Connecting a Brew MP application to a PC through USB/IPort

Objective

This topic describes how to

* Connect a Brew® MP application to a PC via USB/IPort
* Send Attention (AT) commands to the Attention Command Processor (ATCOP)

Requirements

The application needs to use the correct COM port and the correct commands.

Example - Determining which COM port to use

To determine which COM port to use:

1. Open Device Manager.

Under Modems, you should see a modem corresponding to the device.

1. Right-click on the modem and select Properties.
2. In the Properties dialog box for the modem, click the Modem tab.

At the top of the Modem tab, the Port value indicates the COM port to use.

Example - Sending AT commands to the ATCOP

After the application successfully sends an AT$BREW command, the application communicates with the Brew MP Serial Command Processor (BSCOP). The BSCOP commands have numbers prepended to them, such as 01ver or 02app.

If the application opens AEESIO\_PORT\_INCOMING, the application gains control of the port that enters (or is already in BSCOP). The port also only opens when the transition into BSCOP occurs.

Note: If you're writing an internal tool for the device, you may want to consider using Connect. It handles the serial port/connectivity issues, provides reliability, and has other features such as the ability to copy files, send events, and start applets.

The following sample code is from a file called siousage.c. SIO is a protected class. To create it, you must declare a dependency on AEECLSID\_SERIAL in your MIF.

// Our applet's structure

typedef struct SIOUsage

{

AEEApplet a;

IPort \*piPort;

AEECallback cbReadable;

AEECallback cbWriteable;

} SIOUsage;

static int SIOUsage\_Init(SIOUsage \*pme)

{

int nErr;

nErr = ISHELL\_CreateInstance(pme->a.m\_pIShell, AEECLSID\_SERIAL,

(void\*\*)&pSioPort->piPort);

if (nErr != SUCCESS)

{

return EFAILED;

}

// The following line opens the USB serial port. To open the RS-232

// serial port, use AEESIO\_PORT\_SIO1

nErr = IPORT\_Open(pme->piPort, AEESIO\_PORT\_USB1);

if (nErr != SUCCESS && nErr != AEEPORT\_WAIT)

{

// There was some unknown error opening the port

DBGPRINTF("Error opening port");

return EFAILED;

}

// Setup our callbacks. These are called when there is \*probably\* data

// to be written or read.

CALLBACK\_Init(&pme->cbReadable, SIOUsage\_TryRead, (void\*)pme);

CALLBACK\_Init(&pme->cbWriteable, SIOUsage\_TryWrite, (void\*)pme);

IPORT\_Readable(pme->piPort, &pme->cbReadable);

IPORT\_Writeable(pme->piPort, &pme->cbWriteable);

// At this point, if nErr is AEEPORT\_WAIT, the serial port is opening.

// We'll know this is complete when our cbWriteable callback fires.

// Normally, we'll receive AEEPORT\_WAIT instead of SUCCESS.

}

static void SIOUsage\_TryRead(void \*po)

{

SIOUsage \*pme = (SIOUsage\*)po;

int32 dwRead;

char buf[100];

// We can \*probably\* read data here

dwRead = IPORT\_Read(pme->piPort, buf, sizeof(buf));

// Log a message if we did read data

if (dwRead > 0)

{

DBGPRINTF("Read %ld bytes of data", dwRead);

}

// Remember that we need to reset our readable callback

IPORT\_Readable(pme->piPort, &pme->cbReadable);

}

static void SIOUsage\_TryWrite(void \*po)

{

SIOUsage \*pme = (SIOUsage\*)po&

int32 dwWrite&

char \*pBuf = "Hello World!";

// If you want to know if the port successfully got opened, do it here.

// what you want to do is nErr = IPORT\_Open(pme->piPort, NULL);

// We can \*probably\* write data here

dwWrite = IPORT\_Write(pme->piPort, pBuf, STRLEN(pBuf));

// Log a message if we did read data

if (dwWrite > 0)

{

DBGPRINTF("We wrote %ld bytes of data", dwWrite);

}

// Remember that we need to reset our writeable callback

IPORT\_Writeable(pme->piPort, &pme->cbWriteable);

}