Using the Communications Functions

Configuring a Communications Resource

The following example opens a handle to COM1 and fills in a **DCB** structure with the current configuration. The **DCB** structure is then modified and used to reconfigure the device.

```
DCB dcb;
HANDLE hCom;
DWORD dwError;
BOOL fSuccess;

hCom = CreateFile( "COM1",
    GENERIC_READ | GENERIC_WRITE,
    0,    // comm devices must be opened w/exclusive-access
    NULL, // no security attributes
    OPEN_EXISTING, // comm devices must use OPEN_EXISTING
    0,    // not overlapped I/O
    NULL  // hTemplate must be NULL for comm devices
    );

if (hCom == INVALID_HANDLE_VALUE)
{
    dwError = GetLastError();

    // handle error
}

// Omit the call to SetupComm to use the default queue sizes.
// Get the current configuration.

fSuccess = GetCommState(hCom, &dcb);

if (!fSuccess)
{
    // Handle the error.
}

// Fill in the DCB: baud=9600, 8 data bits, no parity, 1 stop bit.

dcb.BaudRate = 9600;
dcb.ByteSize = 8;
dcb.Parity = NOPARITY;
dcb.StopBits = ONESTOPBIT;

fSuccess = SetCommState(hCom, &dcb);

if (!fSuccess)
{
    // Handle the error.
}
```

Monitoring Communications Events

The following example code opens the serial port for overlapped I/O, creates an event mask to monitor

CTS and DSR signals, and then waits for an event to occur. The **WaitCommEvent** function should be executed as an overlapped operation so the other threads of the process cannot perform I/O operations during the wait.

```
HANDLE hCom;
OVERLAPPED o;
BOOL fSuccess;
DWORD dwEvtMask;

hCom = CreateFile( "COM1",
    GENERIC_READ | GENERIC_WRITE,
    0,    // exclusive access
    NULL, // no security attributes
    OPEN_EXISTING,
    FILE_FLAG_OVERLAPPED,
    NULL
    );

if (hCom == INVALID_HANDLE_VALUE)
{
    // Handle the error.
}

// Set the event mask.

fSuccess = SetCommMask(hCom, EV_CTS | EV_DSR);

if (!fSuccess)
{
    // Handle the error.
}

// Create an event object for use in WaitCommEvent.

o.hEvent = CreateEvent(
    NULL,   // no security attributes
    FALSE,  // auto reset event
    FALSE,  // not signaled
    NULL    // no name
    );

assert(o.hEvent);

if (WaitCommEvent(hCom, &dwEvtMask, &o))
{
    if (dwEvtMask & EV_DSR)
    {
         // To do.
    }

    if (dwEvtMask & EV_CTS)
    {
         // To do.
    }
}
```