Visual C++ Serial Communication

Overview

This application demonstrates serial communications using various methods. This simple program, initializes the COM port and sends a Move Relative, Time Based (MRT) command to the attached QuickSilver Device. The device's response is received and displayed.

It is assumed the reader is familiar with Windows, Visual C++, QuickControl®, programming QuickSilver products, and QuickSilver's serial communication. For more information see:

- QCI-TD053 Serial Communications
- SilverLode User Manual

This document is meant to explain the setup and general design of the example. The details concerning communicating with a QuickSilver device are left up to the documents mentioned above and the source code comments.

Microsoft and Windows are registered trademarks of Microsoft Corporation.

Setup

This application demonstrates three different methods of communicating with a QuickSilver device over an RS-232 serial port using Visual C++. They include:

- 8 Bit ASCII Protocol Using MSCOMM.OCX
- 8 Bit ASCII Protocol Using File I/O
- 9 Bit Binary Protocol Using File I/O

The following assumptions are made about the setup:

- The device has already been initialized (using QuickControl) and connected to COM 1.
- 8 Bit Protocol or 9 Bit Binary depending on example
- 57600 Baud Rate
- Unit ID of 16.

Supplied Files

File/Folder	Description
MSCommTest	Source Code using Microsoft's MSCOMM.OCX
FileCommTest	Source Code using file I/O (8 Bit ASCII)
FileCommTest 9 Bit	Source Code using file I/O (9 Bit Binary)
MRT POL	Source Code using a move command and checks for completion
LRP POL	Source Code using a load and run command and checks for completion
LRP POL RRG	Source Code using a load and run command, checks for completion, and checks the data of register 1

Application Overview

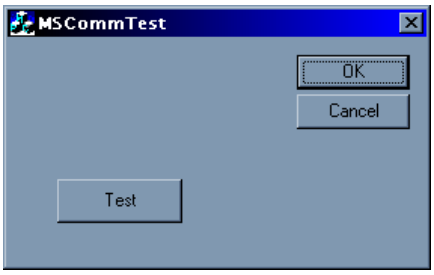
This application will demonstrate three different methods of serial communications (transmission and reception of data) with a QuickSilver device over a serial port. The first method is found in MSCommTest, which uses MSComm (Microsoft Communications Controller-MSCOMM.OCX). The second and third method are found in FileCommTest and FileCommTest 9 Bit and use the file I/O method which does not require the file MSCOMM.OCX .

Both methods will perform the same function, which is verifying communications with the QuickSilver device. Communications are tested by sending an Move Relative, Time Based (MRT) command to the device and looking for an acknowledgment (ACK).

MSCommTest is more of an object oriented approach, but does not allow as much access to the COM port which is why we do not provide an example of the 9 bit protocol using MSCOMM.OCX. If you are only doing simple 8 Bit ASCII communications, we recommend using the MSCommTest. For advanced users, we recommend the FileCommTest methods.

Dialog Box (All Examples)

The dialog box in all the examples is the same. Pressing the 'Test' button will send an MRT command.



Notes On Documentation

Since the Microsoft framework generates some of the code for you, only those classes with QCI specific code and those that are ActiveX Controls will be addressed.

Note: QCI specific code is tagged with the comment character "///QCI". This allows the programmer to quickly search the application for the "important" stuff.

MSCommTest

CMSComm Class

MSComm or Microsoft Communications Controller is an ActiveX Control supplied by Microsoft. The MSComm control provides serial through a serial port.

You can add this control to your project by

1. Selecting Project->Add to project->Components and Controls
2. Select 'Registered ActiveX Controls'
3. Select 'Microsoft Communications Controller, Version 6.0'
4. Click 'Insert'
5. Click 'OK'
6. Click 'OK' to confirm that a new class will be created for this control
7. Close the Components and Controls Gallery.

This inserts the Communications Controller into the dialog editor's toolbox (appearing as a yellow telephone), which you can now drag onto your dialog box to make it part of your project. All of the functions of MSComm are now part of your project.

CMSCommTestDlg Class

This is the dialog box class. It contains the code that enables us to communicate with the device.

When the 'Test' button is pressed, MSCommTestDlg::OnTest() sets up the COM Port and sends out an MRT command to the device. OnTest() then looks for a response and displays it in a Message Box.

FileCommTest

CFileCommTestDlg

This is the dialog box class. It contains the code that enables us to communicate with the device.

CFileCommTestDlg::OnInitDialog() calls CFileCommTestDlg::ConfigCOM to configure COM1.

When the 'Test' button is pressed, CFileCommTestDlg::OnTest() puts the MRT command into strCmd and sends it to CFileCommTestDlg::SendCommand().

SendCommand() first makes sure we have a valid window and our COM port is enabled. The command in strCmd is written to the COM port by the command WriteFile. SendCommand() then looks to receive an Acknowledgement (ACK) from the device using ReadFile.

FileCommTest 9 Bit

CFileCommTestDlg

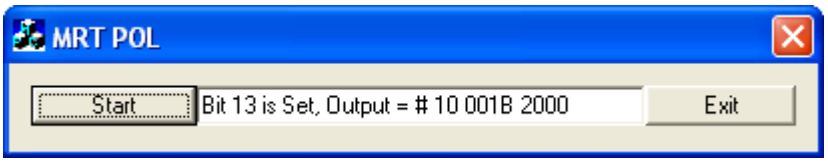
This is the dialog box class. It contains the code that enables us to communicate with the device.

CFileCommTestDlg::OnInitDialog() calls CFileCommTestDlg::ConfigCOM to configure COM1.

When the 'Test' button is pressed, CFileCommTestDlg::OnTest() puts the MRT command into a buffer. Mark Parity is set for the COM port and the first byte is sent. This is how we set the 9th bit to signify an address byte. After the first byte (the address byte) is sent out, parity is set to Space Parity and the rest of the buffer is sent out.

To detect address bytes from the device, we enable the COM port's parity error reporting. CFileCommTestDlg::RxByte() reads in one byte at a time flagging any byte with the 9th bit set (parity error).

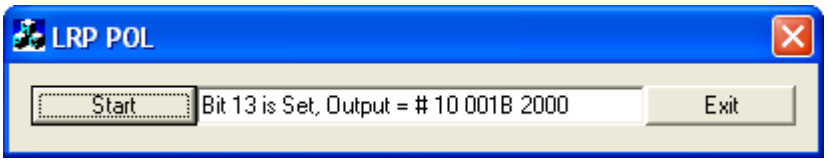
MRT POL



CMSCommTestDlg Class
This is the dialog box class. It contains the code that enables us to communicate with the device.

When the 'Test' button is pressed, MSCommTestDlg::OnTest() sets up the COM Port and sends out an MRT command to the device. OnTest() then looks for a returned completion bit before displaying the complete returned data.

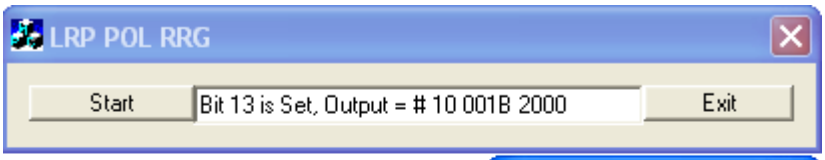
LRP POL

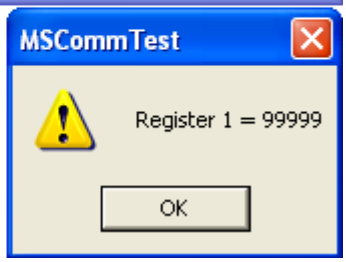


CMSCommTestDlg Class
This is the dialog box class. It contains the code that enables us to communicate with the device.

When the 'Test' button is pressed, MSCommTestDlg::OnTest() sets up the COM Port and sends out an LRP command to the device. OnTest() then looks for a returned completion bit before displaying the complete returned data.

LRP POL RRG





CMSCommTestDlg Class
This is the dialog box class. It contains the code that enables us to communicate with the device.

When the 'Test' button is pressed, MSCommTestDlg::OnTest() sets up the COM Port and sends out an LRP command to the device. OnTest() then looks for a returned completion bit before displaying the complete returned data. Once the program is complete, OnTest() sends out an RRG command, which returns the information stored in register 1.