Last Day talked about how to open an RS232 style connection:

```
Err SrmOpen(UInt32 port, UInt32 baud,
            UInt16* newPortID)
```

- newPortID ← receive port
- port ← can only have one at a time
- 0 if success
  - serErrAlreadyOpen if already open

Err SrmBackground (same prototype) ← more than one but only receive data (Ex Keyboards)

For USB & Bluetooth connections use;

```
Err SrmExtOpen(UInt32 port, SrmOpenConfigType *configP,
               UInt16 configSize, UInt16 *newPortIdP)
```

configP has members
- baud (ignored by USB)
- function (creatorID) → or serFncUndefined, " PPPSession, " SlipSession, Hotsync, Comlib, Telephony, Debugger
- drvrDataP
- drvrDataSize
- sysReserved1
- sysReserved2

Once a connection has been made we should empty receiving serial queue:

```
SrmReceiveFlush(portID, timeoutInTicks);
                  ↑
                UInt16
```

To close a serial port connection:

```
Err error;
error = SrmSendWait(portID);
        // wait till done sending
ErrNonFatalDisplayIf(error == serErrBadPort,
                     "SrmClose: bad port");
if (error == serErrTimeOut)
    FrmAlert(SerialTimeOutAlert);
SrmClose(portID);
```

Sending and receiving data:

Ex)

```
UInt32 bytesSent;
Char buf[] = "hello\n";
SrmSend(portID, buf, StrLen(buf), &error);
SrmSendCheck(portID, &bytesSent);
if(bytesSent != StrLen(buf)
    SrmSendFlush(portID);
```

Receiving Data

Would like

```
Char buf[MaxLength];
SrmReceive (portID, buf, len, timeout, &error);
// Tock

UInt32 bytesReceived;
SrmReceiveCheck (portID, &bytesReceived);

// Also useful
SrmReceiveWait (portId, len, timeout);
```