式为 mm/dd/yyyy 的日期。IMPORT.C 用 PXDateEncode 函数把它转换成 Engine 的日期格式。

表 1.1　IMPORT.C 中的数据结构

字段	类型	格式
Customer Number（顾客编号）	A8	AAAAAAAA
Part Number（部件编号）	A8	AAAAAAAA
Quantity（数量）	N	NNNNN
Date（日期）	D	MM/DD/YYYY

应用程序从命令行上读取两个参数：

```
import  <ascii_file>  <paradox_file>
```

第一个参数 ascii_file 就是前面提到过的那个定长记录的 ASCII 文件；第二个参数 Paradox_file 是要建立的 Paradox 数据表的名称。程序将用 PXTblExist 函数检查此数据表是否存在。

程序的结构

程序首先使用了一组#include 语句。Engine 的头文件是 PXENGINE.H，其名称用双引号括起来（"pxengine.h"），因此编译器徽得从前目录以及标准的蕴含文件目录中去寻找它。

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "pxengine.h"
```

接着，程序中使用了一组 #define 语句，对一些网络参数进行了定义。如果编译时定义了常数 NETWORK，IMPORT.C 就可以支持网络：

- NETUSERNAME：定义用户在网络上登录的名称。
- NETDIR：PARADOX.NET 所在目录的完整路径。
- NETTYPE：当前使用的网络的类型。

```
#ifdef   NETWORK
#define NETUSERNAME   "username"    /* network username */
#define NETDIR        ""            /* net directory */
#define NETTYPE       NOTONNET      /* network type */
#endif
```

样板程序中没有给出专用的网络参数，以便使该程序能够不加修改地在各种系统中运行。如果要在网络上工作，应指定专用的网络参数。例如：

```
#define NETDIR        "\\pdox35\\"
#define NETTYPE       NOVELLNET
```

在网络的条件定义后面，程序进行了其它各种说明，并按照各字段在 Paradox 数据表中的放置顺序定义了这些字段。程序还定义了 ASCII 文件中各字段的长度。最后定义了两个常数：SUCCESS（函数调用成功）和 FAIL（函数调用失败）。

```
/* Field handles of Paradox table */
FIELDHANDLE FieldPartNbr,          /* Part Number */
            FieldCustNbr,          /* Customer Number */
            FieldQuantity,         /* Quantity */
            FieldDate;             /* Date */

/* Size of ASCII table fields */
#define ASCIISIZECUSTNBR  9        /* Custemer Number Length */
#define ASCIISIZEPARTNBR  9        /* Part Number Length */
```

```
#define ASCIISIZEDATE     11        /* Date Length */

/* Function returns */
#define SUCCESS           0         /* function succeded */
#define FAIL              -1        /* function failed */
```

至此，所有的常数值均已定义完毕。应用程序还需要定义一个结构，用于存放从 ASCII 文件中读入的数据。结构中定义了四个字段：顾客编号、部件编号、数量和日期。

```
/* Structure for an ASCII fixed-length record */
typedef struct
{
   char CustNbr[ASCIISIZECUSTNBR];        /* Custemer Number */
   char PartNbr[ASCIISIZEPARTNBR];        /* Part Number */
   int   quantity;                        /* Quantity */
   char date[ASCIISIZEDATE];              /* Date */
} AsciiRecord;
```

接下去,程序定义了几个字符串数组：FieldNames 用于字段名；FieldTypes 用于字段类型（各字段的名称和类型见表 1.1）。IMPORT.C 在建立 Paradox 数据表时要用到这两个数组。

```
/* Field names of Paradox table */
char *FieldNames[] =
{
   "Part Number";
   "Cust Number";
   "Quantity";
   "Date";
};

/* Field types of Paradox table */
char *FieldTypes[] =
{
   "A8";
   "A8";
   "N";
   "D";
};
```

然后，用两个 #define 语句执行简单的计算，以确定数据表中字段的个数和键字段的个数。

```
#define NBRFIELDS       (sizeof(FieldNames) / sizeof(char *))
#define NBRKEYS         2
```

最后，应用程序声明了这个源文件中所用到的全部函数的原型。

- OpenFiles、OpenAsciiFile 和 CreateParadoxFile 用于建立和打开应用程序要使用的各个文件。
- translate 用于从流式文件中读入数据。
- TranslateBuff 把数据传送到一个 Paradox 记录中。
- CloseFiles 关闭程序运行期间打开的所有文件。
- InitFields 用于字段句柄初始化。
- Error 如果程序运行中发生了错误，此函数可显示出错误信息。

各函数的原型如下：

```
/* Function prototypes */
int   main(int, char **);
int   OpenFiles(char **,TABLEHANDLE *, FILE **);
int   OpenAsciiFile(char *,FILE **);
int   CreateParadoxFile(char *);
int   translate(TABLEHANDLE, FILE *);
int   TranslateBuffer(AsciiRecord *, RECORDHANDLE);
void  CloseFiles (TABLEHANDLE, FILE *);
int   InitFields(TABLEHANDLE);
int   Error(int);
```

在下列各节中，我们将详细地讨论每一个函数。

IMPORT.C 的核心

函数 main 控制程序的执行并决定函数的调用顺序。它从命令行中读取两个参数：argc 和 argv。argc 是命令行传给应用程序的参数个数；argv 是一个字符串数组，其中的内容是程序的名称和命令行上传过来的其它参数。

```
int main(int argc, char **argv)
```

main 的第一条语句定义了 ASCII 文件的流式文件指针和 Paradox 数据表的表句柄。

```
FILE   *fpascii                 /* file pointer to ASCII file */
TABLEHANDLE tblHandle           /* table handle to Paradox table */
```

因为命令行传送给 main 两个参数，所以 argc 的值应为 3（由于在 DOS 3.x 或以后的版本中在传送命令行上的参数之前还要传送程序的名称，因此 argc 的值至少为 1。本程序又传送了两个文件名，因此 argc 的值为 3）。如果命令行上输入的参数个数不正确，Engine 会发出一个错误信息。

```
/* Expect two file names on the command line */
if(argc != 3)
{
```

```
    printf("usage: IMPORT <ascii_file> <paradox_file)\n");
    return(FAIL);
}
```

启动 Engine

应该根据 Engine 启动时的环境是单用户还是多用户环境来决定是用 PXInit 还是用 PXNetInit 来初始化 Engine。应当注意：某些单用户的 DOS 扩展器要求用 PXNetInit 进行初始化，以便支持文件的并发存取。

本程序为网络使用了条件定义：如果未定义 NETWORK，程序将调用 PXInit 来启动单用户环境下的 Engine；如果定义了 NETWORK，程序就会则调用 PXNetInit 来进行初始化。PXNetInit 所使用的两个参数已在源程序的开头处作了定义，它们分别代表网络控制文件（PARADOX.NET）的路径、网络的类型和网络用户的名称。

初始化函数只应该调用一次。除非又调用 PXExit 关闭了此环境，否则不应再次进行初始化。如果初始化操作失败了，程序将调用 Error，用它显示出一条错误信息，并终止程序的运行。

```
/* Initialize the Engine */
#ifndef NETWORK
    if(Error(PXInit()))
#else
    if (Error(PXNetInit(NETDIR, NETTYPE, USERNAME)))
#endif
    exit(1);
```

如果初始化成功了，程序就继续执行下一个逻辑步骤：打开文件。首先，程序调用 CreateParadoxFile 以建立 Paradox 数据表。如果由于某种原因而使得建立数据表的操作失败，程序将调用 Exit 并终止运行。

```
if (CreateParadoxFile(argv[2] == FAIL)
    exit(1);
```

如果建立文件的操作成功了，那么 Paradox 数据表就已经存在了。于是程序把命令行参数传送给函数 OpenFiles，以打开 ASCII 文件和 Paradox 数据表。

```
/* Open ASCII file and Paradox file */
if (OpenFiles(argv, &tblHandle, &fpAscii) == SUCCESS)
{
  translate(tblHandle, fpAscii);
  CloseFiles(tblHandle, fpAscii);
}
```

关闭 Engine

程序运行结束时要调用 PXExit 关闭 Engine。若此时发生了错误，将调用 Error 显示错误

信息。

```
return(Error(PXExit()));
```

打开文件

现在，让我们来更详细地研究一下 main。调用 OpenFiles 是为了打开程序将要使用的文件。如果命令行上传送过来的参数个数是正确的，则 argv[1]中是 ASCII 文件的名称；argv[2]中是 Paradox 数据表的名称。这两个参数立即分别被送往 PXTblOpen 和 OpenAsciiFile 函数，以打开这些文件。调用 InitFields 是为了初始化字段句柄。

```
int OpenFiles(char **argv, TABLEHANDLE *tblHandlePtr, FILE **fpAscii)
{
    if (OpenAsciiFile(argv[1], fpAscii) == FAIL)
        return(FAIL);

/* Open the Paradox file */
    if (Error(PXTblOpen(argv[2], tblHandlePtr, 0, 0)))
        return(FAIL);
    if (InitFields(*tblHandlePtr) == FAIL)
        return(FAIL);

    return(SUCCESS);
}
```

调用 PXTblOpen 目的是初始化 TABLEHANDLE 参数。程序传送给此函数四个参数：要打开的数据表的名称和指针 tblHandlePtr，第三个参数用于指定数据表的索引（该值为 0 表示指定主索引），最后一个参数表示要求把对文件所作的一切修改都送入缓冲区中（如果想提高数据的安全度，此参数可作另一种选择：发送数据时将每个记录立即存盘）。如果初始化时调用的是 PXNetInit，最后这个参数将被忽略，对数据记录所作的一切修改都将在发送时直接写入磁盘。

打开 ASCII 流式文件

OpenAsciiFile 调用 fopen 打开流式文件。命令行传送过来的字符串被用来指定要打开的文件；此外还指定了打开文件的目的是读数据。然后，它把文件句柄值（函数 fopen 的返回值）赋给 fpascii。如果返回给 fpascii 的值是 NULL，表示发生了错误。此时程序将调用 C 函数 perror，从标准出错流中打印出错误信息。

```
int OpenAsciiFile(char *fileName, FILE **fpAscii)
{
    if ((*fpAscii = fopen(fileName, "r") == NULL)
    {
```

```
        perror(fileName);
        return(FAIL);
    }
    else
        return(SUCCESS);
}
```

建立 Paradox 数据表

函数 CreateParadoxFile 有三个作用：

■确认指定的文件不存在。

■建立该文件。

■建立主索引文件。

```
int CreateParadoxFile(char *fileName)
{
    int exise, i;
    FIELDHANDLE keys[NBRFIELDS];
```

为了确定该数据表是否存在，程序调用了 PXTblExist。调用时传送的参数是数据表的名称和一个整数型局部变量（Exist）的地址。如果调用成功，Exist 中将是一个非 0 的值（表示 True）。

```
    if (Error(PXTblExist(FILEnAME, &EXIST)))
        RETURN(FAIL);

    if (exist)
    {
        printf("IMPORT: Table %s already exists\n", fileName);
        return(FAIL);
    }
```

如果该数据表确实不存在，CreateParadoxFile 将调用 PXTblCreate 建立该表。第一个参数 fileName 是数据表的名称，接下去的三个参数用于指定数据表中每个记录的字段个数、字段名称及其类型。（有关字段命名的规则，请阅《Paradox 用户指南》的第二章）

```
    if (Error(PXTblCreate(fileName, NBRFIELDS, FieldNames, FieldTypes)))
        return(FAIL);
```

数据表建立之后，程序将调用 PXKeyAdd 为它建立主索引。索引文件的名称与数据表相同，主键字段的个数由 NBRKEYS 指出。第三个参数 keys 是一个整数型数组，代表用于被用作主键的字段句柄。最后一个参数 PRIMARY 表示被建立的索引的类型。有关主索引和键字段的细节，请阅《Paradox 用户指南》的第六章和第十四章。

```
/* Add first two fields as primary key */
for (i=0; i < NBRKEYS; i++)
    keys[i] = i + 1;
if (Error(PXKeyAdd(fileName, NBRKEYS, keys, PRIMARY)))
    return(FAIL);

return(SUCCESS);
```

读入及发送数据

本应用程序的核心是 translate 函数。在从 ASCII 流式文件传输信息之前，需要建立一个临时的记录传输缓冲区，以存贮被传输的信息。调用 PXRecBufOpen 可为这个记录缓冲区分配空间。然后就可以对 ASCII 流式文件进行读入、传输操作并将数据追加到数据表中去。

```
int translate(TABLEHANDLE tblHandle, FILE *fpAscii)
{
    AsciiRecord   asciiBuf;
    RECORDHANDLE recHandle;

    /* Set up a record handle */
    if (Error(PXRecBufOpen(tblHandle, &recHandle)))
        return(FAIL);
```

在 while 循环中调用了函数 feof 以测试 ASCII 流式文件的文件尾。在循环中,调用 fcanf 函数从文件中将两个字符串、一个整数及另一个字符串读到局部变量 asciiBuf 中。此变量的类型为 AsciiRecord。如果 fscanf 的返回值与要读的字段数不相等，循环将终止。

```
/* Read records until end of ASCII file */
while(!feof(fpAscii))
{
    /* Read the buffer and make sure we do not encounter an
       unexpected enf-of-file */
    if (fscanf(fpAscii, "%s %s %d %d", asciiBuf.PartNbr,
        asciiBuf.CustNbr, &asciiBuf.quantity,
        asciiBuf.date) != NBRFIELDS)
        break;
```

接着，TranslateBuffer 将字段值从 asciiBuf 传送到变量 recHandle 中。最后，如果读入及传输均已顺利完成，就调用 PXRecAppend 把新转换过来的数据追加到数据表中。这个函数使用两个参数：数据表句柄和要被追加的记录。

```
if (TranslateBuffer(&asciiBuf, recHandle) == FAIL)
    return(FAIL);
```

由于数据表有索引，新记录将按索引顺序被插入适当位置。如果数据表无索引，新记录将成为数据表的最后一个记录。如果记录插入成功，该记录将成为当前记录。

```
    /* And write it to the Paradox table */
if (Error(PXRecAppend(tblHandle, recHandle)))
    return(FAIL);
```

到达流式文件的尾端后，再没有其他数据需要传输了。于是调用 PXRecBufClose 关闭记录句柄。

```
if (Error(PXRecBufClose(recHandle)))
    return(FAIL);

return(SUCCESS);
```

传输数据

Translate 函数实际上是把数据的传输任务交给 TranslateBuffer 去成的。这个函数的作用是把一个 ASCII 记录传送到 Paradox 的记录缓冲区中。

对于不同类型的 Paradox 字段，可以使用不同的 PXPut...函数。

PXPutAlpha 所使用的参数为：记录传输缓冲区、要修改的字段和要送入 Paradox 数据表的字符串。正如它的英文名称所表示的，这个函数的用途是把字符型的串送入字段。在本程序中，PXPutAlpha 需被调用两次，以便把两个字符串即顾客编号和部件编号送入记录传输缓冲区。

```
/* First the Customer Number */
if (Error(PXPutAlpha(recHandle, FieldCustNbr, buf->CustNbr)))
    return(FAIL);

/* Second the Part Number */
if (Error(PXPutAlpha(recHandle, FieldPartNbr, buf->PartNbr)))
    return(FAIL);
```

PXPutDoub 使用的参数与 PXPutAlpha 基本相同：一个记录句柄、记录中要修改的字段和要放进该字段的数据。唯一的区别是：该数据是 double（双精度）型的。

```
/* Quantity */
if (Error(PXPutDoub(recHandle, FieldQuantity, (double)
     buf->quantity))
    return(FAIL);
```

接下去，TranlateBuff 把数据从流式文件格式转换成 Paradox 格式。首先用 scanf 把装有日期值的字符串分解成三个整数：month、day 和 year。再用 PXDateEncode 把这三个整数值转换成 Paradox 的特殊长整数格式（从公元元年 1 月 1 日起至该日期的天数）。此函数的最后一个参数 PXDate 是一个长整数,用来存放转换后的日期值。

```
sscanf(buf->date, "%2d/%2d/%4d", &month, &day, &year);
if (Error(PXDateEncode(month, day, year, &PXDate)))
    return(FAIL);
```

然后，TranslateBuff 调用 PXPutDate。该函数把上述转换后的 Paradox 日期值送入记录传输缓冲区的指定字段中。

```
if (Error(PXPutDate(recHandle, FIELDDATE, PXDate)))
    return(FAIL);

return(SUCCESS);
```

关闭文件

CloseFields 调用 fclose 关闭 ASCII 流式文件并调用 PXTblClose 关闭 Paradox 数据表。

```
void CloseFiles(TABLEHANDLE tblHandle, FILE *fpAscii)
{
    if (fclose(fpAscii) == EOF)
        fprint(stderr, "cannot close ascii file\n");
    Error(PXTblClose(tblHandle));
}
```

最后，InitFields 用于初始化一个全局变量表（FieldPartNbr, FieldCustNbr,FieldQuantity 和 FieldDate）。这些变量分别代表相应字段的字段句柄。初始化是靠四次调用 PXFldHandle 来完成的，每次调用时传送一个字段名。

```
int InitFields(TABLEHANDLE tblHandle)
{
    if (Error(PXFldHandle(tblHandle, "Part Number", &FieldPartNbr)))
        return(FAIL);
    if (Error(PXFldHandle(tblHandle, "Cust Number", &FieldCustNbr)))
        return(FAIL);
    if (Error(PXFldHandle(tblHandle, "Quantity", &FieldQuantity))
        return(FAIL);
    if (Error(PXFldHandle(tblHandle, "Date", &FieldDate)))
        return(FAIL);
```