# Chapter 3 Communications & Networking

From: rmh@apple.com (Rick Holzgrafe)
**Subject: Re: Zones (how does one determine what zone one is in with code)**

In article <sZhJIpi00WBL03UUlC@andrew.cmu.edu> nf0i+@andrew.cmu.edu (Norman William Franke, III) writes:

> I found out how to read the zones from some sample code from Apple, but
> how does one determine what zone one is in? I thought it was the first
> zone returned, but it just "broke". Does anyone know how to do this?

Yes. The Inside Mac documentation is sketchy and the MPW include files are incomplete, but it can be done and here's how:

```
GetMyZone (char buf[578])
{
   OSErr err;
   short atpRefNum, mppRefNum;
   short myNode, myNet;
   SendReqparms atp;
   BDSElement bds;


   err = OpenDriver ("\P.MPP", &mppRefNum);
   if (err != noErr) /* oops */

   GetNodeAddress (&myNode, &myNet);

   err = OpenDriver ("\P.ATP", &atpRefNum);
   if (err != noErr) /* oops */

   bds.buffSize = sizeof (buf);
   bds.buffPtr = buf;

   atp.ioCompletion = NULL;
   atp.ioRefNum = atpRefNum;
   atp.csCode = 255; // sendRequest

   atp.userData = 0x07000000;// GetMyZone
   atp.atpFlags = 0;
   atp.addrBlock.aNet = myNet;
   atp.addrBlock.aSocket = 6;
   // GetBridgeAddress is new, if not imp. use low-mem global
   // ABusVars.sysABridge
   // If bridge == 0 or net == 0, no bridge and no zones!
   atp.addrBlock.aNode = GetBridgeAddress ();
   atp.reqLength = 0;
   atp.reqPointer = (Ptr)0;
   atp.bdsPointer = (Ptr)&bds;
   atp.filler = 1;// NumOfBuffs -- honest!
   atp.timeOutVal = 1;
   atp.retryCount = 3;
   atp.atpFlags = 0;
   atp.retryCount = 3;

   PBControl ((ParmBlkPtr)&atp, false);
   err = atp.ioResult;

   if (err != noErr) /* oops */

   if (buf[0] > 32) // "Can't happen", it says here
       buf[0] = 32;
   /* buf now contains the name of your zone! */
}
```

The above won't compile as is (you'll need error handling at least), and I may have screwed something up by excerpting this from a larger routine, but it *does* come from a working routine. It's in MPW 3.0 C. Hope i t helps.

-
Rick Holzgrafe

●●●

From: resnick@lees.cogsci.uiuc.edu (Pete Resnick)
**Subject: MacTCP programming with THINK C**
Keywords: think ,mactcp

I have gotten enough request for the changes I have made to the MacTCP header files that I think I will post them here. There are only a few changes that I have made so far, but there may be others that I haven't used yet.

In MacTCPCommonTypes.h, change the "ifndef/include" definition to look like this:

```
#ifndef _MacTypes_
#include <MacTypes.h>
#endif /* _MacTypes_ */
```

In UDPPB.h, comment out the word "PASCAL" in the declaration of UDPNotifyProc so that it looks like this (THINK C doesn't like a typedefed PASCAL function):

```
typedef /*pascal*/ void (*UDPNotifyProc) (
       StreamPtr udpStream,
       unsigned short eventCode,
       Ptr userDataPtr,
       struct ICMPReport *icmpMsg);
```

Do the same thing in TCPPB.h for TCPNotifyProc:

```
typedef /*pascal*/ void (*TCPNotifyProc) (
       StreamPtr tcpStream,
       unsigned short eventCode,
       Ptr userDataPtr,
       unsigned short terminReason,
       struct ICMPReport *icmpMsg);
```

You will not be able to use these typedefs in your own programs; they are only to deal with other declarations in your header files. In your code, declare your notify procs as 'PASCAL void whatever' with all of the parameters as they are declare in the typedef.

That's all I have found so far, except that in my copy of TCPPB.h, instead of the csCode for receive buffer return being TCPBfrReturn as documented, it was TCPRcvBfrReturn. I changed it to TCPBfrReturn.

pr
--
Pete Resnick (...so what is a mojo, and why would one be rising?)

●●●

From: chesley@goofy.apple.com (Harry Chesley)
**Subject: Re: Serial driver - Why is CTSHold being set?**

In article <RANG.90Jun5232240@derby.cs.wisc.edu> rang@cs.wisc.edu (Anton Rang) writes:

> What I do:
>
> 1. Call RAMSDOpen (on port A).

According to a recent tech note, it's now better to do just an OpenDriver, rather than RAMSDOpen. RAMSDOpen was

needed to load the RAM driver in the original Macs, but now that driver is standard. I don't believe this could cause any problems at this point, but OpenDriver is better for the future.

> 2. Call SerReset for the port A input and output drivers.

Triple check the parameters to this call. It's quite possible that something is off, and so you're sending at the wrong baud-rate, number of data bits, or something.

> Calling SerStatus reports that 'holdCTS' is on. Presumably, this
> means that the driver is convinced my modem isn't ready. But (1) my
> modem has CTS forced on, and (2) all the communications software I
> have works fine.

CTS is used for signal flow control. However, I'm quite sure the driver doesn't listen to it unless you explicitly turn it on via SerHShake, so the problem should be earlier than that. If you are trying to talk to a modem, it's possible that the modem isn't listening until some modem signal is asserted (like DTR), though other comm software working suggests that's not it. Don't, however, count on the modem wires going through from the modem to the Mac, as there are as many different cables out there as there are permutations of the wires in the cables (unless you've buzzed your cable out yourself).

> P.S. I'd like to wind up with an application which can handle
> incoming data while in the background under MultiFinder.
> Anyone have any suggestions on the best (or worst) ways
> to go about this? I've thought about setting up a big
> input buffer and periodically checking how many characters
> are in the buffer, using PBRead to extract them, and either
> queueing them for later or processing them. Will this work?

That's the way I'd do it. You can also set up an asynchronous read.

●●●

From: jamesbo@microsoft.UUCP (James Borquist)
**Subject: The prefered AppleTalk interface, is it from hell or what?**

Hi there! I'm trying to use the perfered interface to transmit some information between two computers via ATP. I have been doing this with the alternative interface for quite some time now with the best of success, but for some reason, I just can't seem to get this to work with the prefered interface. I have two programs. They are quite simple tests, they don't do much, and they don't work. Could someone look at the following code fragments and tell me what I am doing wrong before I go completely crazy?

This is from the server:

```
with requestPB { ATPParamBlock } do
  begin
    ioCompletion := nil;
    atpSocket := mySocket; { byte }
    reqLength := 578;
    reqPointer := @requestBuffer; { array[1..578] of byte }
  end;
result := PGetRequest(@requestPB, true);
while 1<>2 do
  if requestPB.ioResult <= 0 then
    begin
      with responsePB { ATPParamBlock } do
        begin
          atpSocket := mySocket;
          atpFlags := atpEOMvalue;
          addrBlock := requestPB.addrBlock;
          bdsPointer := @responseBDS; { BDSType }
          filler0 := 1;
          bdsSize := 1;
          transID := requestPB.reqTID;
        end;
      result := PSendResponse(@responsePB, false);
      with requestPB do
```

```
begin
  ioCompletion := nil;
  atpSocket := mySocket;
```

```
            reqLength := 578;
            reqPointer := @requestBuffer;
          end;
        result := PGetRequest(@requestPB, true);
      end;
```

And this is from the Cient:

```
  with requestPB do
    begin
      atpFlags := atpXOvalue;
      addrBlock := address; { from PLookupName and NBPExtract }
      reqLength := 578;
      reqPointer := @requestBuffer; { array[1..578] of byte }
      bdsPointer := @responseBDS; { BDSType }
      numOfBuffs := 1;
      timeOutVal := 8;
      retryCount := 3;
    end;
  result := PSendRequest(@requestPB, false);
```

Please, somebody help me!!!

Thanks,
James Borquist (uunet!uw-beaver!microsoft!jamesbo)

●●●

From: zben@umd5.umd.edu (Ben Cranston)
**Subject: Problem with Apple's distributed MacTCP dnr.c routine (bug has been fixed)**
Keywords: MacTCP Domain Volume

There is a problem with the version of dnr.c distributed by Apple with early version of MacTCP. It causes applications not to be able to initialize the Domain Name Resolver when running from a volume that is not the system volume (the one containing the current system file).

In routine OpenOurRF() the code does a PBHGetFInfo scan of the system folder looking for a file with type 'cdev' and creator 'mtcp'. When it finds one it calls OpenResFile on the name returned by the GetFInfo and returns the status to its caller, OpenResolver. Due to the vagaries of configuration on 512e machines (:-) the mainline ignores the status.

The name returned by the GetFInfo scan is a "bare" name, e.g., "MacTCP". It is not a pathname like "Harddisk:System Folder:MacTCP". So, the OpenResFile will search the directory containing the executing application and typically not find MacTCP, but then the Poor Man's Search Path will cause the search of the System Folder and MacTCP will be found. The code in OpenResolver will then proceed to read the 'dnrp' resource into memory and call it.

Those who know the PMSP will now understand the problem: the PMSP does NOT cross volume boundaries! So, if you put your program on a floppy it suddenly stops working.

After finding this out I took a look at the NameToAddress Hypercard XCMD and found they had modified the code to solve this problem. What they did, and what I have done, is to do a SetVol to the system folder before the OpenResFile so it will search the System Folder rather than the folder containing the application.

But MY code saves and restores the old SetVol and theirs does not :-).

```
/* ! */ marks new lines:

/* OpenOurRF is called to open the MacTCP driver resources */
short OpenOurRF()
{
   SysEnvRec info;
   HParamBlockRec fi;
```

```
   Str255 filename;
#ifdef ZBEN                /* ! */
```

```
   short savevol;                /* ! */
   OSErr status;           /* ! */
#endif                  /* ! */

   SysEnvirons(1, &info);
   fi.fileParam.ioCompletion = nil;
   fi.fileParam.ioNamePtr = &filename;
   fi.fileParam.ioVRefNum = info.sysVRefNum;
   fi.fileParam.ioDirID = 0;
   fi.fileParam.ioFDirIndex = 1;
   while (PBHGetFInfo(&fi, false) == noErr) {
/* scan system folder for driver resource files of specific type & creator */
    if (fi.fileParam.ioFlFndrInfo.fdType == 'cdev' &&
        fi.fileParam.ioFlFndrInfo.fdCreator == 'mtcp') {
/* found the MacTCP driver file */
#ifdef ZBEN                /* ! */
        GetVol(nil,&savevol);            /* ! */
        SetVol(nil,info.sysVRefNum);     /* ! */
        status = OpenResFile(&filename); /* ! */
        SetVol(nil,savevol);             /* ! */
        return(status);           /* ! */
#else                   /* ! */
#ifdef MPW3.0
        return(OpenResFile(&filename));
#else
        return(OPENRESFILE(&filename));
#endif
#endif                  /* ! */
    }
/* check next file in system folder */
    fi.fileParam.ioFDirIndex++;
    fi.fileParam.ioDirID = 0;
    }
   return(-1);
}
```

●●●