MATLAB-Serial Port Devices

MATLAB®

串口相关功能教程

Serial Port Devices R2012b

Devices connected to your computer's serial port

大家好

本文档是对 MathWorks 串口教程的翻译， 目的在于帮助大家学习 Matlab 串口设备。

教程包含以下 14 个部分，部分链接会直接指向 MathWorks 来源网页，请尽量适应英文。

水平有限，如有错误粗陋，请不吝指正。谢谢！

教程来源：http://www.mathworks.com/help/matlab/serial-port-devices.html

DSP Lab of Northeastern University

Dec, 2012 Boston, USA

如何使用

- 如果您是从零基础开始，既不了解 MATLAB 也从来没有用过串口通信，请在阅读本文档之前，先学习 MATLAB 基础知识和串口通信基础知识。
- 如果您已经具备串口和 MATLAB 的使用经验 ,想通过学习本文档来实现 MATLAB 对串口数据收发控制功能，请从“Getting Started with Serial I/O 开始串口工作”开始阅读。
- 如果您已经具备 MATLAB 串口基本开发能力，建议从“Events and Callbacks 事件和回调用”开始阅读。
- 您可以点击“on this page…本节包含的内容”框内的链接访问对应的英文原文。

目录：教程和实例

Examples and How To

介绍

Introduction

On this page… 本节包含内容：

What Is the MATLAB Serial Port Interface? 什么是 MATLAB 串口接口

MATLAB 串口接口提供了一个对外围设备的直接访问。比如连接在你的计算机上的 modem 打印机和其他技术设备。这个接口实际上建立在你的串口对象上，串口对象支持你的函数和功能，让你可以实现以下的操作：

- Configure serial port communications 完成串口的通信
- Use serial port control pins 使用串口控制指针
- Write and read data 读写数据
- Use events and callbacks 使用事件和回调
- Record information to disk 记录信息到磁盘

设备控制工具箱提供了一个增强的串口功能包，在命令行的基础上，它提供图形化的用户界面，叫做测试和测量工具。你可以用它来与你的串口设备连接，通信，配置，传输数据等操作，而不需要敲入代码。测试测量工具箱可以帮助你生成代码，让你用把代码用在那些需要代码的场合，比如 GUI 设计。工具箱还包含了增强的串口操作功能，包括串口对象的建立，配置，设备的通信等等。你可以用它来和 GPIB 或者 VISA 兼容的设备通信。

如果你想和 pc 兼容的数据采集硬件通信，比如多功能的 IO 板，你就需要数据采集工具箱软件。

你可以访问产品中心：http://www.mathworks.com/products.

Supported Serial Port Interface Standards 支持的串口标准

多年以来，有很多串口通信的标准，被开发出来。比如 RS-232, RS-422, and RS-485, 这些标准都是被 MATLAB 支持的。在他们当中最广泛使用的要数 RS232 标准。

这个向导文件 ,假设你使用的是 RS232 标准 ,在 “串口总览” 部分也默认使用 RS232 标准。

请参考你的计算机和设备文档，弄清楚你在使用哪种标准。

Supported Platforms 支持的平台

- Linux 32-bit
- Linux 64-bit
- Mac OS X
- Mac OS X 64-bit
- Microsoft Windows 32-bit
- Microsoft Windows 64-bit

Using the Examples with Your Device 使用你的设备的例程

这里我们提供的很多例程都是针对特定的外围设备的，我们使用的是 Tektronix® TDS 210 双通道示波器连接在 COM1 接口上，运行在 Windows 平台上，因此，很多文字的命令是针对这个平台和设备的。如果你在使用不同的平台或者使用不同的外围设备，或者使用不同的串口号，请按照你的设备修改例程。

串口总览

Overview of the Serial Port

On this page… 本节包含内容：

1.介绍

对于许多串口通信的应用来讲，你不需要知道串口具体的工作机制，就可以和目标设备进行通信。如果你对上述条目已经很熟悉，你不需要再阅读本章节，你可以直接阅读 ”Getting Started with Serial I/O 开始串口工作” 章节。

2 什么是串口通信

串口通信是适用于一个或多个设备之间的，底层通信协议。通常一个设备是一个计算机，其他设备是一个猫或者一个打印机或者计算机，或者科学技术外围设备，比如示波器，函数信号发生器等等。

此部分有电子或计算机基础的人都已经很熟悉，就暂时跳过这里，请百度或参考源网址：

http://www.mathworks.com/help/matlab/matlab_external/overview-of-the-serial-port.html

3 串口通信标准

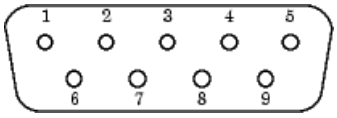
暂时跳过

4 用串口线连接两个设备

暂时跳过

5 串口信号和针脚分布

Serial ports consist of two signal types: data signals and control signals. To support these signal types, as well as the signal ground, the RS-232 standard defines a 25-pin connection. However, most Windows and UNIX[1] platforms use a 9-pin connection. In fact, only three pins are required for serial port communications: one for receiving data, one for transmitting data, and one for the signal ground.
The following diagram shows the pin assignment scheme for a 9-pin male connector on a DTE.



The pins and signals associated with the 9-pin connector are described in the following table. Refer to the RS-232 standard for a description of the signals and pin assignments used for a 25-pin connector.

Serial Port Pin and Signal Assignments

Pin	Label	Signal Name	Signal Type
1	CD	Carrier Detect	Control
2	RD	Received Data	Data
3	TD	Transmitted Data	Data
4	DTR	Data Terminal Ready	Control
5	GND	Signal Ground	Ground
6	DSR	Data Set Ready	Control
7	RTS	Request to Send	Control
8	CTS	Clear to Send	Control
9	RI	Ring Indicator	Control

The term *data set* is synonymous with *modem* or *device*, while the term *data terminal* is synonymous with *computer*.

Note The serial port pin and signal assignments are with respect to the DTE. For example, data is transmitted from the TD pin of the DTE to the RD pin of the DCE.

Signal States

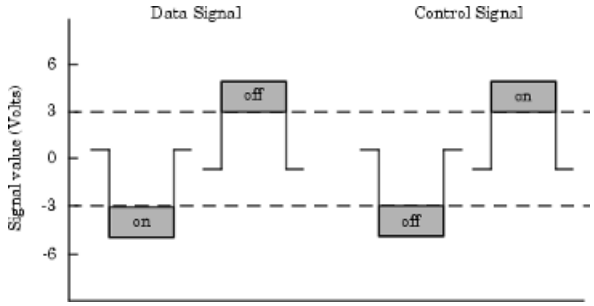
Signals can be in either an *active* state or an *inactive* state. An active state corresponds to the binary value 1, while an inactive state corresponds to the binary value 0. An active signal state is often

described as *logic 1*, *on*, *true*, or a *mark*. An inactive signal state is often described as *logic 0*, *off*, *false*, or a *space*.

For data signals, the `on` state occurs when the received signal voltage is more negative than -3 volts, while the `off` state occurs for voltages more positive than 3 volts. For control signals, the `on` state occurs when the received signal voltage is more positive than 3 volts, while the `off` state occurs for voltages more negative than -3 volts. The voltage between -3 volts and +3 volts is considered a transition region, and the signal state is undefined.

To bring the signal to the `on` state, the controlling device *unasserts* (or *lowers*) the value for data pins and *asserts* (or *raises*) the value for control pins. Conversely, to bring the signal to the `off` state, the controlling device asserts the value for data pins and unasserts the value for control pins.

The following diagram shows the `on` and `off` states for a data signal and for a control signal.



The Data Pins

Most serial port devices support *full-duplex* communication meaning that they can send and receive data at the same time. Therefore, separate pins are used for transmitting and receiving data. For these devices, the TD, RD, and GND pins are used. However, some types of serial port devices support only one-way or *half-duplex* communications. For these devices, only the TD and GND pins are used. This guide assumes that a full-duplex serial port is connected to your device.

The TD pin carries data transmitted by a DTE to a DCE. The RD pin carries data that is received by a DTE from a DCE.

The Control Pins

The control pins of a 9-pin serial port are used to determine the presence of connected devices and control the flow of data. The control pins include

- The RTS and CTS Pins
- The DTR and DSR Pins
- The CD and RI Pins

The RTS and CTS Pins. The RTS and CTS pins are used to signal whether the devices are ready to send or receive data. This type of data flow control—called *hardware handshaking*—is used to prevent data loss during transmission. When enabled for both the DTE and DCE, hardware handshaking using RTS and CTS follows these steps:

1. The DTE asserts the RTS pin to instruct the DCE that it is ready to receive data.
2. The DCE asserts the CTS pin indicating that it is clear to send data over the TD pin. If data can no longer be sent, the CTS pin is unasserted.
3. The data is transmitted to the DTE over the TD pin. If data can no longer be accepted, the RTS pin is unasserted by the DTE and the data transmission is stopped.

To enable hardware handshaking in MATLAB software, see Controlling the Flow of Data: Handshaking.

The DTR and DSR Pins. Many devices use the DSR and DTR pins to signal if they are connected and powered. Signaling the presence of connected devices using DTR and DSR follows these steps:

1. The DTE asserts the DTR pin to request that the DCE connect to the communication line.
2. The DCE asserts the DSR pin to indicate it is connected.
3. DCE unasserts the DSR pin when it is disconnected from the communication line.

The DTR and DSR pins were originally designed to provide an alternative method of hardware handshaking. However, the RTS and CTS pins are usually used in this way, and not the DSR and DTR pins. Refer to your device documentation to determine its specific pin behavior.

The CD and RI Pins. The CD and RI pins are typically used to indicate the presence of certain signals during modem-modem connections.

A modem uses a CD pin to signal that it has made a connection with another modem, or has detected a carrier tone. CD is asserted when the DCE is receiving a signal of a suitable frequency. CD is unasserted if the DCE is not receiving a suitable signal.

The RI pin is used to indicate the presence of an audible ringing signal. RI is asserted when the DCE is receiving a ringing signal. RI is unasserted when the DCE is not receiving a ringing signal (e.g., it is between rings).

6 串口数据格式

暂时跳过

7 为你的平台找到串口的信息

暂时跳过

8 使用虚拟 USB-串口 转接器

暂时跳过

9 参考书目

暂时跳过

开始串口工作

Getting Started with Serial I/O

On this page… 本节包含内容：

Example: Getting Started 例程：开始

这个例子包含了一些基本的串口命令，注意是基于 windows 平台的。如果你有一个连接在 COM1 上的串口设备，要设置波特率在 4800，可以使用以下的代码：

```
s = serial('COM1');
set(s,'BaudRate',4800);
fopen(s);
fprintf(s,'*IDN?')
out = fscanf(s);
fclose(s)
delete(s)
clear s
```

上面代码中 *IDN? 是向设备发送查询身份的信息，如果你使用的设备不支持，可能无法得到返回数据到 out 中 ,请根据你的设备进行相应修改。*IDN?是可编程设备标准命令(SCPI) 语言支持的命令，很多 modem 设备支持，参考你的设备文档，看是否支持 SCPI 语言。(译者按：如有疑问，请继续看串口会话部分，配置相应的参数。)

The Serial Port Session 串口对象会话

这一部分例程提供了一些你从头到尾都要用到的对串口会话的操作，这些步骤包括：

- 建立一个串口对象 -- 使用 `serial` 创建函数对一个特定的串口创建串口对象
  在这里，你可能要用到属性配置，比如波特率，数据位数等等。
- 对设备进行连接 – 要使用 `fopen` 函数来实现串口对象与设备的连接。当设备连接以后可以通过切换合适的值，读或写数据，来更换必要的设备。

- 配置属性 – 使用 `set` 或者 点符号 来建立想要的串口对象的行为属性值，实际上，你在建立对象时或者建立对象以后都可以以此来配置属性。当然，你也可以根据你的设备默认值，跳过串口对象的属性配置。
- 发送和接收数据 -- 发送和数据是通过函数 `fprintf` or `fwrite` 来实现的，接收数据是通过 `fgetl`, `fgets`, `fread`, `fscanf`, or `readasync` 来实现的。串口对象此时的行为都是基于前面的属性配置的。
- 断开连接和清空 -- 当你不再需要的串口对象的时候，断开连接，使用 `fclose function`，清空内存使用 `delete` 函数，从 MATLAB 工作区删除，使用 `clear` 命令。

以上的串口会话操作会经常使用，比如我们开篇的第一个例子。

Configuring and Returning Properties 配置和返回属性

这一个例程将会告诉你怎样显示串口的属性名称和属性值，并且告诉你怎么设置属性值。你可以显示或者设置属性，通过以下函数： `set` 函数， `get` 函数，或者点符号。

显示属性的名称和值：

在你建立了串口对象以后，使用 `set` 函数可在你的命令行窗口中显示所有的可以配置的属性值：

```
s = serial('COM1');
set(s)
    ByteOrder: [ {littleEndian} | bigEndian ]
    BytesAvailableFcn
    BytesAvailableFcnCount
    BytesAvailableFcnMode: [ {terminator} | byte ]
    ErrorFcn
    InputBufferSize
    Name
    OutputBufferSize
    OutputEmptyFcn
    RecordDetail: [ {compact} | verbose ]
    RecordMode: [ {overwrite} | append | index ]
    RecordName
    Tag
    Timeout
    TimerFcn
    TimerPeriod
    UserData
```

```
      SERIAL specific properties:
      BaudRate
      BreakInterruptFcn
      DataBits
      DataTerminalReady: [ {on} | off ]
      FlowControl: [ {none} | hardware | software ]
      Parity: [ {none} | odd | even | mark | space ]
      PinStatusFcn
      Port
      ReadAsyncMode: [ {continuous} | manual ]
      RequestToSend: [ {on} | off ]
      StopBits
   Terminator
```

使用 `get` 函数来显示你的窗口对象的当前属性值：

```
   get(s)
      ByteOrder = littleEndian
      BytesAvailable = 0
      BytesAvailableFcn =
      BytesAvailableFcnCount = 48
      BytesAvailableFcnMode = terminator
      BytesToOutput = 0
      ErrorFcn =
      InputBufferSize = 512
      Name = Serial-COM1
      OutputBufferSize = 512
      OutputEmptyFcn =
      RecordDetail = compact
      RecordMode = overwrite
      RecordName = record.txt
      RecordStatus = off
      Status = closed
      Tag =
      Timeout = 10
      TimerFcn =
      TimerPeriod = 1
      TransferStatus = idle
      Type = serial
      UserData = []
      ValuesReceived = 0
      ValuesSent = 0

      SERIAL specific properties:
```

```
    BaudRate = 9600
    BreakInterruptFcn =
    DataBits = 8
    DataTerminalReady = on
    FlowControl = none
    Parity = none
    PinStatus = [1x1 struct]
    PinStatusFcn =
    Port = COM1
    ReadAsyncMode = continuous
    RequestToSend = on
    StopBits = 1
    Terminator = LF
```

要得到某一个属性值，请把属性的名称提供给 `get` 函数：

```
get(s,'OutputBufferSize')
ans =
   512
```

显示多个属性的值：

```
get(s,{'Parity','TransferStatus'})
ans =
    'none'    'idle'
```

使用点符号来显示属性值：

```
s.Parity
ans =
none
```

设置属性的值：

你可以使用 `set` 函数来设置串口的属性：

```
set(s,'BaudRate',4800);
```

或者使用点号：

```
s.BaudRate = 4800;
```

要设置多个属性值，把多个属性的名称提交给 `set` 函数：

```
set(s,'DataBits',7,'Name','Test1-serial')
```

注意：如果你使用点符号，一次只能设置一个属性。

实际上很多属性是可以在对象创建以后随时都可以设置的，但是，有些属性在串口对象连接以后或者写入数据过程中，不能设置。至于哪些属性在哪些时候可以配置，请参考 Property Reference 属性参考部分。

指定属性名称

串口属性设置的时候可以使用多种名称，这有利于你的工作。你只需要唯一的使用他们，从而你可以缩写他们，例如，你可以把波特率配置写成下面的形式之一：

```
set(s,'BaudRate',4800)
set(s,'baudrate',4800)
set(s,'BAUD',4800)
```

当你在一个文件中包含属性名称，你应该使用全名。这可以避免 MATLAB 软件在以后的版本中，发布同样的缩写时产生问题。

默认的属性值

如果你不指定属性值的话，MATLAB 用它的时候，就会使用默认值。所有的属性配置，都有一个默认值。

注意：你的操作系统提供了一个串口默认属性值，但 MATLAB 会按照它的默认值覆盖之，但不会影响你的串口应用软件。如果你 set 函数里包含的定义字不够，它就会用{}标示出默认值。例如，校验位的属性默认是 None：

```
>> set(s,'Parity')
[ {none} | odd | even | mark | space ]
```

You can find the default value for any property in the property reference pages.

建立一个串口对象

Creating a Serial Port Object

On this page…

1.串口对象概览

建立串口对象的 `serial` 函数要求提供一个连接在你设备上的串口的名称,作为它的输入。

并且你可以在对象建立的过程中，配置它的属性，要想建立一个对应口子的对象，请输入：

```
s = serial('port');
```

这将建立一个对应于’ port’ 的串口对象。 如果 ‘port’ 在使用中，或者不存在，你将不能建立 ‘port’ 到你的设备。 ‘port’ 究竟是什么名字，要根据你的串口所在的平台来确定。设备控制工具箱（The Instrument Control Toolbox function）

```
instrhwinfo('serial')
```

提供了一个可用的串口的列表。以下是在不同平台上的串口对象构造器的例子。

Platform	Serial Constructor
Linux 32 and 64-bit	`serial('/dev/ttyS0');`
Mac OS X and Mac OS X 64-bit	`serial('/dev/tty.KeySerial1');`
MicrosoftWindows 32 and 64-bit	`serial('com1');`
Sun™Solaris™ 64-bit	`serial('/dev/term/a');`

现在，一个名为 s 的串口对象就建立在 MATLAB workspace 中了。你可以用 whos 命令查看 s 的类（class）.

```
>> whos s
   Name      Size         Bytes  Class

   s         1x1            512  serial object

Grand total is 11 elements using 512 bytes
```

Note: The first time you try to access a serial port in MATLAB using the s = serial('port') call, make sure that the port is free and is not already open in any other application. If the port is open in another application, MATLAB cannot access it. Once you have accessed in MATLAB, you can open the same port in other applications and MATLAB will continue to use it along with any other application that has it open as well.

注意： 当你第一次在 MATLAB 中建立 s 对象的时候，请确认串口没有被别的应用程序占用，如果占用了，MATLAB 是不能用的；但是，如果 MATLAB 先占用，其他应用程序依然可以访问这个串口， 他们可以共同使用。

一旦串口建立接下来的属性，就被分配值。根据串口对象的类型，分配通用属性值：

Property Name	Description
Name	Specify a descriptive name for the serial port object
Port	Indicate the platform-specific serial port name
Type	Indicate the object type

在 windows 平台，使用 get 函数来显示这些属性：

```
get(s,{'Name','Port','Type'})
ans =
    'Serial-COM1'    'COM1'    'serial'
```

创建对象的同时，配置对象的属性

你可以在创建对象的时候配置对象的属性， `serial` 将会接受你的属性名称和属性值就像你在设置函数里面配置属性一样，例如你可以分配属性名和属性值。

```
s = serial('port','BaudRate',4800,'Parity','even');
```

如果你分配了非法的属性名，这个对象将不会被创建。如果你分配了非法的值给某个属性

(for example, BaudRate is set to 50)，那么这个对象将会被创建，但是在的使用 fopen 连接的时候，将会出现问题。

串口对象的显示

串口对象提供给你了一个非常方便的信息显示功能，一个对于重要的配置和状态信息的总结。

你可以通过三条途径调用显示这些信息总结：

- Type the serial port object variable name at the command line.
- Exclude the semicolon when creating a serial port object.
- Exclude the semicolon when configuring properties using the dot notation.

你也可以在 workspace 浏览器中显示总结信息。用右键单击设备对象，然后在内容菜单中选择显示总结信息，display summary。

在 windows 平台上显示串口对象 s 的例子：

```
Serial Port Object : Serial-COM1

Communication Settings
   Port:               COM1
   BaudRate:           9600
   Terminator:         'LF'

Communication State
   Status:             closed
   RecordStatus:       off
Read/Write State
   TransferStatus:     idle
   BytesAvailable:     0
   ValuesReceived:     0
   ValuesSent:         0
```

创建一系列的串口对象

在 MATLAB 软件中，你可以通过吧变量们联系在一起，来从一个存在的变量开始，创建一

系列的串口对象。

假设你现在已经创建，两个 s1 和 s2 串口对象，在 Window 平台上：

```
s1 = serial('COM1');
s2 = serial('COM2');
```

你现在可以 MATLAB 语法创建一个串口对象序列，创建行向量 x：

```
x = [s1 s2]

Instrument Object Array

  Index:   Type:       Status:     Name:
  1       serial      closed      Serial-COM1
  2       serial      closed      Serial-COM2
```

或者列序列 y:

```
y = [s1;s2];
```

注意你不能创建有串口对象构成的矩阵。For example, you cannot create the matrix:

```
z = [s1 s2;s1 s2];
??? Error using ==> serial/vertcat
Only a row or column vector of instrument objects can be created.
```

根据你的应用，你也许有时候想要传递串口对象的一组属性值到函数。For example, to configure the baud rate and parity for s1 and s2 using one call to `set`:

```
set(x,'BaudRate',19200,'Parity','even')
```

连接设备

Connecting to the Device

在你开始使用串口对象来写和读数据之前，你必须要使用在 `serial` 函数中分配的串口来连接到你的设备。使用 `fopen` 函数来连接串口对象。

```
fopen(s)
```

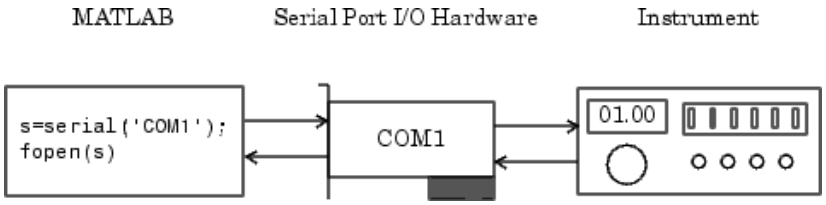
有些串口属性在串口对象连接以后，就不能被修改了，你必须要在使用 `fopne` 函数来连接之前就设置好他们。

Note You can create any number of serial port objects, but you can connect only one serial port object per MATLAB session to a given serial port at a time. However, the serial port is not locked by the session, so other applications or other instances of MATLAB software can access the same serial port, which could result in a conflict, with unpredictable results.

你可以通过检查 Status 来确认这个串口对象是不是连接在设备上：

```
s.Status
ans =
open
```

如下图所示，在设备和串口对象的连接时，数据是可以被写和读的：



配置通讯参数

Configuring Communication Settings

在你使用串口读写数据之前，串口对象和设备必须要有一致的通信设置。配置串口通信包含控制波特率和串口数据格式的一些特殊的属性值。以下的一个表格解释了这些属性：

Property Name	Description
BaudRate	Specify the rate at which bits are transmitted 波特率
DataBits	Specify the number of data bits to transmit 数据位数
Parity	Specify the type of parity checking 奇偶校验
StopBits	Specify the number of bits used to indicate the end of a byte 停止位
Terminator	Specify the terminator character 终结字符

Note If the serial port object and the device communication settings are not identical, data is not readable or writable. 不一致就没法读写。请查看你的设备文档，了解设备串口信息。

读写数据

Writing and Reading Data

On this page…

Before You Begin 开始之前

对于串口应用的开发，有三个重要问题，在你进行数据读写之前要考虑：

- 读写函数会不会阻碍对 MATLAB 命令行的访问？
- 传输的数据是文字 text 还是数字？
- 读写数据完成的条件是什么？

这三个问题在以下的“写数据”和“读数据”部分能找到解答。

Note: All the examples shown below are based on a Windows 32-bit platform. Refer to Overview of a Serial Port Object section for information about other platforms.

Example — Introduction to Writing and Reading Data 读写数据例程

假设你想从一个在 windows 平台上 COM1 连接的示波器返回身份信息，你需要用 `fprintf` 写入`*IDN?` 命令，然后用 `fscanf` 命令读出返回值。

```
s = serial('COM1');
fopen(s)
fprintf(s,'*IDN?')
out = fscanf(s)
```

结果，返回的身份信息是：

```
out =
TEKTRONIX,TDS 210,0,CF:91.1CT FV:v1.16 TDS2CM:CMV:v1.04
```

结束串口会话：

```
fclose(s)
delete(s)
clear s
```

Controlling Access to the MATLAB Command Line 对 MATLAB 命令行的访问控制

通过指定你的读写操作是*同步*的还是*异步的*，你可以控制对 MATLAB 命令行的访问。

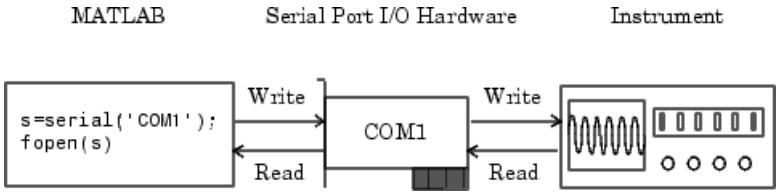
如果是一个同步的操作，那么他就会阻碍对 MATLAB 命令行的访问，直到读写函数完成操作。一个异步的操作不会阻碍命令行，在读写函数执行过程中，你可以执行附加的命令。同步和异步这两个词汇也经常被用在描述串口在硬件层的操作，RS-232 标准支持一个异步通信协议，使用这个协议，每个设备使用他们独自的内部时钟。使用每个字节的起始位来达到数据传输的同步，同时一位或者多位的停止位，来表示每一个字节的结束。For more information on start bits and stop bits, see Serial Data Format. RS-232 标准同样支持一个同步的模式，在这个模式下，所有的传输为都是同步在一个共同的时钟信号下。在硬件层，许多串口操作都是异步的。但是使用许多读写函数的默认行为，你就可以模拟一个同步串口的操作。

Note When used in this guide, the terms *synchronous* and *asynchronous* refer to whether read or write operations block access to the MATLAB command-line. In other words, these terms describe how the software behaves, and not how the hardware behaves. 这里使用同步异步，我们只关心它是否阻碍对 MATLAB 命令行的访问。也就是说，这两个词只描述软件行为，而非硬件行为。

异步读写的好处主要有以下两点：

- 你可以在读写函数执行过程中，执行其他的命令
- 你可以使用所有的支持的 callback 属性（see Events and Callbacks 查看“事件和回调”）

因为串口有不同的读写针脚，你可以同时读写数据，如下图所示：



Writing Data 写数据

这一节将使用三个部分介绍写数据到你的串口设备：

- The Output Buffer and Data Flow describes the flow of data from MATLAB software to the device. 输出缓存和数据流，描述了从 MATLAB 软件到设备的数据流。
- Writing Text Data describes how to write text data (string commands) to the device. 写文字数据，描述了怎样写文字数据（字符串命令）到设备。
- Writing Binary Data describes how to write binary (numerical) data to the device. 写二进制数据，描述了怎样写二进制数据（数字）到设备。

下面的表格描述了写数据时所涉及的函数：

Function Name	Description
fprintf	Write text to the device 写文字到设备
fwrite	Write binary data to the device 写数据到设备
stopasync	Stop asynchronous read and write operations 停止异步读写操作

下面的表格描述了写数据时所涉及的属性：

Property Name	Description
BytesToOutput	Number of bytes currently in the output buffer
OutputBufferSize	Size of the output buffer in bytes
Timeout	Waiting time to complete a read or write operation
TransferStatus	Indicate if an asynchronous read or write operation is in progress
ValuesSent	Total number of values written to the device

输出缓存和数据流

输出缓存是分配给串口对象的计算机内存，用来存储将要写入设备的数据。当输出数据写入你的设备时，数据流遵循以下两步：

- The data specified by the write function is sent to the output buffer. 分配给写数据函数的数据被发送到输出缓存。
- The data in the output buffer is sent to the device.输出缓存内的数据发送到设备。

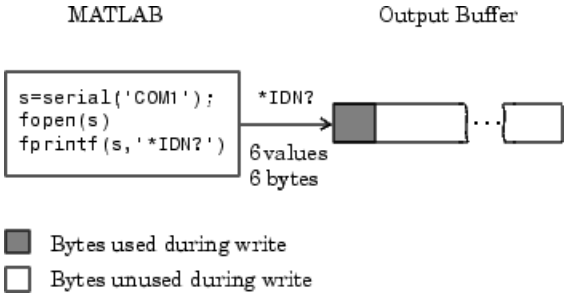
`OutputBufferSize` 属性指定了你要存放在输出缓存中的的数据的最大字节数。

`BytesToOutput` 属性表明了你现存在 output buffer 中的数据的字节数。The default values for these properties are:

```
s = serial('COM1');
get(s,{'OutputBufferSize','BytesToOutput'})
ans =
    [512]    [0]
```

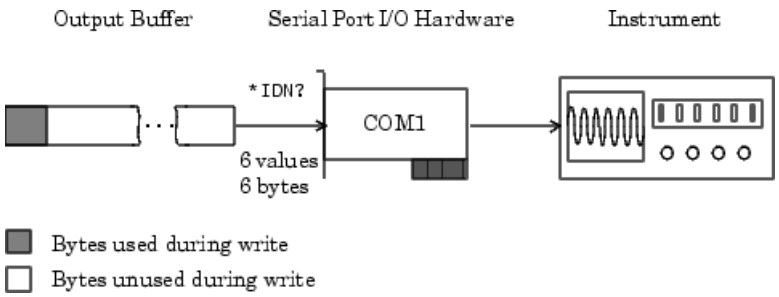
如果你超过了 output buffer 的大小限制，就不能写入数据，错误发生。

For example, suppose you write the string command `*IDN?` to the TDS 210 oscilloscope using the `fprintf` function. As shown in the following diagram, the string is first written to the output buffer as six values.



The `*IDN?` command consists of six values because the terminator is automatically written. Moreover, the default data format for the `fprintf` function specifies that one value corresponds to one byte. For more information about bytes and values, see Bytes Versus Values. `fprintf` and the terminator are discussed in Writing Text Data. 包含 6 个字节的原因是后面跟了一个停止字。 fprintf 指定了一个接一个的字节发送机制。查看更多，请见后面内容。

下面的例子展示了字符串输出到输出缓存，然后通过串口输出到设备：



写文字数据

你使用 `frintf` 函数来把文字数据写入设备。例如，对许多设备来说，写文字数据就是写字符串命令来改变设备的设置，以此来让设备准备返回数值，状态信息等等。

例如，`Display:Contrast` 命令改变了示波器的显示对比度：

```
s = serial('COM1');
fopen(s)
fprintf(s,'Display:Contrast 45')
```

在默认情况下，写函数 `fprintf` 写入数据使用了 `%s\n` 格式。因为很多串口设备，只接受这个文字基础的命令。当然，你也可以使用许多其他的格式，请参考 `fprintf` 网页。

你还可以检查发送了多少数据，通过调用 `ValuesSent` 属性：

```
s.ValuesSent
ans =
    20
```

注意，`ValuesSent` 属性的值中包含了停止字 Terminator，因为发送的命令中，每一个`\n` 的出现，都被替代成停止字的属性值（ `Terminator` property value. ）

```
s.Terminator
ans =
LF  %LF means Line Feedback
```

停止字属性的默认值就是换行符。你设备要求的停止字的值是什么，请查看设备文档。

同步和异步写操作：默认情况下 `fprintf` 函数是同步写入，并且阻碍命令行访问，直到执

行完毕。要想异步写入设备，你必须在 `printf` 的字符串中最后位置指定为 `async`:

```
fprintf(s,'Display:Contrast 45','async')
```

异步操作不会阻碍对 MATLAB 命令行的访问，并且，当一个异步写操作在进行时：

- 1. 可以执行一个异步读操作，因为串口有不同的针脚来完成读和写。
- 2. 可以使用所有支持的回调属性（callback properties）。

你可以使用 `TransferStatus` 属性来查看哪个异步操作是在进行中的。如果没有进行中的异步操作，`TransferStatus is idle.`

```
s.TransferStatus
ans =
idle
```

fprintf 的写操作的完成。使用 fprintf 的写操作在以下条件之一发生时，就完成：

- 数据写入完成。
- `Timeout` 属性指定的时间过了。

用 `stopasync` function 来停止一个异步写操作。

写停止字 `Terminator` 的规则：在 `cmd` 中，所有出现的 `\n` 都会被 `Terminator` 属性的值所替代。因此，当你使用默认的格式 `%s\n` 时，所有写入的命令结尾都要跟一个 `Terminator` 属性的值。参考你的设备文档，看要求什么样的停止字格式。

写二进制数据

使用 `fwrite` function 来写入二进制数据到你的设备。写入二进制数据就是指写入数字值。例如，一个典型的应用就是把二进制数据写入你的设备，比如写入一个任意波形发生器中。

Note Some serial port devices accept only text-based commands. These commands might use the SCPI language or some other vendor-specific language. Therefore, you might need to use the `fprintf` function for all write operations. 如果有些设备只接受文字的命令，比如使用 SCPI 或者厂商指定的语言，你就只能用 `fprintf` 函数来进行所有的写操作。

默认时，`fprintf` 函数使用 uchar 精度来传输数据，你也可以参考函数的说明，设定其他的精度。

默认时，`fwrite` 函数是同步操作的。如果要使用异步，你也需要在 `fwrite` 的末尾部分添加 `async` 参数。要查看更多关于同步异步的内容，请看**写文字数据**部分。关于 `fwrite` 在完成时的操作规则，请看 `fwrite` 的参考文档。

Troubleshooting Common Errors

Use this table to identify common `fprintf` errors.

Error	Occurs when	Troubleshooting
??? Error using ==> serial.fwrite at 199 OBJ must be connected to the hardware with FOPEN.	You perform a write operation and the serial port object is not connected to the device.	Use `fopen` to establish a connection to the device.
??? Error using ==> serial.fwrite at 199 The number of bytes written must be less than or equal to OutputBufferSize-BytesToOutput.	The output buffer is not able to hold all the data to be written.	Specify the size of the output buffer with the `OutputBufferSize`property.
??? Error using ==> serial.fwrite at 192 FWRITE cannot be called. The FlowControl property is set to 'hardware' and the Clear To Send (CTS) pin is high. This could indicate that the serial device may not be turned on, may not be connected, or does not use hardware handshaking	You set the `flowcontrol` property on a serial object to `hardware`. The device is either not connected or a connected device is not asserting that is ready to receive data.	Check your remote device status and flow control settings to see if hardware flow control is causing MATLAB errors.

Reading Data 读数据

这一节使用三个部分介绍如何通过串口从你的设备读出数据：

- The Input Buffer and Data Flow describes the flow of data from the device to MATLAB software.
  输入缓存和数据流，描述了从设备到 MATLAB 软件的数据流。

- Reading Text Data describes how to read from the device, and format the data as text. 读文字数据，描述了怎样从设备读取按文字格式对待的数据。
- Reading Binary Data describes how to read binary (numerical) data from the device. 读二进制数据， 描述了怎样从设备读取二进制（数值）数据。

以下表格展示了与读数据有关的函数：

Function Name	Description
fgetl	Read one line of text from the device and discard the terminator
fgets	Read one line of text from the device and include the terminator
fread	Read binary data from the device
fscanf	Read data from the device and format as text
readasync	Read data asynchronously from the device
stopasync	Stop asynchronous read and write operations

以下表格展示了与读数据有关的属性：

Property Name	Description
BytesAvailable	Number of bytes available in the input buffer
InputBufferSize	Size of the input buffer in bytes
ReadAsyncMode	Specify whether an asynchronous read operation is continuous or manual
Timeout	Waiting time to complete a read or write operation
TransferStatus	Indicate if an asynchronous read or write operation is in progress
ValuesReceived	Total number of values read from the device

输入缓存和数据流

输入缓存（Input Buffer）是串口对象分配的计算机内存，用来存储从设备读到的数据。当从你的设备读数据时，数据流遵循以下两步：

- 从设备读取的数据被存放在 Input buffer 输入缓存中。
- 输入缓存内的数据被返回到读操作函数指定的 MATLAB 变量中。

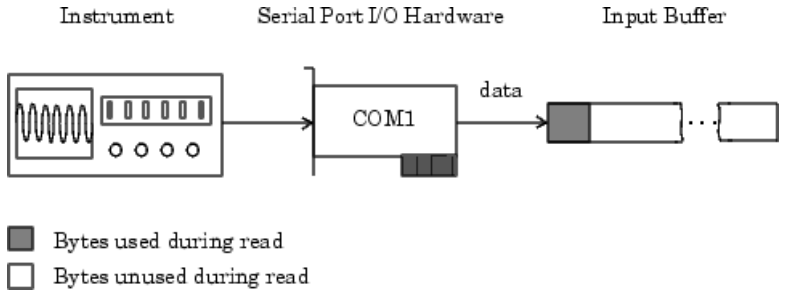
`InputBufferSize` 属性指定了你可以存放在输入缓存中的数据的最大字节数，`BytesAvailable` 属性指定了存放在输入缓存中的，你可以使用的数据的大小。

它们的默认值是：

```
s = serial('COM1');
get(s,{'InputBufferSize','BytesAvailable'})
ans =
    [512]    [0]
```

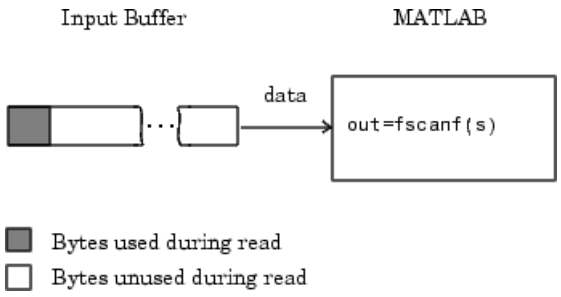
If you attempt to read more data than can fit in the input buffer, an error is returned and no data is read.

例如，假设你使用 `fscanf` 函数来读取一个先前发送给 TDS210 示波器的 "`*IDN?`" 命令的文字返回值，如下图所示，这个返回的文字数据首先通过串口被读入到输入缓存 input buffer 中。



注意，对于一个指定的读操作，你也许并不知道从设备返回上来的数据的字节数。因此，你需要在连接串口对象之前先给 `InputBufferSize` 属性配置一个足够大的值。

入下图所示，在数据存入输入缓存以后，它就被传输到 `fscanf` 函数指定的输出值中。



读文字数据

你将使用 `fgetl`, `fgets`, and `fscanf` 函数来从设备读取数据，并把数据按文字格式对待。

例如，你使用 `fscanf` 函数来读取一个先前发送给 TDS210 示波器的 “`*IDN?`” 命令的文字返回值：

```
s = serial('COM1');
fopen(s)
fprintf(s,'*IDN?')
out = fscanf(s)
out =
TEKTRONIX,TDS 210,0,CF:91.1CT FV:v1.16 TDS2CM:CMV:v1.04
```

默认时， `fscanf` 使用 `%c` 的格式来读取数据，因为许多串口设备返回的数据时基于文字格式的。However, you can specify many other formats as described in the `fscanf` reference pages.

你可以用 `ValuesReceived` 属性来查看读到了多大的数据：

```
s.ValuesReceived
ans =
    56
```

同步和异步读操作：通过 `ReadAsyncMode` 属性，你可以指定读操作采用同步或者异步的方式。`ReadAsyncMode` 属性值有 `continuous` 和 `manual`。如果 `ReadAsyncMode` is `continuous`(默认)， 串口对象连续查询设备，看数据是否可读。如果数据可用，它就被异步地存入输入缓存。要把输入缓存中的数据送到 MATLAB 中，用同步（阻塞）读函数来操作。比如， `fgetl` 或者 `fscanf`。如果数据在输入缓存中可用，这些函数可以很快地返回它们。

```
s.ReadAsyncMode = 'continuous';
fprintf(s,'*IDN?')
s.BytesAvailable
ans =
    56
out = fscanf(s);
```

如果 ReadAsyncMode 的值为 manual， 串口对象不连续查询设备。要异步地读数据，使用 readasync 函数。要把输入缓存中的数据送到 MATLAB 中，用同步读函数来操作。

```
s.ReadAsyncMode = 'manual';
fprintf(s,'*IDN?')
s.BytesAvailable
ans =
    0
readasync(s)
s.BytesAvailable
ans =
    56
out = fscanf(s);
```

异步操作不阻塞命令行，在异步读取数据时，你可以：

- 1. 可以执行一个异步写操作，因为串口有不同的针脚来完成读和写。
- 2. 可以使用所有支持的回调属性（callback properties）。

你可以使用 `TransferStatus` 属性来查看哪一个异步操作是在进行中。如果没有进行中的异步操作，`TransferStatus is idle.`

```
s.TransferStatus
ans =
idle
```

fscanf 函数读操作完成的规则：一个 fscanf 函数在读操作是将要阻塞对 MATLAB 命令行的访问，直到：

- The terminator specified by the `Terminator` property is read. 读到了 Terminator 属性指定的停止字
- The time specified by the `Timeout` property passes. Timeout 属性指定的超时时间到
- The specified number of values specified is read. 达到了指定的读入数据的个数
- The input buffer is filled. 输入缓存满了。

读二进制数据

用 `fread` 函数来从设备读入二进制数据，这意味着你将传递数值给 MATLAB。

例如，你想要返回光标，并且显示对示波器的设置。这需要写入 `CURSOR?` 和 `DISPLAY?` 命

令到设备中。如后读取这些命令的返回结果：

```
s = serial('COM1');
fopen(s)
fprintf(s,'CURSOR?')
fprintf(s,'DISPLAY?')
```

因为 ReadAsyncMode 属性的默认值是 continuous，一旦数据返回，就被异步地读入到输入缓存中。查看一下得到的数据大小：

```
s.BytesAvailable
ans =
    69
```

接下来，使用同步的读函数，来把这些值传输到 MATLAB 变量中。如果你使用的是 fgetl, fgets, or fscanf， 你需要调用两次，因为每个值的后面都跟随了一个停止字，输入缓存中有两个停止字。如果你使用 fread ，你只需要调用它一次就可以把所有数据读入 MATLAB 中。

```
out = fread(s,69);
```

默认下，fread 返回 double 精度的数列。但你也可以指定返回的精度类型，参考 fread 参考文档。你可以把二进制数据转换成文字数据，用 char 函数。

```
val = char(out)'
val =
HBARS;CH1;SECONDS;-1.0E-3;1.0E-3;VOLTS;-6.56E-1;6.24E-1
YT;DOTS;0;45
```

For more information about synchronous and asynchronous read operations, see Reading Text Data. For a description of the rules used by fread to complete a read operation, refer to its reference pages.

Example — Writing and Reading Text Data 例程：写读文字数据

这个例程描述了怎样和设备通过串口进行文字的读写操作。这里使用的是 Tektronix TDS 210 双通道示波器，它连接在 COM1 上。以下的很多命令都跟这个设备对应。一个正弦

波信号输入到示波器的频道 2，要做的是测量输入信号的峰峰值。

1. 建立串口对象，分配在 COM1 上。

```
s = serial('COM1');
```

2. 连接设备 – 因为 ReadAsyncMode 的默认值是 continuous ,一旦有数据可用就会被接收到输入缓存中。

```
fopen(s)
```

3. 写数据和读数据：

```
fprintf(s,'*IDN?')
idn = fscanf(s)
idn =
TEKTRONIX,TDS 210,0,CF:91.1CT FV:v1.16 TDS2CM:CMV:v1.04
```

先确定用的是哪个信号源：

```
fprintf(s,'MEASUREMENT:IMMED:SOURCE?')
source = fscanf(s)
source =
CH1
```

把信号源设定为 channel 2:

```
fprintf(s,'MEASUREMENT:IMMED:SOURCE CH2')
fprintf(s,'MEASUREMENT:IMMED:SOURCE?')
source = fscanf(s)
source =
CH2
```

让示波器测量峰峰值，并且返回数据：

```
fprintf(s,'MEASUREMENT:MEAS1:TYPE PK2PK')
fprintf(s,'MEASUREMENT:MEAS1:VALUE?')
```

用 fscanf 将数据从输入缓存读入 MATLAB

```
ptop = fscanf(s,'%g')
ptop =
2.0199999809E0
```

4. 断开连接 -- 当你不再使用串口对象，断开它和设备的连接，清空内存:`delete(s)`，并移除工作区（`clear s`）：

```
fclose(s)
delete(s)
clear s
```

Example — Parsing Input Data Using textscan 用 textscan 解析输入数据

这个例子描述了怎么使用 `textscan` 函数来获取从设备发送上来的带格式数据。当你想要把字符串数据放入一个或多个变量中时，每个变量都有特定的格式， `textscan` 就非常有用。这里使用的设备还是那个示波器。

1. 建立串口对象

```
s = serial('COM1');
```

2. 连接设备 – 因为 `ReadAsyncMode` 的默认值是 continuous，一旦有数据可用就会被接收到输入缓存中。

```
fopen(s)
```

3. 读写数据 – 用到 `fprintf` 和 `fscanf` 函数。此处 RS232? 是请求设备的 RS232 设置，返回波特率，软件流控制设置，硬件流控制设置，奇偶校验和停止字。

```
fprintf(s,'RS232?')
data = fscanf(s)
data =
9600;0;0;NONE;LF
```

使用 textscan 函数，可以分配 5 个变量到 5 个独立的新变量中：

```
C = textscan(data, '%d%d%d%s%s','delimiter',';');

[br, sfc, hfc, par, tm] = deal(C{:});

br =
```

```
        9600
   sfc =
        0
   hfc =
        0
   par =
       'NONE'
   tm =
       'LF'
```

4. 断开连接 – 当你不再使用串口对象，断开它和设备的连接，清空内存:`delete(s)`，并移除工作区（`clear s`）:

```
   fclose(s)
   delete(s)
   clear s
```

Example — Reading Binary Data 读二进制数据

这个例子描述了你怎样可以从 TDS210 示波器下载显示屏信息到 MATLAB。 使用 windows bitmap 格式，屏幕数据被传输保存到磁盘。这让你可以永久保存你的工作记录，让你容易记录重要的信号和显示参数。

由于传输的数据量可能会相对较大，并且是异步传输，一旦设备有数据可用就会读入数据到输入缓存。这也能让你在数据传输过程中进行别的工作。另外，此处把设备设置为了它最大波特率：19200.

1. 建立串口对象 – 建立一个跟 COM1 对应的串口对象：

```
   s = serial('COM1');
```

2. 配置参数 – 配置输入缓存的容量和波特率：

```
   s.InputBufferSize = 50000;
   s.BaudRate = 19200;
```

3. 连接设备 – 因为 `ReadAsyncMode` 的默认值是 `continuous`，一旦有数据可用就会

被接收到输入缓存中。

```
fopen(s)
```

4. 写数据和读数据 – 让示波器传输显示信息，存为 bitmap 格式：

```
fprintf(s,'HARDCOPY:PORT RS232')
fprintf(s,'HARDCOPY:FORMAT BMP')
fprintf(s,'HARDCOPY START')
```

等待数据传入输入缓存。然后把数据按照 unsigned-interger8 的格式传入 MATLAB：

```
out = fread(s,s.BytesAvailable,'uint8');
```

5. 断开连接 – 当你不再使用串口对象，断开它和设备的连接，清空内存 `delete(s)`，并移除工作区（`clear s`）：

```
fclose(s)
delete(s)
clear s
```

查看 bipmap 数据：

To view the bitmap data, follow these steps:

1. Open a disk file.
2. Write the data to the disk file.
3. Close the disk file.
4. Read the data into MATLAB using the `imread` function.
5. Scale and display the data using the `imagesc` function.

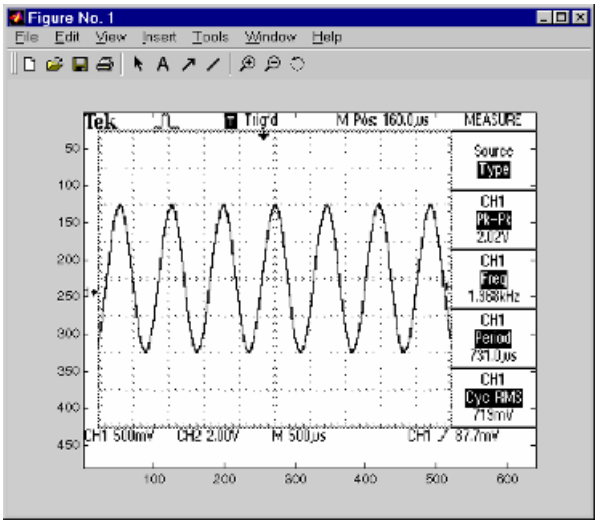
Note that the file I/O versions of the `fopen`, `fwrite`, and `fclose` functions are used.

```
fid = fopen('test1.bmp','w');
fwrite(fid,out,'uint8');
fclose(fid)
a = imread('test1.bmp','bmp');
imagesc(a)
```

Because the scope returns the screen display data using only two colors, an appropriate colormap is selected.

```
mymap = [0 0 0; 1 1 1];
colormap(mymap)
```

The following diagram shows the resulting bitmap image.



事件和回调用

Events and Callbacks

On this page…

Introduction 介绍

通过使用*事件*，你可以使你的串口操作具有更强大的功能和灵活性。当一个条件满足时，事件发生，并且这将会导致一个或多个回调 callback。

当串口对象连接到设备时，你可以使用事件来显示信息，显示数据，分析数据，等等。*回调属性*和*回调函数*控制回调行为。 所有的事件类型都有一个对应的回调属性。回调函数是你用来完成你特定功能的 MATLAB 函数。

当某个事件发生时，通过指定回调函数的名称作为对应回调属性的值，你可以执行一个回调。

Note: All examples in this section on based on a Windows 32-bit platform. For information about other platforms refer to Overview of a Serial Port Object. （本节是基于 32 位 windows 平台，关于其他平台，请参阅“串口总览”部分）

Example — Introduction to Events and Callbacks 例程，事件和回调介绍

这个例子使用了回调函数 instrcallback，当“字节可用”事件发生时，显示一条信息到命令行。当读取到停止字时，事件就被建立了。

```
s = serial('COM1');                    % create serial object s
```

```
fopen(s)                                    % connect device
s.BytesAvailableFcnMode = 'terminator'; % ByteAvailable condition
s
s.BytesAvailableFcn = @instrcallback;   % callback function
fprintf(s,'*IDN?')                          % send *IDN?
out = fscanf(s); % read data
```

MATLAB 中显示：

```
BytesAvailable event occurred at 17:01:29 for the object:
Serial-COM1.
```

结束串口会话：

```
fclose(s)
delete(s)
clear s
```

使用 type 命令，你可以看到内建函数 instrcallback 的代码。

Event Types and Callback Properties 事件类型和回调属性

在下面的表格中你将看到串口事件类型和回调属性，这个表格有 2 列和 9 行，左边第 1 行与右边 2，3，4 行对应，左边 Timer 与 8，9 行对应。

Event Type	Associated Properties
Break interrupt	BreakInterruptFcn
Bytes available	BytesAvailableFcn
	BytesAvailableFcnCount
	BytesAvailableFcnMode
Error	ErrorFcn
Output empty	OutputEmptyFcn
Pin status	PinStatusFcn
Timer	TimerFcn

Event Type	Associated Properties
	TimerPeriod

Break-Interrupt Event 断裂中断事件

当一个串口的断裂中断发生时，这个事件就被建立了。断裂中断是指，当接收的数据处于不活动状态的时间超过了一个字符传输的时间，串口就产生断裂中断。

This event executes the callback function specified for the `BreakInterruptFcn` property. It can be generated for both synchronous and asynchronous read and write operations. 这个事件回调的函数被指定在 `BreakInterruptFcn` 这个属性中。无论是同步异步读写操作，这个事件可以为它们建立。

Bytes-Available Event 字节可用事件

按照 `BytesAvailableFcnMode` 属性中的指定，当一定数量的字节在输入缓存中可用时，这个事件就立即被建立了。

- 如果 `ByteAvailableFcnMode` 的值是 `byte`， 当在输入缓存中的字节数达到 `BytesAvailableFcnCount` 指定的字节数时，字节可用事件就执行在 `BytesAvailableFcn` 属性中指定的回调函数。

- 如果 `BytesAvailableFcnMode` 的值是 `terminator`，当 `Terminator` 属性中分配的值每次被读到时，回调函数就执行。

这个“字节可用事件”只能在异步读的时候被建立。

Error Event 错误事件

当错误发生后，错误事件被立即建立。

这个事件执行的是 `ErrorFcn` 中指定的回调函数。它只能在异步读写操作时建立。当超时发生时，错误事件也会发生。当读写操作没有在 `Timeout` 属性中指定的时间内成功完成时，超时就发生了。
当配置错误发生时，错误事件不会建立。比如，配置了一个错误的属性值。

Output-Empty Event 空输出事件

当输出缓存被清空时，空输出事件发生。
这个事件执行 `OutputEmptyFcn` 属性中指定的回调函数，此事件只有在异步写操作时才能建立。

Pin Status Event 针脚状态事件

当针脚（CD，CTS，DSR，RI）的值发生变化时,一个针脚状态事件建立。这些针脚的详述，请见： Serial Port Signals and Pin Assignments.
这个事件会执行 `PinStatusFcn` 属性中指定的回调函数。这个时间可以在同步或异步读写时建立。

Timer Event 时间事件

当 `TimerPeriod` 属性中指定的时间到时，一个时间事件就被建立。这个时间是从串口对象与设备连接一刻开始算起。

时间事件调用在 `TimerFcn` 属性中指定的回调函数。注意，如果你的系统很慢了或者 `TimerPeriod` 的值太小了，有些时间事件则不会被处理。

Responding To Event Information 事件信息的回馈

在回调函数和记录文件中，你可以对一个事件信息进行反馈。在回调函数中，事件信息是用 `Type` 和 `Data` 两个域来存放的。类型域包含了事件类型，同时数据域包含了事件特定的信息。正如在建立和执行回调函数中描述的，这两个域对应了你在回调函数的函数头中定义的结构体。要想查看怎样在记录文件中记录事件数据信息和事件类型，请看“调试：将信息记录在磁盘上”。

以下表格展示了类型域和对应数据域的值：

Event Information 事件信息表格：

Event Type	Field	Field Value
Break interrupt	`Type`	`BreakInterrupt`
	`Data.AbsTime`	day-month-year hour:minute:second
Bytes available	`Type`	`BytesAvailable`
	`Data.AbsTime`	day-month-year hour:minute:second
Error	`Type`	`Error`
	`Data.AbsTime`	day-month-year hour:minute:second
	`Data.Message`	An error string
Output empty	`Type`	`OutputEmpty`
	`Data.AbsTime`	day-month-year hour:minute:second
Pin status	`Type`	`PinStatus`
	`Data.AbsTime`	day-month-year hour:minute:second
	`Data.Pin`	`CarrierDetect`, `ClearToSend`, `DataSetReady`, or `RingIndicator`
	`Data.PinValue`	`on` or `off`

Event Type	Field	Field Value
Timer	`Type`	`Timer`
	`Data.AbsTime`	day-month-year hour:minute:second

以下各条介绍了数据域的值的含义：

The AbsTime Field

The `AbsTime` field, defined for all events, is the absolute time the event occurred. The absolute time is returned using the `clock` format: day-month-year hour:minute:second.

The Pin Field

The pin status event uses the `Pin` field to indicate if the CD, CTS, DSR, or RI pins changed state. For a description of these pins, see Serial Port Signals and Pin Assignments.

The PinValue Field

The pin status event uses the `PinValue` field to indicate the state of the CD, CTS, DSR, or RI pins. Possible values are `on` or `off`.

The Message Field

The error event uses the `Message` field to store the descriptive message that is generated when an error occurs.

Creating and Executing Callback Functions 建立和执行回调函数

把文件名包含在对应的回调属性的值中，当指定的事件类型发生时，你就可以以此来使分配的回调函数执行。你可以分配回调函数作为一个函数句柄（functional handle）或者作为一个字符串单元阵列元素。函数句柄在 `function handle` 的参考文档中有详述。

例如，要在每次从设备读到停止字时都执行回调函数 mycallback ：

```
s.BytesAvailableFcnMode = 'terminator'; %byte number or terminato
r
s.BytesAvailableFcn = @mycallback; %call callback function
```

或者，你可以把回调函数作为一个阵列组来分配：

```
s.BytesAvailableFcn = {'mycallback'};
```

回调函数要求至少两个输入参数，第一个参数是串口对象，第二个参数是包含事件信息的变量。见“事件信息”表格。这个事件一定要是导致这个回调函数执行的那个事件。比如，回调函数 mycallback 的函数头为：

```
function mycallback(obj,event) %
function mycallback(s,BytesAvailable)
```

通过把回调函数和参数作为阵列组元素来包含，你就可以传递附加的参数到回调函数。例如，传递 MATLAB 的 `time` 变量到 `mycallback` 这个回调函数：

```
time = datestr(now,0); %return string of date, 0 is a format para
s.BytesAvailableFcnMode = 'terminator';
s.BytesAvailableFcn = {@mycallback,time}; % element of cell array
```

也可以用下面的字符串方式来分配回调函数：

```
s.BytesAvailableFcn = {'mycallback',time};
```

此时对应的函数头就应该是：

```
function mycallback(obj,event,time)
```

如上所示，如果你传递了附加的参数进入回调函数，那么这个参数在函数头就要跟在两个必须参数的后面。

Note You can also specify the callback function as a string. In this case, the callback is evaluated in the MATLAB workspace and no requirements are made on the input arguments of the callback function.

Enabling Callback Functions After They Error 错误发生后重新使能回调函数

如果在回调函数执行过程中发生了一个错误，将会发生以下的情况：

- 回调函数将自动被使无能
- 命令行窗口输出错误提示，说回调函数被使无能了。

如果你想重新使能回调函数，把回调的属性设成同样的值 set the callback property to the same value。或者，用 `fclose` 函数断开对象的连接。如果你想用一个不同的回调函数，

当你在配置回调属性到新值的时候，回调函数就被自动使能了。

Example — Using Events and Callbacks 例程，使用事件和回调

当一个字节可用事件或者一个输出空事件发生时，这个例程使用回调函数 instrcallback 来显示事件相关信息到命令行。

1. 建立一个串口对象 -- s 对应于 COM1

```
s = serial('COM1');
```

2. 配置属性 -- 当一个“字节可用”事件或者一个“输出空”事件发生时，执行 instrcallback 函数。因为 instrcallback 函数要求串口对象和事件信息作为输入参数，用“函数句柄”模式指定回调函数即可。

```
s.BytesAvailableFcnMode = 'terminator';
s.BytesAvailableFcn = @instrcallback;
s.OutputEmptyFcn = @instrcallback;
```

3. 连接到设备 -- 因为 ReadAsyncMode 属性的默认值是 continuous，一旦有了数据，就异步返回到输入缓存。

```
fopen(s)
```

4. 写读数据

```
fprintf(s,'RS232?','async')
```

当 RS232? 命令发送以后，以及当收到停止字时，instrcallback 被调用。

```
OutputEmpty event occurred at 17:37:21 for the object:
Serial-COM1.

BytesAvailable event occurred at 17:37:21 for the object:
Serial-COM1.
```

从输入缓存读入数据：

```
out = fscanf(s)
out =
9600;0;0;NONE;LF
```

5. 断开和清空 — When you no longer need s, disconnect it from the instrument and remove it from memory and from the MATLAB workspace.

```
fclose(s)
delete(s)
clear s
```

译者按：此处没有提供回调函数 `instrcallback` 的代码，导致一个疑问，怎样让回调函数对两个不同的事件都作出反应？

使用控制针脚

Using Control Pins

On this page… 本节包含的内容：

Properties of Serial Port Control Pins 串口控制指针的属性

正如“串口信号和针脚分布”中描述的那样，9 针串口包含 6 个控制针脚。下表展示了串口控制针脚对应的属性。

Control Pin Properties

Property Name	Description
DataTerminalReady	State of the DTR pin
FlowControl	Data flow control method to use
PinStatus	State of the CD, CTS, DSR, and RI pins
RequestToSend	State of the RTS pin

Signaling the Presence of Connected Devices 存在连接设备的信号

DTE 和 DCEs 经常使用 CD,DSR,RI 和 DTR 针脚来表示一个设备间的串口连接是否被建立起来。一旦建立起来，你就可以写读数据了。

你可以用 PinStatus 属性来监视 CD,DSR and RI 针脚，用 DataTerminalReady 属性来指派和监视 DTR 针脚的状态。

接下来的例子说明了当 2 个 Modem 连接在一起时，这些针脚是怎样被利用的。

注意：本节所有的例子都是在 32 位 windows 平台上进行的。要得到关于其它平台的更多信息，请参阅“串口对象概览”。

例程 — 连接两个 Modems

这个例程通过同一台电脑连接两个 Modem，解释了你怎样能够监视电脑-Modem 之间的通信状态，以及 Modem-Modem 之间的通信状态。第一个 Modem 连接在 COM1 上，第二个 Modem 连接在 COM2 上。

1. 建立串口对象 -- 在 Modems 上电以后，串口对象 s1 是为第一个 Modem 建立的，s2 是为第二个 Modem 建立的。

```
s1 = serial('COM1');
s2 = serial('COM2');
```

2. 连接设备 -- s1 and s2 连接到 Modem 上。由于 ReadAsyncMode 属性的默认值是 continuous，一旦数据从 Modem 端有效，它们就会被异步接收到输入缓存。

```
fopen(s1)
fopen(s2)
```

因为 DataTerminalReady 属性的默认值是 on，电脑（数据终端）现在已经准备好和 Modem 交换数据。你可以通过 PinStatus 属性来检查 Data Set Ready 针脚，从而检查 Modems（数据集）是否可以和电脑通信。

```
s1.Pinstatus
ans =
    CarrierDetect: 'off'
    ClearToSend: 'on'
    DataSetReady: 'on'
    RingIndicator: 'off'
```

`DataSetReady` 域的值是 `on` ，因为两个 Modem 在连接到对象之前都已上电

3. 配置属性 -- 两个 Modems 都配置在 2400 波特率，停止字为 carriage return(CR).

```
s1.BaudRate = 2400;
s1.Terminator = 'CR';
s2.BaudRate = 2400;
s2.Terminator = 'CR';
```

写数据，读数据 -- 写 `atd` 命令到 1st modem，这个命令把 modem 置为"off the hook"，跟手动提起电话听筒一样的作用。

```
fprintf(s1,'atd')
```

写 `ata` 命令到第二个 modem, 这个命令把 modem 置为应答状态，让它和 1st modem 连接在一起。

```
fprintf(s2,'ata')
```

在两个 modems 连接以后，通过检查 Carrier Detect 针脚，查看 `PinStatus` 属性，确认连接状态。

```
s1.PinStatus
ans =
    CarrierDetect: 'on'
      ClearToSend: 'on'
     DataSetReady: 'on'
    RingIndicator: 'off'
```

确认 modem-modem 连接状态：读取 2nd modem 返回的描述信息。

```
s2.BytesAvailable
ans =
    25
out = fread(s2,25);
char(out)'
ans =
ata
CONNECT 2400/NONE
```

4. 现在要断开两个 modems 之间的连接，配置 `PinStatus` 属性到 off,你可以检查

Carrier Detect 针脚的值来确定 modems 之间是否已经断开。

```
s1.DataTerminalReady = 'off';   % configuration
s1.PinStatus
ans =
    CarrierDetect: 'off'        % verifying, off means discon..
      ClearToSend: 'on'
     DataSetReady: 'on'
    RingIndicator: 'off'
```

5. 断开连接和清空 -- 断开串口对象和 modem 的连接，移除内存占用，以及清空 MATLAB workspace.

```
fclose([s1 s2])
delete([s1 s2])
clear s1 s2
```

Controlling the Flow of Data: Handshaking 控制数据流：握手

数据流控制或 handshaking 握手 是一个用来在 DCE 和 DTE 之间通信时，防止数据传输丢失的方法。例如，假设你的电脑只能接受一定数量的数据，然后必需要处理它们。这就到达一个数据上限 ,此时握手信号传输到 DCE 来停止发送数据。当电脑又可以接收数据时，另一个握手信号传输到 DCE，恢复数据传输。

如果你的设备支持，你可以使用这些方法来进行数据流控制：

- Hardware handshaking
- Software handshaking

注意: 尽管你可以配置你的设备使其同时软件硬件握手 ,但是 MATLAB 不支持这样的行为。

你可以使用 `FlowControl` 属性来指配数据流控制方法。如果 `FlowControl` 是 `hardware`, 就用硬件握手来控制。 如果 `FlowControl` 是 `software`,就用软件。如果

是 none，就没用握手。

硬件握手：

硬件握手时用特定串口的针脚来控制数据流。在多数情况下，它们是 RTS 和 CTS 针脚。

Hardware handshaking using these pins is described in The RTS and CTS Pins.

如果 FlowControl 属性是 hardware，RTS 和 CTS 针脚就自动被 DTE 和 DCE 管理. 你可以通过 PinStatus 属性返回 CTS 针脚的值。通过 RequestToSend 属性，配置和返回 RTS 针脚。

Note Some devices also use the DTR and DSR pins for handshaking. However, these pins are typically used to indicate that the system is ready for communication, and are not used to control data transmission. In MATLAB, hardware handshaking always uses the RTS and CTS pins.

如果你的设备不适用标准的硬件握手，你可能需要手动配置 RequestToSend 属性. 在这种情况下，你应该配置 FlowControl 属性到 none. 如果 FlowControl 是 hardware, 你分配的 RequestToSend 属性值可能不会被关注，参考你的设备文档，查看特殊的针脚行为。

软件握手：

软件握手使用特定 ASCII 字串来控制数据流。下表展示了 Xon 和 Xoff(or XON and XOFF) 这两个字串。

Software Handshaking Characters

Character	Integer Value	Description
Xon	17	Resume data transmission
Xoff	19	Pause data transmission

使用软件握手时，控制字符串像发送正常数据一样传输。因此，需要使用 TD,RD,和 GND 针脚。

软件握手的主要弊端是，在数字量信号写入设备时，握手信号 Xon 或 Xoff 不能被写入。因为数字量可能会包含 17 和 19，这就没法区别究竟是控制字串还是数据。但是，你可以在异步读数据时写入 Xon 或 Xoff, 因为可以同时使用 TD 和 RD 针脚。

例程： 使用软件握手

假设你想用流控制来控制 例程 -- 读取二进制数据 。你需要配置示波器串口对象到“软件流控制”。

```
fprintf(s,'RS232:SOFTF ON')
s.FlowControl = 'software';
```

写 19 到设备，停止数据传输：

```
fwrite(s,19)
```

写 17 到设备，恢复数据传输：

```
fwrite(s,17)
```

调试：将信息记录在磁盘上

Debugging: Recording Information to Disk

On this page… 包含的内容：

Introduction 介绍

把信息记录在磁盘可以让你把串口会话永久记录，也是个简单的调试你应用的方法。在你的串口对象和设备连接以后，你可以记录以下信息到磁盘文件：

- 写入或从设备读取的数据量，以及数据的类型
- 写入或从设备读取的数据
- 事件信息

Recording Properties 记录属性

用 record 函数，可把信息记录在磁盘文件上。下面的表格展示了与记录信息到磁盘文件的操作有关的属性。

Recording Properties

Property Name	Description
RecordDetail	Amount of information saved to a record file
RecordMode	Specify whether data and event information is saved to one record file or to multiple record files
RecordName	Name of the record file

Property Name	Description
RecordStatus	Indicate if data and event information are saved to a record file

Note: All examples in this section are based on a Windows 32-bit platform. For more information about other supported platforms, refer to Overview of a Serial Port Object.

Example: Introduction to Recording Information 例程：记录信息的介绍

这个例程记录了写入设备的值的数量，并写入 myfile.txt 文件中。

```
s = serial('COM1');
fopen(s)
s.RecordName = 'myfile.txt';
record(s)
fprintf(s,'*IDN?')
idn = fscanf(s);
fprintf(s,'RS232?')
rs232 = fscanf(s);
```

结束串口：

```
fclose(s)
delete(s)
clear s
```

你可以使用 type 命令，在命令行窗口显示 myfile.txt

Creating Multiple Record Files 建立多个记录文件

当你使用 record 函数来记录信息， RecordMode 属性决定了是新建一个的记录文件还是新的信息知识附加在一个记录文件之后。

你可以配置 RecordMode 到 overwrite, append, 或者 index. 如果 RecordMode 是 overwrite, 每次声明记录时，就会覆盖以前的记录文件。 如果是 append ，新信

息就会附加到 RecordName 指定的文件的后面。如果是 index ，每次声明记录时就会新建一个记录文件。指定一个记录文件名的规则将会在下一部分讨论。

Specifying a Filename 分配文件名

用 RecordName 属性来指派记录文件的文件名。RecordName 可以赋予任何值，包括你的操作系统提供的文件路径。并且 ,如果 RecordMode 是 index, 文件名将遵循以下规则：

- 按索引编号的文件名是按数字分配的，数字被放在文件名的后缀名之前，每次增加 1.
- 如果指派的第一个记录文件文件名中没有包含数字，如 myfile.txt，则第二个记录文件为：myfile01.txt
- RecordName 在每次记录文件关闭后都被更新。
- 如果指定的文件名已经存在，原先的同名文件将被覆盖。

The Record File Format 记录文件的格式

记录文件是 ASCII 文件，它记录了一个或多个串口会话。利用 RecordDetail 属性来指定保存在记录文件的信息量。

RecordDetail 可以为 compact or verbose. 一个 compace 记录文件包含：写入设备的值数量，读出的值数量，值的数据类型，和事件信息。 verbose 记录文件不仅包含了前面的信息，还包含了传输的数据。

由 uchar, schar, (u)int8, (u)int32 指定的数据格式的二进制数据被记录成 16 进制形式。例如如果整数值 255 按照 16 位整数读入，它的 16 进制形式 00FF 就被记录在记录文件。Single- and double-precision floating-point numbers are recorded as decimal values using the %g format, and as hexadecimal values using the format specified by the IEEE® Standard 754-1985 for Binary Floating-Point Arithmetic.使

用 %g 格式，单精度和双精度浮点数按照十进制的值来记录。如果用 IEEE 标准 754-1985 二进制浮点结构指定的格式，也可以存成 16 进制。（confusing）

IEEE 浮点格式包含三个构成元素 :符号位 sign bit ,指数域 exp field 和 significant field，单精度浮点值有 32 位，下图展示了它们的值：

$$value = (-1)^{sign}(2^{exp-127})(1.significand)$$

双精度浮点值有 64 位，如图：

$$value = (-1)^{sign}(2^{exp-1023})(1.significand)$$

浮点格式的构成元素和对应的单双精度位，见表：

Component	Single-Precision Bits	Double-Precision Bits
sign	1	1
exp	2–9	2–12
significand	10–32	13–64

左起是第一位。

Example: Recording Information to Disk 例程：记录信息到磁盘

这个例子说明了怎样记录在串口对象和 Tektronix TDS 210 示波器之间传输的信息到文件。后面给出了建立的记录文件结构。

1. 建立串口对象

```
s = serial('COM1');
```

2. 连接设备 – 因为 ReadAsyncMode 的默认值是 continuous ,一旦有数据可用就会被接收到输入缓存中。

```
fopen(s)
```

3. 配置属性值 -- 配置 s 到记录信息中，多文件，verbose 格式，初始文件名被定义为 WaveForm1.txt

```
s.RecordMode = 'index';
s.RecordDetail = 'verbose';
s.RecordName = 'WaveForm1.txt';
record(s)
```

4. 写数据和读数据 -- 写入设备的命令和从设备读回的数据都被记录在记录文件中。对于设备的命令解释，请参阅 “例程 -- 写读文字数据”

```
fprintf(s,'*IDN?')
idn = fscanf(s);
fprintf(s,'MEASUREMENT:IMMED:SOURCE CH2')
fprintf(s,'MEASUREMENT:IMMED:SOURCE?')
source = fscanf(s);
```

用 fread 函数来读取 peak-to-peak voltage，注意 fread 返回的数据是用的 16 进制格式。

```
fprintf(s,'MEASUREMENT:MEAS1:TYPE PK2PK')
fprintf(s,'MEASUREMENT:MEAS1:VALUE?')
ptop = fread(s,s.BytesAvailable);
```

把 peak-to-peak voltage 转换成字符串形式：

```
char(ptop)'
ans =
2.0199999809E0
```

记录的状态从 on 转换到 off，因为 RecordMode 属性的值是 index，记录的文件

名也自动更新。

```
record(s)
s.RecordStatus
ans =
off
s.RecordName
ans =
WaveForm2.txt
```

5. 断开连接 -- 当你不再使用串口对象，断开它和设备的连接，清空内存:delete(s), 并移除工作区（clear s）:

```
fclose(s)
delete(s)
clear s
```

记录文件的内容

这里列出了文件 WaveForm1.txt 的内容。因为 RecordDetail 属性是 verbose，收发值的多少，命令和数据都被记录。注意 fread 函数返回的值是 16 进制格式的。

```
type WaveForm1.txt

Legend:
  * - An event occurred.
  > - A write operation occurred.
  < - A read operation occurred.
1     Recording on 22-Jan-2000 at 11:21:21.575. Binary data in...
2    > 6 ascii values.
       *IDN?
3    < 56 ascii values.
       TEKTRONIX,TDS 210,0,CF:91.1CT FV:v1.16 TDS2CM:CMV:v1.04
4    > 29 ascii values.
       MEASUREMENT:IMMED:SOURCE CH2
5    > 26 ascii values.
       MEASUREMENT:IMMED:SOURCE?
6    < 4 ascii values.
       CH2
```

```
7    > 27 ascii values.
       MEASUREMENT:MEAS1:TYPE PK2PK
8    > 25 ascii values.
       MEASUREMENT:MEAS1:VALUE?
9    < 15 uchar values.
       32 2e 30 31 39 39 39 39 39 38 30 39 45 30 0a
10     Recording off.
```

保存和装载

Saving and Loading

On this page... 本节包含的内容：

Using save and load 使用保存和装载

使用 `save` 命令，你可以保存串口对象到文件，就像你保存任何 workspace 的变量一样。

例如，假设你创建了一个与串口 COM1 关联的串口对象，配置了几个属性值，执行了数据写和读操作：

```
s = serial('COM1');
s.BaudRate = 19200;
s.Tag = 'My serial object';
fopen(s)
fprintf(s, '*IDN?')
out = fscanf(s);
```

要保存你的串口对象及其内含的值到文件 `myserial.mat`，使用：

```
save myserial s out  % obj-s and variable - out
```

注意，你可以使用 record 函数把数据和事件信息按照文字保存到磁盘文件。

此时，可以用 load 命令重建 `s` 和 `out` 到 workspace 中：

```
load myserial
```

载入时，只读属性的值将被恢复到它们的默认值。例如，`Status` 属性就被恢复到 `closed`.

因此，要使用 `s`，你还需要用 `fopen` 函数重新连接串口对象到设备。要查看一个属性是否只读，请参阅属性的参考文档。

Using Serial Port Objects on Different Platforms 在不同平台上使用串口对象

你要在不同平台之间使用保存的串口对象，你需要修改 `Port` 属性的值。例如，在 Microsoft Windows 平台上建立了一个串口对象 `s` 关联在 COM1 上。如果你想保存 `s` ，然后把它用在 Linux 平台上，请设置 `Port` 属性的值到符合新平台的值，此处，在对象加载以后，把它设为 `ttyS0` 即可。

断开和清空

Disconnecting and Cleaning Up

On this page... 本节包含的内容：

Disconnecting a Serial Port Object 断开串口对象连接

当你不再需要和设备通信，用 fclose 函数断开它和串口对象的连接。

```
fclose(s)
```

用 Status 属性来检查串口对象与设备是否断开连接：

```
s.Status
ans =
closed
```

当 fclose 执行以后，分配给 s 的串口就重新可用。你可以使用 fopen 连接它到其他的串口对象上。

Cleaning Up the MATLAB Environment 清空 MATLAB 环境

当串口对象不再被需要，可用 `delete` 函数来使它从内存移除。

```
delete(s)
```

在使用 delete 之前，要用 `fclose` 函数来断开它和设备的连接。

一个已经被删除的串口对象是 *非法的* ，意味着你不能用它再连接设备。这种情况下，把它从 MATLAB workspace 中移除吧。 要把它从工作区中移除，请使用 `clear` 命令:

```
clear s
```

若对一个仍然连接在设备上的串口对象运用 `clear` 命令，可把它从工作区移除，但它还是保持连接。此时可用 `instrfind` 函数重新装载一个已经被清空的对象，使它回到MATLAB。

属性参考

Property Reference

On this page... 本节包含内容：

The Property Reference Page Format 属性参考页格式

每一个属性的参考解释都包含了以下条目中的全部或部分：

- 属性名称
- 属性描述
- 属性的特点，包括：
  - 只读 -- 属性是只读的条件

    一个属性可能总是只读的，也可能一直不是只读的，也可能只在串口对象打开时只读，也可能在串口对象记录数据时时只读的。你可以用 set 函数来配置属性的值。你也可以用 get 函数或点符号来返回属性的值。
  - 数据类型 -- 属性的数据类型

    这是你设定属性值时使用的数据的类型。
- 合法的属性值，包括默认属性值

  当用一个预先定义的列表给出属性值时，默认的属性值通常都用{}标示出来。
- 使用这个属性的例子
- 与它有关的属性和函数

Serial Port Object Properties 串口对象属性

下面大致地描述了串口对象属性，并且把它们按用途进行的分类。

在下一节，这些属性将按照字母表顺序排列，并且可以通过链接查看详细的属性参考文档。

Communications Properties	
BaudRate	Rate at which bits are transmitted
DataBits	Number of data bits to transmit
Parity	Type of parity checking
StopBits	Number of bits used to indicate the end of a byte
Terminator	Terminator character

Write Properties	
BytesToOutput	Number of bytes currently in the output buffer
OutputBufferSize	Size of the output buffer in bytes
Timeout	Waiting time to complete a read or write operation
TransferStatus	Indicate if an asynchronous read or write operation is in progress
ValuesSent	Total number of values written to the device

Read Properties	
BytesAvailable	Number of bytes available in the input buffer
InputBufferSize	Size of the input buffer in bytes
ReadAsyncMode	Specify whether an asynchronous read operation is continuous or manual
Timeout	Waiting time to complete a read or write operation
TransferStatus	Indicate if an asynchronous read or write operation is in progress
ValuesReceived	Total number of values read from the device

Callback Properties	
BreakInterruptFcn	Callback function to execute when a break-interrupt event occurs
BytesAvailableFcn	Callback function to execute when a specified number of bytes is available in the input buffer, or a terminator is read
BytesAvailableFcnCount	Number of bytes that must be available in the input buffer to generate a bytes-available event
BytesAvailableFcnMode	Specify if the bytes-available event is generated after a specified number of bytes is available in the input buffer, or after a terminator is read
ErrorFcn	Callback function to execute when an error event occurs
OutputEmptyFcn	Callback function to execute when the output buffer is empty
PinStatusFcn	Callback function to execute when the CD, CTS, DSR, or RI pins change state
TimerFcn	Callback function to execute when a predefined period of time passes
TimerPeriod	Period of time between timer events

Control Pin Properties	
DataTerminalReady	State of the DTR pin
FlowControl	Data flow control method to use
PinStatus	State of the CD, CTS, DSR, and RI pins
RequestToSend	State of the RTS pin

Recording Properties	
RecordDetail	Amount of information saved to a record file
RecordMode	Specify whether data and event information are saved to one record file or to multiple record files
RecordName	Name of the record file
RecordStatus	Indicate if data and event information are saved to a record file

General Purpose Properties	
ByteOrder	Order in which the device stores bytes
Name	Descriptive name for the serial port object
Port	Platform-specific serial port name
Status	Indicate if the serial port object is connected to the device
Tag	Label to associate with a serial port object
Type	Object type
UserData	Data you want to associate with a serial port object

属性 – 按字母表排序

Properties — Alphabetical List