ActiveX Control Robx.ocx

Installation

The installation disk contains the control, the dynamic-link libraries it needs and the installer program setup.exe. Run setup to install and register the components. The program xdem can be used to test the control.

Methods

The following methods are provided. C-Style declarations are used here. If you are using another language then the appropriate declarations should be automatically generated when you add the control to your application.

short OpenComm(short Port, long BaudRate);

Opens the communications port.
Port is 1 for COM1 etc.
BaudRate should normally be 19200
The return value is 1 if successful, 0 for failure. Use the method GetCommErrorString() to get a description of the problem.

void CloseComm();

Call this before quitting the application.

short SendString(LPCTSTR String);

Sends the string to the controller. The string should end with a "Carriage Return" character (ASCII 0D hex).
The return value is 1 if the string was successfully sent, 0 if a communication error occurred. Use the method GetCommErrorString() to get a description of the problem.
The controller must be in the "ready" state (as reported by GetStatus) before invoking this method. After sending the string invoke GetStatus() until the "ready" state is received.

short GetStatus();

Gets the controller status as follows:

0: Waiting
2: Ready, received OK
1: Ready but not OK
-1: Communication error

After receiving status 1 or 2, you can use GetResponse() to get the last line sent by the controller.
If status is -1, use GetCommErrorString() to get a description of the problem.

CString GetResponse();

This gets the controller response. If status was 2 the response string indicates what went wrong.

CString GetCommErrorString();

Gets a description of communication problems.

void AboutBox();

Displays the AboutBox();
Using the methods
Test programs xdem.exe and xdem3.exe are provided. The C++ source code is also provided in the zip files xdem.zip and xdem3.zip.

Any other programming language which supports Activex controls may of course also be used.

First Open the communications port.

Whenever a command is sent to the controller (by SendString()) you should wait for it to complete the command before sending another. The function Command_wait in xdem3dlg.cpp shows how to do this:

```
int CXdem3Dlg::Command_wait(CString command, time_t maxtim){
    if(sstep)
        AfxMessageBox("Ready to send " + command);
    if(!m_robx.SendString(command + "\r")){
        AfxMessageBox("Send string " + m_robx.GetCommErrorString());// problem
sending string
        return 0;
    }
    time_t tim = time(0);
    int stat;
    while(1){
        Sleep(100);
        stat = m_robx.GetStatus();
        if(stat)
            break;
        if((time(0) - tim) > maxtim){
            AfxMessageBox(command + " Timed out");
            return 0;
        }
    }
    switch(stat){
    case -1:
        AfxMessageBox(m_robx.GetCommErrorString());// comm problem
        return 0;
        break;
    case 1:
        AfxMessageBox(m_robx.GetResponse());// not OK
        return 0;
        break;
    case 2:
        return 1;    // OK
        break;
    }
    return 0;
}
```

It sends the command then waits in a loop for GetStatus();

(It is permissible to do this with a pre-emptive multi-tasking operating system which interrupts tasks to give each a share of processing time, the Sleep() function returns control to the operating system each time round the loop to avoid wasting system time)

It is desirable to have a way of getting out of the loop if something goes wrong and the controller never responds. For each command a time is provided in which it is expected to finish.

Some commands, such as sending “CARTESIAN”, complete immediately so one second is sufficient whereas if the robot is commanded to move the controller will not respond with OK until the robot has completed the move so several seconds may be needed.

If the response is not OK then GetResponse() is used to see what the problem was.

The xdem3 program shows a sequence of commands with optional single stepping.

The first command in the sequence is just a “carriage return” character. This checks that the controller is in the “ready” state before sending the first “real” command.

Sequencing commands in this way blocks the main program thread so the user interface does not work while the sequence is running. It may be desirable to start a second “worker thread” to run the command sequence.
Alternatively, rather than writing the command program in C++ itself, these difficulties may be avoided by using a dedicated Task Sequence Control system such as the one we provide.

Useful commands
In addition to the normal ROBOFORTH commands there are some commands especially for use with a supervisor program:

GF returns a text string consisting of the joint positions, however many there are on the particular robot. For example if you send GF you could get back

GF 1000 2000 3000 1500 750 OK

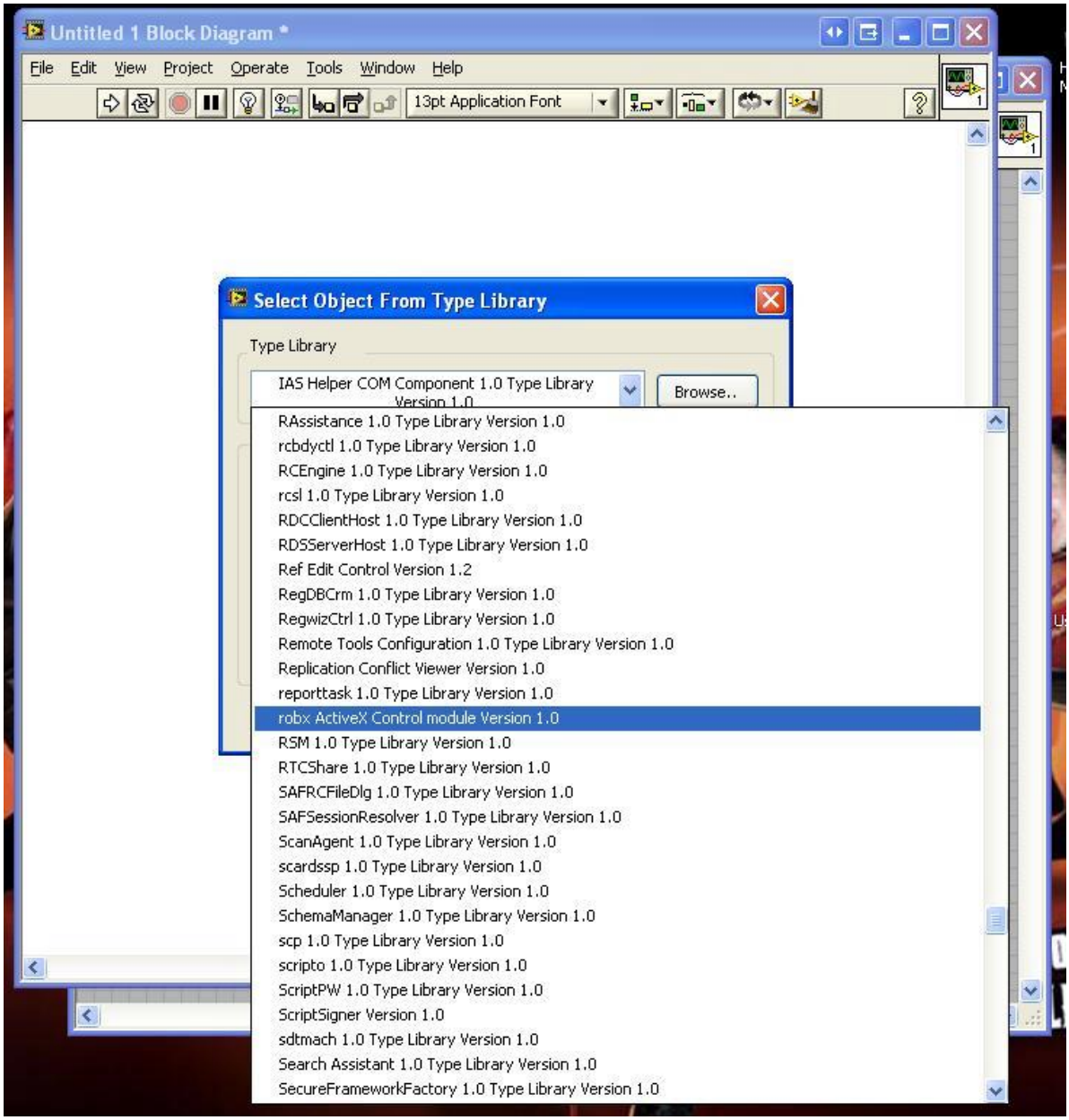
(for a 5-axis robot.)

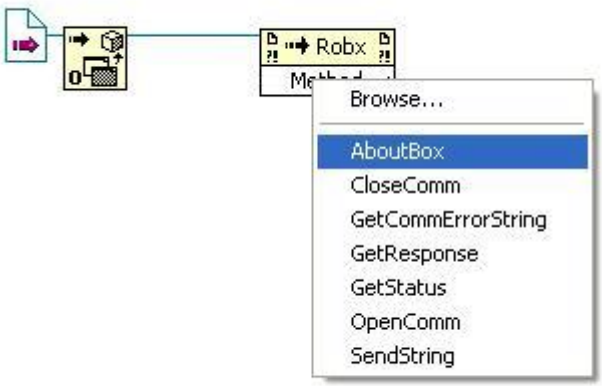
CF returns the Cartesian positions in a similar way: x,y and z as integers in 0.1mm units then two wrist angles in 0.1 degree units.

It is advisable to send the string "DECIMAL" first to ensure that the functions are interpreted correctly and return decimal numbers.

LabView

To use RobX with Labview use “Automation Open” to open a reference to the ActiveX server then “Invoke Node” to list and choose the methods, see screen shots on next page.


Untitled 1 Block Diagram *
File Edit View Project Operate Tools Window Help
13pt Application Font
Select Object From Type Library
Type Library
IAS Helper COM Component 1.0 Type Library Version 1.0
Browse..
RAssistance 1.0 Type Library Version 1.0
rcbdyctl 1.0 Type Library Version 1.0
RCEngine 1.0 Type Library Version 1.0
rcsl 1.0 Type Library Version 1.0
RDCClientHost 1.0 Type Library Version 1.0
RDSServerHost 1.0 Type Library Version 1.0
Ref Edit Control Version 1.2
RegDBCrm 1.0 Type Library Version 1.0
RegwizCtrl 1.0 Type Library Version 1.0
Remote Tools Configuration 1.0 Type Library Version 1.0
Replication Conflict Viewer Version 1.0
reporttask 1.0 Type Library Version 1.0
robx ActiveX Control module Version 1.0
RSM 1.0 Type Library Version 1.0
RTCShare 1.0 Type Library Version 1.0
SAFRCFileDlg 1.0 Type Library Version 1.0
SAFSessionResolver 1.0 Type Library Version 1.0
ScanAgent 1.0 Type Library Version 1.0
scardssp 1.0 Type Library Version 1.0
Scheduler 1.0 Type Library Version 1.0
SchemaManager 1.0 Type Library Version 1.0
scp 1.0 Type Library Version 1.0
scripto 1.0 Type Library Version 1.0
ScriptPW 1.0 Type Library Version 1.0
ScriptSigner Version 1.0
sdtmach 1.0 Type Library Version 1.0
Search Assistant 1.0 Type Library Version 1.0
SecureFrameworkFactory 1.0 Type Library Version 1.0


Robx
Browse...
AboutBox
CloseComm
GetCommErrorString
GetResponse
GetStatus
OpenComm
SendString