WinCon 3.3

Hard Real-time Performance at your Fingertips

User's Guide

Version 1.0

Another Real-Time Product
by Quanser Consulting

Quanser
ADVANCED CONTROL SYSTEMS

How to contact Quanser Consulting:

	(905) 940-3575	Telephone
	(905) 940-3576	Facsimile
	80 Esna Park Drive Markham, ON Canada L3R 2K8	Mail
	http://www.quanser.com http://www.wincon.quanser.com	Web
	mailto://tech@quanser.com mailto://info@quanser.com	Support for WinCon General information

All Rights Reserved

Printed in Canada.

Disclaimer

While every effort has been made to ensure the accuracy and completeness of all information in this document, Quanser, Inc. assumes no liability to any party for any loss or damage caused by errors or omissions or by statements of any kind in this document, its updates, supplements, or special editions. Quanser, Inc. further assumes no liability arising out of the application or use of any product or system described herein; nor any liability for incidental or consequential damages arising from the use of this document. Quanser, Inc. disclaims all warranties regarding the information contained herein, whether expressed, implied or statutory, including implied warranties of merchandability or fitness for a particular purpose. Quanser Inc. reserves the right to make changes to this document or to the products described herein without furhter notice.

Trademarks

WinCon is a trademark of Quanser Consulting, Inc.
MATLAB, Simulink and Real-Time Workshop are registered trademarks and Target Language Compiler is a trademark of The MathWorks, Inc.
Windows and Visual C++ are registered trademarks of Microsoft Corporation.
RTX is a registered trademark of VenturCom, Inc.
Intel and Pentium are registered trademarks of Intel Corporation.
Other brands and their products are trademarks or registered trademarks of their respective holders and should be noted as such.

Licencing Rules

You must acquire and dedicate a license for WinCon for each computer on which WinCon is used.

Each WinCon package consists of one copy of WinCon Server and one copy of WinCon Client. If you want to run configurations that require more than one PC, you will need as many copies of WinCon. One WinCon license is required per PC on which a WinCon component is installed, whether it is WinCon Server, WinCon Client, or both.

For complete details, please consult the End-User Licence Agreement (EULA) provided with the WinCon software.

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Figures

Table of Figures

Table of Figures

About WinCon 3.3

WinCon™ is a real-time Windows 98/NT/2000/XP application. It allows you to run code generated from a Simulink diagram in real-time on the same PC (also known as local PC) or on a remote PC. Data from the real-time running code may be plotted on-line in WinCon Scopes and model parameters may be changed on the fly through WinCon Control Panels as well as Simulink. The automatically generated real-time code constitutes a stand-alone controller (i.e. independent from Simulink) and can be saved in WinCon Projects together with its corresponding user-configured scopes and control panels.

WinCon software actually consists of two distinct parts: *WinCon Client* and *WinCon Server*. They communicate using the TCP/IP protocol. WinCon Client runs in hard real-time while WinCon Server is a separate graphical interface, running in user mode. This guide describes how to install and use WinCon.

New Features and Updates

CAUTION: In WinCon 3.3, the project file format has changed. Old project files will not work and MUST be rebuilt. Therefore, to convert a non-backward compatible WinCon project, one must **save that project's visual layout (e.g. WinCon Displays and Control Panels) and settings** in order to **visually reproduce it** in the new WinCon 3.3 project format. Refer to Section WinCon Projects, on page 89, for more details.

The new features and updates that have been made since the WinCon 3.2 release include:

- **An improved WinCon Project file format**. New WinCon Projects now include **more information** on each project component and are now **portable** (to different disk locations). Object-versioning information has also been added, to allow for **backward compatibility** with future releases of WinCon.
- WinCon **Displays** and **Control Panels** now **reassociate their variables by name** when loading a WinCon Project or when the real-time code is reloaded (e.g. because of a rebuild). Hence, it is now possible to change the Simulink model and reload a previous WinCon project and/or newly-generated real-time code with the previously defined Display(s) and Control Panel(s).
- **Improved** WinCon **Scope** (with new auto-scaling capabilities and faster graphics), **XYGraph, Digital Meter** and **Thermometer**.
- Two **new Controls** have been added to WinCon Controls: **Text** and **Picture**. Pictures and multi-line text can now be placed on WinCon Control Panels.
- **WinCon Control Panels** now **write their current variable values to the controller when a project is loaded or the model is rebuilt**, instead of the other way around.
- Under Windows NT/2000/XP, WinCon Client can now be **run from a non-Administrator account**, for higher security. Due to a new NT Service, called **WinCon Service**, Regular Users (i.e. with no Administrator privileges) have full use of WinCon.
- **WinCon Task Manager** enables the **monitoring of real-time tasks**.
- **Improved MultiQ-PCI drivers**. All 16 Analog-To-Digital channels can now be read at once.
- **RTX is no longer required to be installed prior to installing WinCon Server**, under Windows NT/2000/XP. However RTX **MUST be installed** to run real-time code on Windows NT/2000/XP.

Local and Remote Configurations

WinCon supports two possible configurations: the local configuration (i.e. a single machine) and the remote configuration (i.e. two or more machines). In the local configuration, WinCon Client, executing the real-time code, runs on the same machine and at the same time as WinCon Server (i.e. the user-mode graphical interface). In the remote configuration, WinCon Client runs on a separate machine from WinCon Server. The two programs always communicate using the TCP/IP protocol. Each WinCon Server can communicate with several WinCon Clients, and reciprocally, each WinCon Client can communicate with several WinCon Servers.

Local Configuration

The local configuration is shown below in Figure 1. The data acquisition card, in this case the MultiQ, is used to interface the real-time code to the plant to be controlled. The user interacts with the real-time code via either WinCon Server or the Simulink diagram. Data from the running controller may be plotted in real-time on the WinCon scopes and changing values on the Simulink diagram automatically changes the corresponding parameters in the real-time code. The real-time code, i.e. WinCon Client, runs on the same PC. However, the real-time code takes precedence over everything else, so hard real-time performance is still achieved!

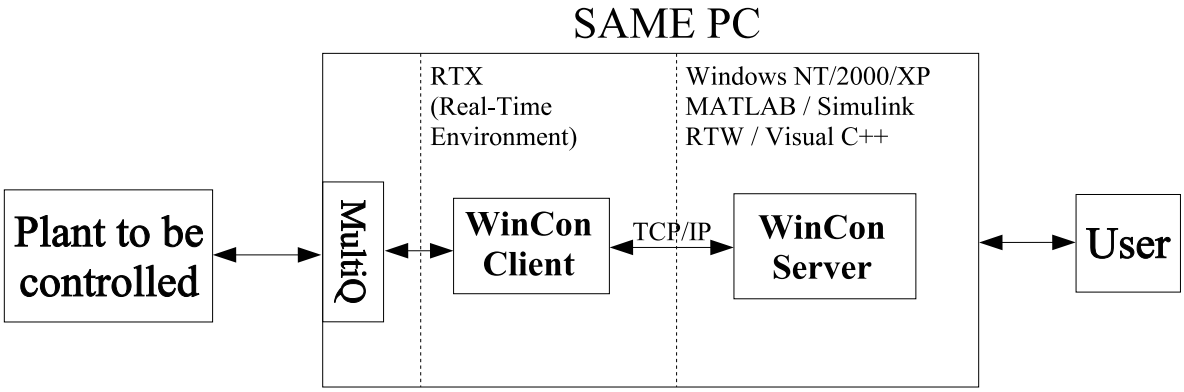


Figure 1 Local configuration: 1 PC

Remote Configurations

In the remote configurations, WinCon Client, and hence the real-time code, runs on a separate platform than Simulink and WinCon Server (i.e. the user interface).

The minimal remote configuration is called remote configuration #1 and requires two different PC's, as depicted below in Figure 2.

A more general case is remote configuration #2, where N PC's are involved as illustrated by Figure 84. Communication between the different machines is over a network (e.g. Local Area Network, Intranet, Internet). Note that if the Internet is the intervening network then the machines could be a world apart!

Finally, the most general case is remote configuration #3, which has multiple WinCon Servers and multiple WinCon Clients running as nodes on a TCP/IP network. For example WinCon Server S1 can download code to WinCon Client C1 and other code to WinCon Client C2. WinCon Server S2 can then connect to WinCon Client C1 and observe data in real-time from it. WinCon Server S2 can also download code to WinCon Client C3. The difference between WinCon Servers S1 and S2 is that S1 is a **Primary Server** for WinCon Clients C1 and C2 and can start and stop the controllers it downloaded. WinCon Server S2, however, is a **Secondary Server** for WinCon Clients C1 and C2 and can only collect data from them. It is, however, the primary server for WinCon Client C3.

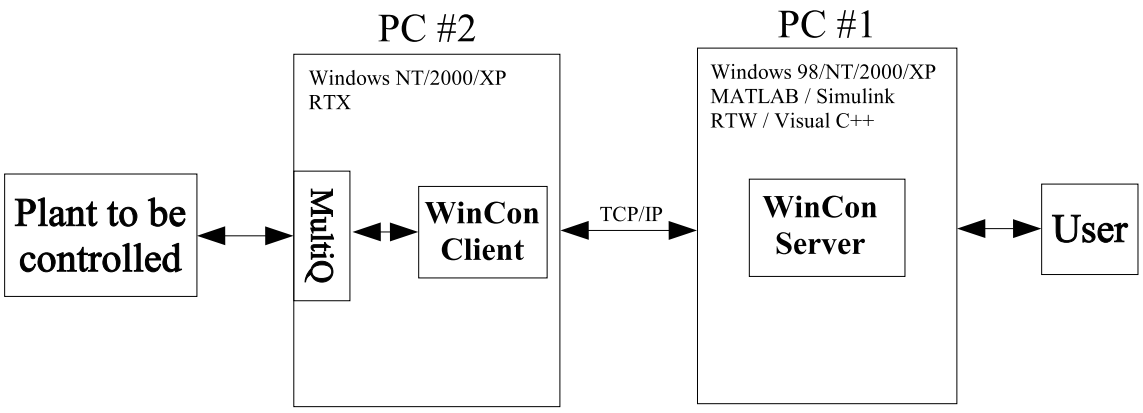


Figure 2 Remote configuration #1: 2 PC's

The advantage of these configurations is that PC's #2 to #(N-1) are not burdened by any other tasks. Thus, you can achieve the fastest possible performance in these configurations. You could, of course, get similar performance in the local configuration because the real-time code takes precedence over everything else – but for very fast sampling rates the controller would consume all the CPU time, leaving no time left for plotting or changing parameters! The remote configuration avoids this problem by moving the user interface portion to another machine.

These configurations are also convenient for remote monitoring and tuning of the plants. For example, WinCon Server could be running in a remote office while the WinCon Clients are located in a factory or laboratory setting.

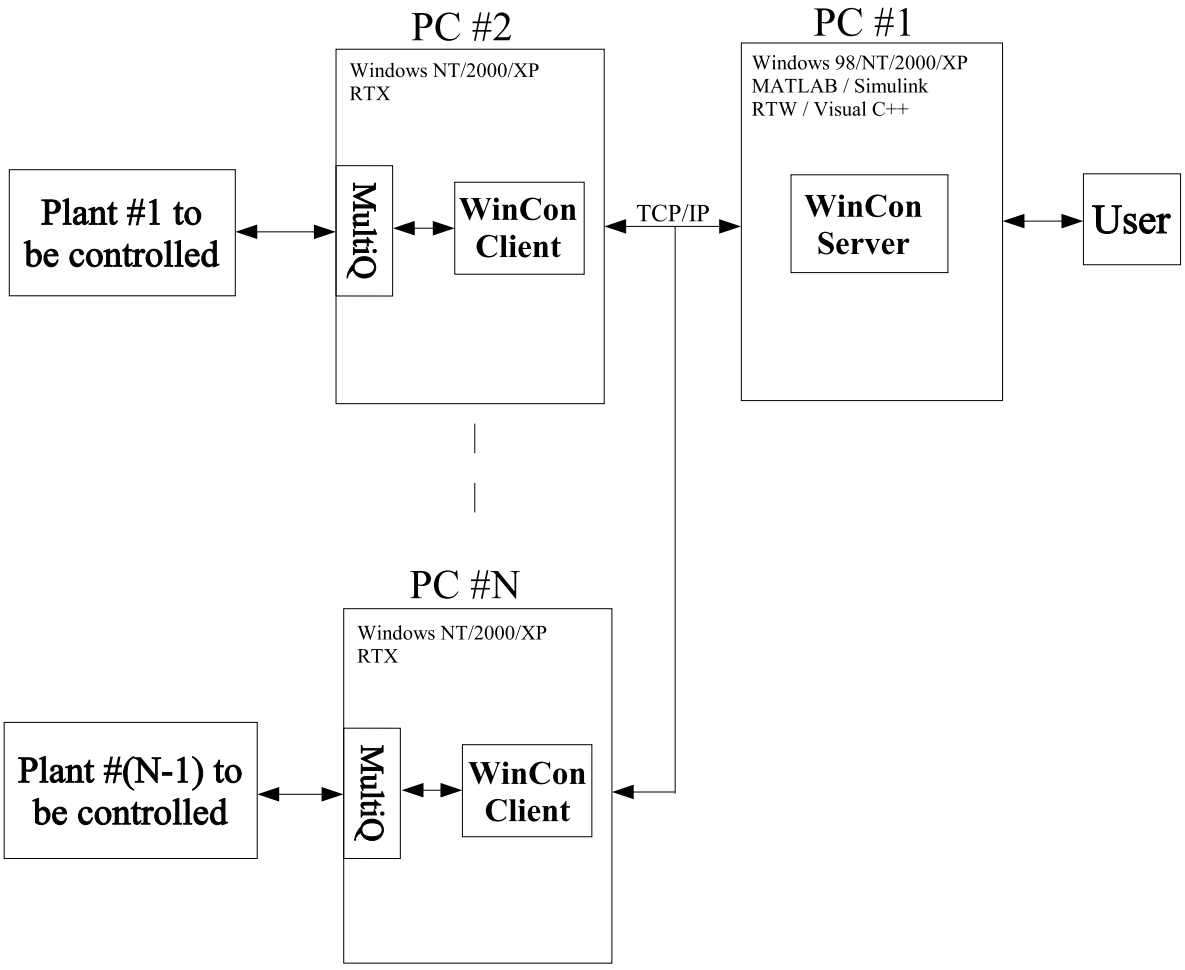


Figure 3 Remote configuration # 2: N PC's

WinCon Requirements

This chapter describes the software that must be installed prior to installing WinCon. It is important that the instructions in this document be followed carefully.

Note: The installation is designed to be relatively straightforward, but failing to read this guide can make things unnecessarily complicated!

Hardware Compatibilities

The standard data acquisition cards supported by WinCon are:

- The Quanser MultiQ-3 (a.k.a. MQ3) series.
- The Quanser MultiQ-PCI series.
- The Keithley MetraByte DAS-1600/1400 series.
- The Quanser MultiQ (a.k.a. MultiQ-2) series.

If you have an I/O board other than the ones specified above that you would like to use, you must obtain or write your own device drivers, as Simulink blocks, for it and link them with WinCon. Please refer to the *Real-time Workshop User's Guide* for details on writing drivers. Quanser Consulting can assist you in writing drivers for a nominal fee. Please contact Quanser Consulting directly to inquire about this service.

As implied in Section Local and Remote Configurations, the data acquisition card of your choice MUST be installed on the PC connected to the plant to be controlled, which is also where WinCon Client runs. In the local configuration, WinCon Server is also installed on that same PC.

Software Requirements

Depending on the WinCon Server / WinCon Client configuration, the following software packages don't have to be all installed on the same PC.

Installation Checklist for WinCon Server

WinCon Server runs under Windows 98/NT/2000/XP. WinCon can run its own Controller Library files (i.e. *.wcl files) and projects (i.e. *.wcp files) as a standalone system. However, to edit and modify/re-design controllers and re-compile real-time code, WinCon Server uses Simulink® and Real-Time Workshop® (RTW) by The MathWorks, and Visual C++® by Microsoft. If you want to design your own real-time controllers, these associated packages must be installed prior to installing WinCon Server. If you have not installed

these packages yet, then read the Sections Windows Installation, Microsoft Visual C++ Installation, and MATLAB Installation for suggestions on installing these programs that will make the installation of WinCon easier. The following checklist indicates the different packages that you must have installed on your computer in order to install WinCon.

- ❑ Microsoft **Windows 98/NT/2000/XP**.
- ❑ Microsoft **Visual C++ 5.x or 6.x** Professional and Enterprise editions.
- ❑ **MATLAB** Version **5.3.x or 6.x** (i.e. Release 11.x or 12.x, respectively), with Simulink and the Real-Time Workshop.

Installation Checklist for WinCon Client

WinCon Client also runs on a PC with Microsoft Windows 98/NT/2000/XP. Under Windows NT/2000/XP, WinCon Client employs the RTX® real-time kernel for Windows provided by VenturCom. Hence under Windows NT/2000/XP, RTX has to be installed for WinCon Client to run. If the MultiQ-PCI is used under Windows 98, the driver for the Sensoray 626 board MUST be installed. These associated packages must be installed prior to installing WinCon Client. If you have not installed these packages yet, then read the Sections Windows Installation, RTX Installation, and Installing the Sensoray SDK 626 Driver for suggestions to make the complete installation easier. The following checklist indicates the different packages that you must have installed on your computer in order to install WinCon Client.

- ❑ Microsoft **Windows 98/NT/2000/XP**.
- ❑ VenturCom **RTX** RunTime (RT) or Software Development Kit (SDK) version, when installing WinCon Client under **Windows NT/2000/XP**.
- ❑ Sensoray **SDK 626 driver**, with **Windows 98 and the MultiQ-PCI**.

Windows Installation

WinCon Client and WinCon Server can run on a different PC, and each PC can run a different version of Microsoft Windows amongst 98/NT/2000/XP.

However, when RTX is to be used (i.e. for Windows NT/2000/XP), it is recommended that the **latest Windows Service Pack (SP)** corresponding to your Operating System (O/S) be installed. No non-Microsoft modifications to the installation should be made. This is important as RTX requires a default installation of NT, 2000, or XP.

The **TCP/IP protocol** is a required element in order for WinCon to run. This is typically present whenever a modem or network interface card (NIC) is installed in the PC. It is also installed automatically with Microsoft Windows 98, NT, 2000, and XP. To check for the installation under Windows 2000, for example, go to: *Start* | *Settings* | *Control Panel* | *Network and Dial up Connections* | *Local Area Connection* | *Properties*.

RTX Installation

RTX is required by WinCon Client to run under Windows NT/2000/XP. The following concerns only a **uniprocessor** (UP) installation of the **RTX RunTime (RT) Environment** from VenturCom. Please contact Quanser Sales for multiprocessor (MP) and/or RTX SDK licencing.

Software Requirements

The minimum required RTX versions for each operating system are presented in Table 1 below. For complete system requirements information, please refer to the *RTX Installation Guide* provided on the CD, and, in particular, to the file entitled *RTX 5.1 Release Notes.pdf*.

Supported Version of Microsoft Windows	*RTX Version*
Windows XP Professional Edition	RTX 5.1
Windows 2000 Server and Professional Ed., SP2	RTX 5.1
Windows NT 4.0 Workstation and Server Ed., SP6a	RTX 5.1
Windows 2000 Server and Professional Ed., SP1	RTX 5.0
Windows NT 4.0 Workstation and Server Ed., SP6	RTX 5.0
Windows NT 4.0 Workstation and Server Ed., SP6	RTX 4.3.2.1
Windows NT 4.0 Workstation and Server Ed., SP3 or 4	RTX 4.2

Table 1 RTX Compatibility Table

Installing RTX

At this point, please review the applications that are currently installed on your system. You should ensure that there are no applications that will run a real-time process. This includes, but is not limited to the following:

- LabView with real-time control components.
- Real-Time Windows Target.
- Any multimedia audio or video capture or encoding programs.
- Virus Protection software that has a real-time monitoring function.
- Activity logging or monitoring software may also be problematic.

These items should ideally be deleted or at the very least completely disabled and removed from the Windows Start menu before proceeding to the RTX installation. Failure to

complete this step will likely result in a failed installation and/or inoperation of the core real-time process you are trying to set-up.

Full Administrator privileges are required for successfully installing and uninstalling the RTX products.

CAUTION:
Please refer to the *RTX Installation Guide* provided on the CD (for example, in the file *RTX 5.1 Release Notes.pdf*) for a fully detailed installation procedure.

In general, the RTX license shipped with the CD concerns only the **uniprocessor RTX RunTime (RT) Environment** (and not the SDK version). For example, to install RTX 5.1 RT, double click on the file *RTX 5.1 Runtime.exe* and follow the instructions. Install to the default path on the C drive and reboot after completion.

Do not attempt to install the RTX SDK (i.e. Do not install *RTX51.exe*), unless you have acquired a license for the RTX SDK.

To check the success of the RTX installation, run the System Response Time Measurement utility program, *srtm.rtss*. To do so, use Windows Explorer to launch the program: *C:\Program Files\VenturCom\RTX\Samples\srtm.rtss*. The program produces a steady 1kHz tone for 15 seconds. To run properly, this sample application must be executed from an account with full administrative privileges.

If you wish to **upgrade** your version of the VenturCom RTX, it is imperative that all the WinCon components present on your system be uninstalled *before* removing and upgrading RTX. You may then proceed to either re-install your previous WinCon components or upgrade to the latest WinCon. Please refer to Section Uninstalling or Upgrading WinCon for complete details.

Configuring RTX

Please refer to the *RTX Runtime Documentation* (e.g. The file *RTX 5.1 Runtime.pdf*) for complete details on the RTX configuration. To configure RTX, launch the application: *Start | Programs | VenturCom RTX | RTX Properties*, and select the *Settings* tab. The dialog depicted in Figure 4 will be displayed. Adjust the settings as follows:

- The RTX Startup setting **must** be configued as on *Demand*.
- Change the HAL timer period to **100 microseconds**. The HAL extension timer period corresponds to the minimum sampling interval that a controller can achieve when it is based on the software timer. In this case, the maximum sampling frequency is 10kHz. Faster sampling rates can be achieved by using a hardware clock. **If the Windows clock no longer runs at the proper time when RTX is**

running, then adjust the HAL timer period to compensate. Some versions of RTX fail to update the system time correctly for certain values of the HAL time period. The buttons available under the Control tab may be used to start and stop RTX.

- ❑ In the RTX Properties window, also select the *Debug* tab and make sure that the *Process Exception Disposition* is set to *Terminate the faulting process*.

Restart the computer after clicking OK to save the new settings.

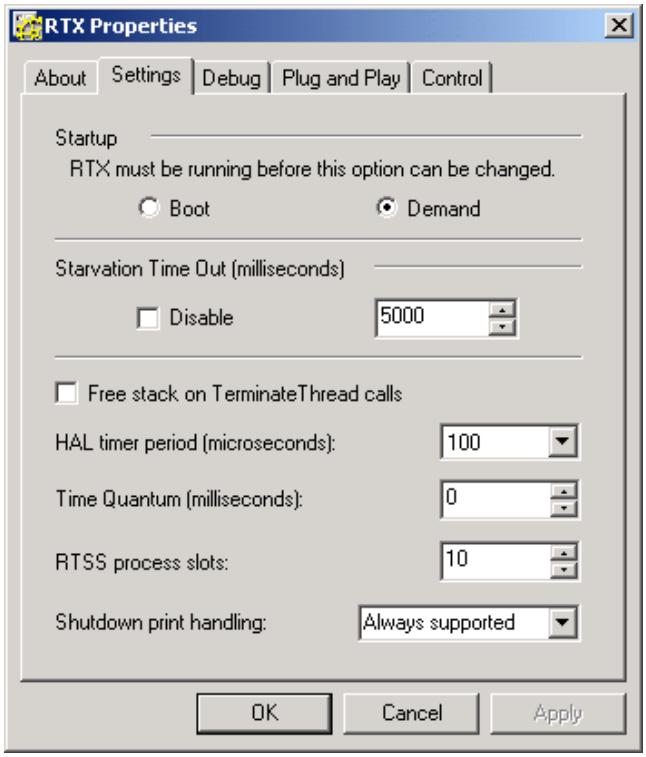


Figure 4 RTX Properties Window

Microsoft Visual C++ Installation

Software Requirements

The PC running WinCon Server must have **Microsoft Visual C++ Professional or Enterprise** Edition installed.

The Student, or Learning, Edition is not compatible.

The version of Microsoft Visual C++ that is required on the WinCon Server machine **depends on the operating system under which WinCon Client is running**, since the real-time code generated on the WinCon Server machine will be executed on the WinCon Client computer.

If WinCon Client runs under **Windows 98** (i.e. RTX is not required), **Microsoft Visual C++ 5.x and 6.x** are supported.

If WinCon Client runs under **Windows NT/2000/XP**, determine the version of RTX that is installed on the WinCon Client computer (also refer to section RTX Installation). Table 2 enumerates the minimum requirements for the Microsoft Visual C++ compiler according to the RTX version installed on the WinCon Client computer. For complete software requirements, please refer to the *RTX Installation Guide*, provided on the CD.

RTX Version	*Supported Version of Microsoft Visual C++*
RTX 5.1	Microsoft Visual C++ 6.0 Professional and Enterprise Editions, SP 5
RTX 5.0	Microsoft Visual C++ 6.0 Professional and Enterprise Editions, SP 4
RTX 4.3.2.1	Microsoft Visual C++ 6.0 Professional and Enterprise Editions, SP 3
RTX 4.2	Microsoft Visual C++ 5.0 or 6.0 Professional and Enterprise Editions

Table 2 Microsoft Visual C++ Compatibility Table

Installing Visual C++

Please refer to the *Microsoft Visual C++ Installation Guide* to properly install the version of the Microsoft Visual C++ compiler required for your system. Any questions regarding the installation of Visual C++ should be directed to the Microsoft technical support team – not to Quanser Consulting.

WinCon will continue to be upgraded to the most recent Visual C++ releases. There are currently no plans to support other compilers. Previous versions of WinCon supported the Borland and Watcom compilers. However, with the advent of Windows '95, support for these compilers had to be discontinued due to their lack of support for generating virtual

device drivers (*.vxd files) without additional 3rd party tools. In constrast, Visual C++ provided the necessary support and thus was adopted in their stead.

MATLAB Installation

Software Requirements

The PC running WinCon Server must have a compatible version of The MathWorks' MATLAB installed, in addition to Simulink, and the Real-Time Workshop toolbox. WinCon presently supports **MATLAB 5.3.x or 6.x** (i.e. Release 11.x or 12.x, respectively) with the corresponding Simulink (i.e. Version 3.0.x or 4.x, respectively) and Real-Time Workshop (i.e. Version 3.0.x or 4.x, respectively). Additionally, the Control System Toolbox can be useful for controller design.

Note that WinCon only supports the MATLAB full release – the student version of MATLAB is not compatible.

To check the MATLAB, Simulink, and toolbox versions, execute the **ver** command at the MATLAB prompt. For example:

```
>> ver
-----------------------------------------------------
MATLAB Version 6.1.0.450 (R12.1) on PCWIN
MATLAB License Number: 116767
-----------------------------------------------------
MATLAB Toolbox Version 6.1 (R12.1) 18-May-2001
Quanser Consulting Device Drivers Version 1.2.0, 26-Sep-2000
WinCon Toolbox Version 3.2.0, 26-Sep-2000
Quanser Toolbox Version 1.2.0, 26-Sep-2000
Simulink Version 4.1 (R12.1) 06-Apr-2001
Real-Time Workshop Version 4.1 (R12.1) 18-May-2001
Real-Time Windows Target Version 2.1 (R12.1) 02-Feb-2001
MATLAB Compiler Version 2.2 (R12.1) 30-Mar-2001
Control System Toolbox Version 5.1 (R12.1) 18-May-2001
```

Installing MATLAB

Install MATLAB to the default path on the C drive. If there were ever any incorrect versions installed on your system, it is imperative that all evidence of these installs be removed before proceeding to the new MATLAB installation. Please refer to the *MATLAB Installation Guide for Windows* for full details on the installation of MATLAB. Any questions regarding the installation of MATLAB should be directed to The MathWorks technical support team – not to Quanser Consulting.

CAUTION:

Perform a **local installation** ONLY. There should be **no** network shared components.

Configuring MATLAB

MATLAB must be configured to use the Microsoft Visual C++ (a.k.a. MSVC++) compiler, as follows.

Launch MATLAB and run **mex -setup** in the MATLAB command window to set the parameters for Real-Time Workshop (RTW). Refer to the *Real-Time Workshop User's Guide* supplied by The MathWorks for full details. Be sure to specify MSVC++ as the compiler of choice.

For example:

```
>> mex -setup
Please choose your compiler for building external interface (MEX)
files:

Would you like mex to locate installed compilers [y]/n? y

Select a compiler:
[1] Digital Visual Fortran version 6.0 in C:\Program Files\Microsoft
Visual Studio
[2] Lcc C version 2.4 in E:\MATLAB6P1\sys\lcc
[3] Microsoft Visual C/C++ version 6.0 in E:\Program Files\Microsoft
Visual Studio

[0] None

Compiler: 3

Please verify your choices:

Compiler: Microsoft Visual C/C++ 6.0
Location: E:\Program Files\Microsoft Visual Studio

Are these correct?([y]/n): y

The default options file:
"E:\Documents and Settings\Administrator.QCI\Application
Data\MathWorks\MATLAB\R12\mexopts.bat"
is being updated from E:\MATLAB6P1\BIN\WIN32\mexopts\msvc60opts.bat...

Installing the MATLAB Visual Studio add-in ...

  Updated E:\Program Files\Microsoft Visual
Studio\common\msdev98\template\MATLABWizard.awx
    from E:\MATLAB6P1\BIN\WIN32\MATLABWizard.awx
  Updated E:\Program Files\Microsoft Visual
Studio\common\msdev98\template\MATLABWizard.hlp
```

```
    from E:\MATLAB6P1\BIN\WIN32\MATLABWizard.hlp
  Updated E:\Program Files\Microsoft Visual
Studio\common\msdev98\addins\MATLABAddin.dll
    from E:\MATLAB6P1\BIN\WIN32\MATLABAddin.dll
  Merged E:\MATLAB6P1\BIN\WIN32\usertype.dat
    with E:\Program Files\Microsoft Visual
Studio\common\msdev98\bin\usertype.dat

Note: If you want to use the MATLAB Visual Studio add-in with the
MATLAB C/C++
      Compiler, you must start MATLAB and run the following commands:

      cd(prefdir);
      mccsavepath;

      (You only have to do this configuration step once.)
```

Installing the Sensoray SDK 626 Driver

With **Windows 98** and the **MultiQ-PCI data acquisition system**, the Sensoray SDK 626 Windows driver *must* be installed on the PC running WinCon Client. It *must* also be installed **prior to** WinCon Client.

The SDK 626 driver installation package is provided on the WinCon installation CD. However, it may not be the latest version. The latest version should be downloaded from the Sensoray web site at: http://www.sensoray.com/html/download_file.htm. Scroll down the web page and click on the SDK626 link to download the file *sdk626.zip*. After unzipping it, follow the installation procedure described in the file *.\Windows\626 DLL Manual.PDF*.

Installation Notes Regarding WinCon

Any virus scanner should be disabled during the installation procedure of any WinCon component (i.e. Server, Controls, and Client). Once completed, the WinCon installation can be tested using the WinCon example files: type *wcdemos* in the MATLAB command window.

Local Configuration

In the local configuration, WinCon Server and WinCon Client are installed on the same PC, as defined in section Local and Remote Configurations. In this situation, the WinCon installation steps are briefly enumerated below:

Step1. Install WinCon Server.

Step2. Install WinCon Controls.

Step3. Install WinCon Client.

Step4. Restart the computer.

These steps are fully detailed in the next sections.

Remote Configuration

In a remote configuration, WinCon Server and WinCon Client are installed on different PC's, as defined in section Local and Remote Configurations. There can only be one running WinCon Server per PC and one running WinCon Client per PC (due to the use of a dedicated TCP/IP port for Client-Server communications). However, each PC can run a different version of Microsoft Windows, amongst 98/NT/2000/XP.

Be sure to check each PC network connection. Prior to installation of any network configuration of WinCon, you must ensure that each PC involved in the WinCon operation is on a fully functional network with the TCP/IP protocol installed and configured. If this is not the case, contact your Network Administrator or the person(s) responsible for the network and inform them of your requirements.

Ensure that you have the **correct number of licenced copies of WinCon** for any network installation. One WinCon license is required per PC on which a WinCon component is installed (i.e. either WinCon Server, or WinCon Client, or both).

In the case of the very general Remote Configuration #3 described in Local and Remote Configurations (with multiple WinCon Servers and multiple WinCon Clients), the WinCon installation steps are briefly enumerated below:

Step1. For each PC interacting with the user, install WinCon Server and WinCon Controls, and then restart the computer.

Step2. For each PC connected to a plant to be controlled, install WinCon Client and restart the computer.

These steps are fully detailed in the next sections.

Installing WinCon Server

If you are upgrading WinCon, please refer to section Uninstalling or Upgrading WinCon. Older versions of WinCon must be uninstalled prior to installing a new version.

In the local configuration (i.e. one PC), **WinCon Server** *must* be installed **before WinCon Client**. As described in section Local and Remote Configurations, WinCon Server runs on the user's PC. If you intend to generate real-time code for WinCon Client, please ensure that the appropriate versions of MATLAB, Simulink, and Real-Time Workshop, as well as RTX (for Windows NT/2000/XP) and Microsoft Visual C++ are installed on the system.

The installation steps for WinCon Server are as follow:

Step1. Insert the **WinCon Installation CD** into the CD-ROM drive.

Step2. In the Windows Explorer window, double-click on **D:\WinCon Server 3_3\SETUP.EXE** to launch the WinCon Server installer. If your CD-ROM is not drive D, then substitute the appropriate drive letter.

Step3. Follow the installation instructions. Make sure the WinCon Server **serial number** has been entered correctly. The serial number is supplied to you on a separate form.

Step4. Choose the **Setup Type** corresponding to the configuration you desire. *Typical* is recommended. In the *Typical* configuration, all the components necessary to run WinCon Server and to generate real-time code from a Simulink diagram are installed. In the *Compact* configuration, WinCon Server runs without the possibility of generating real-time code. In the *Remote* configuration, WinCon Server can build real-time code for a remote WinCon Client (i.e. located on another machine). For example, the latter configuration is standard for the Wireless Ethernet Embedded Control System (WEECS).

Step5. **Choose to restart your computer LATER** if you intend to install WinCon Controls. Proceed to the installation of WinCon Controls, as described in Section Installing WinCon Controls, on the same PC. If you do not intend to install WinCon Controls, but choose to install WinCon Client on the same PC (i.e. local configuration), also elect to restart your computer later and proceed to the installation of WinCon Client, as described in Section Installing WinCon Client.

Step6. **Choose to restart your computer NOW** if you do not intend to install WinCon Controls or WinCon Client on the same PC. Proceed to the WinCon Client installation, as described in Section Installing WinCon Client, on the other PC.

Installing WinCon Controls

If you are upgrading WinCon, please go to the section entitled Uninstalling or Upgrading WinCon to uninstall WinCon Controls before installing a newer version.

WinCon Controls must be installed *after* WinCon Server, and must always be installed on the same PC. WinCon Controls adds ActiveX control widgets for the WinCon Server Control Panel. These controls allow model parameters to be changed on the fly, while the real-time code is running, without the use of Simulink.

The installation steps for WinCon Controls are as follows:

Step1. Insert the **WinCon Installation CD** in the CD-ROM drive.

Step2. In the Windows Explorer window, double-click on **D:\WinCon Controls 3_3\SETUP.EXE** to launch the WinCon Controls installer. If your CD-ROM is not drive D then substitute the appropriate drive letter.

Step3. Follow the installation instructions. Make sure the WinCon Controls **serial number** has been entered correctly. The serial number is supplied to you on a separate form.

Step4. **Choose to restart your computer LATER** if you intend to install WinCon Client on the same computer (i.e. local configuration). Proceed to the installation of WinCon Client, as described in section Installing WinCon Client.

Step5. **Choose to restart your computer NOW** if you do not intend to install WinCon Client on that same computer (i.e. remote configuration). Proceed to the WinCon Client installation, as described in section Installing WinCon Client, on the other PC.

Installing WinCon Client

If you are repairing or upgrading an existing WinCon installation, please go to section Uninstalling or Upgrading WinCon to uninstall WinCon prior to installing the newer version.

WinCon Client is always installed on the PC directly connected, through the data acquisition board of your choice, to the plant to be controlled. If the local configuration is used, WinCon Client should be installed on the same PC as WinCon Server. In remote configuration #3 (i.e. multiple WinCon Servers and multiple WinCon Clients), the WinCon Client installation procedure needs to be repeated on as many PC's as required, depending on the number of plants to be controlled. In the latter configuration, please ensure that you have the **correct number of licenced copies of WinCon**. One WinCon license is required per PC on which a WinCon component is installed (i.e. either WinCon Server, or WinCon Client, or both).

Under Windows NT/2000/XP, the WinCon Client Installation also installs **WinCon Service**, described in section WinCon Service. WinCon Service handles the loading of real-time code on behalf of WinCon Client. WinCon Service thus allows **regular users**, i.e., *without* Administrative privileges, to run real-time controllers. The WinCon Client installer also installs the **WinCon Task Manager** application, which is described in Section WinCon Task Manager.

The installation steps for WinCon Client are as follow:

Step1. Insert the **WinCon Installation CD** in the CD-ROM drive.

Step2. In the Windows Explorer window, double-click on **D:\WinCon Client 3_3\SETUP.EXE** to launch the WinCon Client installer. If the CD-ROM is not drive D, then substitute the appropriate drive letter.

Step3. Follow the installation instructions. Make sure the WinCon Client **serial number** is entered correctly. The serial number is supplied to you on a separate form.

Step4. Choose to **restart** your computer now, under Windows NT/2000/XP, to get WinCon Service running.

Uninstalling or Upgrading WinCon

Upgrading or Repairing WinCon

If you are upgrading or repairing an existing version of WinCon installed on your system, you ***must*** completely uninstall WinCon ***first*** (i.e. WinCon Server, WinCon Controls, and WinCon Client) as described in the section entitled Uninstalling WinCon. In the case of an upgrade, ensure that your system specifications still meet the software requirements of the latest WinCon; otherwise, you will have to upgrade to the latest WinCon pre-requisites (e.g. VenturCom RTX, Microsoft Visual C++) before continuing any further. **Note that WinCon should be uninstalled *prior* to upgrading Matlab, RTX or Visual C++ to meet the requirements of the WinCon upgrade**. Please refer to the *Installation Guide* particular to the software(s) you wish to uninstall and/or upgrade. Once the system specifications are met, you can proceed to the installation of WinCon. To do so, refer to the following preceding sections: Installing WinCon Server, Installing WinCon Controls, and Installing WinCon Client.

Uninstalling WinCon

Full Administrator privileges are required to uninstall (and re-install) WinCon. To uninstall all the different components of the WinCon software, follow the steps described below:

Step 1. **Exit WinCon Link**. On the PC running WinCon Server, the WinCon Link icon is found in the system tray (near the clock). To terminate WinCon Link, either left-click on the icon to open its window and select *Exit*, or right-click on it and choose *Exit* from the context menu that appears. Close all open applications.

Step 2. **Uninstalling WinCon Server**. On the PC running WinCon Server, open the *Start* | *Settings* | *Control Panel* | *Add/Remove Programs* window. Select, for example, *WinCon Server 3.2*, and click on the *Change/Remove* button to remove it from that computer. Click *Yes* to the *Confirm File Deletion* window to proceed with the removal of WinCon Server. If a warning message appears indicating that WinCon Link is still running, abort the WinCon Server uninstall, close the Add/Remove Programs window and Control Panel and return to *Step 1*, to first shut down WinCon Link, before proceeding any further.

Step 3. **Uninstalling WinCon Controls**. On the PC running WinCon Controls, (the same PC as WinCon Server), open the *Start* | *Settings* | *Control Panel* | *Add/Remove Programs* window. Select, for example, *WinCon Controls 3.2*, and click on the *Change/Remove* button to remove it from that computer. Click *Yes* to the *Confirm File Deletion* window to proceed with the removal of WinCon Controls. Click *Yes To All* in the *Remove Shared File?* warning window, as shown in Figure 5, to remove all of files corresponding to Control Panel widgets,

namely: *QVSlider.ocx*, *QEdit.ocx*, *QKnob.ocx*, *QSelect.ocx*, *QSlider.ocx*, *QButton.ocx*, *QText.ocx*, and *QPicture.ocx*. Also click *Yes* to the *Remove Shared File?* confirmation window. The uninstall will then complete automatically.

Step 4. **Reboot**. To finish the WinCon Server uninstall, it is recommended that the computer be restarted. **If the uninstaller indicates that not all files could be removed, then the computer *must* be rebooted prior to installing another version of WinCon Server.** Otherwise, the new installation may be corrupted because the uninstaller removes problematic files the first time the computer is restarted.

Step 5. **Uninstalling WinCon Client**. On the PC running WinCon Client (possibly the same PC as WinCon Server), open the *Start* | *Settings* | *Control Panel* | *Add/Remove Programs* window. Select, for example, *WinCon Client 3.2*, and click on the *Change/Remove* button to remove it from the computer. Click *Yes* to the *Confirm File Deletion* window to proceed with the removal of WinCon Client.

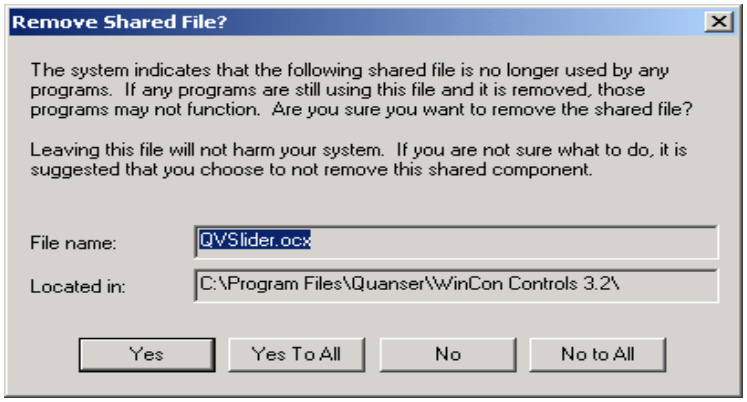


Figure 5 Remove Shared File? Window

WinCon Principles of Operation

WinCon Server

WinCon Server is the software component that performs the following functions:

- ❑ Conversion of a Simulink diagram to C source code using The MathWorks' Real-Time Workshop (RTW).
- ❑ Compilation and linking of the generated code, using Microsoft Visual C++, to create a real-time WinCon Controller Library (*.wcl) executable.
- ❑ Downloading the WinCon Controller Library file (.wcl) to WinCon Client for execution.
- ❑ Starting and stopping the real-time code on WinCon Client (remotely, in the case of a remote configuration).
- ❑ Maintaining TCP/IP communications with any connected WinCon Clients.
- ❑ Maintaining communication with Simulink to perform real-time changes to controller parameters. For example, if value of a Gain block is changed in Simulink, WinCon Server passes this information to WinCon Client and the change is made immediately in the running controller.
- ❑ Making changes to controller parameters using user-defined Control Panels.
- ❑ Plotting the data streamed from the real-time code via WinCon Client.
- ❑ Saving real-time data to disk and/or the MATLAB workspace.

WinCon Client

WinCon Client is the real-time software component that runs the code generated from the Simulink diagram at the sampling rate specified. It can perform the following tasks:

- ❑ Reception of the controller code in the form of a WinCon Controller Library file (.wcl) from WinCon Server.
- ❑ Running the controller in real-time.
- ❑ Maintaining communications with any connected WinCon Servers.
- ❑ Streaming real-time data to any WinCon Server(s) requesting it.
- ❑ Management of a user-defined threshold setting to control the maximum amount of CPU time allotted to the real-time code.

WinCon Server's Graphical Interface

The WinCon Server graphical user interface is shown below, in Figure 6.

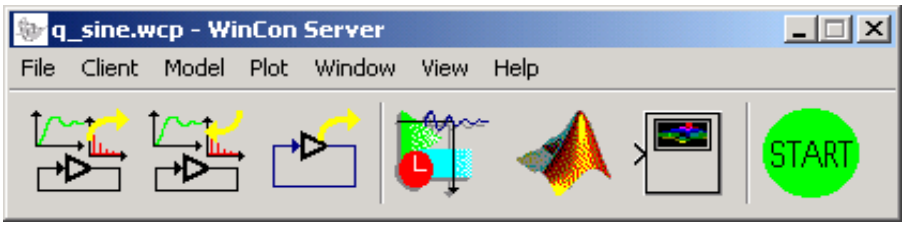


Figure 6 WinCon Server Graphical Interface

WinCon Server's *File* Menu Options

WinCon Server supports project files to save the plots and control panels designed by the user. The WinCon project files (i.e. .wcp) are managed through the use of WinCon Server's *File* menu, whose items are listed and described in Table 3.

File Menu Option	*Description*
New	Start a new WinCon project.
Open...	Open an existing WinCon project.
Save	Save a WinCon project.
Save As...	Save the current WinCon project under a new name.
Close	Close the current WinCon project.
[Recent Projects List]	Quick access list of recently opened projects.
Exit	Exit WinCon Server.

Table 3 WinCon Server's File Menu Options

WinCon Server's *Client* Menu Options

The WinCon Client information relevant to the Server is managed by using WinCon Server's *Client* menu options, which are enumerated in Table 4.

Client Menu Option	*Description*
Connect...	Connect to a desired client. Enter the name of the remote machine or its IP address and remote port.
Disconnect	Disconnect from the primary WinCon Client.
Recent Clients	List of the last three Clients to which the Server was connected.
Close on Exit	Close the local Client upon exit from the Server. (A local Client is run automatically by the Server when needed.)
Threshold...	Set the CPU time Threshold for the current Client.
Network...	Launches Microsoft Windows' Network dialog interface in order to obtain information about the network.
[Clients List]	List of Clients that are currently connected to this Server. A check mark (i.e. √) at the beginning of the Client's address indicates that this Client is the currently active Client. All operations act upon the currently active Client only, with the exception of the Start All and Stop All commands. A star (i.e. *) at the end of the name indicates that this is a Secondary Server for that Client. A secondary server cannot change controller parameters.

Table 4 WinCon Server's Client Menu Options

WinCon Server's *Model* Menu Options

The Simulink model files and the WinCon Controller files are managed via the WinCon Server's *Model* menu options, which are listed and described in Table 5.

Model Menu Option	*Description*
Open...	Open a WinCon controller (.wcl) or a Simulink model (.mdl).
Close	Close the current model.
Reload	Reload the current model.

Model Menu Option	*Description*
Build	Build the real-time code for the currently active model. Launch Simulink if it is not already up and running. The generated code is downloaded automatically to the currently active Client.
Download	Download the real-time code for the currently active model to the currently active Client.
Start	Start running the downloaded code in the currently active Client.
Stop	Stop running the real-time code in the currently active Client.
Start All	Start the real-time code in all Clients currently connected.
Stop All	Stop the real-time code in all Clients currently connected.
[Models List]	List controllers which are loaded into Clients. A controller has a check mark (i.e. √) if it is the currently active model. All operations refer to the currently active model, on the currently active Client, with the exception of the Start All and Stop All commands.

Table 5 WinCon Server's Model Menu Options

WinCon Server's *Plot* Menu Options

The WinCon real-time plots are managed through use of the WinCon Server's *Plot* menu, whose commands are enumerated in Table 6. For detailed information on the plotting capabilities of WinCon, refer to section Plotting On-Line Data.

Plot Menu Option	*Description*
New	Open a new plot. Possible types are: Scope, X-Y Graph, Digital Meter, and Thermometer. The display does not need to be defined in the Simulink diagram.
Open...	Open an existing Scope, Digital Meter, or Thermometer defined in the Simulink diagram.
Properties...	Open the Plot Properties dialog for setting the default properties of the Scope, X-Y Graph, Digital Meter and Thermometer.
Close All	Close all open plots.

Table 6 WinCon Server's Plot Menu Options

WinCon Server's *Window* Menu Options

The navigation to other windows or software related to WinCon is done through the WinCon Server's *Window* menu options, which are listed in Table 7.

Window Menu Option	*Description*
Always on Top	Select whether you want the WinCon Server Toolbar to stay on top of all other windows.
Matlab	Launch the MATLAB command window.
Simulink	Launch Simulink and open the Simulink diagram for the currently active model, if any.
Control Panel	Open the Control Panel Window, used for parameter tuning without Simulink.
External Interface Window	Open the External Interface Window (EIW) to allow other applications to connect to WinCon, and for external sinks and sources.
[Displays List]	Navigate through the different open displays.

Table 7 WinCon Server's Window Menu Options

WinCon Server's *View* Menu Options

The appearance of the WinCon Server window is controlled through using the WinCon Server's *View* menu options, which are listed and described in Table 8.

View Menu Option	*Description*
Toolbar	Show or hide the WinCon Server toolbar.

Table 8 WinCon Server's View Menu Options

WinCon Server's *Help* Menu Options

The WinCon Server's *Help* menu options are listed and described in Table 9.

Help Menu Option	*Description*
About WinCon Server...	Displays the WinCon Server version and build numbers.

Table 9 WinCon Server's Help Menu Options

WinCon Server's Toolbar Buttons

The shortcuts offered by the WinCon Server's toolbar buttons, are described in Table 10.

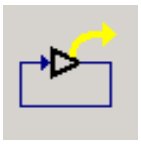
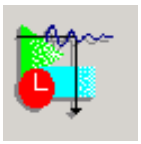
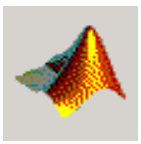
Toolbar Button	*Description*
	Open an existing WinCon Project.
	Save to a WinCon Project. It saves the running controller, plots, and control panels to a *.wcp* file. An *.ocp* file may also be saved as part of the project.
	Open a Simulink (.mdl) model or a WinCon (.wcl) Controller Library.
	Launch Simulink. Open the Simulink diagram associated with the current model, if any.
	Launch MATLAB. If a model is currently opened, start MATLAB in that model's directory.
	Open a Plot which is defined directly in the current (Simulink) model, through the use of a Scope, Display, Thermometer or X-Y Graph block.
	Start/stop the WinCon Controller (.wcl) on the currently active Client.

Table 10 WinCon Server's Interface Icons

WinCon Server's Keyboard Shortcuts

The possible WinCon Server's keyboard shortcuts, also known as hot-keys, are listed and described in Table 11.

Hotkey	*Action*
Ctrl + N	Create a new WinCon Project.
Ctrl + O	Open an existing WinCon Project.
Ctrl + S	Save a WinCon Project.
Ctrl + M	Open a Simulink (.mdl) model or WinCon Controller (.wcl).
Ctrl + R	Reload the current model.
Ctrl + B	Build the current model (to generate a .wcl file). Launch Simulink if it is not already up and running.
Ctrl + D	Download the real-time code for the current model (i.e. controller library file - .wcl) to the currently active Client.
Ctrl + P	Open a Plot which is defined in the current Simulink model.
Alt + Pause	Start the controllers in all connected Clients.
Pause	Stop the controllers in all connected Clients.

Table 11 WinCon Server's Keyboard Shortcuts

File Extensions Generated and Used by WinCon Server

WinCon Server generates and/or uses files with the extensions listed in Table 12.

File Extension	*Definition*
.wcp	WinCon Project file.
.wcl	WinCon real-time Controller Library file.
.ocp	WinCon Control Panel (using Active X) file.
.mdl	Simulink Model file.
.m	MATLAB script file.

Table 12 File Extensions used by WinCon Server

WinCon Client's Graphical Interface

The WinCon Client's graphical interface is depicted in Figure 7 below. The WinCon Client window is usually minimized (by default), as you do not normally need to interact with it. In the event that you wish to ensure that the WinCon Client is running, you can maximize this window and observe its content. The WinCon Client window displays some of the real-time performance variables discussed in section Timing of Real-Time Events. The window allows you to monitor the status of the real-time controller. A WinCon Client can only execute one model at a time.

The most commonly used feature is the *File* | *Network...* menu option, which provides network information for the PC running WinCon Client. The WinCon Client window also allows you to change the Threshold parameter, which ensures that enough processor time is allocated to foreground tasks, as discussed in section Choice of Threshold.

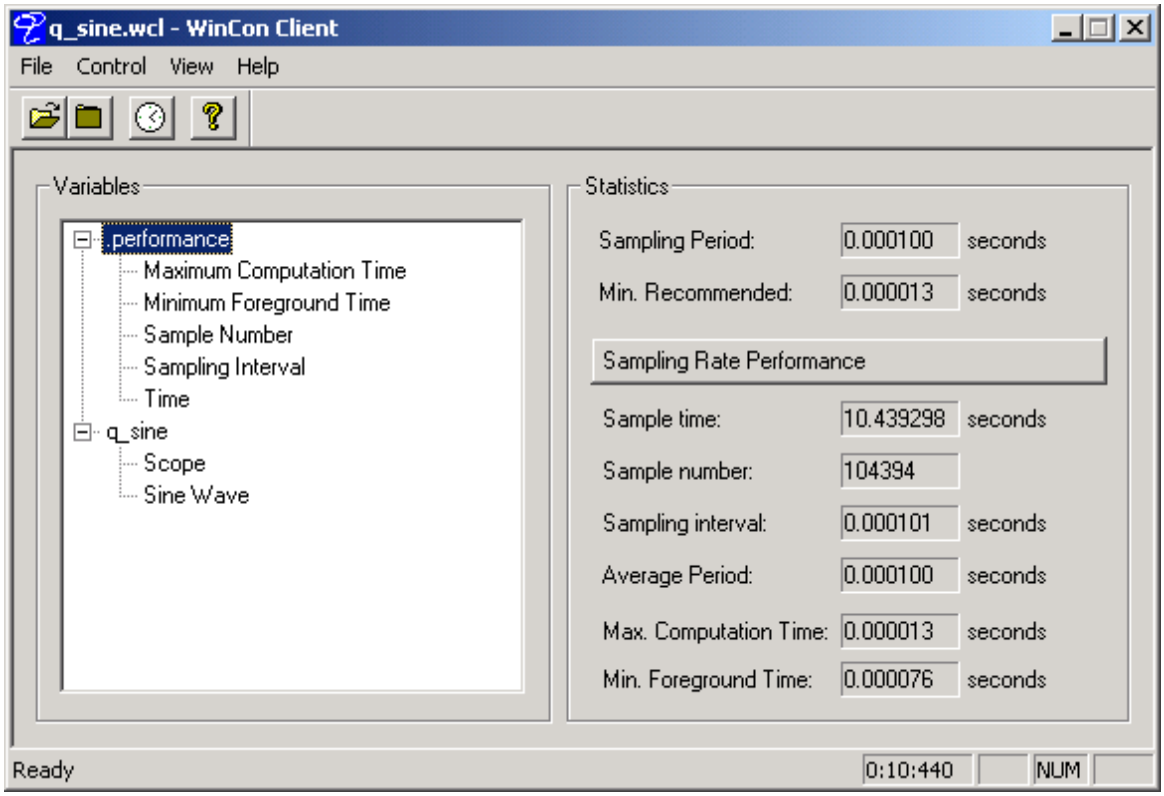


Figure 7 WinCon Client Graphical Interface

WinCon Client's *File* Menu Options

The WinCon Client's *File* menu options are listed and described in Table 13.

File Menu Option	*Description*
Open...	Open a WinCon controller (i.e. a .wcl file).
Close	Close the current WinCon controller. The controller is stopped prior to unloading it from the real-time kernel.
Servers...	Display a list of the WinCon Servers connected.
Network...	Obtain network information for the computer on which WinCon Client is running. It launches Microsoft Windows' Network dialog interface.
Port...	Set the TCP/IP port used for Client-Server communications. The default port is 17255. Only change the port number if you are running other software that interferes with the default port.
Checksum	Display the model's 128-bit checksum. It is used to verify that the loaded WinCon controller matches its associated Simulink diagram.
[Recent Controllers List]	List of recently loaded real-time controllers (i.e. .wcl files).
Exit	Exit the WinCon Client. The controller is stopped, if necessary, prior to exit.

Table 13 WinCon Client's File Menu Options

WinCon Client's *Control* Menu Options

The WinCon Client's *Control* menu options are listed and described below, in Table 14.

Control Menu Option	*Description*
Threshold...	Set the Threshold, which is the minimum processor time allotted to foreground tasks. If the running controller uses too much CPU time, WinCon Client stops the controller. The threshold value is saved whenever the Client is exited. It is not saved with the project. **Note that device drivers also steal processor time from foreground tasks. Hence, the threshold does not always reflect the processor time used up the controller alone, and it may be necessary to disable threshold checking by setting the threshold to zero on some systems.**
Frame Rate...	Frame rate for streaming data to WinCon Server [Hz]. WinCon Client streams data in "frames" to make data streaming more efficient over a network. This rate is different from the sampling rate at which data is collected, which is dependent on the sampling rate of the controller.
Start	Start the real-time controller.
Stop	Stop the real-time controller.

Table 14 WinCon Client's Control Menu Options

WinCon Client's *View* Menu Options

The appearance of the WinCon Client window is controlled via the WinCon Client's *View* menu options, which are listed and described below, in Table 15.

View Menu Option	*Description*
Toolbar	Show or hide the toolbar.
Status Bar	Show or hide the status bar.

Table 15 WinCon Client's View Menu Options

WinCon Client's *Help* Menu Options

The WinCon Client's *Help* menu options are listed and described below, in Table 16.

Help Menu Option	*Description*
About WinCon Client...	Display the version number of WinCon Client.

Table 16 WinCon Client's Help Menu Options

WinCon Client's Keyboard Shortcuts

The possible WinCon Client's keyboard shortcuts, also known as hot-keys, are listed and described in Table 17.

Hotkey	*Action*
Ctrl + O	Open a WinCon Controller (.wcl).
Ctrl + C	Close the current WinCon Controller.
Pause	Stop the running WinCon Controller.

Table 17 WinCon Client's Keyboard Shortcuts

WinCon Client's Displayed Statistics

As shown on Figure 7, the WinCon Client window is divided into two parts. The left-hand side displays all the variables defined in the loaded WinCon controller that may be plotted in WinCon Server. The right-hand side displays some statistics related to the running, by WinCon Client, of the real-time controller. These statistics provide key information for the evaluation of the real-time performance of the controller. The statistical parameters are described below, in Table 18. Many of these statistics are also available as variables that may be plotted in WinCon Server, to track their variation over time. These "performance" variables are accessible under the `.performance` tree when selecting a variable for plotting.

Displayed Statistics	*Description*
Sampling Period	Desired sampling period specified in the Simulink options.
Min. Recommended	Minimum recommended sampling period for the running controller based on the observed computation time, T_C, and specified Threshold. Running the controller faster than this rate will result in a foreground time, T_F, that will drop below the specified threshold and thus stop the controller.

Displayed Statistics	*Description*
Sample time	Elapsed time that the controller has been running. If this value does not match the actual elapsed time, according to a stop watch, then the PC is not fast enough to sustain the sampling rate specified for the controller.
Sample number	Integer number of samples taken during the sample time.
Sampling interval	Instantaneous sampling period, as measured by an independent clock source.
Average Period	Average of the actual sampling intervals, over the sample number.
Max. Computation Time	Maximum computation time, T_{C_max}, observed during the controller run. This gives a measure of the worst case computation delay.
Min. Foreground Time	Minimum foreground time, T_{F_min}, observed during the running of the controller.

Table 18 WinCon Client's Displayed Statistics

Discrete Time Considerations

Timing of Real-Time Events

The timing diagrams in Figure 8 illustrates the principles of real-time operation. The system or hardware clock/timer generates interrupts at fixed intervals T_S. T_S is the user-selected sampling period. Since the processor may be servicing other hardware interrupts or direct memory access (DMA) requests, it takes an indeterminate amount of time to service the interrupt. This duration is termed the latency period T_L. Foreground tasks indirectly affect this latency period as they make use of hardware resources, such as the video driver or hard disk. While all foreground operations, and even device driver operations, in Windows NT/2000/XP run at a lower priority than the real-time code, operations performed directly by the hardware, such as DMA, can still affect this latency period.

At the end of the latency period, the controller performs its computations for that sampling instant, taking a duration T_C. After the controller completes its calculations and I/O, the processor resumes executing the foreground tasks, such as Simulink, until another interrupt occurs. The effective sampling period, T_E, is the time taken between two Interrupt Service Routines (ISR's). This time is variable and depends on the latency time.

Tests indicate a **latency time in the tens of microseconds** under Windows NT/2000/XP, that is to say within the RTX real-time environment. This value depends on the foreground tasks that are running, their use of hardware resources, and the speed of the system. When running in remote configurations, where WinCon Client runs on a dedicated PC, the latency time can be below 10 microseconds. The computational delay, T_C, depends mostly on the complexity of the real-time controller that is running. Note that the scale of Figure 8 is exaggerated. Controllers that take over 50% of the processor time and result in such asymmetry are very unlikely.

One important factor to note is that the latency is typically relatively constant, reflecting the time required for the processor to recognize the assertion of an interrupt by the hardware time-base (either the PC's internal timer or a time-base on a data acquisition card), to load the interrupt vector and to invoke the interrupt service routine. Thus, if the latency is approximately 10 µs for the first sample, it will likely be around 10 µs for subsequent samples. Hence, the actual sampling period T_E is very close to the desired period T_S - the samples are only delayed by T_L. Furthermore, since the sampling period is ultimately governed by a hardware timer, the average sampling period is very accurate.

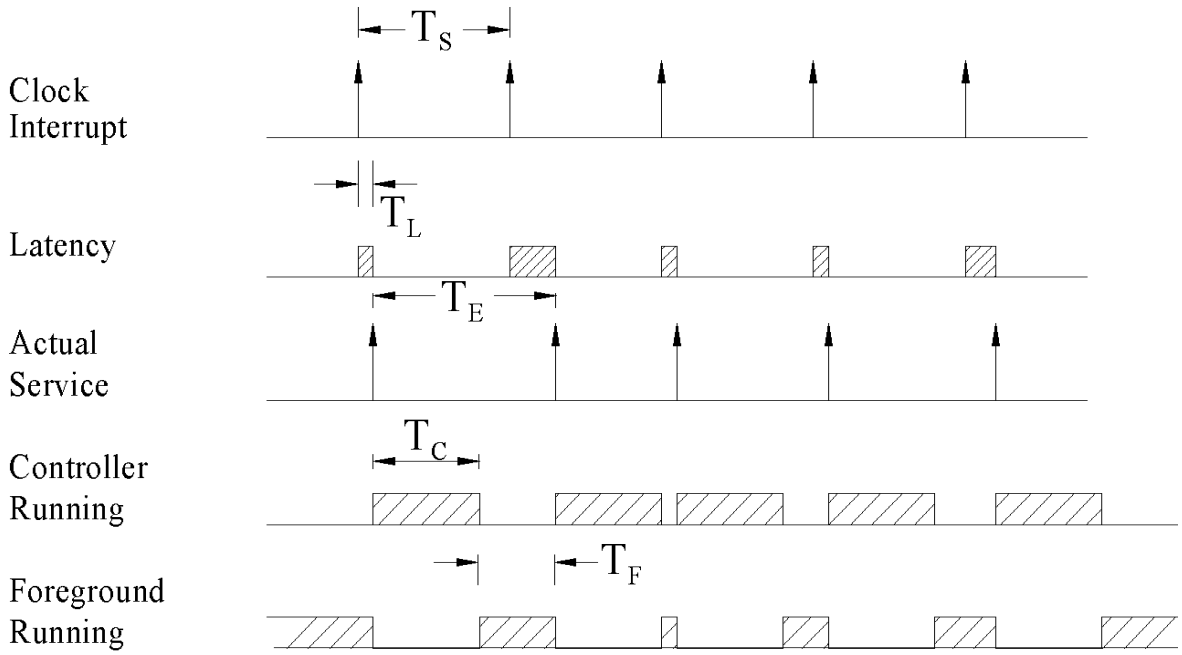


Figure 8 Timing of Real-Time Events

Choice of Sampling Period

Determining the fastest sampling frequency possible depends on how much computing power you are willing to relinquish to foreground tasks, where $T_F = T_E - T_C$ is illustrated by Figure 8, especially for real-time plotting. You should always start with the slowest possible sampling frequency for the process that you want to control. If you want faster sampling, increase it slowly and note how much this affects your foreground tasks. If the sampling frequency is too high, the mouse cursor and real-time plots slow down appreciably. This degradation is an indication that the controller is running too fast for the computer that is used by WinCon Client. The WinCon Client window gives statistics concerning the performance of the running real-time controller, as described in Section WinCon Client's Graphical Interface.

In WinCon, the sampling period is set as a Real-Time Workshop (RTW) option, by accessing the WinCon menu appearing in the controller Simulink diagram. As described in Section Setting Real-Time Workshop Options , the sampling period, T_S, is set by entering its value in seconds in the text field *WinCon* | *Options...* | *Solver* | *Fixed step size*.

Choice of Threshold

Each WinCon Client is assigned a *Threshold* parameter that ensures that there is enough foreground time allocated for the operating system (i.e. Windows 98/NT/2000/XP) to perform its diverse functions. The WinCon Client Threshold can be set in the Client window by selecting *Control* | *Threshold...* from the Client menu bar or *Client* | *Threshold...* from the Server menu bar. The threshold value is saved by WinCon Client only. It is not part of the WinCon project format. The functionality is as follows: the Client saves the threshold when it is exited. The next time it is run, the Client will restore the saved threshold. Hence, the threshold retains its most recent setting, even across sessions.

The foreground time T_F is computed at each sample as the time difference between the initiation of the present ISR and the completion of the previous ISR, as illustrated in Figure 8. The real-time controller is forced to stop if the foreground time drops below the specified threshold. Therefore, **WinCon Client stops the controller if: T_F < *Threshold.*** The default value for *Threshold* is **zero**. However, this value can be changed from either the WinCon Server or the WinCon Client window. The *Threshold* value is rather arbitrary but tests under Windows NT/2000/XP have shown that 20 μs is enough to allow foreground tasks to operate normally. Sometimes, the *Threshold* value will be crossed at startup only, and once the real-time code has been put into cache memory it will not happen again. In these cases, the real-time code will only be able to run after the first try. Selecting a *Threshold* value that is too low and/or a sampling rate that is too fast may crash your system. It is recommended that the equation $T_S = 20e\text{-}6 + T_C$ (in second) to be used as an indication of the fastest possible sampling period possible for a given controller.

Choice of Time Base

WinCon Client achieves the user-defined sampling period, as described in Section Setting Real-Time Workshop Options , by programming the PC's system timer or a hardware time-base in a data acquisition card, to generate interrupts at the prescribed frequency. The PC's system timer is referred to as the **System clock**. A timer on a data acquisition card is referred to as a **Hardware clock**.

1. **System Clock**. The system clock, or timer, derives a real-time interrupt from the Windows system timer. Under Windows NT/2000/XP, VenturCom RTX provides the real-time kernel environment. The real-time performance achieved using the system clock is comparable to that obtained when using a hardware clock. However, the sampling frequency achievable under RTX is limited to 10kHz, due to the minimum RTX HAL Timer period of 100 microseconds. For sampling rates higher than 10 kHz, with Windows NT/2000/XP, a hardware clock should be used. Under Windows 98, since the RTX environment is not present, the sampling rate is limited to 1kHz (a 1 millisecond sampling period).

For sampling rates higher than 1 kHz, a hardware clock should be used with Windows 98.

2. **Hardware Clock**. The hardware clock, or timer, is a PC-independent clock which is connected to an interrupt line on the PC bus. For example, the MQ3 board has three such clocks which can be used to generate interrupts. The Simulink-integrated drivers found in the Quanser Toolbox for Simulink includes Time-Base blocks to control the real-time clock of the MultiQ-2, MQ3, and MultiQ-PCI cards. To use a hardware clock, simply include its corresponding Time-Base block in the Simulink diagram.

   When including a MQ3 (or MultiQ-2) timer block in the diagram, ensure that the IRQ level selected in the block is compatible with the jumper configuration on the board. **The MQ3 board is configured for IRQ #5 timer #1 at the factory.** Also, ensure that no other devices are connected to the IRQ level you are using for the MQ3. Specifically, **ensure that your modem, network card or sound card does not conflict with the MQ3 board**.

   On the MultiQ-PCI board, there are six counters that can be used as a clock source. However, these counters are usually used to acquire encoder counts. Hence when using the Multi-PCI Time-Base block (under Windows 98 or NT), **ensure that the Clock Source parameter does not conflict with a channel number used in an Encoder Input block**.

Generating Real-Time Code

Before a Simulink model may be run in real-time, you must first generate the real-time code. WinCon Server employs Real-Time Workshop to generate the real-time code.

WinCon Link and WinCon Menu

After WinCon Server has been installed, and the computer rebooted, a *WinCon Link* icon, automatically appears in the Windows system tray, next to the clock, as shown in Figure 9. The WinCon Server installation configures WinCon Link to be automatically launched on start-up. WinCon Link creates a new Simulink menu, called *WinCon*, that appears in the Simulink window menu bar, as depicted in Figure 10. The WinCon menu enables you to generate and run the real-time code seamlessly from your Simulink block diagram. The *WinCon* menu options are listed and described below, in Table 19.

Figure 9 WinCon Link Icon

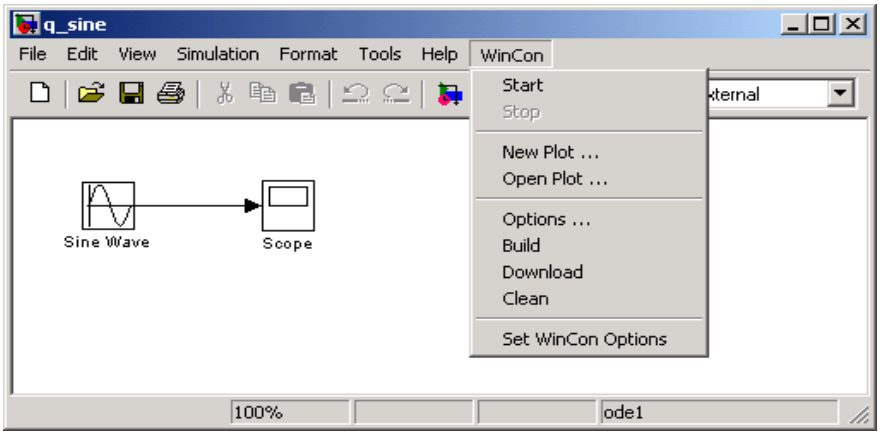


Figure 10 WinCon Menu in Simulink

WinCon Menu Option	*Description*
Start	Start the real-time controller (i.e. .wcl file).
Stop	Stop the running real-time controller.
New Plot...	Open the New Plot dialog box. Allows one to monitor the output of **any** Simulink block in the block diagram, through plotting of the data on a WinCon Scope, an X-Y Graph, a Digital Meter, or even a Thermometer.
Open Plot...	Open a Simulink Scope, Display, Thermometer, or WinCon X-Y Graph (found in the wctools/Extra Sinks library) defined in the controller block diagram, if any, as a WinCon Scope, Digital Meter, Thermometer or X-Y Graph respectively.

WinCon Menu Option	*Description*
Options...	Open the dialog box to configure the Real-Time Workshop settings. Same as selecting the menu item *Tools* | *Real-Time Workshop* | *Options...* in the Simulink window menu bar.
Build	Generate, compile and link the real-time controller code (i.e. .wcl file) equivalent to the Simulink block diagram.
Download	Download the real-time controller code to the currently active WinCon Client to which WinCon Server is connected.
Clean	Delete the temporary compilation directory where all the files generated by the Build command are located. This function forces a clean build (i.e. "from scratch"). This can be useful, since a normal Build only re-compile functions that have changed since the last Build. Occasionally however these functions may get corrupted and you will need to re-compile everything. In this case, simply click on the Clean command to delete all the intermediate files and to ensure that you are re-compiling all the necessary files.
Set WinCon Options	Choosing this menu option will automatically set the Real-Time Workshop Options to the WinCon defaults corresponding to your system.

Table 19 WinCon Menu Options

Setting Real-Time Workshop Options

Before building your WinCon real-time controller code from Simulink, you need to configure the Real-Time Workshop options appropriately. WinCon configures the default options for all new Simulink diagrams to have the appropriate options, so this step is normally unnecessary. The menu option *WinCon* | *Set WinCon Options*, from the Simulink window menu bar, as described in Table 19 above, may also be used to automatically configure the appropriate RTW settings. However, this section outlines the important settings, in case they are not set correctly. **Always be sure to set the desired sampling period**.

To configure the Simulink model to be built using WinCon, select *WinCon* | *Options...* from the Simulink window menu bar. This command opens a dialog box similar to the one shown in Figure 11. The Real-Time Workshop parameters can also be accessed through the *Simulation* | *Simulation parameters... Ctrl+E* option from the Simulink window menu bar.

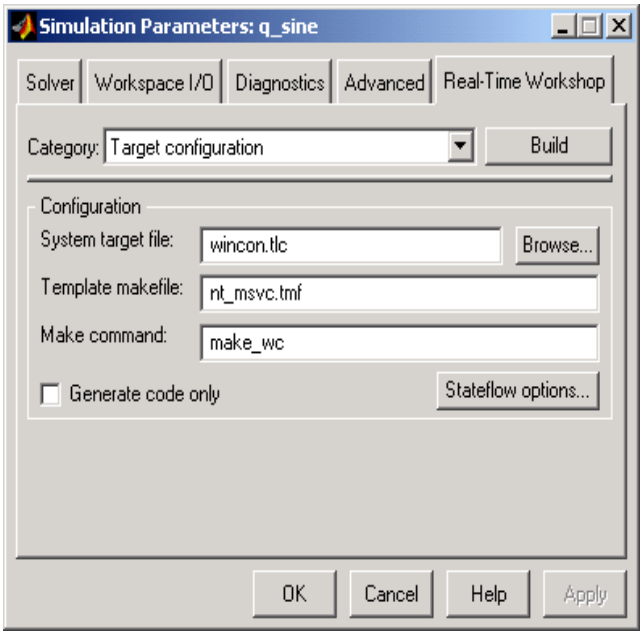


Figure 11 Real-Time Workshop Settings Dialog Box

The following Real-Time Workshop settings, shown in Table 20, ***must*** be set (and saved) in the Simulink diagram for which you wish to generate real-time code. Improper settings will result in the failure of WinCon to generate real-time code.

RTW Setting	*Description*
System target file (*Real-Time Workshop* Tab)	Select the Target Language Compiler file to be ***wincon.tlc***, as shown in Figure 11. The ***wincon.tlc*** file can be found, for example, under: C:\MATLAB6p1\rtw\c\Wincon\.
Template makefile (*Real-Time Workshop* Tab)	To generate code for a WinCon Client running under Windows NT/2000/XP, choose: ***nt_msvc.tmf***, as shown in Figure 11. For a WinCon Client running under Windows 98, choose: ***win_msvc.tmf***.
Make command (*Real-Time Workshop* Tab)	Choose ***make_wc*****,** as shown in Figure 11.
Solver Fixed-Step Size (*Solver* Tab)	Set to the desired sampling period T_S (in seconds), as shown in Figure 12.
Solver options (*Solver* Tab)	The choice of integration method depends, in some cases, on the controller. Only fixed step integration methods are supported. The ode1 (Euler) method is typically used.
Solver Mode (*Solver* tab)	Set to **Single-tasking**.

Table 20 Real-Time Workshop Settings

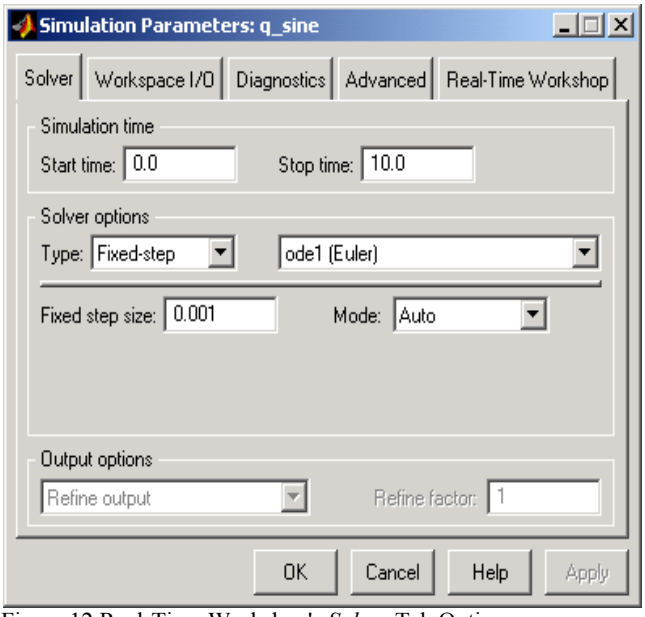


Figure 12 Real-Time Workshop's *Solver* Tab Options

Setting Simulink Options

Prior to building your WinCon real-time controller code from Simulink, you also need to verify the configuration of the following Simulink settings of the model for which you wish to generate real-time code:

- **Simulation | External**. Ensure that the option *Simulation* | *External* is checked from the Simulink window menu bar. If *Normal* is selected, then Simulink will only perform a simulation instead of communicating with the real-time code via WinCon Server.

- **External Target Interface.** From the Simulink window menu bar, selecting the item *Tools* | *External mode control panel...* | *Target interface...* brings up the *External Target Interface* window. Make sure that the *MEX-file for external interface* is set to: ***wc_comm***, as shown in Figure 13.

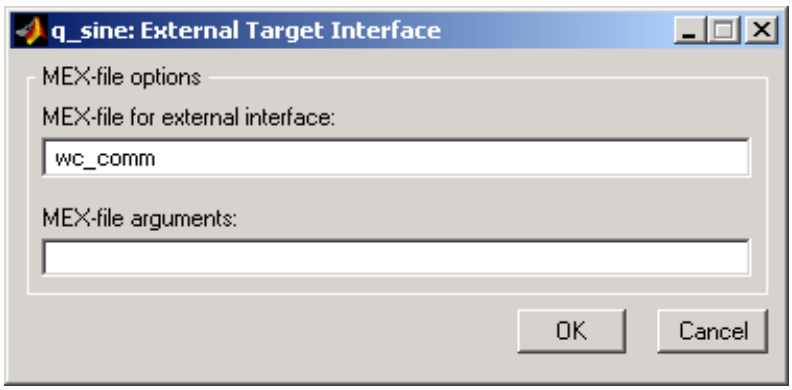


Figure 13 External Target Interface

Real-Time Code Generation

Before the controller real-time code can be generated, the Simulink and RTW options of the diagram you want to run in real-time must first be set and checked accordingly to sections Setting Real-Time Workshop Options and Setting Simulink Options. The WinCon real-time code can then be generated by selecting the *WinCon* | *Build* item from the Simulink window menu bar, as shown in Figure 14.

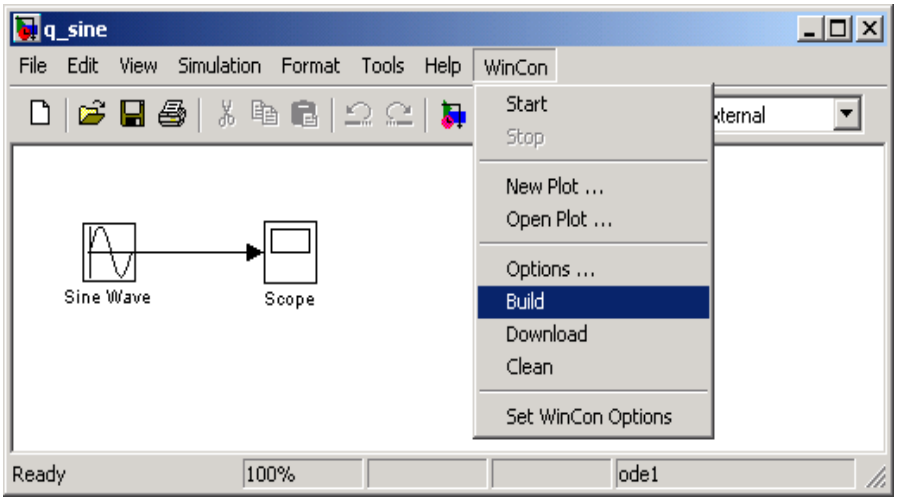


Figure 14 Generating the Real-Time Code

The Build procedure generates a WinCon Controller Library (i.e. .wcl) file. When code generation and compilation is complete, a message appears in the MATLAB window indicating that a *wcl* file has been successfully created. For example, for the **q_**sine model, the message that appears is:

```
### Created executable: q_sine.wcl
### Successful completion of Real-Time Workshop build procedure for
model: q_sine
```

After successful completion of the Build procedure, WinCon Server automatically downloads the generated controller file (i.e. .wcl) to WinCon Client, where it can be executed. If required, please refer to section Connecting to WinCon Client for instructions on how to connect WinCon Server to different WinCon Clients.

Restrictions on Real-Time Code

While WinCon supports a very broad range of Simulink blocks, as well as user-defined blocks, it is important to note that there are some restrictions on Simulink diagrams that are used in real-time. This section enumerates some of the more important restrictions. There are also differing restrictions depending on the target operating system. For example, Windows '98 is more restrictive than NT, 2000 or XP. Quanser recommends using an operating system based on the NT kernel, such as NT, 2000 or XP.

Calling Win32 Functions in Real-Time Code

On all platforms, WinCon supports user-defined S-functions written in C. This capacity is tremendously useful, particularly when interfacing with custom hardware. However, because the real-time code is executed in a real-time kernel environment (essentially an

operating system buried underneath Windows), ***Windows operating system functions cannot be called.*** Calling functions such as MessageBox, for example, is not possible, because the S-function code is running in a different (hidden) operating system.

In Windows NT/2000/XP, certain Win32 functions are emulated for your convenience, although some of the functions offer a limited subset of the full functionality available under the Win32 environment. The complete list of functions available with RTX 5.1 are provided in Table 21. A function is labelled "Deterministic" if the elapsed time for the call is less than 5 microseconds. For deterministic performance, non-deterministic functions should only be called at model start or termination.

In Windows '98, none of the Win32 functions are emulated.

Win32 Function	*Deterministic*	*Win32 Function*	*Deterministic*
AbnormalTermination	No	HeapDestroy	No
CloseHandle	No	HeapFree	No
CreateDirectory	No	HeapReAlloc	No
CreateFile	No	HeapSize	Yes
CreateThread	No	InitializeCriticalSection	No
DeleteCriticalSection	No	LeaveCriticalSection	Yes
DeleteFile	No	LoadLibrary	No
DeviceIoControl	No	RaiseException	No
EnterCriticalSection	Yes	ReadFile	No
ExitProcess	No	RemoveDirectory	No
ExitThread	No	ResumeThread	Yes
FreeLibrary	No	SetFilePointer	No
GetCurrentProcessId	Yes	SetLastError	Yes
GetCurrentThread	Yes	SetThreadPriority	Yes
GetCurrentThreadId	Yes	SetUnhandledExceptionFilter	Yes
GetExceptionCode	Yes	Sleep	Yes
GetExceptionInformation	Yes	SuspendThread	Yes
GetExitCodeThread	Yes	TerminateThread	No
GetLastError	Yes	TlsAlloc	Yes

Win32 Function	*Deterministic*	*Win32 Function*	*Deterministic*
GetProcAddress	No	TlsFree	Yes
GetProcessHeap	No	TlsGetValue	Yes
GetThreadPriority	Yes	TlsSetValue	Yes
HeapAlloc	No	UnhandledExceptionFilter	Yes
HeapCreate	No	WriteFile	No

Table 21 Win32 functions supported on NT/2000/XP

Calling Standard C Library Functions in Real-Time Code

For the same reason that most Win32 functions are not available in the real-time environment, there is also only a limited subset of the standard C library functions available under NT/2000/XP. Some functions only have a limited subset of their full functionality. The complete list of functions is enumerated in Table 22. A function is labelled "Deterministic" if the elapsed time for the call is less than 5 microseconds. Deterministic functions marked with an asterisk are only deterministic for small input sizes (e.g., short strings). For deterministic performance, non-deterministic functions should only be called at model start or termination.

In Windows '98, none of the standard C library functions are available, save those that are inlined by the compiler, such as strcpy or memset. Note that memory allocation is not possible in the Windows '98 real-time environment.

Function	*Det.*	*Function*	*Det.*	*Function*	*Det.*	*Function*	*Det.*
abs	Y	isalnum	Y	memset	Y	strtok	Y
acos	Y	isalpha	Y	modf	Y	strtol	Y
asin	Y	iscntrl	Y	perror	N	strtoul	Y
atan	Y	isdigit	Y	pow	Y	tan	Y
atan2	Y	isgraph	Y	printf	N	tanh	Y
atof	Y	islower	Y	putc	N	tolower	Y
atoi	Y	isprint	Y	putchar	N	toupper	Y
atol	Y	ispunct	Y	qsort*	Y	towlower	Y
bsearch*	Y	isspace	Y	rand	Y	towupper	Y

* Deterministic only for small input sizes

Function	*Det.*	*Function*	*Det.*	*Function*	*Det.*	*Function*	*Det.*
calloc	N	isupper	Y	realloc	N	ungetc	Y
ceil	Y	iswalnum	Y	rewind	N	va_start	Y
cos	Y	iswalpha	Y	setjmp	Y	vsprintf	N
cosh	Y	iswascii	Y	signal	N	wcscat	Y
difftime	Y	iswcntrl	Y	sin	Y	wcschr	Y
div	Y	iswctype	Y	sinh	Y	wcscmp	Y
exit	N	iswgraph	Y	sprintf	Y	wcscpy	Y
exp	Y	iswlower	Y	sqrt	Y	wcscpsn	Y
fabs	Y	iswprint	Y	srand	Y	wcsftime	Y
fclose	N	iswpunct	Y	sscanf	N	wcslen	Y
fflush	N	iswspace	Y	strcat	Y	wcsncat	Y
fgets	N	iswupper	Y	strchr	Y	wcsncmp*	Y
floor	Y	iswxdigit	Y	strcmp	Y	wcsncpy*	Y
fmod	Y	isxdigit	Y	strcpy*	Y	wcspbrk	Y
fopen	N	labs	Y	strcspn	Y	wcsrchr	Y
fprintf(stderr)	N	ldexp	Y	strerror	Y	wcsspn	Y
fputc	N	ldiv	Y	strlen	Y	wcsstr	Y
fputs	N	log	Y	strncat	Y	wcstod	Y
fread	N	log10	Y	strncmp*	Y	wcstol	Y
free	N	longjmp	Y	strncpy*	Y	wcstoul	Y
frexp	Y	malloc	N	strpbrk	Y	wprintf	N
fseek	N	memchr	Y	strrchr	Y	wtof	Y
ftell	N	memcmp*	Y	strspn	Y	wtoi	Y
fwrite	N	memcpy*	Y	strstr	Y	_controlfp	N
getc	N	memmove*	Y	strtod	Y	_fpreset	N

Table 22 Standard C library functions supported under NT/2000/XP

Using Dynamic Link Libraries in Real-Time Code

Win32 dynamic link libraries (DLLs) contain executable code that can be loaded dynamically by applications as required. Most of the Windows operating system is implemented as dynamic link libraries. The use of dynamic link libraries allows more than one application to share the same code, such as a standardized library of functions like the standard C library. While these dynamic libraries are often used, they are incompatible for real-time code. Real-time code must be compiled for the real-time kernel. Since dynamic link libraries are essentially Win32 executables, containing code compiled for the Win32 environment, they cannot be executed in the real-time kernel.

Hence, dynamic link libraries cannot be used in real-time code.

Since some third party numerical libraries are implemented as dynamic link libraries, as well as the programming interface to most device drivers, this restriction is unfortunate. If static link library versions of the same DLLs could be obtained, then this restriction can likely be overcome, provided these static libraries do not, in turn, depend on other DLLs.

Matlab Scripts

While MATLAB scripts may be used to control WinCon, they cannot be used in real-time code. MATLAB is an interpretive language and cannot run in real-time. Furthermore, the MATLAB engine and numerical libraries have not been compiled for the real-time environment, so neither is not possible to call MATLAB from real-time code.

Thus, do not use the "MATLAB Fcn" block from "Functions & Tables" in a Simulink diagram that will be used to generate real-time code.

The Matlab Compiler

MATLAB is a convenient language in which to write numerical algorithms. Hence, it is tempting to write S-functions using MATLAB instead of C. In order to run these S-functions in real-time, the assumption is often made that by using the MATLAB Compiler, these MATLAB-based S-functions may then be run in real-time. Unfortunately, that is not the case.

Code generated by the MATLAB Compiler cannot be run in real-time.

The problem is not in the concept, but in the implementation of the MATLAB Compiler. The MATLAB Compiler makes extensive use of dynamic link libraries for the numerical routines that it needs to execute MATLAB code as a C program. Unfortunately, the use of Win32 dynamic link libraries is not allowed in the real-time kernel environment. Hence, the code generated by the MATLAB compiler cannot be run in real-time. If The MathWorks provided these same dynamic link libraries as static link libraries instead, then there is a good chance that the MATLAB Compiler would then work for real-time code generation.

Until The MathWorks makes this option available, however, the MATLAB Compiler cannot be supported by WinCon (or any other real-time target).

Running the Real-Time Code

In order to run the controller on a desired WinCon Client, WinCon Server must first **connect** to that Client. When connected, WinCon Server can then download the generated code. Once the code resides on the Client, WinCon Server can instruct WinCon Client to run the real-time code. WinCon Server can connect to as many WinCon Clients as necessary, but there is **only one active Client at any instant**.

Most of the WinCon Server functions operate only on the model currently loaded onto the active Client. For example, the Start/Stop button on the toolbar starts and stops the controller on the currently active Client. Changing the active Client by select a new Client from the list of connected Clients (under the WinCon Server *Client* menu) will make the Start/Stop button control the newly selected Client instead. See section The Active Client for more details on the concept of an active Client.

Connecting to WinCon Client

Connecting to the Local Client

In the local configuration, WinCon Server and WinCon Client run on the same PC. To connect to the local WinCon Client, enter ***localhost*** in the *Client* | *Connect...* dialog box from the WinCon Server menu bar, as shown in Figure 15. Alternatively, you can also enter ***127.0.0.1,*** which is the standard local IP address.

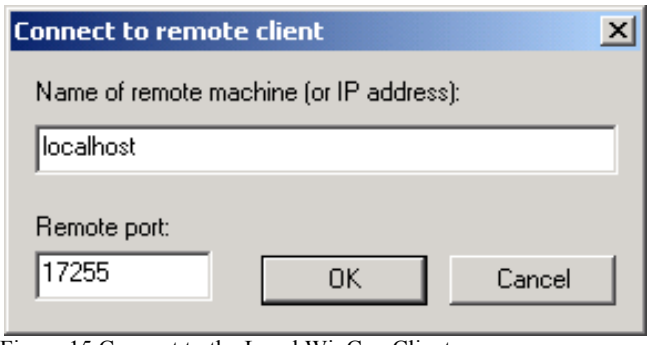


Figure 15 Connect to the Local WinCon Client

If WinCon Server does not already have a Client connection, opening a model in WinCon Server or generating real-time code from the Simulink diagram (i.e. using the *WinCon* | *Build* command) will automatically launch and connect to the local WinCon Client.

Connecting to a Remote Client

In a remote configuration, WinCon Server and WinCon Client run on two different PC's. Both PC's must have an IP address assigned to them. If one of the computer's IP address still needs to be set, please consult your *Windows Installation Guide* and/or your network administrator. Under Windows NT/2000/XP, administrative rights are required to access these properties.

The following steps describe the sequence of operations to connect WinCon Server to WinCon Client over a network (e.g. LAN, Internet):

Step 1. **Launch** both **WinCon Server** (on the PC where the user is) and **WinCon Client** (on the PC connected to the plant to be controlled). In order to connect to a remote WinCon Client, WinCon Server must have the Client computer's IP address or host name. If your Client PC's IP address is not known, proceed to Step 2, otherwise go to Step 3.

Step 2. Determine the **Client PC's IP address**. To do so, select the ***File*** | ***Network...*** item from the WinCon Client menu bar. This command opens the Microsoft Windows Network Connections control panel, which shows the status of the network. It also often displays the Client PC's IP address. Alternatively, you can open a Windows Command Prompt and type the following command: ***ipconfig***. This displays the Windows IP configuration for the Client PC. To launch a Command Prompt under Windows NT/2000/XP, select *Start* | *Programs* | *Accessories* | *Command Prompt* from the Windows Taskbar. Under Windows 98, launch an MS-DOS Prompt instead, by selecting *Start* | *Programs* | *MS-DOS Prompt* from the Windows Taskbar. As a remark, the Server PC's IP address can be determined in a similar manner by selecting the *Client* | *Network...* item from the WinCon Server menu bar. This command also opens the Microsoft Windows Network Connections control panel.

Step 3. **Connect to the WinCon Client**. Once the Client PC's IP address is known, WinCon Server can then connect to that Client. To do so, select the *Client* | *Connect...* item from the WinCon Server menu bar. This command open a dialog box similar to the one shown in Figure 15. In that *Connect to remote client* window, enter the IP address or computer name of the desired WinCon Client's machine. Remember that you must have the Client open on the remote machine before connecting it to a Server.

When WinCon Server is connected to the desired Client, it can then download the real-time code for the selected model to that Client, as described in Section Downloading the Real-Time Code. Once done, WinCon Server can connect to another Client on another PC on the network, and download the same model or a different model to it. The procedure can be repeated for as many Clients as necessary.

To see the WinCon Clients to which a WinCon Server is currently connected, refer to the *Client* menu from the WinCon Server window. There is a list of connected Clients at the bottom of that *Client* menu. This list displays the computer names or IP addresses of all connected WinCon Clients. The list will be empty if there are no connection. A check mark (i.e. √) to the left of a Client indicates it is the active Client. An asterisk (i.e. *) to the right of a Client indicates that the WinCon Server you are using is not the primary Server connected to that Client. Only the primary Server can change controller parameters.

The Active Client

Rather than have one Start/Stop button, one Plot button, etc. for each connected Client, WinCon Server uses the notion of an *active Client* to control the Client to which its various operations are referring. For example, the Start/Stop button only controls the currently active Client. To Start or Stop a different Client, change the active Client first. The Start/Stop button will then control the new active Client.

Changing the active Client does not change the status of the real-time code on the other connected Clients. For example, suppose Client #1 is the currently active Client. Clicking on the Plot button will open a plot for the model loaded on that Client. Clicking Start will run that model, data streamed from Client #1 will be displayed on the plot and the Start button will change to the red Stop button. For the benefit of this example, let this plot be referred to as Plot #1.

Now suppose Client #2 is selected from the list of Clients under the Client menu. Client #2 then becomes the active Client. Client #1 will still be running, and its data will still be displayed continuously on Plot #1. The Start/Stop button will change back to the the green Start graphic because the controller is not running yet on Client #2. Opening a plot using the Plot button will now display data from the model loaded on Client #2 (If a different model is loaded on Client #2, then the plot variable list will change to reflect the variables defined in the model on Client #2). Let this plot be called Plot #2. Clicking on the Start/Stop button will start the controller on Client #2. Plot #2 will begin to display real-time data streamed from Client #2. Now controllers are running on both Client #1 and Client #2, and both Plot #1 and Plot #2 are drawing data streamed from their respective Clients.

Now suppose the Stop All menu item is selected, or the Pause key is pressed. WinCon Server will now instruct both Client #1 and Client #2 to stop running their controllers. Hence, the Start/Stop button will change back to the green Start graphic and the plots will stop, because the real-time code on both Clients has stopped and data is no longer being streamed from the real-time code. Clicking the Start All menu item will start the controllers on both Client #1 and Client #2 at the same time, and data will again be plotted on both plots.

Downloading the Real-Time Code

When WinCon Server has established a TCP/IP connection to the WinCon Client on which you want to run the real-time controller code, a WinCon Controller Library file (i.e. .wcl file) can be downloaded to the desired Client by selecting *Model* | *Download* from the WinCon Server menu bar or *WinCon* | *Download* from the Simulink menu bar. In both cases, the corresponding model must be open in WinCon Server prior to being downloaded. Note however that building code in Simulink for WinCon will automatically open the model in WinCon Server and download it to the currently active Client.

Running WinCon Client

Once the controller code has been downloaded to WinCon Client, it can be run in real-time on that Client. Starting or stopping the real-time code can be achieved using any of the following interfaces:

- ❑ START or STOP button on the WinCon Server toolbar.
- ❑ *Model* | *Start* or *Model* | *Stop* from the WinCon Server menu bar.
- ❑ *WinCon* | *Start* or *WinCon* | *Stop* from the Simulink menu bar.
- ❑ START or STOP button on the WinCon Client toolbar.
- ❑ *Control* | *Start* or *Control* | *Stop* from the WinCon Client menu bar.
- ❑ On the Server PC, *Pause* to stop or *Alt* + *Pause* to start the controllers on all Clients connected to WinCon Server.
- ❑ On the Client PC, *Pause* to stop or *Alt* + *Pause* to start the controller currently loaded in WinCon Client.

WinCon Task Manager

The WinCon Task Manager application can be launched from the *Start* | *Programs* | *WinCon 3.3* | *WinCon Task Manager* menu item accessible from the Windows taskbar. WinCon Task Manager, as shown in Figure 16, lists all the real-time tasks that are currently loaded into VenturCom's real-time kernel (RTX). The list is updated once a second. WinCon Task Manager can be seen updating the list when, for example, WinCon Client loads a controller into the RTX kernel. WinCon Task Manager is only installed on Windows NT/2000/XP.

WinCon Task Manager also allows you to kill any real-time task (e.g. .wcl file). However, the user **should not kill** the RTX's Win32 supplemental C library task called **W32_dll.rtss** at PID=010. WinCon Client automatically detects when a task has been killed.

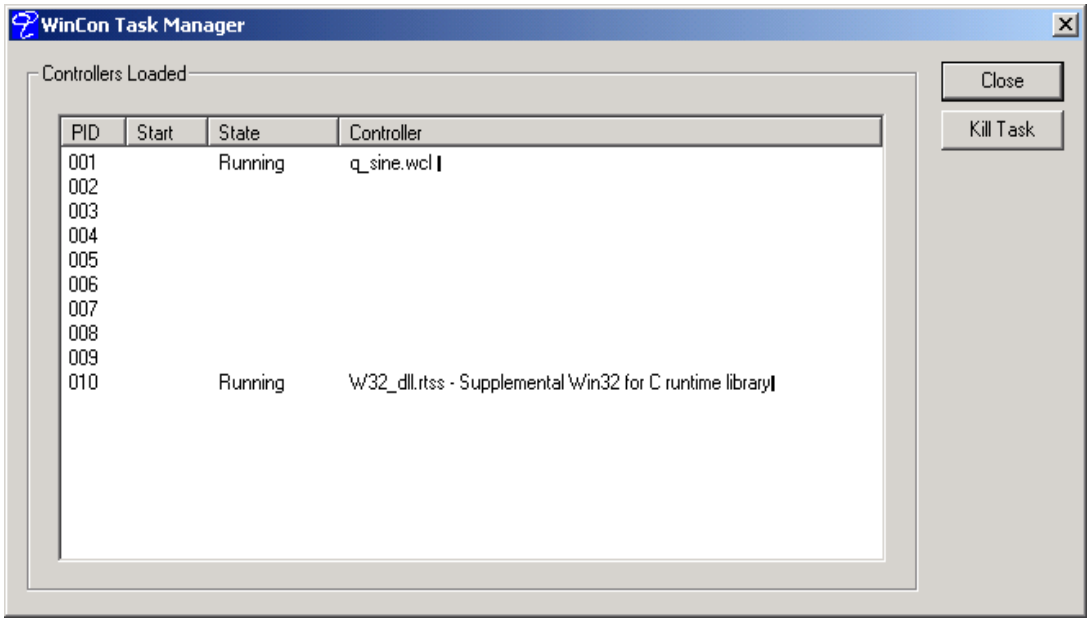


Figure 16 WinCon Task Manager

WinCon Service

For greater security, WinCon Client can be run from a non-Administrator account, under Windows NT/2000/XP. This added security is provided by the NT service called WinCon Service. WinCon Service allows regular users, ***without*** Administrator privileges, to run (and kill) real-time controllers from WinCon. WinCon Service loads real-time code, on behalf of WinCon Client, into the RTX environment. Note that WinCon Service must be running for WinCon Client to be able to run real-time code.

WinCon Service is installed during the WinCon Client installation and is automatically configured to run at boot time. As it is launched every time the machine is booted, the user does not have to run it manually. Note that WinCon Service loads the necessary RTX components as soon as it runs. WinCon Service must be running for WinCon Client to operate. Under Windows 2000/XP, NT services are accessed from: *Start* | *Settings* | *Control Panel* | *Administrative Tools* | *Services*, and under Windows NT from: *Start* | *Control Panel* | *Services*. Users with Administrator privileges may start and stop WinCon Service from the *Services* control panel applet.

Plotting On-Line Data

WinCon allows you to plot live data from the running model in real-time. As described in Section Saving On-Line Data, the data may also be saved to disk or to the MATLAB workspace. Data collection is flexible enough to satisfy your data acquisition requirements.

With WinCon, **you do not need to include any Simulink Scope block, or other "Sink", in your model to plot or save data!** Outputs of any Simulink block are available from WinCon Server for plotting and saving. Using the *Plot* | *New* selection list available from the WinCon Server menu bar or the *WinCon* | *New Plot...* item from the Simulink diagram menu bar, you can choose the best type of display to visualize the desired data. The following WinCon displays, described in the forthcoming sections, are currently available: WinCon Scope (i.e. data vs. time), WinCon X-Y Graph (i.e. an x versus y plot), WinCon Digital Meter (e.g. like a digital voltmeter), and WinCon Thermometer. Once the type of display is chosen, you may then select any model variable(s) that you desire to display in real-time.

If you have defined Simulink Scopes and/or Displays, as well as Quanser X-Y Graphs and/or Thermometers, in your model they can also be accessed, and their data plotted, by using the *Plot* | *Open...* selection list from the WinCon Server menu bar or the *WinCon* | *Open Plot...* item from the Simulink diagram menu bar. These blocks are accessed by WinCon Server as, respectively, WinCon Scopes, Digital Meters, X-Y Graphs, and/or Thermometers. Note that the Quanser X-Y Graph and Thermometer blocks can be found in the WinCon Toolbox of the Simulink library browser as a part of the Extra Sinks subsystem, as described in Section Extra Sinks on page 92.

WinCon displays can be created and changed on-the-fly (just like WinCon control panels). It is recommended however that display be opened when the real-time controller is not running.

Right-clicking on any WinCon display gives you quick access to the most useful menu options for that WinCon display, otherwise available only though that display's menu bar.

WinCon displays and control panels are saved as part of a WinCon Project. They re-associate their variables or parameters by name when loading from a WinCon project. Hence, it is possible to change the Simulink diagram and have the displays (as well as the control panels) continue to work when you reload a WinCon project designed using the older diagram, provided the names of the variables or parameters used have not changed.

WinCon Scope

A typical WinCon Scope is depicted in Figure 28. It displays the data collected from the running code in real-time. Refer to section Plotting On-Line Data to see how to open, or create, a WinCon Scope.

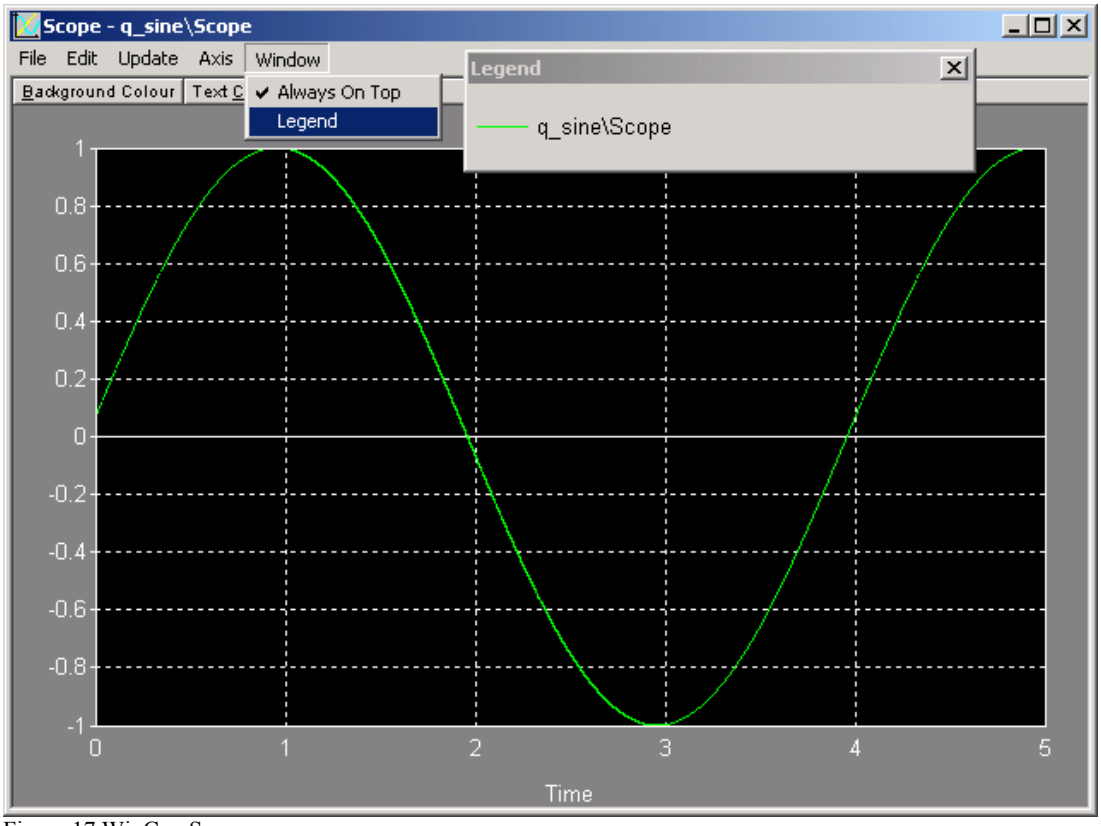


Figure 17 WinCon Scope

As seen in Figure 28, the default plot background is black. However, to preserve ink, plots printed from WinCon Scopes (as well as WinCon X-Y Graphs) always have a white background with black text and lines. Table 23 describes how to print the content of a WinCon Scope. The Scope legend, corresponding to the different signals being plotted, is always displayed in a separate window, as illustrated in Figure 28. Select *Window | Legend* from the WinCon Scope menu bar to show the legend window, as explained in Table 26. To avoid having an excessively wide legend, the variable/signal names, from the Simulink model, are intelligently truncated to fit within the legend window. In many cases, ellipses indicate what part of the name was removed (typically, parts of the intervening subsystem name). Only in the case in which the variable/block name itself (i.e. without the hierarchy) is extremely long, will the name itself be truncated.

The WinCon Scope colour and font toolbar uses the standard Microsoft Windows colour chooser and font chooser dialogs, respectively, so that they can be set by the user. A quick and easy way to include the content of a WinCon Scope into a document is to use the *Edit | Copy* item from the Scope menu bar. This operation copies a bitmap of the current plot to the clipboard, for subsequent pasting.

For plotting on a Scope, you can choose the output(s) of any block(s) of the Simulink diagram (corresponding to the running code) by selecting *File* | *Variables...* from the WinCon Scope menu bar. Table 23, below, gives a description of all the WinCon Scope's *File* menu options.

File Menu Option	*Description*
Save	Open a submenu to save the acquired data, as explained in Section Saving On-Line Data.
Variables...	Open a selection list from which to choose the block(s) (from the Simulink diagram) whose output(s) you desire to plot.
Print...	Print the WinCon Scope content, with a white background and black text and lines.
Close	Close the WinCon Scope.

Table 23 WinCon Scope's File Menu Options

The WinCon Scope's *Update* menu options are listed and described below, in Table 24.

Update Menu Option	*Description*
Real-Time	When checked, this option makes the plot return to Time 0 at the end of a trace and restart drawing the new data.
Freeze Plot	If selected when the real-time plot is running, this option makes the plot trace immediately stop (freeze). The full extent of the sampled data is plotted. This mode is referred to as **fixed mode**. If selected when the real-time plot is not running, the plot enters **fixed sweep mode** or **one-shot mode.** In this case, the plot will collect one buffer full of data when the controller is started, and then freeze.
Freeze All Plots	When checked, this option applies the "Freeze Plot" command (described above) to all plots at the same time.
Frequency ...	Open a dialog box to select the *plot* sampling frequency (or period). The plot sampling frequency is the frequency at which data is decimated for real-time display *only*. It is used for efficient plotting and does not affect the data displayed when the plot is frozen or saved. When a plot is frozen or saved, the data is not decimated, but reflects data collected at the *controller* sampling frequency (different from the plot frequency).
Buffer ...	Duration of time for which data is buffered for each plot.

Table 24 WinCon Scope's Update Menu Options

It is important to note that the plotting on WinCon Scopes is **decimated**, but **only in real-time mode**. In other words, the collected data is automatically displayed at a rate that minimizes the number of pixels to be drawn. This rate is known as the plot sampling frequency, or decimation frequency. This decimation is done deliberately in order to ensure that the plots can keep up with the real-time data stream and to maximize the number of plots that can be opened at any instant. However, this decimation may result in **plot aliasing during real-time display**. Therefore, you must be careful in selecting the time scale of the data that is displayed in real-time mode, so that the plots are not misinterpreted.

When WinCon Client is decimating the data and streaming it to the WinCon Scopes, it also collects data for the save variable at the *controller* sampling frequency. For example, suppose the controller sampling frequency is 1kHz and the plot sampling frequency is 200Hz. WinCon Client will collect data every 1 millisecond, but only send every fifth data point to the WinCon Scope during real-time plotting. However, as soon as the plot is frozen and its data saved, WinCon Client transmits *all* the data, collected at 1kHz, to the Scope. **Thus, as soon as a plot is frozen, it redraws its display with *all* the available data and the effects of plot aliasing at the plot sampling frequency, if any, disappear.** The same process occurs when the plot buffer is saved. *All* the available data is saved to disk, collected at the 1kHz sampling frequency. Hence, no data is lost due to the temporary decimation during real-time plotting.

Since the amount of data displayed on a plot can be quite large (hundreds of thousands of points), WinCon Scopes perform on-the-fly data compression when plotting, so that (very) large numbers of data points are handled quickly and efficiently.

The WinCon Scope's *Axis* menu options are listed and described below, in Table 25. They allow the user to change the plot axes.

Axis Menu Option	*Description*
Auto-Scaled	When checked, this option automatically scales the vertical (i.e. Y) axis of the plot at the end of each sweep, according to the data collected during the previous sweep.
Fixed	When checked, this option fixes the scale of the vertical (i.e. Y) axis of the plot. Auto-scaling is not performed at the end of each sweep.
Auto-Scale Range...	Open a dialog box to set the minimum Y range used during auto-scaling, so that the Y range does not go to zero for constant signals.
Fixed Range...	Open a dialog box to set the range of the fixed Y axis.

Axis Menu Option	*Description*
Time...	Open a dialog box to set the time interval for the time axis. This *Time* frame cannot exceed the *Buffer* duration defined in Table 24. In fixed mode or fixed sweep mode, a *Time* interval less than the *Buffer* length enables a scroll bar which allows the user to examine the data more closely, as shown in Figure 18.
Grid	When checked, it adds major grid lines to the axes.

Table 25 WinCon Scope's Axis Menu Options

In both fixed modes (namely the fixed sweep mode and the fixed mode, as defined in Table 24), the data plotted on the Scope is not decimated and every point that was collected (at the sampling frequency of the real-time code) is drawn. In fixed sweep mode, one buffer's worth of data is collected from WinCon Client (i.e. real-time code) and displayed on the plot. To examine that data closely, the timescale on the plot can be changed by using the *Axis | Time...* item from the Scope menu bar, as mentioned in Table 25. A time frame strictly smaller than the buffer length results in the plot being magnified for the time axis. A scroll bar also appears at the top of the Scope to allow the entire buffer of collected data to be navigated, as pictured in Figure 18. By dragging the slider on the top of the graph, the user may examine the collected data in detail.

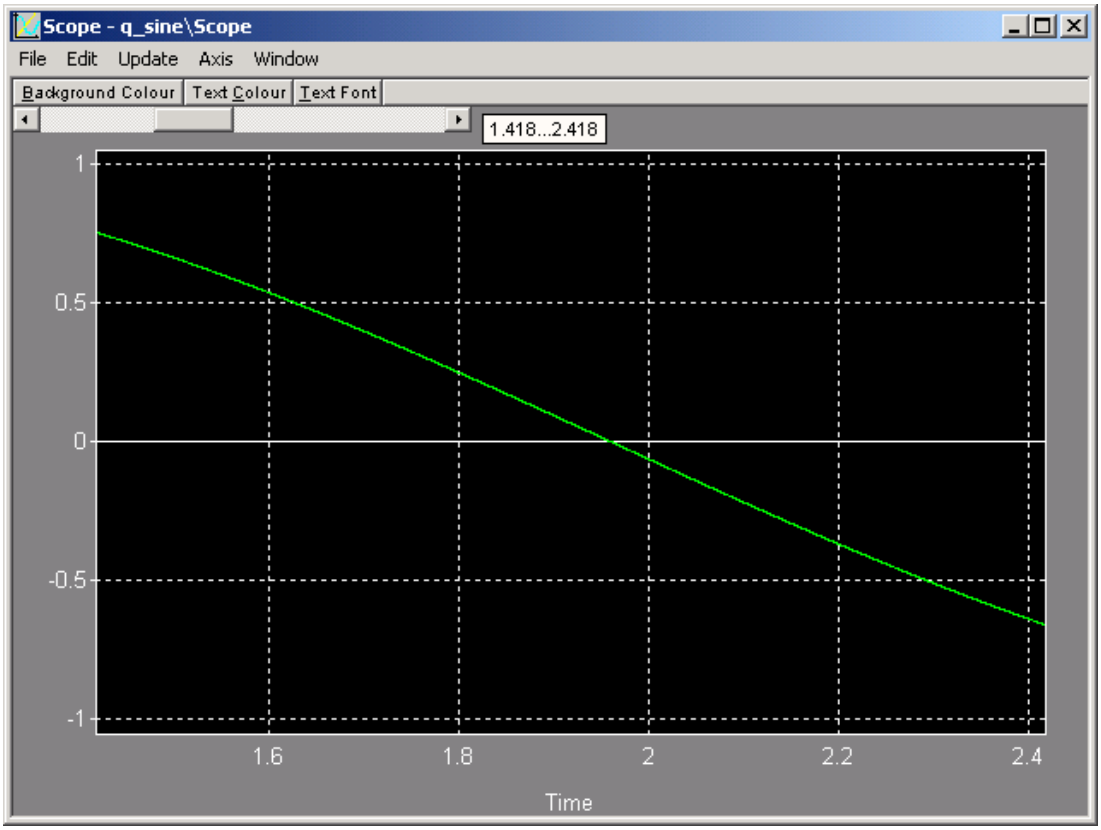


Figure 18 WinCon Scope in Fixed Mode

The WinCon Scope's *Window* menu options are listed and described below, in Table 26.

Window Menu Option	*Description*
Always on top	Select whether you want the WinCon Scope plot window to stay on top of all other windows.
Legend	Open a window containing the legend information. You can change line colors, used to trace signals, by clicking on the corresponding legend name.

Table 26 WinCon Scope's Window Menu Options

WinCon X-Y Graph

A typical WinCon X-Y Graph is depicted in Figure 19. It displays two variables (plotted against each other) collected from the running code in real-time. Refer to Section Plotting On-Line Data to see how to open, or create, a WinCon X-Y Graph.

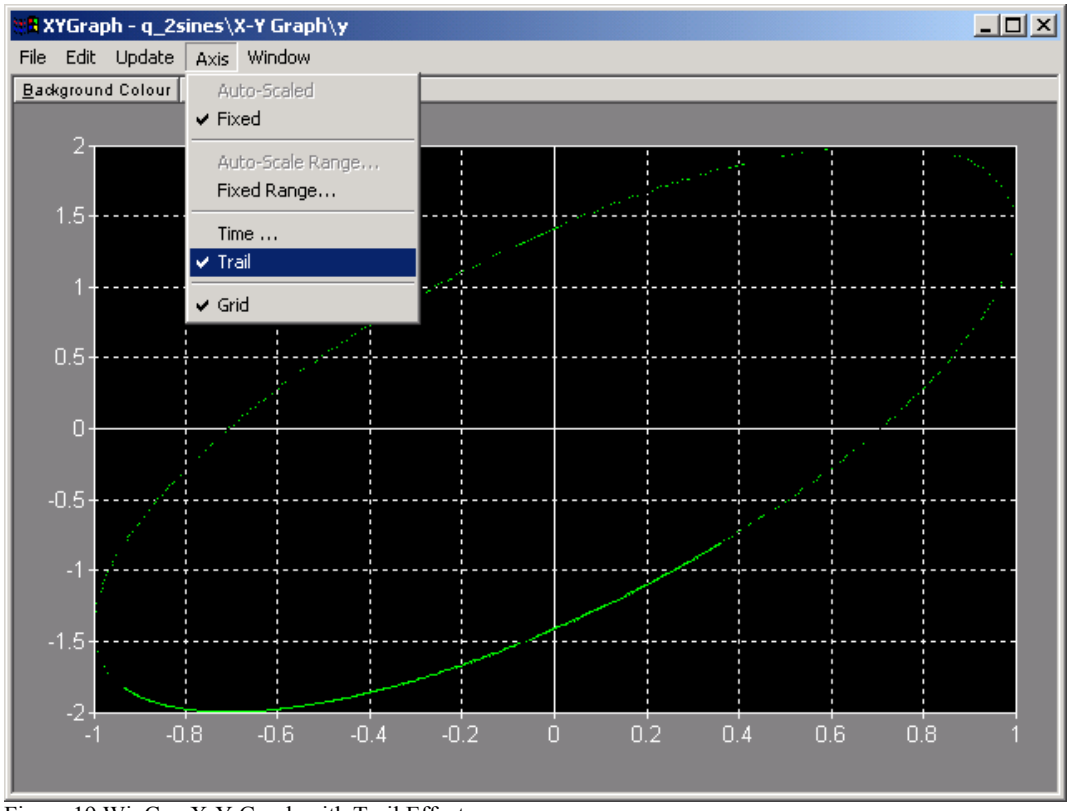


Figure 19 WinCon X-Y Graph with Trail Effect

In real-time mode, the x-y plot erases itself at the end of the buffer of collected data, unless the *Axis* | *Trail* item from the WinCon X-Y Graph menu bar is selected. In this case, the X-Y plot leaves a trail, as depicated in Figure 19. The WinCon X-Y Graph interface and functionality are very similar to the operation of the WinCon Scope, described in section WinCon Scope.

WinCon Digital Meter

A typical WinCon Digital Meter is depicted in Figure 20. It digitally displays the instantaneous value of a variable collected from the running code in real-time. This display is especially useful for accurately monitoring slowly varying variables. The Digital Meter

displays -INF or +INF when the value is infinite. Refer to section Plotting On-Line Data to see how to open, or create, a WinCon Digital Meter.

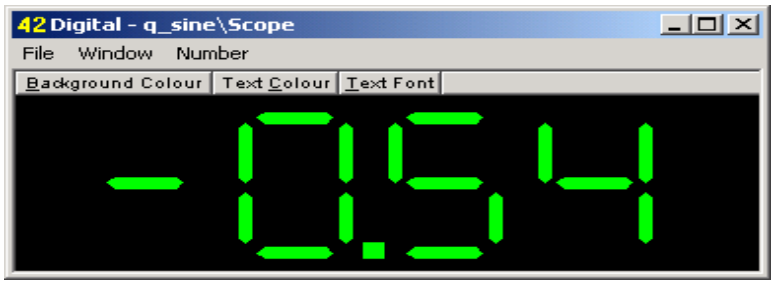


Figure 20 WinCon Digital Meter

The WinCon Digital Meter's *Number* menu options are enumerate in Table 27 below.

Number Menu Option	*Description*
Decimal Places...	Open a dialog box to select, automatically or manually, the number of decimal places. This option sets the accuracy of the display.
Use Segments	If checked, print each digit using 7 segments. Otherwise, each digit is printed according to the selected *Text Font* from the toolbar.
Scientific Notation	If checked, use scientific notation instead of decimal format.
Decimal Format	If checked, use decimal format instead of scientific notation.

Table 27 WinCon Digital Meter's *Number* Menu Options

The value displayed within the WinCon Digital Meter window can also be **displaced**. To do so, move the mouse cursor over the displayed number. When the mouse cursor changes to a horizontal double arrow (similar to ⇔), left click on the display and drag the mouse left or right. The digits will move also.

The WinCon Digital Meter's *Window* menu options are listed in Table 28 below.

Window Menu Option	*Description*
Always On Top	Select whether you want the WinCon Digital Meter window to stay on top of all other windows or not.
Hide Menu	Show or hide the Digital Meter's menu bar.
Hide Toolbar	Show or hide the Digital Meter's toollbar.

Table 28 WinCon Digital Meter's *Window* Menu Options

WinCon Thermometer

A typical WinCon Thermometer is depicted in Figure 21. The mercury level of the thermometer reflects the instantaneous value of a variable collected from the running code in real-time. The WinCon Thermometer is the analog counterpart of the WinCon Digital Meter, described in Section WinCon Digital Meter. This display is especially useful for monitoring slowly varying variables. Refer to Section Plotting On-Line Data to see how to open, or create, a WinCon Thermometer.

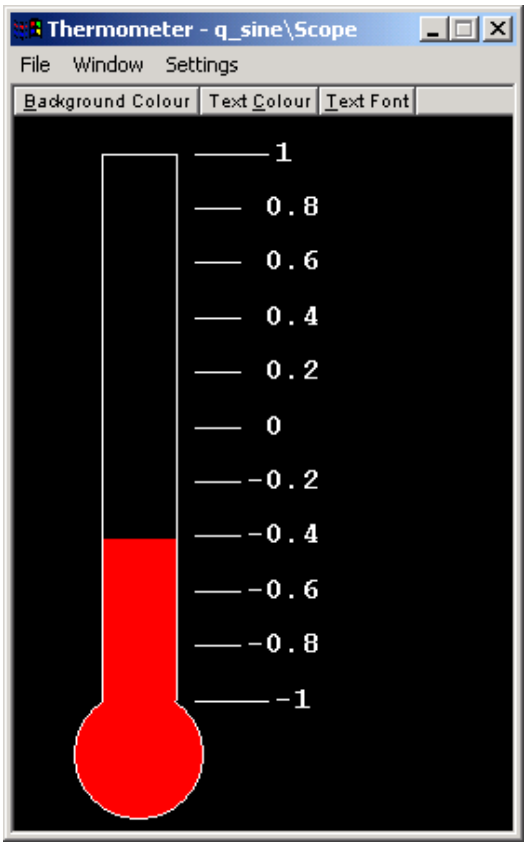


Figure 21 WinCon Thermometer

The WinCon Thermometer's *Settings* menu options are explained below in Table 29.

Settings Menu Option	*Description*
Range...	Open a dialog window to set the thermometer range (i.e. minimum and maximum values).
Thermometer Colours...	Open the standard Microsoft Windows Color selection window to choose the thermometer colour.

Table 29 WinCon Thermometer's Settings Menu Options

Performance Monitoring

The performance of each WinCon controller running in real-time can be monitored by plotting, on any of the WinCon displays, any of the performance variables available. The supplied performance parameters are listed in the *Variables* window of WinCon Client, under the field *.performance*, as depicted in Figure 7. You can review section Timing of Real-Time Events to familiarize yourself with the terminology used in the following paragraphs.

One of the most important controller performance variables is the *Sampling Interval*, which shows the actual sampling interval achieved between samples (i.e. T_E). It should be as close as possible to the desired sampling interval (i.e. T_S). If this variable diverges significantly from the desired sampling period, then the controller performance will not be as expected. In Figure 22, the monitoring in real-time of the *Sampling Interval* is achieved through a WinCon Digital Meter.

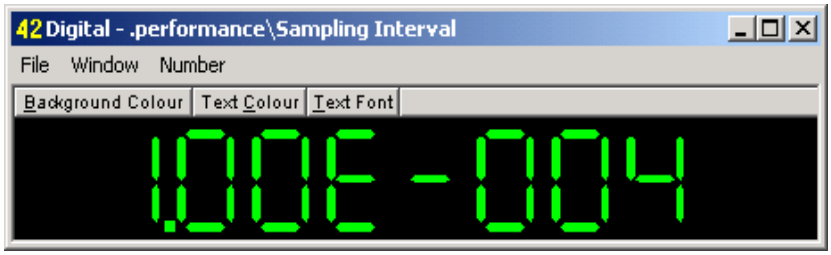


Figure 22 Monitoring of the Actual Sampling Time

Another important performance parameter is the computation delay (i.e. T_C). It shows how long it takes the PC to execute the real-time controller code at every sample. Knowing the maximum value for T_C allows the user to determine the fastest sampling frequency possible for that particular controller on that particular PC. Specifically, knowing the maximum computation delay *Max(T_C)* and wishing to keep a minimum time *Min(T_F)* for foreground tasks results in the following fastest sampling rate possible:

$$F_{MAX} = 1 / (\text{Min}(T_F) + \text{Max}(T_c))$$

This maximum computation delay, *Max(T_C)*, can also be monitored in real-time on a WinCon display and is labelled *Maximum Computation Time* under the *.performance* node of the variable list in, for example, the WinCon Client window.

Saving On-Line Data

CAUTION: The following Simulink blocks, presented below in Table 30, are **non-deterministic** in nature and consequently are not compatible with real-time operations. Specifically, these blocks are the Simulink *To File* and *To Workspace* blocks. Therefore, they cannot be included in a controller diagram destined to run in real-time. (Actually, the To Workspace block may be included, but it will simply define a variable that may be plotted by a WinCon Scope – it will not save the data to the MATLAB workspace automatically). However, the WinCon Scope, as well as the WinCon X-Y Graph, offer an alternative to those Simulink blocks. Table 30, below, presents some of the WinCon Scope (and WinCon X-Y Graph) solutions to saving on-line data.

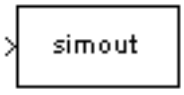
Simulink Block	*Equivalent WinCon Scope (or X-Y Graph) Solution*
	In order to achieve this functionality, the desired data must be acquired in the WinCon Scope (or X-Y Graph) buffer and then saved to a MAT-file using the *File* | *Save* | *Save As MAT-file...* option from the WinCon Scope (or X-Y Graph) menu bar, as described in Table 31.
	In order to achieve this functionality, the desired data must be acquired in the WinCon Scope (or X-Y Graph) buffer and then saved to the MATLAB workspace using the *File* | *Save* | *Save To Workspace* option from the WinCon Scope (or X-Y Graph) menu bar, as described in Table 31.

Table 30 Non-Real-Time Simulink Blocks and their Equivalent WinCon Solutions

The on-line data being saved from a WinCon Scope or a WinCon X-Y Graph is not decimated. The sampling frequency for the real-time code is also the frequency at which the selected data is acquired by the WinCon Scope or X-Y Graph buffer. Therefore, you should ensure that the buffer duration of your WinCon Scope or X-Y Graph is long enough to accommodate your data acquisition needs. The buffer duration can be accessed through the *Update* | *Buffer...* option from the WinCon Scope or X-Y Graph menu bar, as specified in Table 24. WinCon Scopes are optimized for handling large quantities of data and thus are the preferred option for saving data.

WinCon Scope and X-Y Graph offer three different ways to save on-line data. These saving options are described below, in Table 31. They are listed in a submenu of the *File* | *Save* item from the WinCon Scope or X-Y Graph menu bar.

File | Save SubMenu Option	*Description*
Save As M-file...	This command saves the buffered data to an **M-file** (i.e. a script with a .m extension) of the user-selected name. Typing that M-file name in the MATLAB window will generate a MATLAB plot of the saved data and bring the saved variables into the workspace using variable names constructed from the model name and signal names (e.g. *q_sine_Scope*).
Save As MAT-file...	This command saves the buffered data to a **MAT-file** (i.e. a binary file with a .mat extension) of the user-selected name. That MAT-file can then be loaded into the MATLAB workspace by using the MATLAB load command. The variable names of the saved data are constructed from the model name and the name of the block outputs that you had selected (e.g. *q_sine_Scope*).
Save To Workspace	The result of this is equivalent to saving the desired data to a MAT-file first, and then loading it into the MATLAB workspace.

Table 31 WinCon Scope and X-Y Graph Data Saving Capabilities

Changing Parameters On-Line

There are two ways of changing model parameters in the running controller: either through the Simulink model itself, or using a WinCon Control Panel.

Any parameter that you can change in a Simulink diagram can be changed in the running controller on-the-fly. In Simulink, to modify a parameter, double-click on the block whose parameter(s) you wish to change and type in the new value(s). As soon as you click OK or Apply, the new parameter value(s) is downloaded to the real-time code and you can see the impact immediately! Additionally, variables computed, or typed, in the MATLAB workspace may be used as parameter values in Simulink blocks, and as such may also be modified. To do so, enter the new parameter value in the MATLAB workspace and select the *Edit* | *Update diagram* item from the Simulink window menu bar. The new parameter values will be download to the real-time code. Alternatively, you may also press *Ctrl + D* to cause Simulink to re-read the MATLAB workspace variables and download any resulting changes in block parameters to the real-time code.

If the user desires to run the real-time code without Simulink, but still wants to change controller parameters, the WinCon Control Panel may be used. The WinCon Control Panel was explicitly designed for tuning controllers without Simulink being launched (or even installed). Control Panels allow the user to associate control widgets to real-time model parameters. **The user is not expected to run both a WinCon Control Panel and Simulink at the same time.** Use of the WinCon Control Panel is intended for the final stage when you have completed the structure of your controller and are tuning it without Simulink. Keep in mind that there is no connection to Simulink through the Control Panel, so if you are using the Control Panel and have the Simulink diagram open at the same time, parameter values in the Simulink diagram may not correspond to parameter values in the Control Panel.

WinCon Control Panel

A sample WinCon Control Panel is illustrated in Figure 23.

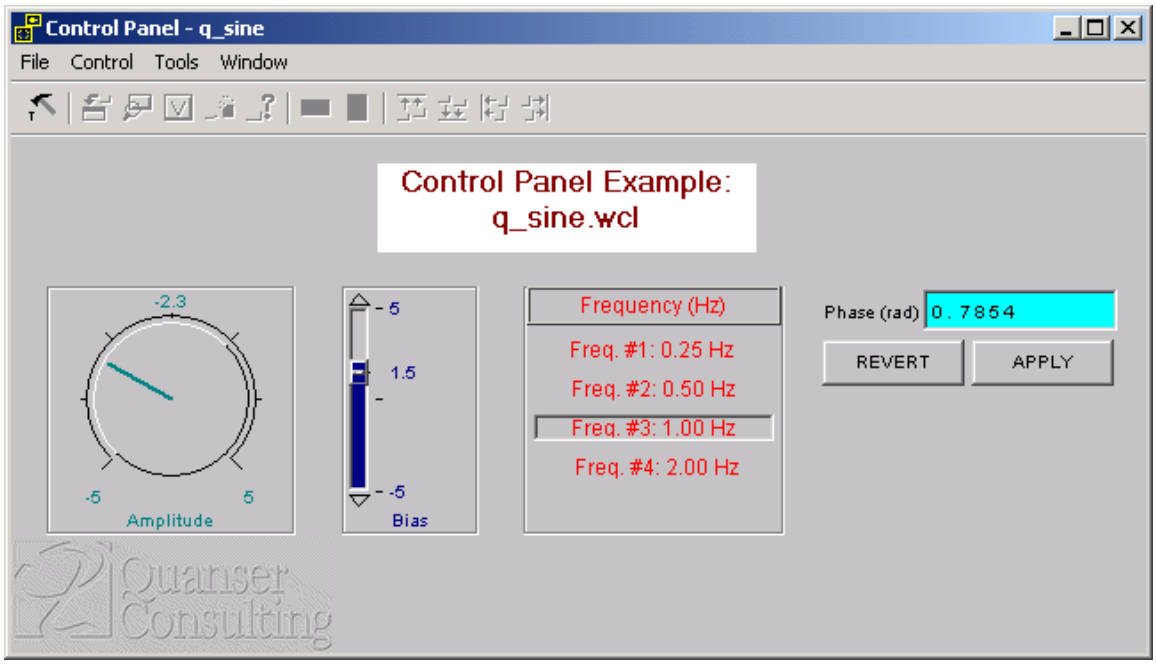


Figure 23 WinCon Control Panel

WinCon Control Panels accept ActiveX controls. The controls, like WinCon displays, are saved as part of a WinCon Project, as described in section WinCon Projects, below, together with an ActiveX file (having a .ocp extension).

Like WinCon displays, WinCon Control Panels also re-associate their variables/parameters by name when loading a WinCon project or when the real-time code is reloaded (e.g. because of a rebuild). Hence, it is possible to change the Simulink model and have the Control Panel(s), as well as the display(s), work when you reload a previous WinCon project and/or newly-generated real-time code (i.e. .wcl file), as long as the associated variable names remain unchanged.

WinCon Server 3.3 also has Control Panels that write their current variable values to the controller when a project is loaded or the model is rebuilt. Therefore, WinCon Control Panels set the initial parameter values in the controller when loaded, instead of the other way around.

CAUTION: The drawback to this approach can appear in the case of a single WinCon Client (e.g. unique experiment) / multiple WinCon Server (e.g. multiple student) configuration. In this remote configuration, WinCon Projects with different Control Panels

must not be opened while the controller is already running! However, usually only one student connects to one experiment at a time.

Building WinCon Control Panels

WinCon Control Panels, like WinCon Displays, can be created and changed on-the-fly. It is recommended, however, that any components be created when the real-time controller is not running.

To create a WinCon Control Panel, use the *Window* | *Control Panel* item from the WinCon Server menu bar. This operation opens up a new control panel, as an empty canvas. To insert a control widget onto it, you must first ensure you are in Design Mode by activating the *Window* | *Design Mode* menu item from the Control Panel menu bar, as described in Table 32 below. Table 32 lists the WinCon Control Panel's *Window* menu options.

Window Menu Option	*Description*
Design Mode	If checked, the Control Panel is in Design Mode. The communication with the real-time code is then interrupted and the Control Panel widgets can be edited and/or created. If unchecked, the parameter communication with the running code is reestablished.
Menu	Show or hide the Control Panel's menu bar.
Toolbar	Show or hide the Control Panel's toolbar.
Always On Top	Select whether you want the WinCon Control Panel window to stay on top of all other windows.

Table 32 Control Panel's *Window* Menu Options

Use the menu option *Control* | *Insert Control*... to start the insertion process, as explained in Table 33 below, whichs enumerates the Control Panel's *Control* menu options. Note that these menu options are only available if the WinCon Control Panel is in Design Mode. The WinCon Control Panel's *Control* menu allows the user to configure the behaviour (e.g. associated variable, default value) and the appearance (e.g. label, colour) of the supplied control widgets (listed in Table 34).

Control Menu Option	*Description*
Insert Control...	Open a selection list from which to choose the type of control you wish to use. Next, it automatically calls the *Model Parameters* selection window (also available from the *Associate Variable...* menu item), where the variable to associate with the control is chosen. This variable must be a scalar, i.e. of size (1,1). Once done, the control will be inserted at the top left corner of the control panel, where you can drag it to another location.
Delete	This option deletes the selected control(s) from the Control Panel. Pressing the *Delete* key has the same effect. To select one control, click on it when the Control Panel is in Design Mode. To select multiple controls, hold the *Ctrl* key and click on each control when the Control Panel is in Design Mode.
About...	Open an information window about the selected control. To select a control, click on it when the Control Panel is in Design Mode.
Properties...	Open the *Control Properties* dialog window corresponding to the desired control that has been selected. To select a control, click on it when the Control Panel is in Design Mode. Each control has its own set of properties which can be modified. Typically, a control's properties include items such as appearance, colour, value range, label, and the like.
Associate Variable...	Open the *Model Parameters* selection window, in which the variable to (re-)associate with the selected control is chosen. This variable must be a scalar, i.e. of size (1,1). To do this, ensure that the proper control, for which you wish to create or change the variable association, has been selected. To select a control, click on it when the Control Panel is in Design Mode.

Table 33 Control Panel's *Control* Menu Options

The ActiveX controls currently available to WinCon Control Panels are described below, in Table 34.

Control	*Description*
On/Off Button (a.k.a. QButton)	On/Off button, to switch a variable between two values.
Edit Control (a.k.a. QEdit)	Input text box, to manually type in the desired value for a given variable.
Horizontal Slider (a.k.a. QSlider)	Horizontal slider, to set a variable value between a user-defined minimum and maximum, with the possibility of using increments.
Vertical Slider (a.k.a. QVSlider)	Vertical slider, to set a variable value between a user-defined minimum and maximum, with the possibility of using increments.
Knob Control (a.k.a. QKnob)	Knob, to set a variable value between a user-defined minimum and maximum.
Multiple select (a.k.a. QSelect)	Selection list from which to choose the value to which to set the associated variable.
Picture (a.k.a. QPicture)	Displays a picture, such as a bitmap, icon, or metafile.
Text (a.k.a. QText)	Displays text of user-defined justification, font, and colour. For multi-line text, a carriage return may be added by pressing *Ctrl* + *Enter* in the *Text:* edit box.

Table 34 Control Panel's Available Controls

Once all the desired controls have been inserted and configured in the Control Panel, their size and location still needs to be set.

To move an individual control, select it (by left-clicking on it) and drag it to the desired location on the Control Panel, and release the mouse button. To move a group of controls, select the desired multiple controls, drag them to their desired new location on the Control Panel, and release the mouse button. To select multiple controls, hold the *Ctrl* key and click on the desired controls to select, when the Control Panel is in Design Mode. To size an individual control, select it and move your mouse cursor to one of its corners or edges so that the resize cursor appears. Left click on the corner or edge and drag the mouse to modify the size of the control. Release the mouse when the control is the desired size.

To size and align multiple controls relatively to each other, the WinCon Control Panel's *Tools* menu items should be used. Note that these menu options are only available if the WinCon Control Panel is in Design Mode and multiple controls have been selected. The control selected first acts as the reference. The Control Panel's *Tools* menu options are listed and described in Table 35 below.

Tools Menu Option	*Description*
Make Same Width	This option makes all the selected controls the same width as the first selected control.
Make Same Height	This option makes all the selected controls the same height as the first selected control.
Top Align	This option makes the top of all the selected controls align with the top of the first selected control.
Bottom Align	This option makes the bottom of all the selected controls align with the bottom of the first selected control.
Left Align	This option makes the left edge of all the selected controls align with the left edge of the first selected control.
Right Align	This option makes the right edge of all the selected controls align with the right edge of the first selected control.

Table 35 Control Panel's *Tools* Menu Options

If you wish to to start over and clear the Control Panel of all of its controls, select *File* | *Clear* from the Control Panel menu bar.

Right-clicking on a WinCon Control Panel gives you quick access to the most useful menu options for inserting and/or modifying controls, otherwise only available though the Control Panel's menu bar.

Once you are satisfied with your newly designed Control Panel, exit the Design Mode by de-selecting the *Window* | *Design Mode* item from the Control Panel menu bar, as described in Table 32. The WinCon Control Panel now allows you to interact on-the-fly with the selected parameters of your real-time code (e.g. controller). Depending on your requirements, you may then re-iterate the control insertion/editing process as many times as necessary.

WinCon Projects

A WinCon project file (i.e. with a .wcp extension) is created by saving a session in WinCon Server. Such a session can contain several Client connections, controllers, displays, and a control panel. If you re-load a saved WinCon Project, all Client connections are re-established, controllers downloaded, and displays and control panel re-opened. In order to successfully open a complex WinCon Project with multiple Client connections, make sure that all the required WinCon Clients are up and running prior to loading the project.

CAUTION: In WinCon 3.3, the project file format has changed from earlier versions. **Old project files** (prior to WinCon 3.3) **will not work** (i.e. cannot be properly loaded) from WinCon 3.3 and onward **and MUST be rebuilt**. However, object-versioning information for each project component has been added to the WinCon project file format, so that future releases of WinCon can maintain backward compatibility with project files from WinCon 3.3 onward. Therefore, if you wish to keep a non-backward compatible WinCon project, you will have to **save that project's visual layout (e.g. WinCon Displays and Control Panels) and settings** in order to **visually reproduce them** in WinCon 3.3 project format. **Saving a Project's visual layout** can be achieved by either taking screen captures of that Project's different WinCon Displays and/or Control Panels, or keeping that project on another machine together with the corresponding (older) WinCon so that it can still be opened and looked at.

More information is now stored in WinCon Projects. Projects have also been revised to make them portable to different disk locations. This change was done by saving the WinCon model path (in the project file) as a relative path (relatively to the location of the project file), instead of an absolute path. Hence, it is now possible to share WinCon Projects.

To copy, or move, a WinCon Project to a different disk location, ensure that you copy, or move, the .wcp project file with its corresponding .wcl model file and .ocp ActiveX control panel files to the desired location.

To open a WinCon Project once the WinCon Server is launched, you can either use the *File | Open...* item from the WinCon Server menu bar, or drag the corresponding .wcp project file from Windows Explorer and drop it onto the WinCon Server window. If the WinCon Server window is not open, you can open the WinCon Project by double-clicking on the corresponding project file (i.e. with a .wcp extension).

The WinCon Toolbox

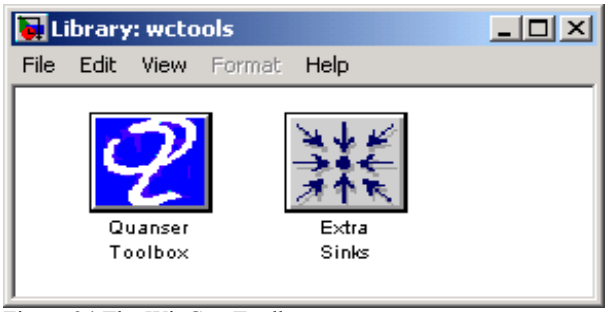


Figure 24 The WinCon Toolbox

There are two ways to access the WinCon Toolbox library. The first way is to type the following command at the MATLAB prompt:

```
wctools
```

This command opens up the WinCon Toolbox as shown in Figure 24. The other way to access the WinCon Toolbox is through the standard Simulink Library Browser, as depicted in Figure 25. You can get to it by selecting the *Library Browser* button in the Simulink window, and then choosing the WinCon Toolbox icon.

As seen in Figure 24, the WinCon Toolbox provides access to the Quanser Toolbox and to a few additional Extra Sinks. While the Quanser Toolbox is immediately portable between WinCon, under Windows 98/NT/2000/XP, and SimuLinux-RT (SLX), under Linux, the WinCon Toolbox's Extra Sinks are specific to WinCon and cannot be compiled for SimuLinux-RT (SLX). In other words, any controller diagram using blocks from the Quanser Toolbox can be used, with no change, in both WinCon and SimuLinux-RT (SLX). SimuLinux-RT (SLX) allows hard real-time performance and control under Real-Time Linux. More information on SimuLinux-RT (SLX) can be obtained on Quanser's website, at: http://www.simulinux.com.

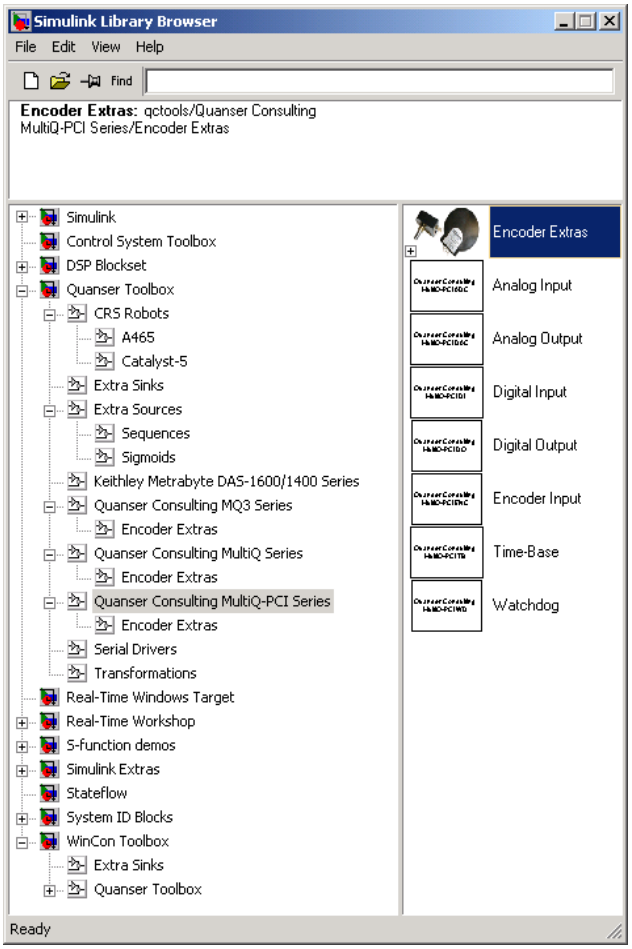


Figure 25 The WinCon and Quanser Toolboxes within the Simulink Library

The Quanser Toolbox

The Quanser Toolbox is fully described in section Interfacing to Hardware: The Quanser Toolbox.

Extra Sinks

Sinks are Simulink blocks, like Scopes, that take an input but generally do not have output ports. Quanser Consulting provides additional sinks to enhance the Simulink environment. Double-click on the Extra Sinks icon in the WinCon Toolbox, wctools, to access these sinks. The extra sinks are illustrated in Figure 26. Two sinks are currently provided: the

Thermometer and the X-Y Graph blocks.

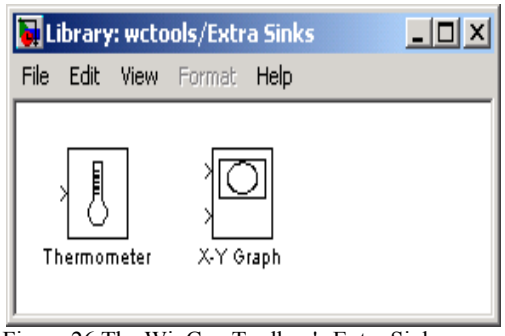


Figure 26 The WinCon Toolbox's Extra Sinks

Thermometer

The Thermometer block defines a thermometer display for WinCon, similar to that described in section WinCon Thermometer. The dialog box of the Thermometer block is shown in Figure 27. It lets the user set the maximum and minimum thermometer values to be used when the Thermometer is opened in WinCon Server. The Thermometer will appear in the plot list when the Plot button is pressed on the WinCon Server toolbar.

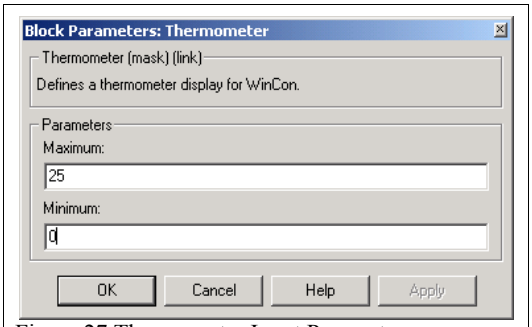


Figure 27 Thermometer Input Parameters

X-Y Graph

The X-Y Graph block defines an X-Y Graph display for WinCon, similar to that described in section WinCon X-Y Graph. The dialog box of the X-Y Graph block is shown in Figure 79. It lets the user set the range of the X and Y axes when the X-Y Graph is opened in WinCon Server. The X-Y Graph will appear in the plot list when the Plot button is pressed on the WinCon Server toolbar.

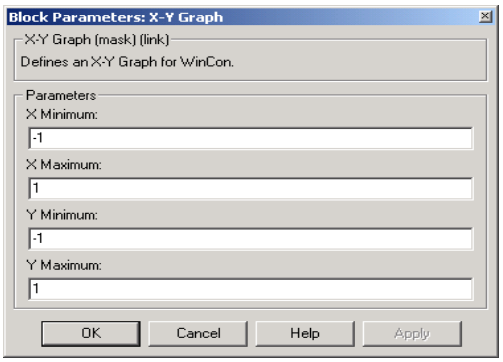


Figure 28 X-Y Graph Input Parameters

Interfacing to Hardware: The Quanser Toolbox

A real-time package finds its real use when it can be interfaced with hardware. Fortunately, WinCon may be used with a variety of data acquisition cards, including, of course, Quanser Consulting's own MultiQ-2, MultiQ-3, and MultiQ-PCI data acquisition cards. Accessing hardware becomes as simple as placing blocks into your Simulink diagram. The blocks for the Quanser products, as well as some additional blocks, are available in the Simulink-integrated Quanser Toolbox library, described in this section.

There are two ways to access the Quanser Toolbox library. The first way is to type the following command at the MATLAB prompt:

```
qctools
```

This command opens up the Quanser Toolbox as shown in Figure 26. The other way to access the Quanser Toolbox is through the standard Simulink Library Browser, as depicted in Figure 25. You can get to it by clicking on the *Library Browser* button in the Simulink window, and then clicking on the Quanser Toolbox icon.

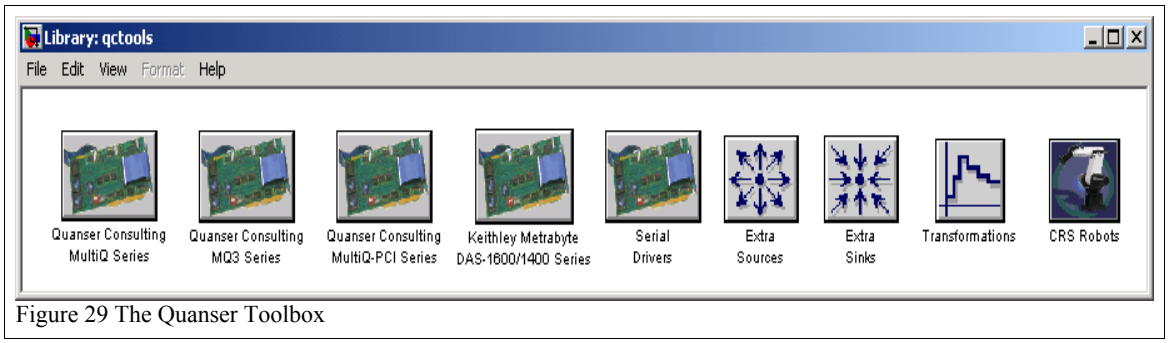


Figure 29 The Quanser Toolbox

As briefly mentioned previously, the Quanser Toolbox library contains all the Simulink blocks you need to use Quanser Consulting's MultiQ-2, MultiQ-3, and MultiQ-PCI data acquisition cards. There are also additional blocks to increase functionality, such as extra sources (e.g. sigmoids, sequences), extra sinks, serial drivers, data conversion blocks, and convenient discretizing transfer function blocks. Be sure to see what is available – it can save you time!

Quanser Consulting MQ3 Series

This section describes the blocks used with the MultiQ-3. The blocks for the MultiQ series are identical and the Keithley Metrabyte blocks are similar. Each block comes with online help as well.

Double-click on the Quanser Consulting MQ3 Series block to open a folder of all the blocks that may be used with the MultiQ-3 data acquisition card. This library is illustrated in Figure 30 below. There are blocks for analog input, analog output, digital input, digital output, encoders and a hardware time-base. These blocks may be used to interface with more than one MultiQ card in the same Simulink diagram, if desired. The block are discussed in the following subsections.

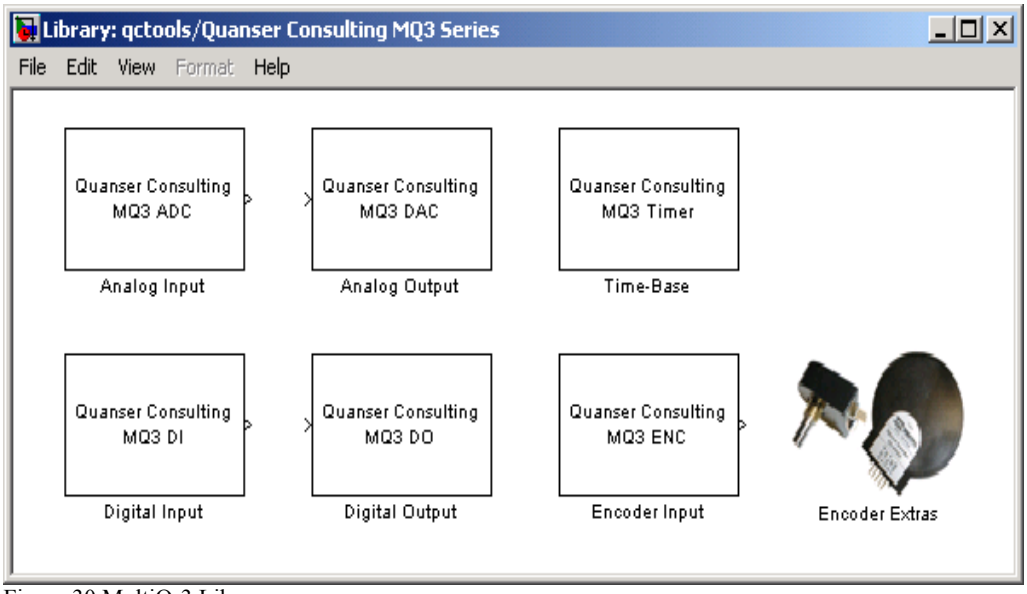


Figure 30 MultiQ-3 Library

Analog Input

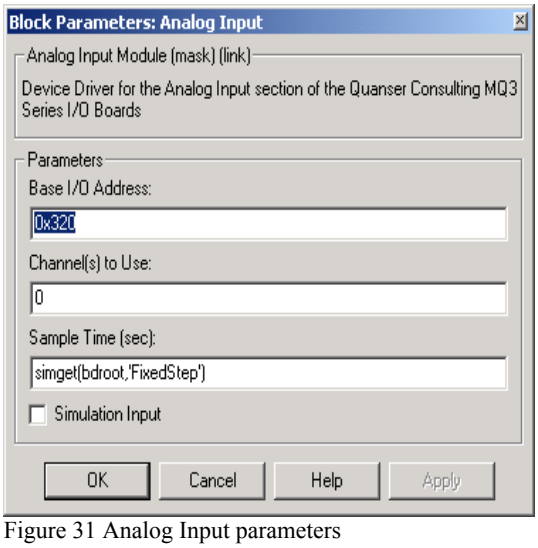


Figure 31 Analog Input parameters

The Analog Input block is used to interface with the analog-to-digital converters on the MultiQ. The analog input block may be used to read more than one analog input at a time. In this case, the output from the analog input block will be a vector with one entry for each analog input read. You may put more than one analog input block into your diagram, if you prefer. The only restriction is that no two analog input blocks can read the same channels. However, they can read different channels on the same I/O card. To change the analog input parameters, double-click on the block after

dragging it into your Simulink diagram. The dialog box shown in Figure 31 below will appear.

The first field allows you to specify the base address of the MultiQ I/O card. The MultiQ card is an I/O mapped ISA card. If there is more than one MultiQ card in your system, simply specify the base address of the card you want. All the MultiQ blocks have a Base I/O Address parameter.

The second field identifies the analog input channel, or channels, you wish to read. Specify a scalar number from 0 to 7 to read one of the eight analog input channels. To read more than one channel, specify a vector. The channels will be read in the order that they appear in the vector. For example, entering [5 2] will cause the Analog Input block to read analog channel 5 first and then to read analog channel 2. The output from the Analog Input block will be a vector, where the first element corresponds to analog channel 5, and the second element corresponds to analog channel number 2. The output units are volts.

In most cases, you will probably want to specify the channels in ascending order. You can do this using the convenient ':' vector shorthand notation of MATLAB. For example, to read analog channels 2 through 4, simply enter 2:4 into the Channel(s) to Use field.

The third parameter is the sample time. All the MultiQ blocks include a sample time parameter so that they may be used in discrete-time systems. The default sample time is set to the base sampling rate of your system. You will not have to change this default unless you are developing a multi-rate system. The expression simget(bdroot,'FixedStep') returns the base sampling period as a real number, so you can multiply this expression by an integer to get a multiple of the base sampling period. Alternatively, you can simply enter a real number corresponding to the sampling period for this block. The sampling period must be positive.

The final parameter is the Simulation Input check-box. This parameter has no effect on the real-time code. It is only used for simulation. When this box is checked, the Analog Input block will have an input port. During normal simulation, this input will be quantized according to the A/D resolution and fed to the output port. Thus, a model of the plant may be connected to this input so that the performance of the system may be simulated with quantization effects taken into account.

For example, suppose you are balancing an inverted pendulum and the joint angle of the pendulum is measured using a potentiometer. Put a model of the inverted pendulum into your Simulink diagram and connect the joint angle in the model to the input of the Analog Input block. Connect the output of the Analog Input block to your control system, as you would when controlling the actual hardware.

Now when you simulate the system, the Analog Input block reads the output of your inverted pendulum model, quantizes it according to the resolution of the MultiQ analog inputs, and outputs the result. To your controller, its output will look like the signal it

would read from the real hardware, including the quantization. Hence, you can simulate the behavior of your system accurately.

When the diagram is run in real-time, the Analog Input block ignores its input and reads from the actual hardware. Thus, the same block diagram may be used for both simulation and real-time operation.

If you wish to prevent the real-time code from executing your inverted pendulum model, you can enclose the inverted pendulum computations in an enabled subsystem and enable the subsystem using the Is Simulation? block provided by Quanser. This block is found in the Extra Sources folder of the Quanser Toolbox.

Analog Output

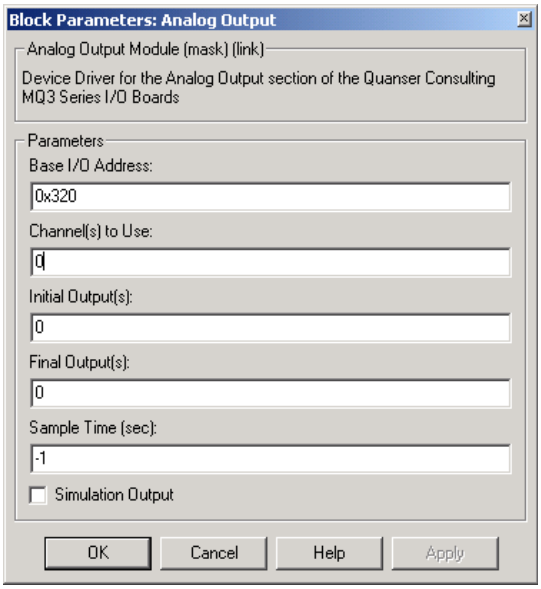


Figure 32 Analog Output parameters

The Analog Output block is used to interface with the digital-to-analog converters on the MultiQ data acquisition card. The analog output block may be used to write to more than one analog output at a time. In this case, the input to the analog output block must be a vector with one entry for each analog output written. You may put more than one analog output block into your diagram, if you prefer.

To change the analog output parameters, double-click on the block after dragging it into your Simulink diagram. The dialog box shown in Figure 32 below will appear. Like the Analog Input block, there are fields for the Base I/O Address and Sample Time. See the Analog Input section for a discussion of the Base I/O Address parameter.

The Channel(s) to Use field identifies the analog output channels that will be written by this block. Specify a scalar number from 0 to 7 to write to one of the eight analog output channels. To write more than one channel, specify a vector. The channels will be written in the order that they appear in the vector. For example, entering [5 2] will cause the Analog Output block to write to analog channel 5 first and then to write analog channel 2. The input to the Analog Output block must be a vector, where the first element corresponds to analog channel 5, and the second element corresponds to analog channel number 2. The input units are volts.

In most cases, you will probably want to specify the channels in ascending order. You can do this using the convenient ':' vector shorthand notation of MATLAB. For example, to

write to analog channels 2 through 4, simply enter 2:4 into the Channel(s) to Use field.

The Initial Output(s) field specifies the value(s) that will be written to the analog output channels when the real-time code is initialized, before it starts running. If you specify a scalar, then all analog output channels in the Channel(s) to Use field will be initialized to the same value. To initialize each channel to a different value, specify a vector containing the initial values for each channel.

The Final Output(s) field specifies the value(s) that will be written to the analog output channels when the real-time code is stopped. If you specify a scalar, then all analog output channels in the Channel(s) to Use field will be set to the same value. To set each channel to a different value, specify a vector containing the final values for each channel.

The second-last parameter is the sample time. All the MultiQ blocks include a sample time parameter so that they may be used in discrete-time systems. The default sample time is set to -1 in the case of the Analog Output block. A sampling time of -1 indicates that the sampling rate is inherited from the input signal driving the block. Thus, you will usually not have to change this parameter, even in a multi-rate system. The sample time can also be specified as a positive scalar. The expression simget(bdroot,'FixedStep') returns the base sampling period as a real number, so you can multiply this expression by an integer to get a multiple of the base sampling period. Alternatively, you can simply enter a real number corresponding to the sampling period for this block. The sample time must be -1 or positive.

The Simulation Output checkbox is similar to the Simulation Input option of the Analog Input block. Checking this box puts an output port on the Analog Output block. This output port is only used during simulation. It is not used during real-time execution. During simulation, this output port produces a quantized version of the data at the input port, using the resolution of the MultiQ digital-to-analog converters. Like the Simulation Input option of the Analog Input block, this feature may be used to combine a simulation of your hardware and the real-time code into a single Simulink diagram.

See the discussion of the Analog Input block's Simulation Input parameter for more details.

Digital Input

The digital input block is used for reading the digital inputs on the MultiQ card. It has the same parameters as the Analog Input block. The only difference between the two blocks is that the Digital Input block reads the digital inputs, of which there are eight, instead of the analog inputs. See the Analog Input section for a discussion of its parameters. Note however that the output of the Digital Input block is zero or one, not a voltage.

Digital Output

The Digital Output block is used to write to the digital outputs of the MultiQ data acquisition card. There are eight digital outputs. The Digital Output block has the same parameters as the Analog Output block. The only difference between the two blocks is that the Digital Output block takes inputs that are zero or one, not a voltage, and sends them to the digital outputs of the MultiQ instead of the analog outputs. See the Analog Output section for a discussion of the parameters.

Time Base

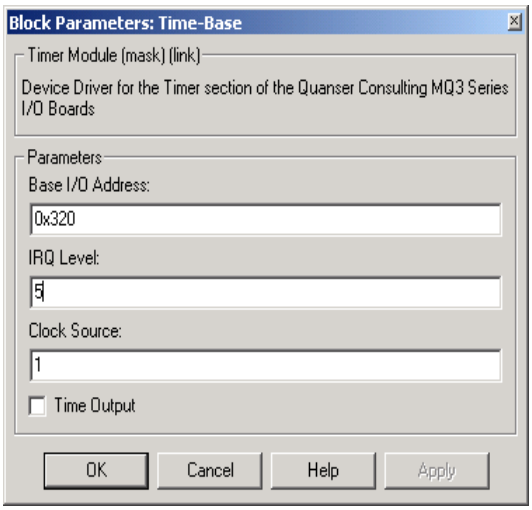


Figure 33 Time Base parameters

The Time Base block allows a hardware timer to be used to generate the sampling rate for the system. When no Time Base block is present in the diagram, the system timer is used. The system timer under Linux is fast and accurate, supporting sub-millisecond sampling rates, so this block is not strictly necessary, but it is available for those users wanting to use the hardware time-base on their data acquisition card. In Windows, the Time Base block is necessary for sampling rates beyond 1kHz on Windows '98 and for sampling rates beyond 10kHz on Windows NT/2000/XP.

The parameters for the MultiQ Time Base block are shown in Figure 33. See the Analog Input section for a discussion of the Base I/O Address parameter.

The IRQ Level parameter must be set to the interrupt level selected on the MultiQ card. The MultiQ-3 card does not support software programmable interrupt levels, so this setting must match the jumper setting on the MultiQ card.

The Clock Source parameter determines which clock on the MultiQ card will be used for the hardware time-base. This setting is also determined by a jumper on the MultiQ card, so this parameter must match the hardware setting for this block to work.

The Time Base block normally has no inputs or outputs. Checking the Time Output box puts an output port on the Time Base block. This port outputs the time since the simulation began. Hence, this output may be used instead of a separate Clock block (the Clock block is a Simulink block found in the Sources library).

Note that only one Time Base block may appear in a Simulink diagram, since only one hardware timer is needed to produce the base sampling rate. If more than one Time Base block is placed in a diagram then an error message will be issued when generating code.

Encoder Input

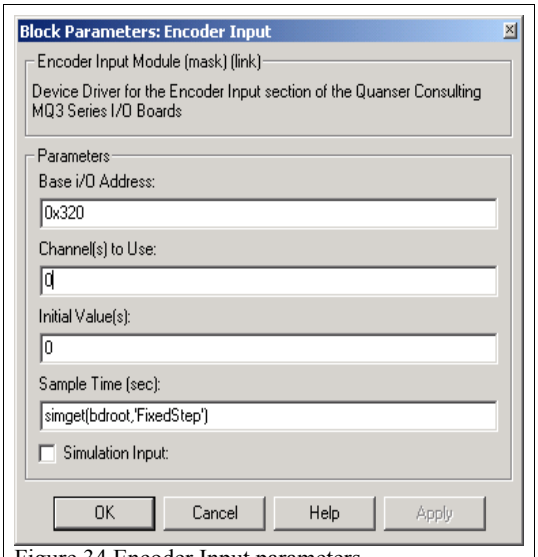


Figure 34 Encoder Input parameters

The Encoder Input block is used to read the counter values of the MultiQ encoder inputs. The number of encoder inputs available depends on the model of MultiQ card purchased. The output of the encoder block is the 24-bit encoder count(s). More than one encoder channel may be read at the same time and more than one Encoder Input block may be present in your Simulink diagram. Indeed, the Encoder Input block is analogous to the Analog Input block except that it reads the encoders and outputs counter values instead of processing analog inputs. The parameters of the Encoder Input block are illustrated in Figure 34. Refer to the Analog Input section for details on the parameters to the Encoder Input block. If the Simulation Input is used, the modulus operator is applied to the input signal to produce a 24-bit signed value (reflecting a genuine encoder count).

Encoder Extras

It is often convenient to calibrate the encoders in the real-time code in response to some event, such as a limit switch being hit or at a certain time. The Encoder Extras library provides a set of Encoder Reset blocks for performing such calibration. One block resets the encoder value based on a level-sensitive input. Two others reset the encoder value based on an edge-triggered input. Since the Encoder Input block sets the initial value of the encoder count, the Encoder Reset blocks are not necessary to set the initial count value. However, they are very useful for calibration. The Encoder Extras library is depicted in Figure 35 below.

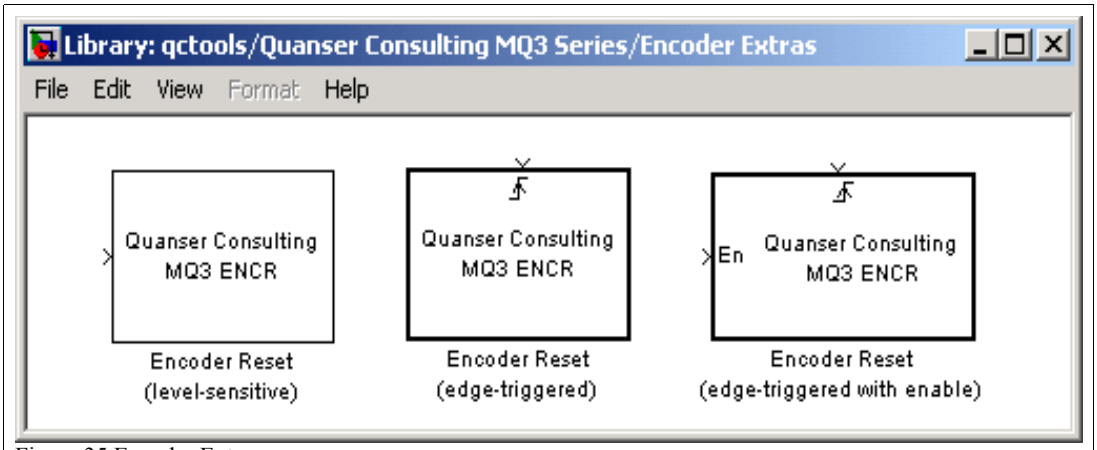


Figure 35 Encoder Extras

Encoder Reset (level-sensitive)

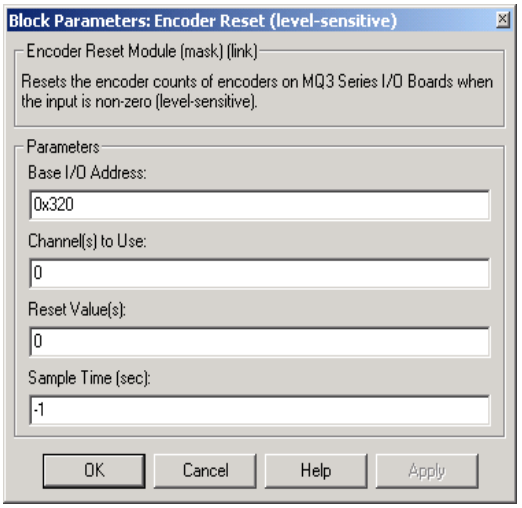


Figure 36 Encoder Reset (level-sensitive) parameters

The level-sensitive Encoder Reset block resets the encoder value when its input is non-zero. Like the Encoder Input block, the Encoder Reset (level-sensitive) block has parameters for the base I/O address and encoder channels to use. Refer to the Encoder Input block, as shown in Figure 36, for a discussion of these parameters.

The Reset Value(s) parameter operates in the same way as the Initial Value(s) parameter of the Encoder Input block except that it specifies the values to which the encoder counts are reset when the input to the block is non-zero. Since the input is level-sensitive, the encoder counts are reset every sample time instant until the input returns to zero. This block may be used to reset the encoder counts until a certain amount of time has transpired for example. For example, with Quanser's inverted pendulum experiment, it could be used to give the user a few seconds to hold the pendulum upright before the controller kicks in. It can also be used in triggered subsystems so that all your calibration may be performed in response to a single trigger.

The Sample Time parameter is used to set the sampling time for the block. The default value is -1, indicating that the sample time is inherited from the input signal driving the block. The sample time must be inherited to use the block in a triggered subsystem. The sample time can be set to a positive real scalar outside of a triggered subsystem.

Encoder Reset (edge-triggered)

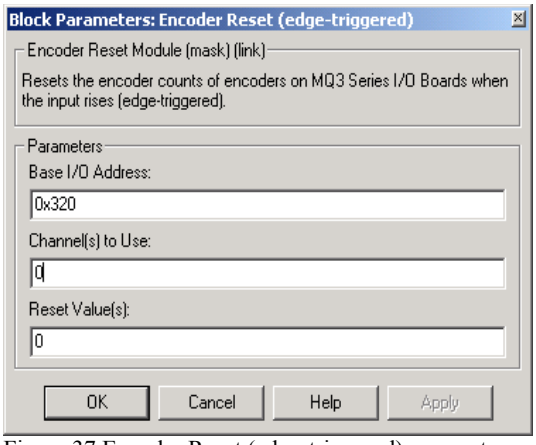


Figure 37 Encoder Reset (edge-triggered) parameters

The edge-triggered Encoder Reset block resets the encoder counts in response to a rising edge at its input. The encoder value is reset only at each rising edge. The block will not continue to reset the encoder counts after the rising edge. This block is useful for resetting the encoder counts at a particular instant in time (but not before) or when a particular event occurs.

For example, a controller designed to find the limits of the workspace could use velocity control to slowly move the apparatus around its workspace. When a limit switch was hit, the Encoder Reset (edge-triggered) block could be used to reset the encoder count to calibrate it. The controller could then switch to position control.

The parameters for the block are illustrated in Figure 37. The parameters are identical to the Encoder Reset (level-sensitive) block with one exception: the Sample Time parameter is missing because the block is event-based. It always derives its sample time from the trigger signal input.

Encoder Reset (edge-triggered with enable)

The Encoder Reset (edge-triggered with enable) block is identical to the Encoder Reset (edge-triggered) block except that it has an enable input as well. The encoder counts will only be reset when the trigger occurs *and* the enable input is non-zero. See the Encoder Reset (edge-triggered) block for a discussion of the block parameters.

Quanser Consulting MultiQ-PCI Series

This section describes the blocks used with the MultiQ-PCI card. Each block comes with online help as well, for your convenience.

Double-click on the Quanser Consulting MultiQ-PCI Series block to open a library of all the blocks that may be used with the MultiQ-PCI data acquisition card. This library is illustrated in Figure 38 below. There are blocks for analog input, analog output, digital input, digital output, encoders, a hardware time-base and a watchdog timer. These blocks may be used to interface with more than one MultiQ-PCI card in the same Simulink diagram, if desired. The block are discussed in the following subsections.

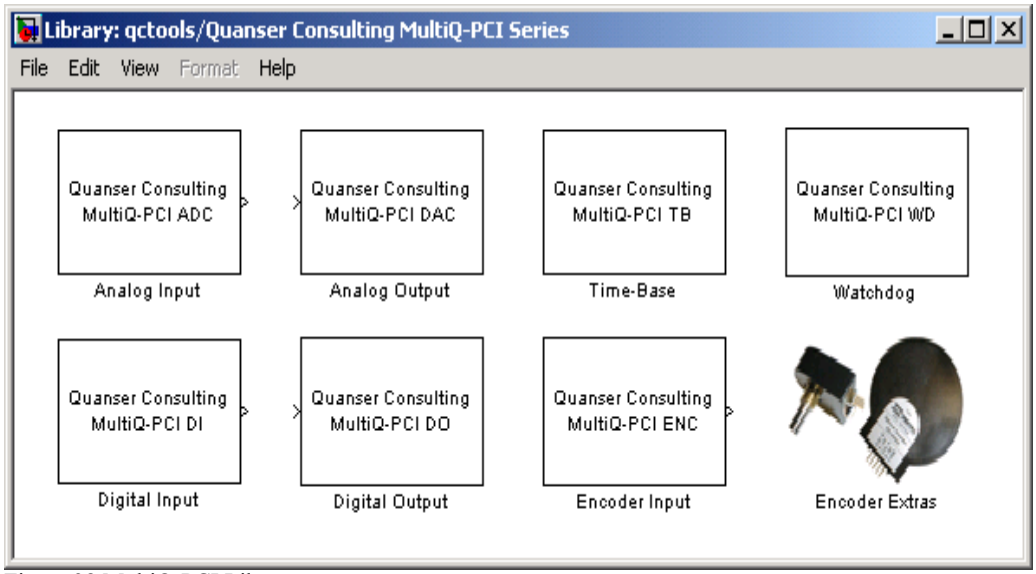


Figure 38 MultiQ-PCI Library

Analog Input

The Analog Input block is used to interface with the analog-to-digital converters on the MultiQ-PCI. **The Analog Input block is also used to read more than one analog input at a time.** In this case, the output from the analog input block will be a vector with one entry for each analog input read. **Note that there can only be one MultiQ-PCI Analog Input block (per card) in a Simulink diagram.** To change the analog input parameters, double-click on the block after dragging it into your Simulink diagram. The dialog box shown in Figure 39 below will appear.

The first field allows you to specify which MultiQ-PCI I/O card to use. The MultiQ-PCI card is a memory-mapped PCI card. The first card is referred to as board number 0 by WinCon. The next card is referred to as board number 1, etc. Hence, if there is more than one MultiQ-PCI card in your system, simply specify the board number of the card you

want. Since the correspondence between the board number and the slot in which the I/O cards are placed depends on the computer manufacturer, the easiest way to determine which card is board number 0, and which card is board number 1, etc. is to output different constant voltages on each board. Which card is which may then be determined easily using a voltmeter. All the MultiQ-PCI blocks have a Board Number parameter.

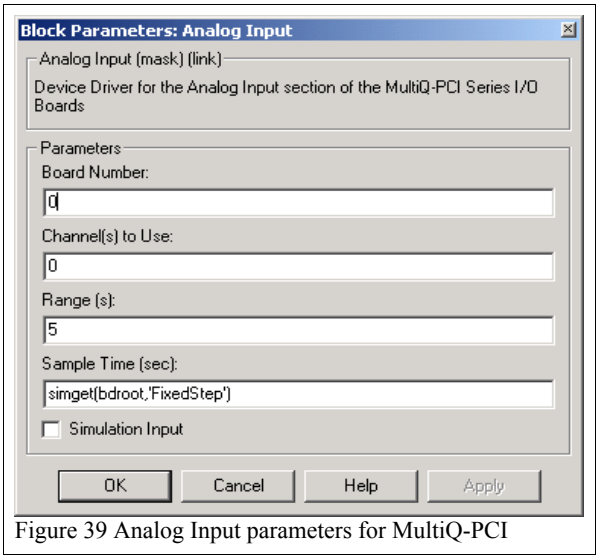


Figure 39 Analog Input parameters for MultiQ-PCI

The second field identifies the analog input channel, or channels, you wish to read. If a MultiQ terminal board is being used, then specify a scalar number from 0 to 7 to read one of the eight single-ended analog input channels. Otherwise, specify a scalar number from 0 to 15 to read one of the sixteen differential analog inputs. To read more than one channel, specify a vector. The channels will be read in the order that they appear in the vector.

For example, entering [5 2] will cause the Analog Input block to read analog channel 5 first and then to read analog channel 2. The output from the Analog Input block will be a vector, where the first element corresponds to analog channel 5, and the second element corresponds to analog channel number 2. The output units are volts.

In most cases, you will probably want to specify the channels in ascending order. You can do this using the convenient ':' vector shorthand notation of MATLAB. For example, to read analog channels 2 through 4, simply enter 2:4 into the Channel(s) to Use field.

The third parameter identifies the voltage range of the input. The analog inputs of the MultiQ-PCI support either a ±5V range or ±10V range. The range is software-programmable. Specify a 5 to use the ±5V range and specify 10 to use the ±10V range. Different ranges may be specified for each input channel by entering a vector for the range parameter. For example, if the Channel(s) to Use field is set to [2, 6] and the range is set to [10 5] then the Analog Input block will read from analog input channel 2 using the ±10V range and it will read from channel 6 using a ±5V range. Range values other than 5 or 10 are not permitted.

The fourth parameter is the sample time. All the MultiQ-PCI blocks include a sample time parameter so that they may be used in discrete-time systems. The default sample time is set to the base sampling rate of your system. You will not have to change this default unless you are developing a multi-rate system. The expression simget(bdroot,'FixedStep') returns the base sampling period as a real number, so you can multiply this expression by

an integer to get a multiple of the base sampling period. Alternatively, you can simply enter a real number corresponding to the sampling period for this block. The sample time must be a positive scalar value.

The final parameter is the Simulation Input check-box. This parameter has no effect on the real-time code. It is only used for simulation. When this box is checked, the Analog Input block will have an input port. During normal simulation, this input will be quantized according to the A/D resolution and fed to the output port. Thus, a model of the plant may be connected to this input so that the performance of the system may be simulated with quantization effects taken into account. See the MultiQ-3 Analog Input block description on page 96 for a more detailed description of this parameter.

Analog Output

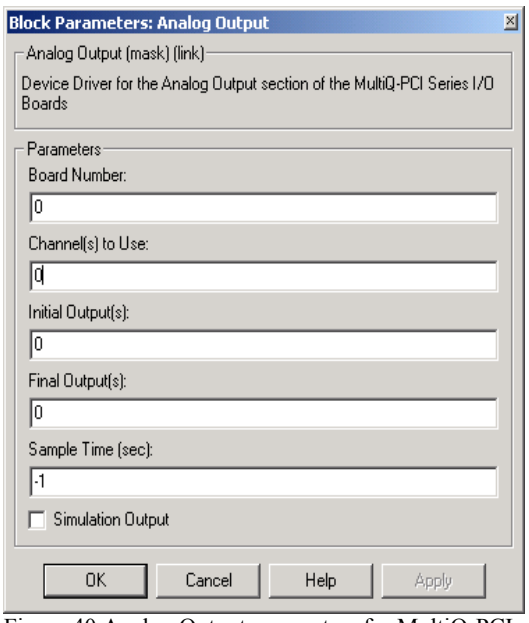


Figure 40 Analog Output parameters for MultiQ-PCI

The Analog Output block is used to interface with the digital-to-analog converters on the MultiQ-PCI data acquisition card. The analog output block may be used to write to more than one analog output at a time. In this case, the input to the analog output block must be a vector with one entry for each analog output written. You may put more than one analog output block into your diagram, if you prefer.

To change the analog output parameters, double-click on the block after dragging it into your Simulink diagram. The dialog box shown in Figure 40 will appear. Like the Analog Input block, there is a field for the Board Number. See the Analog Input section for a discussion of this parameter.

The Channel(s) to Use field identifies the analog output channels that will be written by this block. Specify a scalar number from 0 to 3 to write to one of the four analog output channels. To write more than one channel, specify a vector. The channels will be written in the order that they appear in the vector. For example, entering [5 2] will cause the Analog Output block to write to analog channel 5 first and then to write analog channel 2. The input to the Analog Output block must be a vector, where the first element corresponds to analog channel 5, and the second element corresponds to analog channel number 2. The input units are volts.

In most cases, you will probably want to specify the channels in ascending order. You can do this using the convenient ':' vector shorthand notation of MATLAB. For example, to

write to analog channels 2 through 4, simply enter 2:4 into the Channel(s) to Use field.

The Initial Output(s) field specifies the value(s) that will be written to the analog output channels when the real-time code is initialized, before it starts running. If you specify a scalar, then all analog output channels in the Channel(s) to Use field will be initialized to the same value. To initialize each channel to a different value, specify a vector containing the initial values for each channel.

The Final Output(s) field specifies the value(s) that will be written to the analog output channels when the real-time code is stopped. If you specify a scalar, then all analog output channels in the Channel(s) to Use field will be set to the same value. To set each channel to a different value, specify a vector containing the final values for each channel.

The second-last parameter is the sample time. All the MultiQ blocks include a sample time parameter so that they may be used in discrete-time systems. The default sample time is set to -1 in the case of the Analog Output block. A sampling time of -1 indicates that the sampling rate is inherited from the input signal driving the block. Thus, you will usually not have to change this parameter, even in a multi-rate system. The sample time can also be specified as a positive scalar. The expression simget(bdroot,'FixedStep') returns the base sampling period as a real number, so you can multiply this expression by an integer to get a multiple of the base sampling period. Alternatively, you can simply enter a real number corresponding to the sampling period for this block. The sample time must be -1 or positive.

The Simulation Output checkbox is similar to the Simulation Input option of the Analog Input block. Checking this box puts an output port on the Analog Output block. This output port is only used during simulation. It is not used during real-time execution. During simulation, this output port produces a quantized version of the data at the input port, using the resolution of the MultiQ-PCI digital-to-analog converters. Like the Simulation Input option of the Analog Input block, this feature may be used to combine a simulation of your hardware and the real-time code into a single Simulink diagram. See the discussion of the Analog Input block's Simulation Input parameter for more details.

Digital Input

The digital input block is used for reading the digital inputs on the MultiQ card. The dialog box of the Digital Input block is shown in Figure 41. Like the Analog Input block, there are fields for the Board Number, Sample Time and Simulation Input. See the Analog Input section for a discussion of these parameters.

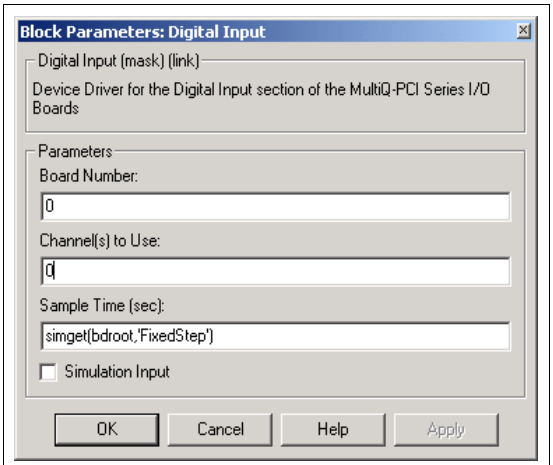


Figure 41 Digital Input parameters for MultiQ-PCI

The output of the Digital Input block is zero or one, not a voltage. The digital input channels to read are specified in the Channel(s) to Use parameter, much like the Analog Input block. A vector may be used to specify more than one input channel.

The Digital Input block supports up to 48 digital inputs. However, in this case, *the digital input channels are shared with the Digital Output block*. Thus, a Digital Input block and a Digital Output block cannot share the same channel numbers because the common channels would be using the same digital I/O pin. For example, if a Digital Input block was reading channel 0 then a Digital Output block could not be writing to channel 0.

Digital Output

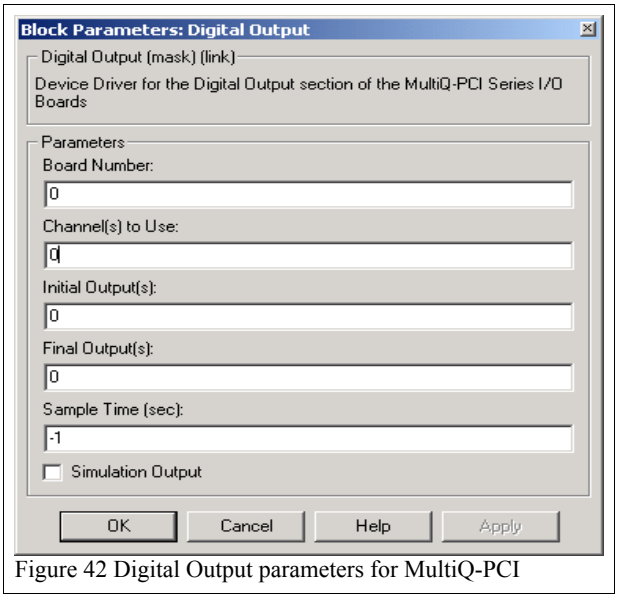


Figure 42 Digital Output parameters for MultiQ-PCI

The Digital Output block is used to write to the digital outputs of the MultiQ-PCI data acquisition card. The dialog box of the Digital Input block is shown in Figure 42. Like the Analog Output block, there are fields for the Board Number, Initial Output(s), Final Output(s), Sample Time and Simulation Input. See the Analog Output section for a discussion of these parameters. The Digital Output block takes inputs that are zero or one, not a voltage, and sends them to the digital outputs of the MultiQ-PCI.

The digital output channels to write are specified in the Channel(s) to Use parameter, much like the Analog Output block. A vector may be used to specify more than one output channel.

The Digital Output block supports up to 48 digital outputs. However, in this case, *the digital output channels are shared with the Digital Input block*. Thus, a Digital Output block and a Digital Input block cannot share the same channel numbers because the common channels would be using the same digital I/O pin. For example, if a Digital Output

block was writing to channel 0 then a Digital Input block could not be reading from channel 0.

Time Base

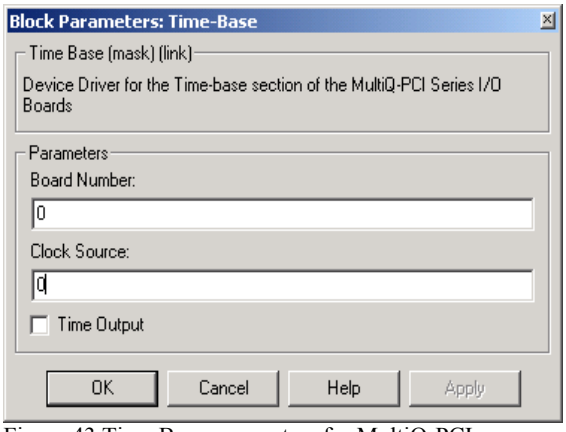


Figure 43 Time Base parameters for MultiQ-PCI

The Time Base block allows a hardware timer to be used to generate the sampling rate for the system. When no Time Base block is present in the diagram, the system timer is used. The system timer under Linux is fast and accurate, supporting sub-millisecond sampling rates, so this block is not strictly necessary, but it is available for those users wanting to use a hardware time-base on their data acquisition card.

In Windows '98, the Time Base block is required for sampling rates beyond 1kHz. The Time Base block is required In Windows NT/2000/XP for sampling rates beyond 10kHz.

The parameters for the MultiQ-PCI Time Base block are shown in Figure 43 below. See the Analog Input section for a discussion of the Board Number parameter.

The Clock Source parameter determines which counter on the MultiQ-PCI card will be used for the hardware time-base. There are six counters on the MultiQ-PCI card, so the clock source may be a scalar integer from 0 to 5. Note however that the same counters are used for the encoder inputs. Hence, *the Clock Source parameter cannot conflict with a channel number used in an Encoder Input block*. Clock source 0 corresponds to the same counter as encoder channel 0.

The Time Base block normally has no inputs or outputs. Checking the Time Output box puts an output port on the Time Base block. This port outputs the time since the simulation began. Hence, this output may be used instead of a separate Clock block (the Clock block is a Simulink block found in the Sources library).

Note that only one Time Base block may appear in a Simulink diagram, since only one hardware timer is needed to produce the base sampling rate. If more than one Time Base block is placed in a diagram then an error message will be issued when generating code.

Watchdog Timer

The Watchdog Timer block is used to program the watchdog timer on the MultiQ-PCI card. If the watchdog timer expires before the next sampling instant then the MultiQ-PCI card is

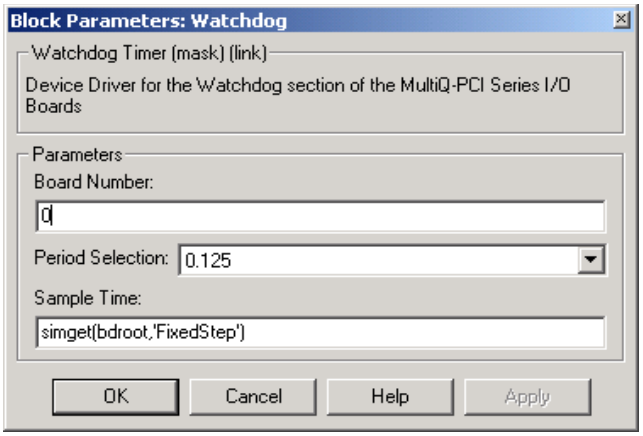


Figure 44 Watchdog Timer parameters for MultiQ-PCI

reset and the model is stopped. An error message will be issued in this case indicating that the watchdog timer expired. Note that resetting of the MultiQ-PCI card is done in hardware. Hence, even if the system has crashed, the card will still be reset when the watchdog timer expires. This block provides a useful safety feature. However, Quanser strongly recommends that other safety features also be implemented in your Simulink diagrams. An unstable controller would continue to reset the watchdog timer (because it hasn't crashed) yet would drive the hardware out of control! Thus, other safety features should also be incorporated. For example, limit switches may be polled using a Digital Input block and fed to a Stop Simulation block to halt the real-time code in the event of a limit switch being hit.

The parameters of the Watchdog Timer block are illustrated in Figure 44. See the Analog Input section for a discussion of the Board Number and Sample Time parameters.

The Period Selection parameter determines the period of the watchdog timer. There is a choice of four periods: 0.125 seconds, 0.5 seconds, 1 second and 10 seconds. The period should always be longer than the base sampling period of the system, yet as small as possible. Thus, in the event of a crash, the watchdog timer will reset the card as soon as possible, but it will not interfere with the normal operation of the controller. For example, if the controller sampling period is 0.010 seconds, then the watchdog period should be set to 0.125.

Encoder Input

The Encoder Input block is used to read the counter values of the MultiQ-PCI encoder inputs. Up to six encoder inputs are supported, number 0 through 5. Note that with the MultiQ terminal board, encoder channels 6 and 7 are not used for encoders. The output of the encoder block is the 24-bit encoder count(s). More than one encoder channel may be read at the same time and more than one Encoder Input block may be present in your Simulink diagram. The parameters of the Encoder Input block are illustrated in Figure 45. Like the Analog Input block, there are fields for the Board Number, and Sample Time. See the Analog Input section for a discussion of these parameters.

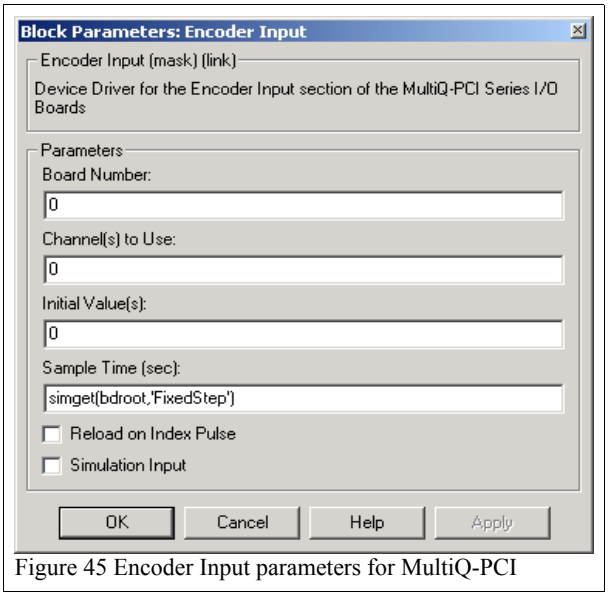


Figure 45 Encoder Input parameters for MultiQ-PCI

The Channel(s) to Use parameter specifies which encoder channels are to be read. More than one encoder channel may be specified by entering a vector for this parameter. Note however that encoders are handled using the counters on the MultiQ-PCI board. These counters are also used for the Time Base block. Hence, no channel number may conflict with the clock source of a Time Base block in the same diagram. Clock source 0 corresponds to the same counter as encoder channel 0.

The Initial Value(s) parameter is used to preload the encoder counts when the model starts. For example, setting this parameter to 0 will cause all encoders to read a value of 0 initially. The initial value for each encoder channel may be specified separately by entering a vector for this parameter. For example, if the Channel(s) to Use parameter is [2, 4] and the Initial Value(s) parameter is [32768, -8192] then encoder channel 2 will have an initial value of 32768 and channel 4 will have an initial value of -8192 when the model is run.

The final parameter, the Simulation Input check-box, puts an input port on the Encoder Input block when it is selected. This input port may be used much like the Simulation Input on the Analog Input block. However, in this case, the input is a count value that is fed to the output (during simulation only) after restricting it to a 24-bit value using the modulus operator. The simulation input is ignored in real-time code.

Encoder Extras

It is often convenient to calibrate the encoders in the real-time code in response to some event, such as a limit switch being hit or at a certain time. The Encoder Extras library provides a set of Encoder Reset blocks for performing such calibration. One block resets the encoder value based on a level-sensitive input. Two others reset the encoder value based on an edge-triggered input. Since the Encoder Input block sets the initial value of the encoder count, the Encoder Reset blocks are not necessary to set the initial count value. However, they are very useful for calibration. For a full discussion of these blocks, refer to the section on the MultiQ-3 Encoder Extras library. The blocks are identical for the MultiQ-PCI except that a Board Number parameter is used in place of a Base I/O Address parameter. See the MultiQ-PCI Analog Input block description for details on the Board Number parameter.

Serial Drivers

The Serial Drivers library provides blocks for accessing the serial ports of your PC. The Serial Driver library is depicted in Figure 46 below. The blocks recognize COM1 through COM4. Baud rates of up to 115,200 baud are supported. The blocks also support all the built-in data types, so they may even be used for tele-operation across a null modem cable or communication with embedded systems.

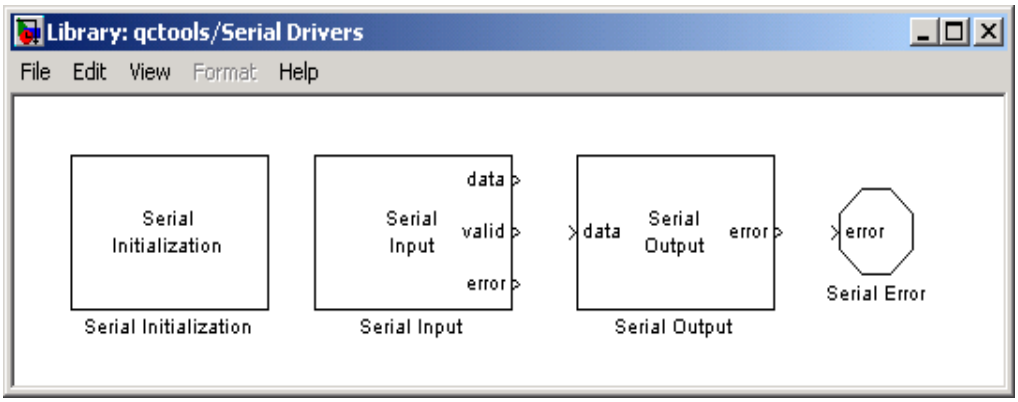


Figure 46 Serial Drivers

Serial Initialization

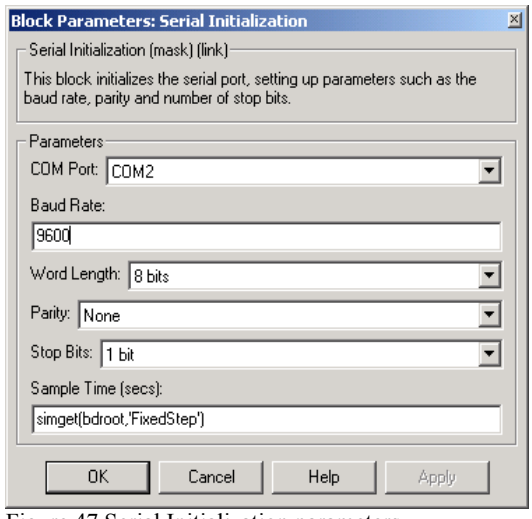


Figure 47 Serial Initialization parameters

The Serial Initialization block is used to set the serial port parameters such as baud rate, parity, stop bits, etc. A separate block is employed because there are settings that are typically only set once at the start of execution. This technique greatly reduces the number of parameters of the other serial driver blocks. Only one Serial Initialization block may be placed in the diagram per serial channel.

The parameters to the Serial Initialization block are shown in Figure 47. The COM Port parameter selects which serial port to initialize. The driver assumes that a 16550A UART or equivalent is available on your system. The UART fifos are used to maximize throughput.

The Baud Rate parameter is used to set the baud rate. Baud rates as low as 50 and as high as 115,200 are possible. Non-standard baud rates in this range are also permitted, although are not recommended. The default baud rate is 9600 baud.

The Word Length parameter sets the number of bits per character sent. If a word length other than 8 is used, then the bytes in the input signal will be truncated to the number of bits in the word length. Hence, an 8 bit word length should be used for all data types except the 8-bit data types int8 and uint8. The default word length is 8 bits.

The Parity parameter sets the parity to even, odd, or none. The default is none.

The Stop Bits parameter determines the number of stop bits. The choices are 1, 1.5 or 2 stop bits. However, 2 stop bits is only possible with a word length of 5. If a word length other than 5 is used then the stop bits must be set to 1 or 1.5.

The Sample Time parameter is only available so that the block acts as a discrete-time block. If all the blocks in your diagram are discrete then more integration methods are allowed by Simulink. You should never need to change this parameter because it has no impact on the real-time code. The Serial Initialization block only executes when the model is started and when it terminates. No code is executed for the Serial Initialization block while the model is running.

Serial Output

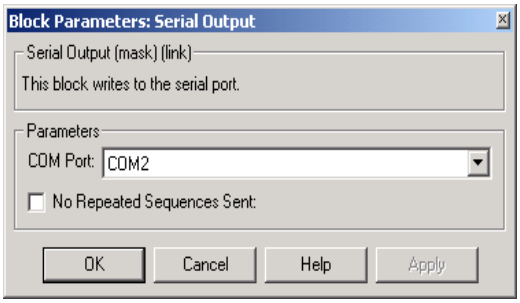


Figure 48 Serial Output parameters

The Serial Output block is used to write data to the serial port. It writes all the data at its input to the serial port each sampling instant. Hence, if the input is a vector, the entire vector is written to the serial port.

The Serial Output block also supports any of the standard data types at its input. The 8-bit data types, int8 and uint8, are written to the serial port as characters. The 8-bit data types are useful for printing messages to the serial port. The larger data types, such as double, are written to the serial port as a series of bytes. For example, a double value is sent as 8 bytes, while a single precision value is sent as 4 bytes. The Serial Input block can be programmed to recognize any of the standard data types as well, so the Serial Input and Output blocks may be used for tele-operation over a null modem cable, or for communicating with embedded systems via a serial link. The To Bytes block can be used to combine data types of different sizes into a single write operation.

The Serial Output block has two parameters, as shown in Figure 48: COM Port and No Repeated Sequences Sent. The COM Port parameter identifies the serial port to which to write. A Serial Initialization block should be used to initialize this COM port. The No Repeated Sequences Sent parameter is used to prevent redundant information from being sent out the serial port. It is particularly useful for communicating with embedded systems, such as the Quanser PIC-based boards. When the No Repeated Sequences

Sent box is checked, the Serial Output will only write its input to the serial port when that input changes (and the initial value). Hence, with this option enabled, you may send commands to an embedded system without worrying that the same command will be sent every sampling instant.

The Serial Output block also has an error signal output. The output is normally zero, but it goes to one if the Data Set Ready signal has not been asserted on the serial port. Absence of Data Set Ready generally means that the software at the other end of the connection is not running or has not made a connection. Alternatively, it could mean that the serial link is broken. For example, in Windows, the HyperTerminal program does not assert Data Set Ready until you "connect". Rather than stop the model with an error, the Serial Output block gives the user control over this scenario by supplying the error output. To stop the model on this error, connect the output to a Stop Simulation block or Quanser's Stop With Error block.

Serial Input

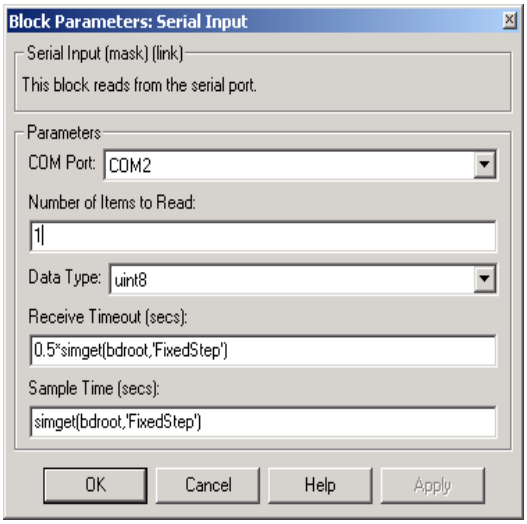


Figure 49 Serial Input parameters

The Serial Input block reads from the serial port. Every sampling instant it reads the specified number of items. The data type of each item read can also be specified so that the Serial Input block is fully compatible with the Serial Output block. It may also be used to read different data types from an embedded system connected by a serial link to the PC, such as Quanser's PIC-based modules.

To prevent the Serial Input block from starving the operating system for CPU time while it waits for data, the Serial Input block uses a time-out interval. If not enough data arrives within the time-out interval then the data is considered invalid and this failure is indicated by the valid output signal.

The Serial Input dialog box of the Serial Input block is shown in Figure 49. The COM Port parameter specifies the serial port from which to read. A Serial Initialization block should be present to initialize this serial port.

The Number of Items to Read parameter indicates the number of data items to read from the serial port. The size of each data item depends on the data type selected. For example, if the data type is set to double, then each item requires 8 bytes. Thus, if the number of items is set to 2 then 16 bytes will be read from the serial port. The 16 bytes will be interpreted as two 8-byte double precision values and these values will be output as a 2-vector to the data

output of the Serial Input block. The From Bytes block can be used to separate data types of different sizes read in a single read operation.

The Receive Timeout parameter indicates how long the Serial Input block will wait (polling) until it considers the read to have failed. Since the Serial Input block reads every sampling instant, this parameter should be set to less than the sampling time. The default value is half the base sampling rate of the model. The time-out is for all the data combined – not for each individual data item.

The Sample Time parameter specifies the sampling period to use for this block. The default value is the same as the base sampling period of the model. Other sampling times may be specified. The sample time must be a positive scalar. For example, to read the serial port every 1/10th of a second, set the Sample Time to 0.1.

The Serial Input block has several outputs. The data that has been read is output on the data output port. It will be a vector whose length is the same as the number of data items specified and of the data type indicated in the Data Type parameter. If data was not read successfully from the serial port, then the previous data is output.

The valid output is a scalar output. When the data is read successfully, the valid output is one. However, if the receive time-out expired before all the data arrives then the valid output goes to zero. Hence, the valid signal may be used to ignore invalid data or to flag an error. The valid output also goes to zero if an error occurs reading the serial port, such as a framing error, parity error or overrun error.

The error signals provides more detail about why a read failed. It is a vector of six elements, with each element corresponding to a different error. When the read completes successfully, the error output is all zero. When an error occurs, the appropriate element in the error vector goes to one. Table 36 enumerates the meaning of each error vector element.

Element	*Error Description*
1	No connection – Data Set Ready not asserted.
2	Overrun error – baud rate setting is likely incorrect
3	Parity error – parity setting incorrect or line is noisy
4	Framing error – stop bit setting incorrect or line is noisy
5	Break interrupt – break signal sent by remote terminal. Connection being broken?
6	Receive time-out – all the data did not arrive within the time-out interval

Table 36 Serial Input errors

Serial Error

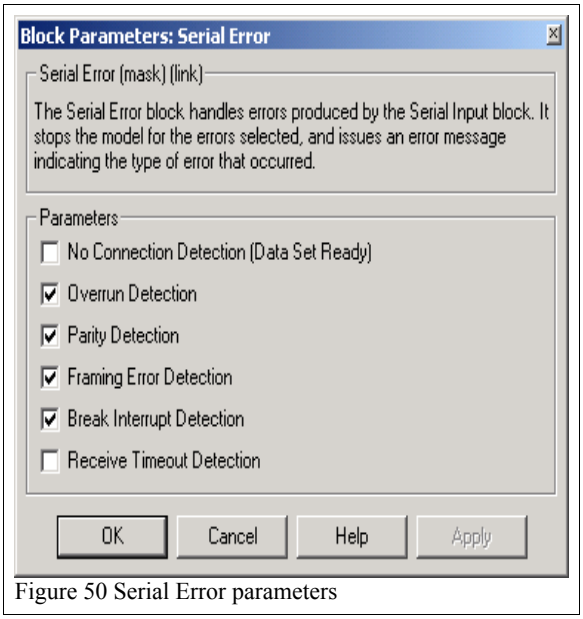


Figure 50 Serial Error parameters

The Serial Error block is designed to be used with the Serial Input block. When the error signal from the Serial Input block is connected to the Serial Error block then the model will be stopped with an appropriate error message whenever one of the selected serial input error occurs.

Double-click on the Serial Error block to open the dialog shown in Figure 50. The Serial Error block will detect any error for which the associated check-box is selected. See the Serial Input block for a description of the different types of errors that can be detected. When a selected error is detected, the model will be stopped and an error-specific message will be issued to the user.

The No Connection error and Receive Time-out errors are not detected by default because these errors are likely to occur before the remote system has been initialized. If you always know that the remote system will be running prior to starting the model, then you can enable the detection of these errors. Note that a receive time-out will occur unless the remote system is sending data at the same rate as the model is expecting to receive data.

Extra Sources

Sources are Simulink blocks, like the Sine Wave block, that produce an output that is not a function of any input. Quanser Consulting provides additional sources to enhance the Simulink environment. Double-click on the Extra Sources icon in the Quanser Toolbox, qctools, to access these sources. The extra sources are illustrated in Figure 51.

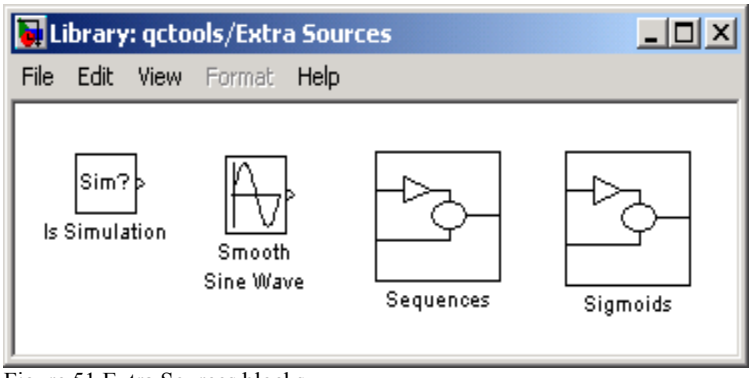


Figure 51 Extra Sources blocks

Is Simulation?

The Is Simulation? block may be used in a Simulink diagram to determine when the model is running as a normal simulation or when it is running as real-time code. The output of the block is one when the model is running as a normal simulation. It is zero when running as real-time code.

The block is most useful with enabled subsystems, because it allows you to enable or disable the execution of certain code depending on whether it is running in simulation or real-time. For example, in conjunction with blocks like the MultiQ Analog Input with Simulation Input block, the Is Simulation? block could be used to execute a model of the hardware in simulation, but prevent that model code from executing when the diagram was run in real-time. Hence, the same diagram could be used for simulation as well as the real-time code.

Smooth Sine Wave

The Smooth Sine Wave block generates a sine wave, with an optional cosine output. The sine wave output remains continuous even when the amplitude and frequency are changed during simulation. This block is useful as a reference signal for real-time systems. Note that the output cannot be a vector because it is not generally possible to maintain continuity of multiple sinusoids while keeping the sinusoids synchronized.

The dialog box of the Smooth Sine Wave block is shown in Figure 52. The block input parameters are the amplitude of the sine wave, its frequency in Hertz, its initial phase in

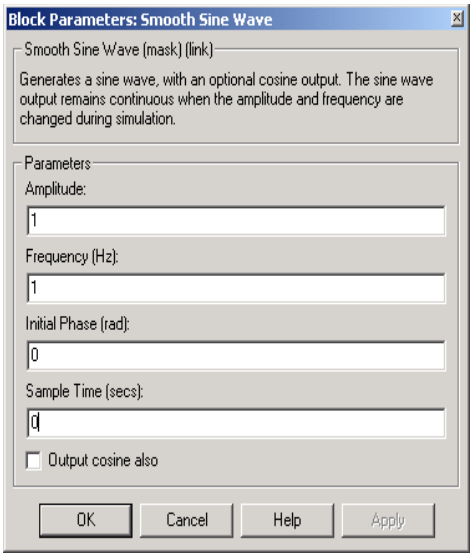


Figure 52 Smooth Sine Wave Parameters

radians, and the sampling rate for this block. If the amplitude is modified during simulation then the phase of the output is adjusted to ensure that the curve is always continuous. If the frequency is changed during simulation then the phase of the sine wave is adjusted to ensure that the output is always continuous. The initial phase parameter only determines the phase at time t=0. Changing this parameter during simulation has no effect on the output. The Sample Time may be zero to make it a continuous-time block or -1 to inherit the sample time from the block to which the sine wave is connected. Otherwise, specify a positive value to indicate a discrete sampling time.

If the Output cosine also option is checked then the block will have an additional output. This output will produce the cosine. Note that very little extra code is required to produce the cosine in addition to the sine. Note, however, that the cosine is not guaranteed to be continuous under parameter changes.

Sequences

Figure 53 shows the different Sequences blocks provided as part of the Quanser Extra Sources.

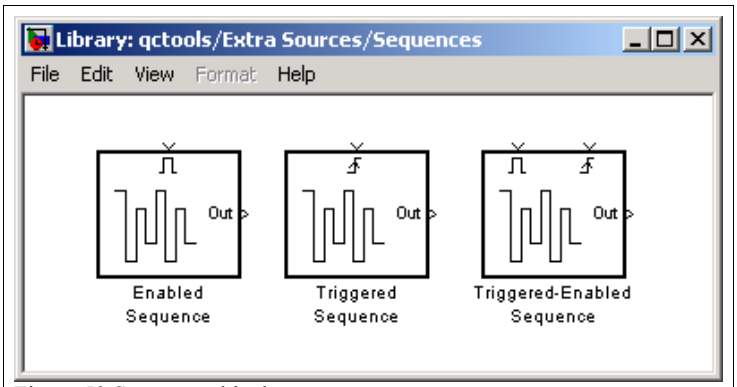


Figure 53 Sequences blocks

Enabled Sequence

The Enabled Sequence block sequentially outputs the elements of a vector. Each sampling instant that it is enabled, it outputs the next element in the vector. The dialog box of the

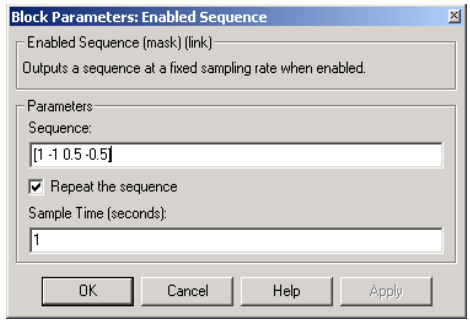


Figure 54 Enabled Sequence Parameters

Enabled Sequence block is shown in Figure 54. If the Repeat Sequence option is checked, then the sequence is repeatedly output. Otherwise, the output stays at the last value in the vector when all elements have been output. The Enabled Sequence block is simpler to use than the Repeating Sequence block, since no time vector is required. It also does not need to perform linear interpolation.

Triggered Sequence

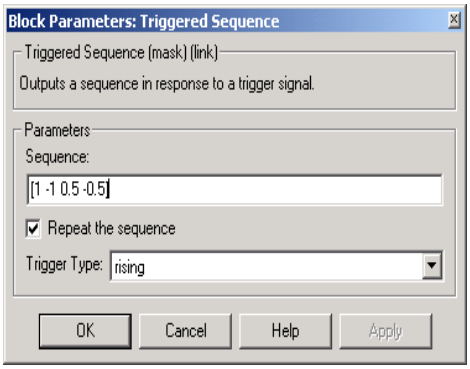


Figure 55 Triggered Sequence Parameters

The Triggered Sequence block sequentially outputs the elements of a vector. Each time the trigger is fired, it outputs the next element in the vector. The dialog box of the Triggered Sequence block is shown in Figure 55. If the Repeat Sequence option is checked, then the sequence is repeatedly output. Otherwise, the output stays at the last value in the vector when all elements have been output. The Triggered Sequence block is useful when a series of reference positions need to be output, but the time to reach each target is unknown in advance. This block is typically used in conjunction with the Sigmoid or Triggered Sigmoid blocks.

Triggered-Enabled Sequence

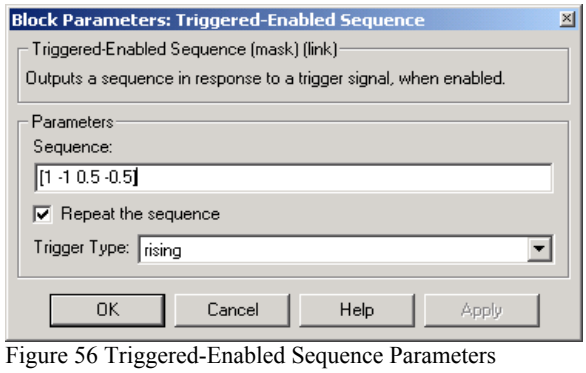


Figure 56 Triggered-Enabled Sequence Parameters

The Triggered-Enabled Sequence block sequentially outputs the elements of a vector. Each time the trigger is fired and the enable input is asserted, it outputs the next element in the vector. The dialog box of the Triggered-Enabled Sequence block is shown in Figure 56. If the Repeat Sequence option is checked, then the sequence is repeatedly output. Otherwise, the output stays at the last value in the vector when all elements have been output. The Triggered-Enabled Sequence block is useful when a series of reference positions need to be output, but the time to reach each target is unknown in advance. This block is typically used in

conjunction with the Sigmoid or Triggered Sigmoid blocks.

Sigmoids

Quanser supplies Sigmoid blocks, as shown in Figure 57, that can be used to command robot joints. These sigmoid blocks generate trapezoidal velocity profiles, i.e. rectangular acceleration profiles. These result is smooth (i.e. parabolic) position profiles. The maximum acceleration and velocity to be reached are finite and user-specified.

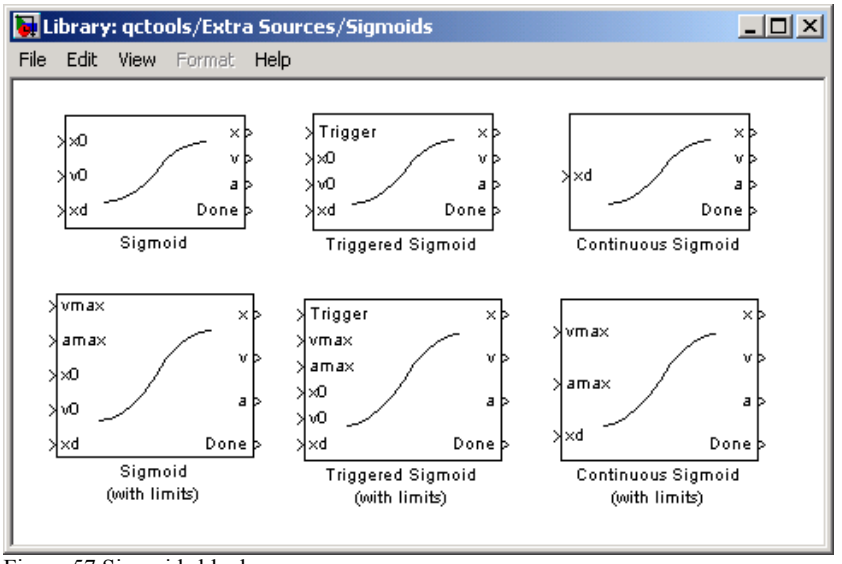


Figure 57 Sigmoids blocks

As a reminder, applying a step position input to a robot joint controller is never recommended. The reason is that this would request infinite velocity and acceleration and therefore would excite flexible modes that are not taken into consideration by the robot motion controller and could potentially damage the robot itself and its joints.

Sigmoid

The Sigmoid block generates a smooth trajectory from the initial position (and velocity) to the target position. The velocity at the target position will be zero. It uses the specified maximum velocity and acceleration to derive the fastest possible trajectory from the initial position to the target position.

The dialog box of the Sigmoid block is shown in Figure 58. The Sigmoid block samples its inputs at regular intervals, determined by the Input Sample Time parameter.

Each time the inputs are sampled, the trajectory is re-computed. Hence, the sample time should normally be slower than the time taken to traverse the trajectory.

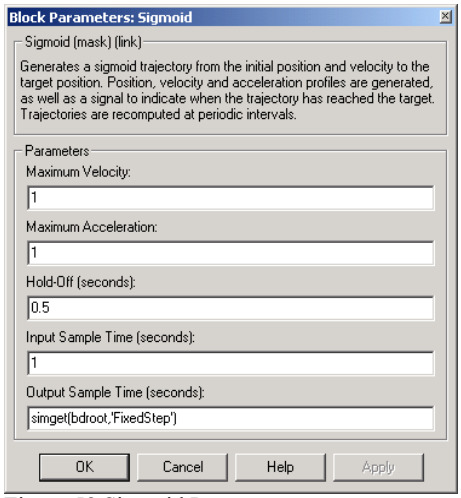


Figure 58 Sigmoid Parameters

The output of the Sigmoid block is generated using the sampling rate specified by the Output Sample Time parameter. The Output Sample Time may be set to zero for a continuous-time trajectory.

The Sigmoid block provides the Done signal to indicate when the trajectory reaches its target. This output may be used to trigger another event, or reset encoder counts, for example.

Sigmoid (with limits)

The Sigmoid (with limits) block generates a smooth trajectory from the initial position (and velocity) to the target position. The velocity at the target position will be zero. It uses the specified maximum velocity and acceleration to derive the fastest possible trajectory from the initial position to the target position. It differs from the Sigmoid block in that the maximum velocity and acceleration are inputs to the block rather than parameters. Hence, they may be time-varying.

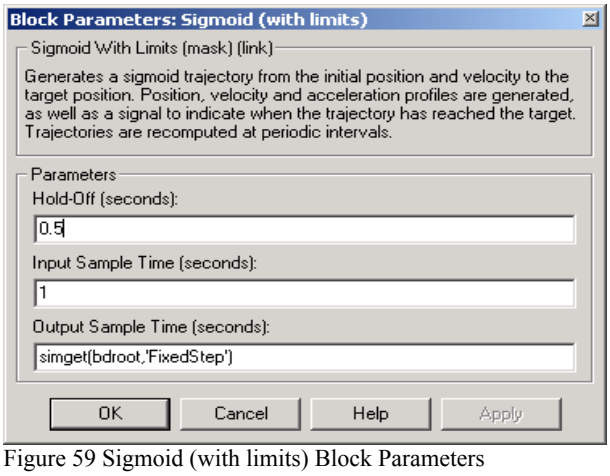


Figure 59 Sigmoid (with limits) Block Parameters

The dialog box of the Sigmoid (with limits) block is shown in Figure 59. The Sigmoid (with limits) block samples its inputs at regular intervals, determined by the Input Sample Time parameter. Each time the inputs are sampled, the trajectory is re-computed. Hence, the sample time should normally be slower than the time taken to traverse the trajectory.

The output of the Sigmoid (with limits) block is generated using the sampling rate specified by the Output Sample Time parameter. The Output Sample Time may be set to zero for a continuous-time trajectory.

The Sigmoid (with limits) block provides the Done signal to indicate when the trajectory reaches its target. This output may be used to trigger another event, or reset encoder counts, for example.

Triggered Sigmoid

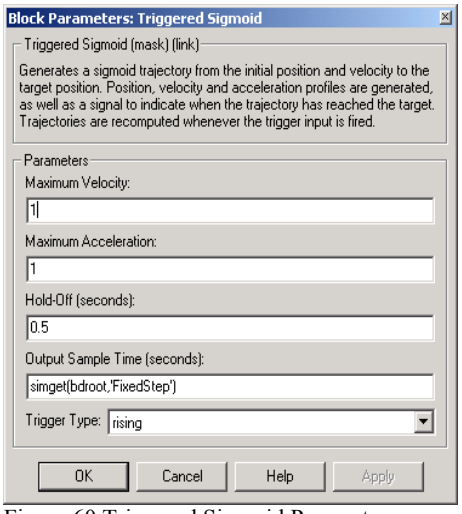


Figure 60 Triggered Sigmoid Parameters

The Triggered Sigmoid block generates a smooth trajectory from the initial position (and velocity) to the target position. The velocity at the target position will be zero. It uses the specified maximum velocity and acceleration to derive the fastest possible trajectory from the initial position to the target position.

The dialog box of the Triggered Sigmoid block is shown in Figure 60. The Triggered Sigmoid block samples its inputs when the trigger input fires. Each time the inputs are sampled, the trajectory is re-computed.

The output of the Triggered Sigmoid block is generated using the sampling rate specified by the Output Sample Time parameter. The Output Sample Time may be set to zero for a continuous-time trajectory.

The Triggered Sigmoid block provides the Done signal to indicate when the trajectory reaches its target. This output may be used to trigger another event, or reset encoder counts, for example.

Triggered Sigmoid (with limits)

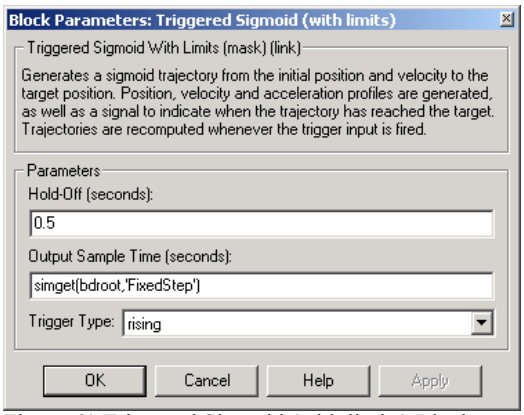


Figure 61 Triggered Sigmoid (with limits) Block Parameters

The Triggered Sigmoid (with limits) block generates a smooth trajectory from the initial position (and velocity) to the target position. The velocity at the target position will be zero. It uses the specified maximum velocity and acceleration to derive the fastest possible trajectory from the initial position to the target position. It differs from the Triggered Sigmoid block in that the maximum velocity and acceleration are inputs to the block rather than parameters. Hence, they may be time-varying.

The dialog box of the Triggered Sigmoid (with limits) block is shown in Figure 61. The

Triggered Sigmoid (with limits) block samples its inputs when the trigger input fires. Each time the inputs are sampled, the trajectory is re-computed.

The output of the Triggered Sigmoid (with limits) block is generated using the sampling rate specified by the Output Sample Time parameter. The Output Sample Time may be set to zero for a continuous-time trajectory.

The Triggered Sigmoid (with limits) block provides the Done signal to indicate when the trajectory reaches its target. This output may be used to trigger another event, or reset encoder counts, for example.

Continuous Sigmoid

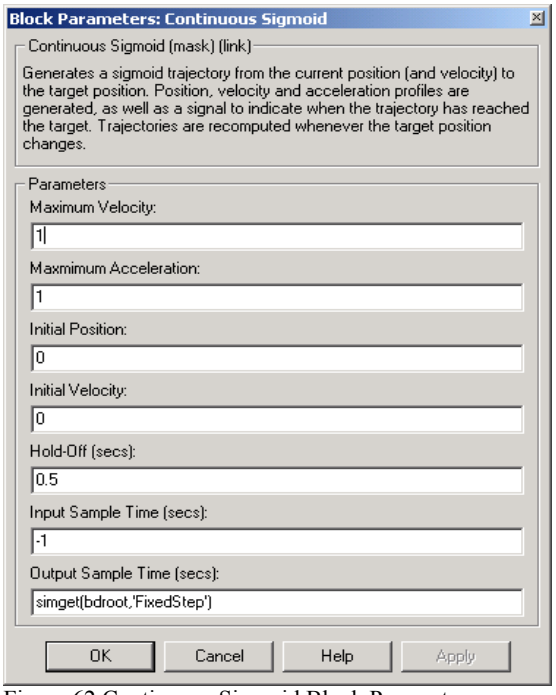


Figure 62 Continuous Sigmoid Block Parameters

The Continuous Sigmoid block generates a smooth trajectory from the current position (and velocity) to the target position. The velocity at the target position will be zero. It uses the specified maximum velocity and acceleration to derive the fastest possible trajectory from the current position to the target position. The current position and velocity are determined by the current trajectory and the current time. The initial position and velocity (when the model is started) may be specified as parameters.

The dialog box of the Continuous Sigmoid block is shown in Figure 62. The Continuous Sigmoid block samples its inputs at regular intervals, determined by the Input Sample Time parameter. Each time the inputs are sampled and the target position has changed, the trajectory is re-computed.

Hence, the sample time should normally be slower than the time taken to traverse the trajectory.

The output of the Continuous Sigmoid block is generated using the sampling rate specified by the Output Sample Time parameter. The Output Sample Time may be set to zero for a continuous-time trajectory.

The Continuous Sigmoid block provides the Done signal to indicate when the trajectory reaches its target. This output may be used to trigger another event, or reset encoder counts, for example.

Continuous Sigmoid (with limits)

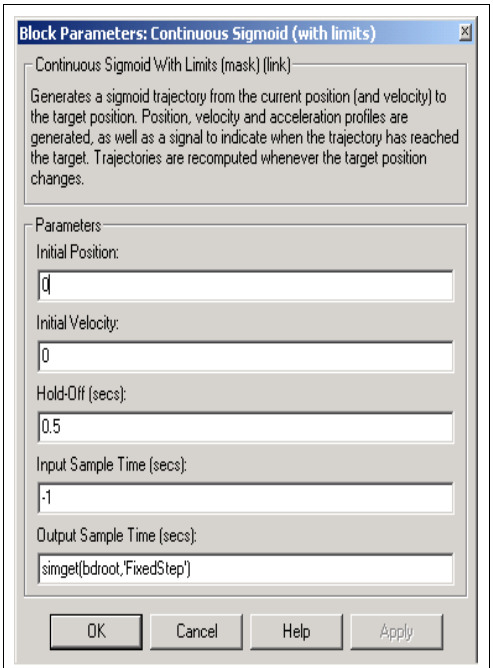


Figure 63 Continuous Sigmoid (with limits) Block Parameters

The Continuous Sigmoid (with limits) block generates a smooth trajectory from the current position (and velocity) to the target position. The velocity at the target position will be zero. It uses the specified maximum velocity and acceleration to derive the fastest possible trajectory from the current position to the target position. The current position and velocity are determined by the current trajectory and the current time. The initial position and velocity (when the model is started) may be specified as parameters. It differs from the Continuous Sigmoid block in that the maximum velocity and acceleration are inputs to the block rather than parameters. Hence, they may be time-varying.

The dialog box of the Continuous Sigmoid (with limits) block is shown in Figure 63. The Continuous Sigmoid (with limits) block samples its inputs at regular intervals, determined by the Input Sample Time parameter. Each time the inputs are sampled and the target position has changed, the trajectory is re-computed. Hence, the sample time should normally be slower than the time taken to traverse the trajectory.

The output of the Continuous Sigmoid (with limits) block is generated using the sampling rate specified by the Output Sample Time parameter. The Output Sample Time may be set to zero for a continuous-time trajectory.

The Continuous Sigmoid (with limits) block provides the Done signal to indicate when the trajectory reaches its target. This output may be used to trigger another event, or reset encoder counts, for example.

Extra Sinks

Sinks are Simulink blocks, like Scopes, that take an input but generally do not have output ports. Quanser Consulting provides additional sinks to enhance the Simulink environment. Double-click on the Extra Sinks icon in the Quanser Toolbox, qctools, to access these sinks. The extra sinks are illustrated in Figure 64. Two sinks are currently provided: the Stop With Error block and the PC Speaker block.

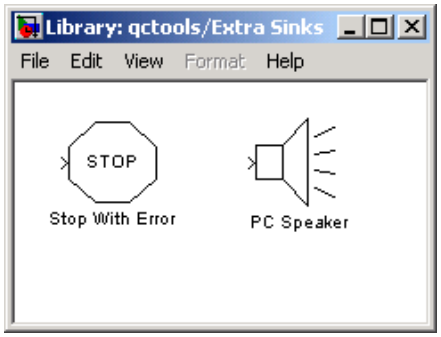


Figure 64 Extra Sinks

Stop With Error

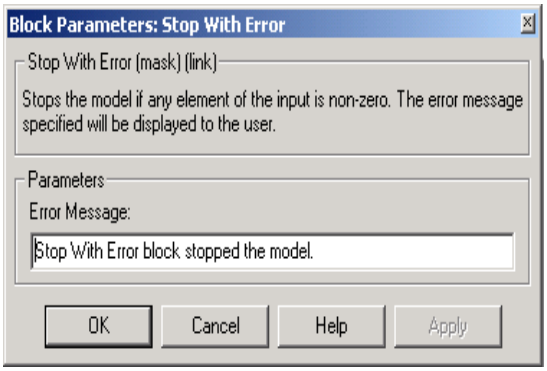


Figure 65 Stop With Error parameters

The Stop With Error block is quite useful for reporting error conditions in your model. It stops the model when its input is non-zero and issues an error. The user can specify the error message that is issued. Hence, this block is useful for reporting different errors in your model. For example, the model could be set up to stop when a limit switch is hit and to issue a different error message depending on the limit switch encountered.

The Stop With Error block takes a single parameter, the error message string, as illustrated in Figure 65. Quotes are not required around the string because the Error Message parameter is not evaluated. The error message specified will be reported to the user when the input to the block becomes non-zero.

PC Speaker

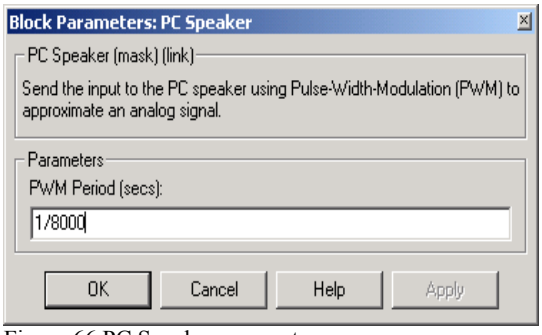


Figure 66 PC Speaker parameters

The PC Speaker block allows you to produce sound using the PC Speaker. The block is primarily for demonstration purposes since no sound card is required. However, novel uses may be found for this block. The dialog box of the PC Speaker block is shown in Figure 66.

The speaker in the PC allows only very limited control. It is really designed for generating a single, grating tone. However, the PC Speaker block uses Pulse-Width Modulation (PWM) techniques to produce an arbitrary analog waveform from the speaker. PWM involves using a square wave of varying duty cycle to emulate an analog signal level. Since the mechanical speaker system acts as a low-pass filter, this technique does in fact produce the desired result.

The PC Speaker block takes a single parameter – the period of the PWM waveform. The base sampling rate of your model must be at least as fast as the PWM period in order to vary the duty cycle of the PWM waveform. The higher the base sampling rate of your system, the better the resolution of the PWM. The default value corresponds to an 8 kHz PWM frequency. Hence, at the default setting the user will hear the 8 kHz tone on top of the audio waveform being output. Use a faster PWM period to move this tone out of hearing range. Writing to the PC speaker is very fast, so impressive sampling rates may be used.

Transformations

Some operations are common when implementing real-time models. For example, converting a transfer function from continuous-time to discrete-time is a typical operation, especially when implementing a multi-rate control system. The Transformations group in the Quanser Toolbox provides a set of Simulink blocks to perform these kinds of functions for you. The block set is depicted in Figure 67.

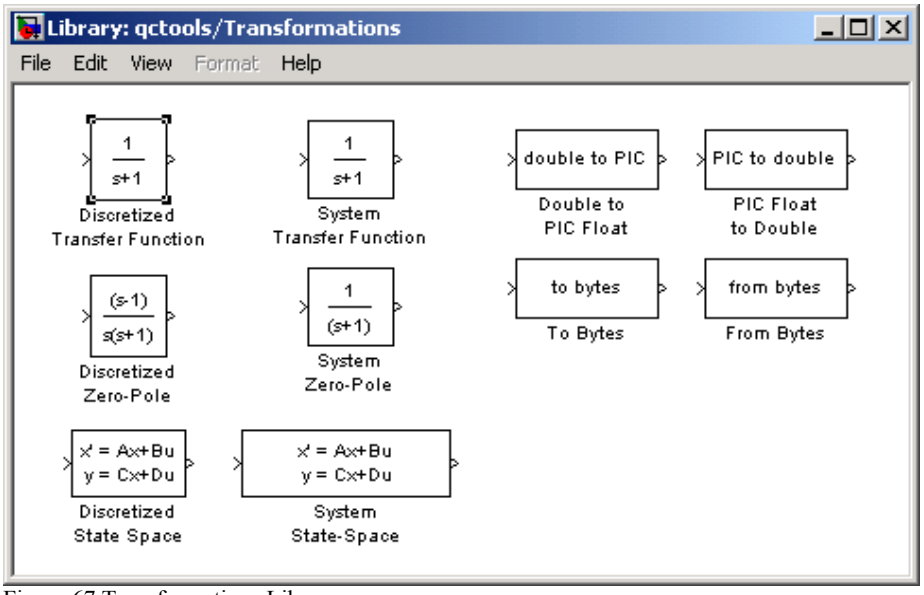


Figure 67 Transformations Library

Discretized Transfer Function

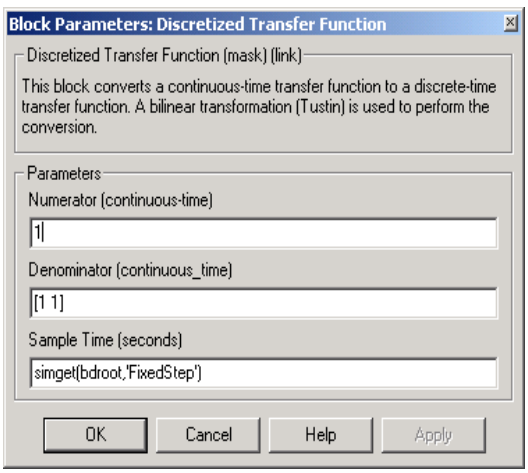


Figure 68 Discretized Transfer Function parameters

This Discretized Transfer Function block converts an analog transfer function to a discrete time transfer function for a specified sampling period. The parameters of the block reflect the continuous-time transfer function but the *implementation* is discrete-time. The conversion is performed using the Tustin bilinear transformation. The analog transfer function is displayed in the block.

The parameters of the Discretized Transfer Function block are shown in Figure 68. The numerator field is a vector representing the numerator polynomial of the *continuous-time* transfer function. Similarly, the denominator

parameter is a vector representing the denominator polynomial of the continuous-time transfer function. These two parameters are exactly the same as the continuous-time Transfer Function block provided in Simulink.

The last parameter is the sample time. The continuous-time transfer function identified by the first two parameters will be discretized for this sampling period. The resulting discrete-time transfer function will be used in the real-time code and in simulation. The default sampling period is the base sampling period of your model.

Discretized Zero-Pole

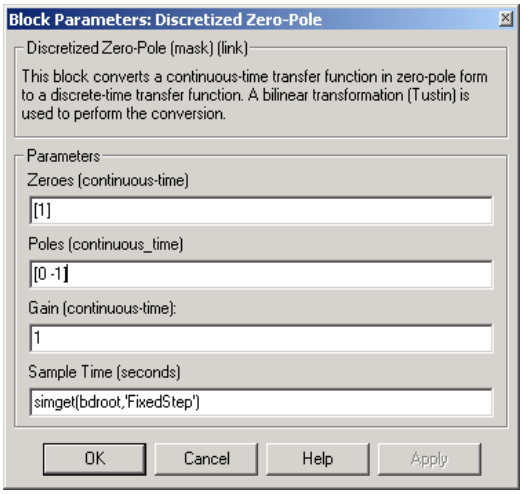


Figure 69 Discretized Zero-Pole Block parameters

The Discretized Zero-Pole block converts a continuous-time transfer function in zero-pole form to a discrete-time transfer function. A bilinear transformation (Tustin) is used to perform the conversion. The dialog box of the Discretized Zero-Pole block is shown in Figure 69.

Discretized State Space

The Discretized State Space block converts a continuous-time LTI system in state-space form to a discrete-time state-space system. A bilinear transformation (Tustin) is used to perform the conversion. The dialog box of the Discretized State Space block is shown in Figure 70. The block parameters are the matrices A, B, C, and D for the continuous-time system. These matrices appear in the state equations: x' = Ax + Bu and y = Cx + Du.

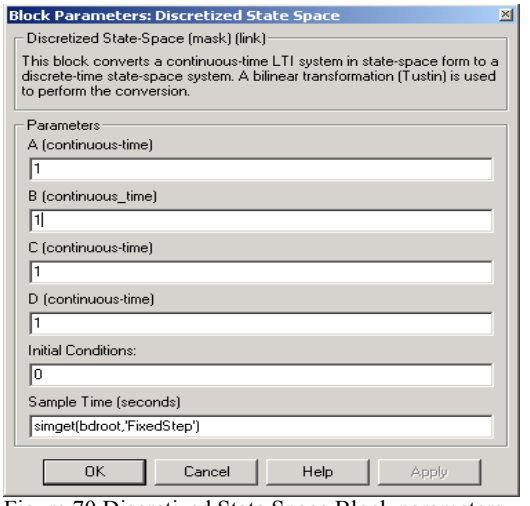


Figure 70 Discretized State Space Block parameters

System Transfer Function

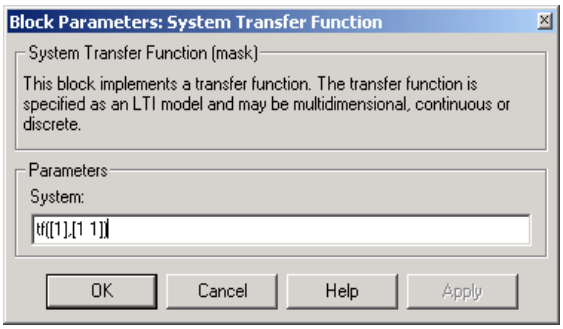


Figure 71 System Transfer Block parameters

The System Transfer Function block implements a transfer function. The transfer function is specified as an LTI model and may be multidimensional, continuous or discrete. The transfer function is typically created using the tf command, but any of the LTI model commands may be used. Figure 71 represents the dialog box of the System Transfer Function block.

System Zero-Pole

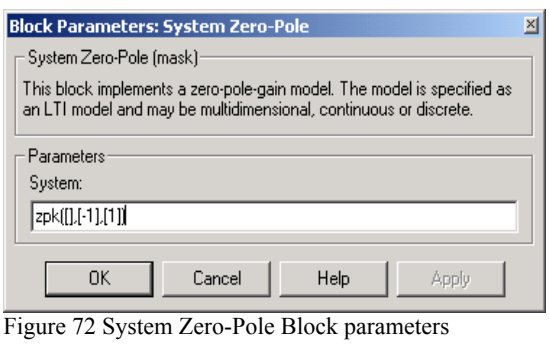


Figure 72 System Zero-Pole Block parameters

The System Zero-Pole block implements a zero-pole-gain model. The model is specified as an LTI model and may be multidimensional, continuous or discrete. The model is typically created using the zpk command, but any of the LTI model commands may be used. The dialog box of the System Zero-Pole block is shown in Figure 72.

System State-Space

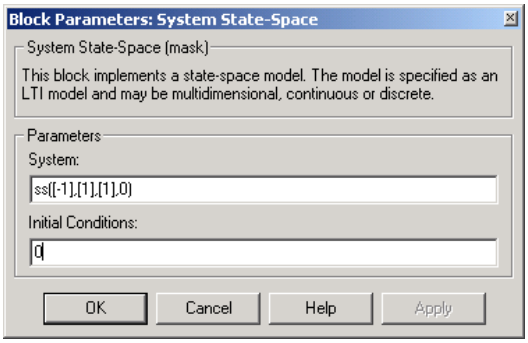


Figure 73 System State-Space Block parameters

The System State-Space block implements a state-space model. The model is specified as an LTI model and may be multidimensional, continuous or discrete. The model is typically created using the `ss` command, but any of the LTI model commands may be used. The dialog box of the System State-Space block is shown in Figure 73.

Double to PIC Float

The Double to PIC Float block converts from the IEEE-standard double precision floating-point format used by Simulink to the floating-point format used by Microchip for the PIC. Since the floating-point format for the PIC is a 32-bit format, an unsigned 32-bit integer is used to represent the value in Simulink. This block is typically used in conjunction with the Serial Output block to send floating-point values over a serial link to a PIC module.

PIC Float to Double

The PIC Float to Double block converts from the floating-point format used by Microchip for the PIC to the IEEE-standard double precision floating-point format used by Simulink. This block is typically used in conjunction with the Serial Input block to retrieve floating-point values over a serial link from a PIC module.

To Bytes

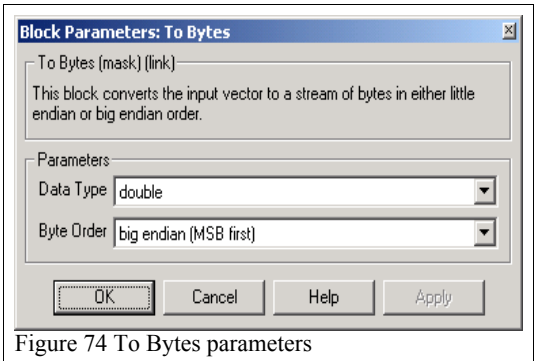


Figure 74 To Bytes parameters

The To Bytes block converts its input to a stream of bytes. The input can be any of the standard data types. For example, if a scalar double value is fed into the block, then 8 bytes will be output from the block. Whether the first byte output is the least-significant or most-significant byte is user-selectable. The To Bytes block actually takes two parameters: the input data type and the byte-ordering.

The dialog box of the To Bytes block is represented in Figure 74. The Data Type parameter is used to select the type of the input. (The type cannot be determined automatically from the input because Simulink resolves data types after it resolves port widths).

The Byte Order parameter selects whether the bytes are output in little endian or big endian order. For example, suppose the input is the single 32-bit integer 0x12345678 in hexadecimal. If the big endian byte order is chosen, then the output will be the vector [0x12, 0x34, 0x56, 0x78]. When the little endian byte order is selected, then the output is [0x78, 0x56, 0x34, 0x12].

From Bytes

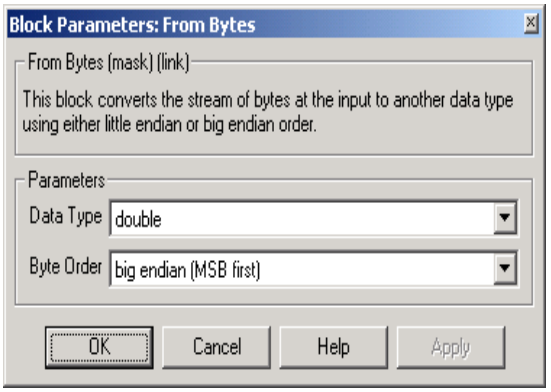


Figure 75 From Bytes parameters

The From Bytes block performs the reverse operation to the To Bytes block. It takes a vector of bytes and combines them into another data type. The output can be any of the standard data types. For example, if the double type is selected then when 8 bytes are fed to the input, a scalar double will be output from the block. Whether the first byte input is the least-significant or most-significant byte is user-selectable. The From Bytes block actually takes two parameters: the output data type and the byte-ordering.

The dialog box of the From Bytes block is depicted in Figure 75. The Data Type parameter is used to select the type of the output. The vector of bytes at the input must be a multiple of the data type size so that the From Bytes block has the right number of bytes to construct the data type at the output. Hence, when the double type is selected, the input must contain a multiple of 8 bytes.

The Byte Order parameter selects whether the bytes at the input are in little endian or big endian order. For example, suppose the type is set to uint32 and the input is the vector [0x12, 0x34, 0x56, 0x78] in hexadecimal. When the big endian byte order is selected, then the output is 0x12345678. If the little endian byte order is chosen, then the output is 0x78563412.

CRS ROBOTS

The Quanser Toolbox also interfaces to the two following CRS robots: the A465 and the Catalyst-5 (a.k.a. CRS 255). The Quanser library concerning those two CRS robots is shown in Figure 76. It provides kinematic and coordinate-convertion functions for the two robots in order for the user to implement an open architecture controller. Open architecture means that the controller directly sends voltages to control the motors.

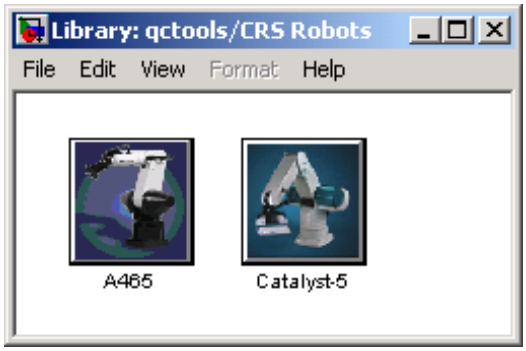


Figure 76 CRS Robots Libraries

CRS A465 Blocks

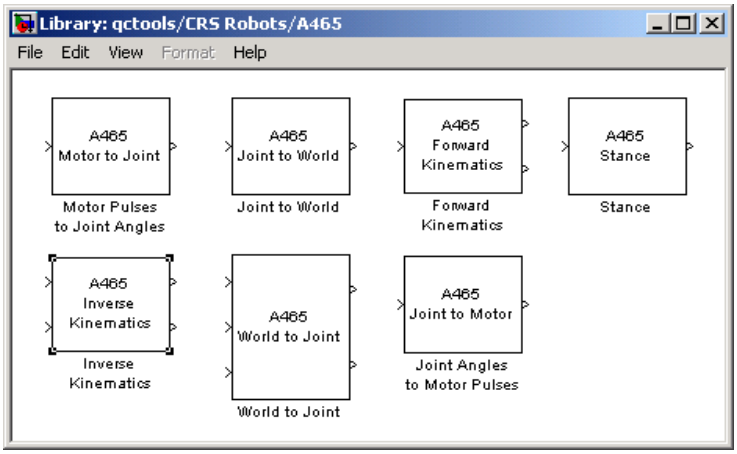


Figure 77 CRS A465 Blocks

The Quanser blocks interfacing to the CRS A465 are depicted in Figure 77. The CRS A465 is a 6-DOF articulated robot. It has 6 joints powered by 6 motors. Quanser provides Simulink blocks in order to seamlessly interface a user-defined open architecture controller to the CRS C500 controller provided with the robot.

A465 Motor Pulses to Joint Angles

The A465 Motor Pulses to Joint Angles block converts encoder counts read from all six CRS A465 joints into joint angles in radians.

The block inputs are the motor encoder pulses. The encoder counts read from the six A465 motors. The encoder counts should be supplied as a 6-vector in order of the joints i.e. base, shoulder, elbow, forearm roll, wrist pitch and wrist roll.

As an output, the A465 Motor Pulse to Joint Angles block produces the joint angles, in radians, for each of the joints. These joint angles in radians, corresponding to the given encoder counts, are contained in a 6-element vector. They are supplied in the same order as the input i.e. base, shoulder, elbow, forearm roll, wrist pitch and wrist roll.

The A465 Motor Pulses to Joint Angles block has no parameters.

A465 Joint to World

The A465 Joint to World block converts joint angles to world coordinates for the CRS A465 robot.

The block inputs are the joint angles in radians for the six A465 joints. The joint angles should be supplied as a 6-vector in order of the joints i.e. base, shoulder, elbow, forearm roll, wrist pitch and wrist roll.

The A465 Joint to World block produces the position and orientation of the robot end-effector as a 6-vector in the world coordinates. The vector elements represent the X, Y, and Z cartesian coordinates of the end-effector in millimeters, followed by the yaw, pitch and roll respectively, in radians.

The A465 Joint to World block has no parameters.

A465 Forward Kinematics

The A465 Forward Kinematics block converts joint angles to world coordinates for the CRS A465 robot.

As an input to the block, the joint angles should be supplied as a 6-vector in order of the joints i.e. base, shoulder, elbow, forearm roll, wrist pitch and wrist roll.

The A465 Forward Kinematics block produces the position and orientation of the robot end-effector as a 6-vector, as well as a stance signal that represents the current robot pose. The stance output is included because the robot can in general achieve the same position and orientation of the end-effector via eight different joint configurations. The stance output differentiates between these configurations and is useful as an input to the A465 Inverse Kinematics block. The world coordinates are supplied in a 6-vector containing the position and orientation of the robot end-effector. The vector elements represent the X, Y, and Z cartesian coordinates of the end-effector in millimeters, followed by the yaw, pitch, and roll respectively, in radians. The stance is supplied as a 3-vector representing the desired stance of the robot. The elements and their possible values are defined in Table 37.

The A465 Forward Kinematics block has no parameters.

Element	*0*	*1*	*-1*
1	Reaching Backward.	Reaching Forward.	Let Inverse Kinematics choose element configuration.
2	Elbow Down.	Elbow Up.	
3	Wrist Flipped.	Wrist Not Flipped.	

Table 37 Representation of the CRS A465 Stance

A465 Stance

The A465 Stance block outputs the stance corresponding to the given joint angles, for the CRS A465 robot.

The block input is a vector containing the joint angles in radians for the six A465 joints. The joint angles, supplied as a 6-vector, should be in order of the joints i.e. base, shoulder, elbow, forearm roll, wrist pitch and wrist roll.

The A465 Stance block produces a stance signal that represents the current robot pose. The stance output is useful because the robot can in general achieve the same position and orientation of the end-effector via eight different joint configurations. The stance output differentiates between these configurations and is useful as an input to the A465 Inverse Kinematics or A465 World to Joint blocks. Note that the A465 Forward Kinematics block also provides a stance output and is generally used instead of this block.

The stance of the robot is represented by a 3-element vector, whose elements and their possible values are defined in Table 37. The A465 Stance block has no parameters.

A465 Inverse Kinematics

The A465 Inverse Kinematics block converts world coordinates to joint angles for the CRS A465 robot. Since this mapping is not unique, additional information must be supplied to indicate which of the eight possible solutions (in general) is desired. The block uses the previous joint angles and the desired stance to arrive at a suitable configuration. Typically, the stance supplied by the A465 Forward Kinematics block is used for the stance input. The previous joint angles are maintained by the block itself. It uses the joint angle output from the previous iteration. Look under the block mask for more details.

A465 World to Joint

The A465 World to Joint block converts world coordinates to joint angles for the CRS A465 robot. Since this mapping is not unique, additional information must be supplied to indicate which of the eight possible solutions (in general) is desired. The block uses the previous joint angles and the desired stance to arrive at a suitable configuration. Typically, the stance supplied by the A465 Forward Kinematics block is used for the stance input. Note that the A465 Inverse Kinematics block is generally easier to use because it keeps track of the previous joint angles automatically.

A465 Joint Angles to Motor Pulses

The A465 Joint Angles to Motor Pulses block converts joint angles in radians to encoder counts for all six CRS A465 joints.

The block input is a 6-element vector containing the joint angles, in radians. The joint angle elements should be supplied, in the 6-vector, ordered in accordance with the robot joints i.e. base, shoulder, elbow, forearm roll, wrist pitch and wrist roll.

The A465 Joint Angles to Motor Pulses block produces the encoder counts for each of the motors from the given joint angles. The encoder counts are generated in a 6-vector with the elements in the same order as the input i.e. base, shoulder, elbow, forearm roll, wrist pitch and wrist roll.

The A465 Joint Angles to Motor Pulses block has no parameters.

CRS Catalyst-5 Blocks

The CRS Catalyst 5DOF (also known as CRS A255) is a 5-DOF articulated robot. It has 5 joints powered by 5 motors. The Quanser blocks interfacing to the CRS Catalyst-5 are depicted in Figure 78.

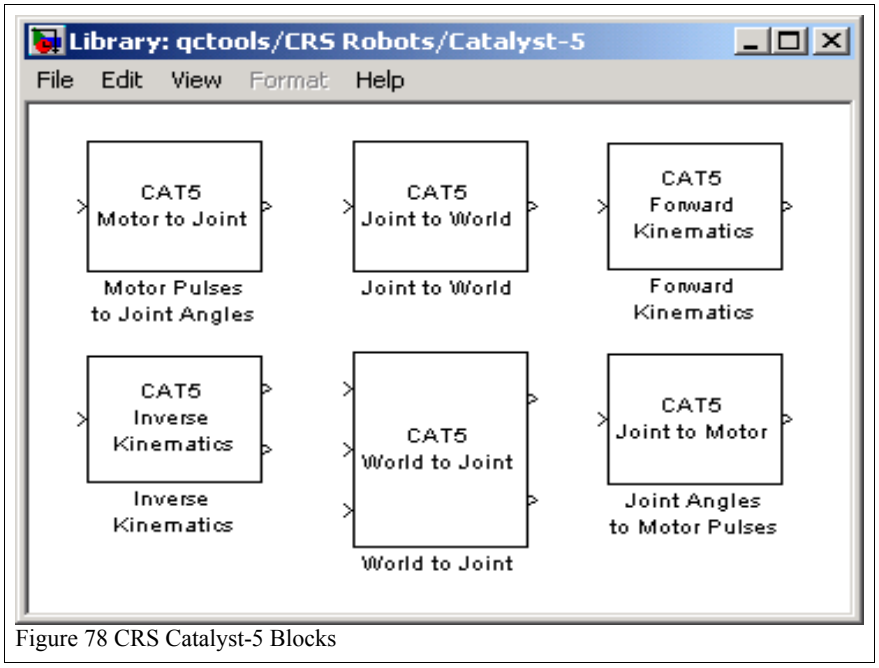


Figure 78 CRS Catalyst-5 Blocks

CAT5 Motor Pulses to Joint Angles

The CAT5 Motor Pulses to Joint Angles block converts encoder counts read from all five CRS Catalyst-5 joints into joint angles in radians.

CAT5 Joint to World

The CAT5 Joint to World block converts joint angles to world coordinates for the CRS Catalyst-5 robot.

CAT5 Forward Kinematics

The CAT5 Forward Kinematics block converts joint angles to world coordinates for the CRS Catalyst-5 robot. This block is functionally equivalent to the CAT5 Joint to World block.

CAT5 Inverse Kinematics

The CAT5 Inverse Kinematics block converts world coordinates to joint angles for the CRS Catalyst-5 robot. Since there are singularities in the workspace, the block uses the previous joint angles to arrive at a suitable configuration. The previous joint angles are maintained by the block itself. It uses the joint angle output from the previous iteration. Look under the block mask for more details.

CAT5 World to Joint

The CAT5 World to Joint block converts world coordinates to joint angles for the CRS Catalyst-5 robot. Since there are singularities in the workspace, the block uses the previous joint angles to arrive at a suitable configuration. Note that the CAT5 Inverse Kinematics block is generally easier to use because it keeps track of the previous joint angles automatically.

CAT5 Joint Angles to Motor Pulses

The CAT5 Joint Angles to Motor Pulses block converts joint angles in radians to encoder counts for all five CRS Catalyst-5 joints.

Interfacing to Other Hardware

WinCon supports the device driver blocks supplied in the Quanser Toolbox. The list of supported devices continues to expand. You can also write your own device driver blocks. Device driver blocks are written as S-functions. If you wish to make your code more efficient, you should also write Target Language Compiler files for your device driver blocks. However, this step is only necessary if you want to extract the maximum performance from your system. Refer to the documentation provided by The MathWorks with your MATLAB package for information on how to write device driver blocks and to implement inline code using the Target Language Compiler.

WinCon Scripting Commands in MATLAB

Quanser Consulting provides numerous MATLAB scripts (i.e. functions) which can be used to control WinCon from the MATLAB window and your own MATLAB programs. All commands have online help that you can read simply by typing help followed by the command name at the MATLAB prompt. For example, help wc_setoptions. You can list all the commands by typing help wincon at the MATLAB prompt.

These functions give you the maximum flexibility in controlling your experiments, including the ability to change parameters on the fly. You can construct a Simulink diagram within a MATLAB script, generate real-time code for WinCon, run the code in real-time, change gains while running, gather data, stop the real-time code, analyze the data, redesign the system and repeat the process. These are very useful for applications which require automated execution such as gain scheduling, data collection, adaptive learning control and parameter estimation.

wc_build

Formats:

```
wc_build
wc_build('model')
wc_build('model','path')
```

Parameters:

model	Name of the Simulink model for which to build real-time code.
path	Directory containing the Simulink model.

Description:

The wc_build command generates and compiles the real-time code for a Simulink model and automatically downloads it to the target machine. It also changes directory to the model directory, or the path specified, prior to building the code so that all intermediate files and the final real-time module reside in the same directory as the Simulink model. The default target is the local machine.

With no model specified, the currently open model is built. In the second form of the command, the model to build may be specified explicitly. The final form of the command also allows you to specify the path of the directory containing the model. The model path is only required if the model is not open and it is not in the MATLAB path (see the commands path and wc_path).

Examples:

wc_build	Builds the real-time code for the currently open Simulink model.
wc_build('test')	Builds the real-time code for model 'test.mdl'.

wc_clean

Formats:

```
wc_clean
wc_clean('model')
```

Parameters:

model	Name of the Simulink model for which to clean up intermediate files.

Description:

The wc_clean command may be used to delete all temporary files created during the building of real-time code. Note that it does not blindly delete source files or header files. It only deletes the source files and header files automatically generated by the Real-Time Workshop. It does delete all object files and other intermediate files, as well as generated real-time code (i.e. `*.wcl` file). wc_clean uses wc_path to get the path for the model.

If no model is specified, the command cleans up intermediate files for the model that is currently open. The wc_clean command also allows you to specify the model explicitly.

This command is useful under two circumstances. First, you may wish to archive your work and clear out all unnecessary files. The wc_clean command will do that for you. Second, there may be unusual circumstances where the code is not building cleanly because of old intermediate files left in the model directory. In this case, wc_clean may be used to ensure that no old files are left around.

Examples:

wc_clean	Clean up intermediate files for the currently open Simulink model.
wc_clean('test')	Clean up intermediate files for the files generated when building model 'test.wcl'.

wc_close

Format:

```
wc_close
```

Description:

The wc_close command closes the current WinCon project. The user is prompted to save the project if it has been modified.

wc_disconnect

Formats:

```
wc_disconnect
wc_disconnect('model')
wc_disconnect('model', 'path')
```

Parameters:

model	Name of the WinCon model to disconnect.
path	Directory containing the real-time model (see wc_path).

Description:

The wc_disconnect command disconnects a model from WinCon. If no model is specified then the current system is used. The model path is only required if the model is not open or if it is not in the MATLAB path (see the commands path and wc_path). This function is only useful in MATLAB Releases 11 or above. This function will do nothing in earlier versions. wc_disconnect does not assume that WinCon Server is running.

wc_download

Formats:

```
r = wc_download

r = wc_download('model')
r = wc_download('model','path')
```

Parameters:

model	Name of the Simulink model to download.
path	Directory containing the real-time module (see wc_path).

Return Values:

r	Boolean: 1 on success, 0 to indicate an error.

Description:

The wc_download command runs the WinCon Server automatically, if it is not already running, and downloads real-time code into WinCon. If no model is specified, the current system is used. The path need not be specified if the model is open or is in the MATLAB path. The command also allows you to specify the path of the directory containing the model. The model path is only required if the model is not open or if it is not in the MATLAB path (see the commands path and wc_path). If the download is successful, a value of 1 is returned. Otherwise a value of 0 is returned.

wc_examples

Formats:

```
r = wc_examples
wc_examples
```

Return Value:

r	String containing the full path to the WinCon examples directory.

Description:

The wc_examples command returns the full path to the WinCon examples directory. If a return argument is specified, the returned path is stored as a string into that parameter. If no return argument is specified, the wc_examples command change to the WinCon examples directory.

wc_isrunning

Formats:

```
r = wc_isrunning
r = wc_isrunning('model')
r = wc_isrunning('model', 'path')
```

Parameters:

model	Name of the WinCon model (.wcl).
path	Directory containing the real-time model (see wc_path).

Return Values:

r	Boolean: 1 if true, 0 if false.

Description:

The wc_isrunning command tests whether the real-time code is currently running. If no model is specified then the current system is used. The model path is only required if the model is not open or if it is not in the MATLAB path (see the commands path and wc_path). wc_isrunning returns 1 if the model is running, 0 otherwise.

wc_newplot

Formats:

```
wc_newplot
wc_newplot('model')
```

Parameters:

model	Name of the Simulink model.

Description:

The wc_newplot command opens a new plot in WinCon. If no model is specified, wc_newplot uses the current system. If no type is specified, the user will be presented with a list of the different plot types to choose from (e.g. Scope, X-Y Graph, Digital Meter, Thermometer). wc_newplot runs WinCon Server automatically, if it is not already running.

wc_open

Format:

```
r = wc_open('project')
```

Parameter:

project	WinCon project filename.

Return Value:

r	Boolean: 1 on success, 0 to indicate an error.

Description:

The wc_open command opens the specified WinCon project. wc_open runs WinCon Server automatically, if it is not already running. wc_open returns 1 on success, 0 otherwise.

Example:

wc_open('q_p')	Opens the WinCon project 'q_p.wcp'.

wc_openplot

Formats:

```
wc_openplot
wc_openplot('model')
wc_openplot('model', 'plot')
```

Parameters:

model	Name of the Simulink model.
plot	Name of plot to open.

Description:

The wc_openplot command opens in WinCon a Simulink display (e.g. Scope, X-Y Graph, Display) defined in the given model. If no model is specified, wc_openplot uses the current system. If no plot parameter is specified, the user is presented with a list of displays, as defined in the model, to choose from. wc_openplot runs WinCon Server automatically, if it is not already running.

Example:

wc_openPlot('q_p', 'q_p\Angle')	opens in WinCon the display called 'Angle' in the model 'q_p.mdl'.

wc_path

Formats:

```
p = wc_path
p = wc_path('model')
```

Parameters:

model	Name of the Simulink model for which to return the path.

Return Values:

p	String containing the full path name to the directory where the model is.

Description:

The wc_path command is used internally by many of the other WinCon scripts. It returns the full path to the directory in which the given model is stored. It is also used by the code generation modules to ensure that the generated code is always produced in the same directory as the Simulink model – even if the current directory in MATLAB is different!

However, you may find this command useful if you wish to write your own MATLAB scripts that need to know where the Simulink model is stored. A primary example might be a script that automatically runs through a series of tests, modifying the controller gains each time and storing data to the same directory as the model, or to a subdirectory.

With no model specified, the command returns the path to the currently open model. The command also allows the model to be specified explicitly. If the model is not open, then it must be in the MATLAB path (see the MATLAB path command). Additionnally, if the model has yet to be saved then an empty string is returned. Use the isempty command to test for an empty matrix.

Examples:

p = wc_path	Returns the path to the currently open model in the variable p.
p = wc_path('test')	Returns the path to the Simulink model called 'test.mdl' in the variable p.

wc_refresh

Formats:

```
r = wc_refresh
```

Return Values:

r	Boolean: 1 on successful refresh, 0 to indicate an error.

Description:

The wc_refresh command refreshes the state of Simulink so that its state is updated correctly during the execution of a MATLAB script. wc_refresh returns 1 if Simulink is refreshed or 0 if Simulink could not be refreshed.

wc_run

Formats:

```
r = wc_run
```

Return Values:

r	Boolean: 1 on success, 0 to indicate an error.

Description:

The wc_run command runs the WinCon Server. If it is already running, it brings it to the

foreground. wc_run returns 1 if WinCon had not been running or 0 if WinCon was already running or failed to run.

wc_save

Formats:

```
wc_save
wc_save('file')
```

Parameter:

file	Filename to save the WinCon project to.

Description:

The wc_save command saves the current WinCon project. If a filename is specified then the project is saved to the specified file. If no extension is specified, then an appropriate extension (i.e. wcp) is added.

Example:

wc_save('q_p')	Saves the current WinCon project (e.g. model and associated plots and control panels) to the project file named 'q_p.wcp'.

wc_saveplot

Formats:

```
wc_saveplot('plot title', 'filename')
```

Parameters:

plot title	Name of the WinCon plot window.
filename	Name of the MAT file to save data to.

Description:

The wc_saveplot command saves the WinCon plot data to a MAT file. wc_saveplot looks for the plot with the given title and tells it to save its data to the given file in MAT format. Note that the **.mat** extension **MUST** be specified in the filename. It is not added automatically. The WinCon plot name is shown in the taskbar of the plot.

Also note that when you save data from a plot, WinCon switches it to fixed mode temporarily in order to collect the data from the Server without any decimation **It is important to wait until all the data has been collected before you start the controller again.** See the script example q_p_script.m, in Section ,that ensures that the data has been saved before you start a second run.

As a reminder, any data saved in a MAT file can then be loaded into the MATLAB workspace by using the load command.

Example:

wc_savePlot('Scope – q_p\Angle', 'scope.mat')	Saves the data displayed in 'Scope – q_p\Angle' to the file 'scope.mat'.

wc_select

Formats:

```
wc_select
wc_select('model')
wc_select ('model', 'path')
```

Parameters:

model	Name of the Simulink model to select.
path	Directory containing the real-time model (see wc_path).

Description:

The wc_select command selects the given model as the active WinCon model and loads the corresponding real-time code into WinCon Client. If no model is specified then the current system is used. The model path is only required if the model is not open or if it is not in the MATLAB path (see the commands path and wc_path). wc_select will run WinCon Server automatically, if it is not already running.

wc_setoptions

Formats:

```
wc_setoptions
wc_setoptions('model')
wc_setoptions('model' , 'os_opt')
```

Parameters:

model	Name of the Simulink model for which to set the options. May also be 0.
os_opt	Name of the operating system. Valid values are: 'win', 'dos' or 'nt'.

Description:

wc_setoptions sets the build options for a model in order to generate real-time code for WinCon.

In its first form of the command, when no option is specified, the wc_setoptions command sets the default options for new Simulink models created during that MATLAB session. wc_setoptions is automatically executed during MATLAB startup (i.e. called in matlabrc.m) so that the default Real-Time Workshop options are configured for WinCon whenever a new Simulink model is created. That call also makes WinCon the default for real-time code generation.
The second form of the command, with the name of a Simulink model as an argument, changes the settings for an existing Simulink model to those required for that machine operating system. **When a Simulink model is specified, it must be open** at the time the wc_setoptions command is invoked. If the model is set to the number zero in this case (not a string), then the wc_setoptions command sets the defaults for all new Simulink models created during the current MATLAB session. If the name of the operating system is not specified, a default suitable for the user's machine is assumed (set during installation). The command will not overwrite the base sampling period if you have already specified one. If the integration method is variable step then wc_setoptions changes the integration method to an "equivalent" fixed step integration method. Otherwise it leaves the integration method unchanged.
The final form of the command allows you to specify a different operating system.

Examples:

wc_setoptions('test', 'win')	set the options for the 'test.mdl' diagram to build real-time code for a Windows 98 WinCon Client.
wc_setoptions(0, 'nt')	set the default options to build real-time code for a Windows NT/2000/XP WinCon Client.

wc_start

Formats:

```
wc_start
wc_start('model')
wc_start('model','path')
```

Parameters:

model	Name of the model to start.
path	Directory containing the real-time module (see wc_path).

Description:
The wc_start command launches the WinCon Server, if it is not already running, and starts the real-time code for the model. If no model is specified then the current system is started.

The model path is only required if the model is not open or if it is not in the MATLAB path (see the commands path and wc_path). The model code can then be stopped using the wc_stop command.

wc_stop

Formats:

```
wc_stop
wc_stop('model')
wc_stop('model', 'path')
```

Parameters:

model	Name of the Simulink model to stop.
path	Directory containing the real-time module (see wc_path).

Description:

The wc_stop command stops the real-time code running under the WinCon Client. If no model is specified then the current system is stopped. The model path is only required if the model is not open or if it is not in the MATLAB path (see the commands path and wc_path). wc_stop assumes WinCon Server is running. It will not issue an error if the model has already stopped running. In fact, if a script exits because wc_isrunning returns false, then wc_stop should still be called to update the Simulink diagram status.

wc_update

Formats:

```
wc_update
wc_update('model')
wc_update('model', 'path')
```

Parameters:

model	Name of the model to update.
path	Directory containing the real-time model (see wc_path).

Description:

The wc_update command updates workspace variables in the real-time code. wc_update runs WinCon Server automatically, if it is not already running. If no model is specified then the current system is used. The model path is only required if the model is not open or if it is not in the MATLAB path (see the commands path and wc_path).

Model Examples

Please read the following section on accessing the supplied WinCon examples before proceeding to try any of the subsequent example models.

WinCon Demonstrations Window

WinCon comes with sample Simulink-based models to help you begin using WinCon. The model examples can be accessed by typing:

```
wcdemos
```

at the MATLAB prompt. This command opens the *WinCon Demonstrations Window*, as pictured in Figure 8.

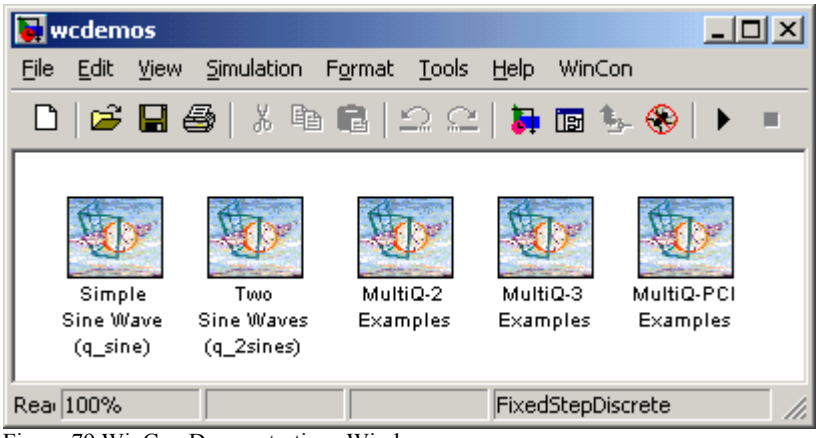


Figure 79 WinCon Demonstrations Window

The WinCon Demonstrations are example Simulink-based diagrams which illustrate how to use WinCon. Some of the examples are hardware-specific and require the presence of a particular data acquisition card or a specific Quanser experiment. The same examples are provided for each data acquisition card. Other examples do not require any hardware.

Figure 80 is obtained by double-clicking on the *MultiQ-PCI Examples* block of the *WinCon Demonstration* window illustrated in Figure 79. It displays the library of demonstrations available for the Quanser MultiQ-PCI board. If you are using another type of Quanser's data acquisition card (e.g. MultiQ or MultiQ-3), then open the subsystem corresponding to your data acquisition card to acess the associated examples.

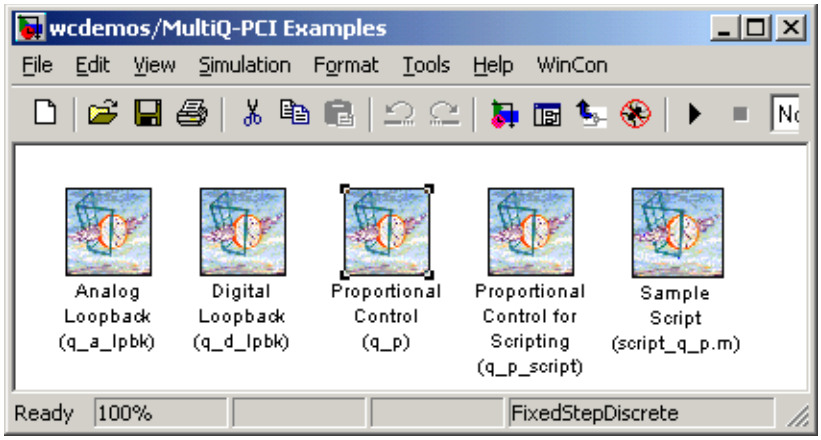


Figure 80 Library of Demonstrations for Quanser's MultiQ-PCI board

NOTE: If you are not using a MultiQ-series board, ensure that you have the data acquisition blocks required to run the desired example and possibly a timer block for your particular board. You will need to replace any of the MultiQ-series blocks with the equivalent blocks appropriate for your data acquisition card. See the Real-Time Workshop manuals that came with your MATLAB package for details on writing your own drivers for your board(s), or contact Quanser.

Also note that the Simulink files corresponding to the WinCon Demonstration window are stored in the directory returned by the function wc_examples, such as C:\MATLAB6p1\work\Examples. Hardware-specific examples are stored in subdirectories of the Examples directory. Table 38 lists the different subdirectories and what data acquisition card is required by the examples in that directory.

Subdirectory	*Data Acquisition Card*
MULTIQ	Quanser's MultiQ-2 data acquisition card (supplanted by MultiQ-3).
MULTIQ-3	Quanser's MultiQ-3 data acquisition card.
MULTIQ-PCI	Quanser's MultiQ-PCI data acquisition card.

Table 38 Hardware-Specific Example Directories

Analog Loopback Example: q_a_lpbk.mdl

The Analog Loopback model, q_a_lpbk.mdl, is an example that everyone should try. Not only does it introduce you to the concepts of WinCon and interaction with external hardware, but it also tests that your data acquisition board, together with its A/D and D/A channels, has been configured correctly.

The loopback model (or "controller") generates a sine wave and outputs the sine wave to analog output channel 0. It reads whatever voltage arrives at analog input channel 0 and writes the data to a Scope so that the result can plotted. Follow the subsequent steps to successfully execute the Analog Loopback example.

1. Before using the analog loopback example, you need to **wire** analog output channel 0, i.e. D/A #0, on your data acquisition card to analog input channel 0, i.e. A/D #0. This configuration is illustrated in Figure 81 for the MultiQ-PCI card. No other hardware is required for this example.

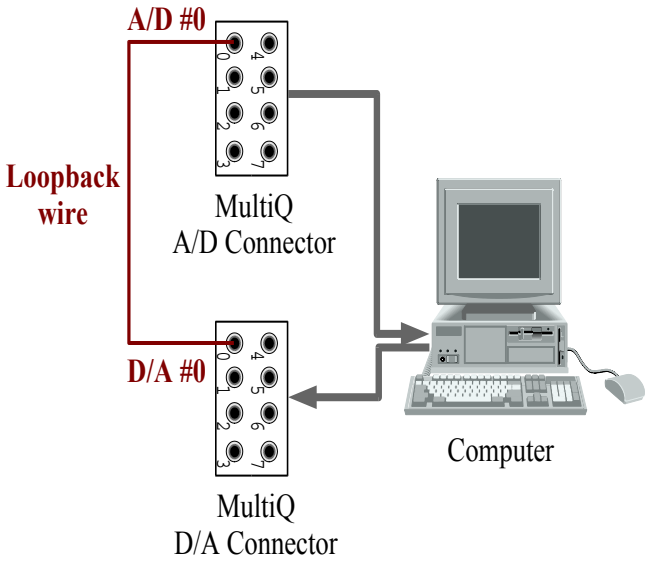


Figure 81 Loopback Wiring Configuration

2. To **open the model**, follow the instructions given in Section WinCon Demonstrations Window and double-click on the *Analog Loopback (q_a_lpbk)* block that corresponds to your data acquisition board. For the MultiQ-PCI, the model window depicted in Figure 82 will open. Note that you could have created this model yourself using Simulink and the Quanser Toolbox, as described in Interfacing to Hardware: The Quanser Toolbox on page 95.

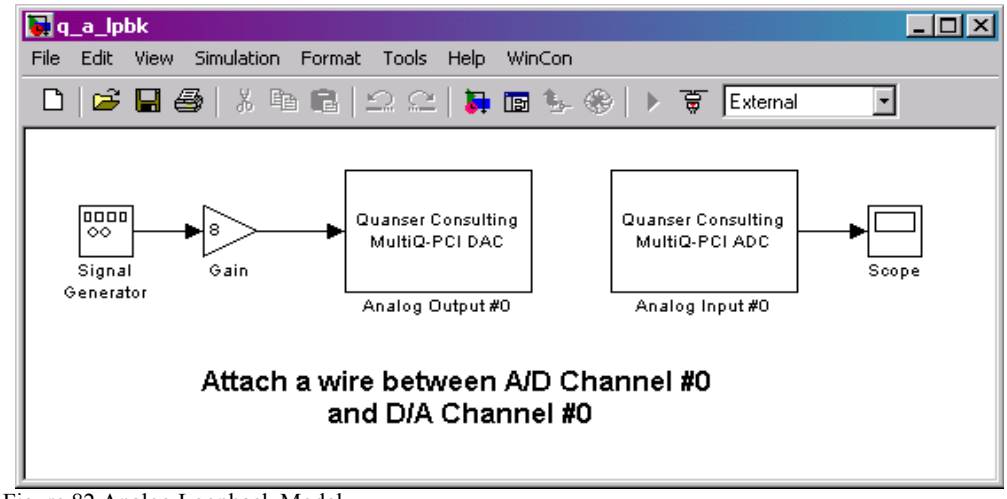


Figure 82 Analog Loopback Model

3. You must first ensure that all your options are set correctly according to the operating system under which the real-time code is going to run. The WinCon Server installer automatically configures the examples so this step is not strictly necessary. (Simulink is also configured to such that the default options are appropriate for WinCon). However, this step is required for pre-existing models which may not be configured for WinCon. To configure the model for WinCon, select *WinCon* | *Set WinCon Options* from the Simulink menu bar, as seen in Figure 83 below.

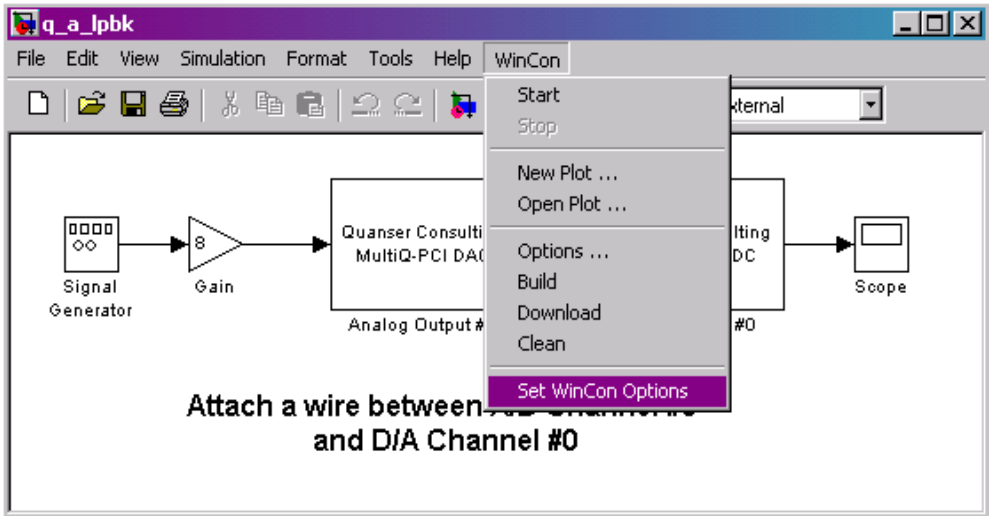


Figure 83 Setting the default WinCon options

Note: If you are running the client on a different PC, you must start the WinCon Server and Connect to the remote WinCon Client before building or downloading the model.

4. You are now ready to **Build** the system. Select *WinCon* | *Build* from the Simulink menu bar, as shown below:

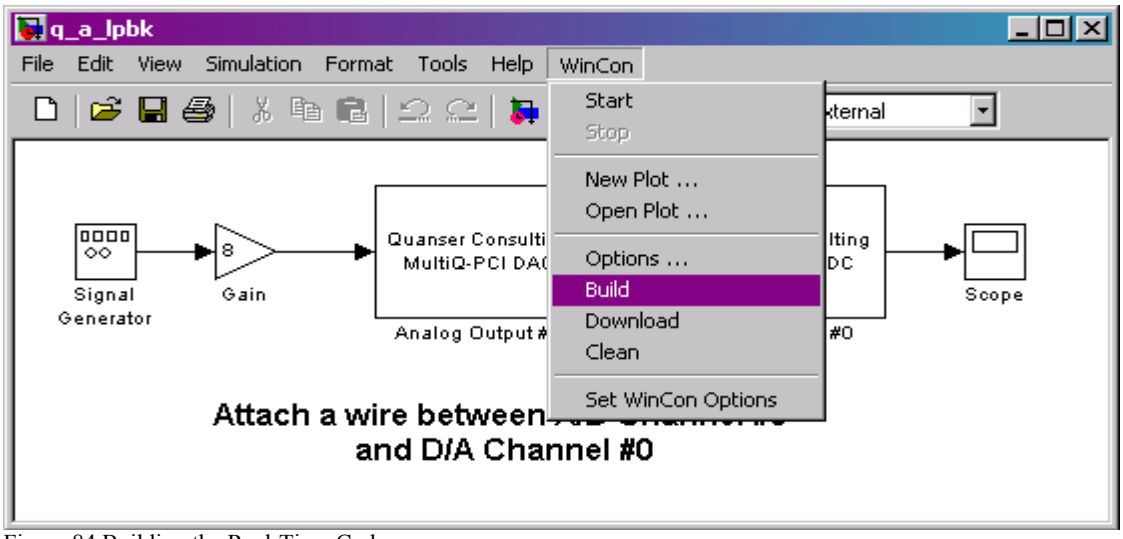


Figure 84 Building the Real-Time Code

This step generates the real-time code for the diagram. **Wait until the compilation is complete.** The MATLAB window displays the progress of the code generation task and when it is complete, the following message appears:

Successful completion of Real-Time Workshop build procedure for model: q_a_lpbk

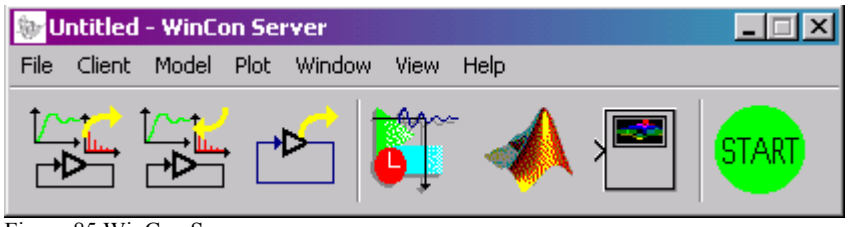


Figure 85 WinCon Server

Following the code generation, WinCon Server and WinCon Client are automatically started. The generated code is automatically downloaded to the Client and the system is ready to run.

5. Click on the **Start** button in the WinCon Server toolbar (shown Figure 85). This action starts the controller, which runs at a sampling frequency of 200 Hz, as set in the Simulink model. Click on the **Open Plot** button in WinCon Server. The names of all displays in the Simulink model diagram appear in a Multiple Select Variable Tree. You can then select the displays you would like to plot. In this case, select **Scope**. You should now see a clipped sinosoid as shown in Figure 86.

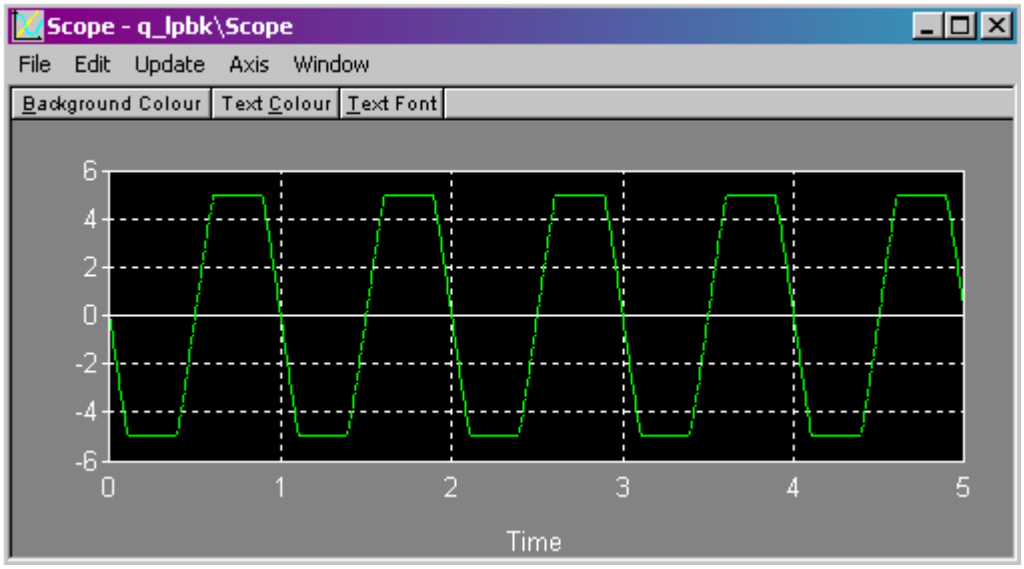


Figure 86 Clipped Sinosoid

The sine wave has been limited by the A/D input channel, which only has a range of ±5 volts.

6. Click **Go To Simulink** from the WinCon Server toolbar. Double-Click on the **Gain** block and change it to 4. Note the changes in your Scope plot. This shows that the output of the D/A is being measured by the A/D block. You should also note that you were able to change the gain in **real-time!**

7. Now **Stop** the controller via the WinCon Server toolbar. Go back to your Simulink model and select *WinCon* | *Options...* from the menu bar. Under the *Solver* tab, change the *Fixed step size* from *0.005* to *0.05* (thus changing the sampling frequency from 200Hz to 20Hz). Re-run the controller from WinCon Server. You will get an error message telling you that the model needs to be rebuilt. When you change the base sampling period it affects all discrete time blocks in your system. Hence, Simulink insists that you rebuild the real-time code when the sampling period is changed.

   Rebuild the real-time code. **Note that the code builds much faster** because only one file has to be recompiled – the source code generated from your model.

   Now you can run the controller again. Notice the difference in the plotted sine wave. Since the sampling frequency has been lowered, the signal is more discretized than the previous one. **Stop** the controller. You should now be familiar with setting the sampling period for a model. Go back to the *WinCon* | *Options...*, and reset the step size to 0.005 (i.e. 200Hz). Re-build your controller and plot the Scope once again.

8. In the **Scope** window, select *File* | *Save* | *Save As M File*. You will be presented with a dialog box of where to save the file. Choose a desired location and save the M-file. The

data from the WinCon Scope buffer will be output to a MATLAB M-file. In WinCon Server, click on the **Go to MATLAB** icon. In MATLAB, locate the M-file you just created, and run it. You should see a plot of your scope that looks similar to Figure 87.

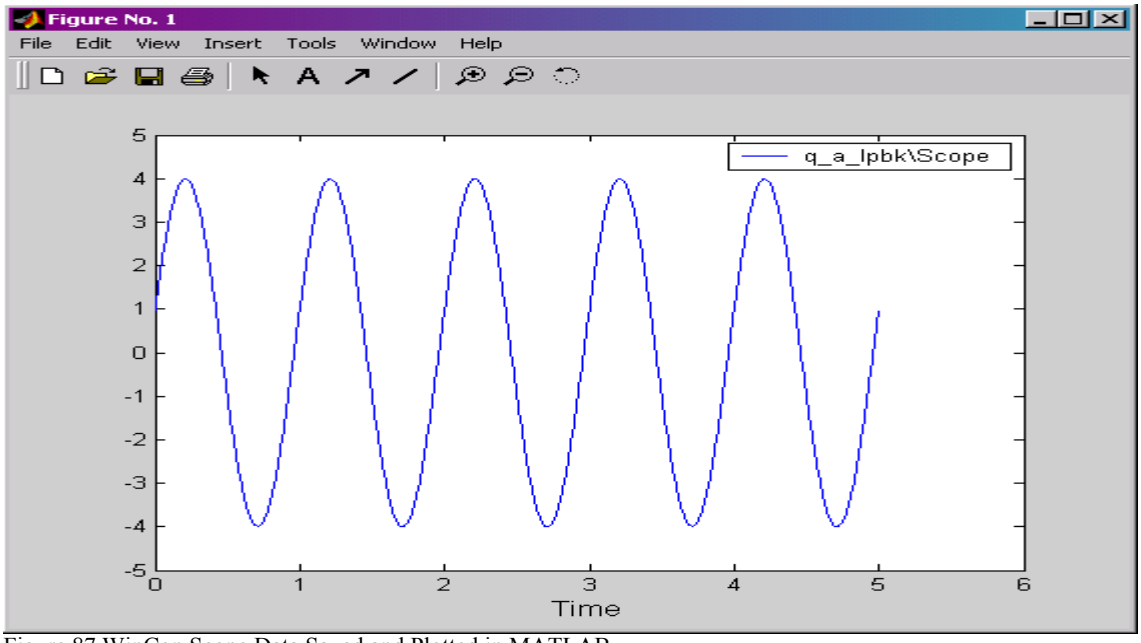


Figure 87 WinCon Scope Data Saved and Plotted in MATLAB

You can now **Stop** the controller if you have not done so already. This brings to an end the Analog Loopback Example. From this example, you should have an understanding of the following:

- How to **build** a Simulink model and **download** it to WinCon Client.
- How to **plot** model variables in WinCon Scope.
- How to **save** Wincon Scope data for later use in the MATLAB environment.

Proportional Control Example: q_p.mdl

The proportional control example illustrates the setting of control loop parameters through the MATLAB or Simulink environments, or directly from WinCon. You will be introduced to WinCon Control Panels and WinCon Projects as well.

To run this example, the following hardware is required:

- A Quanser UPM 2405 /1503 Power Module or equivalent.
- A Quanser MultiQ PCI/MQ3 or equivalent.

- A Quanser SRV02-E servo plant.

1. Before using this example, wire the system as follows:
   - Connect a cable of **gain 5** (should be labelled with a #5) from the UPM **To Load** connector to **1 – Motor Drive** of the SRV02-E as labelled in Figure 88 below.
   - Connect a cable from **2 – Encoder Angle** (as shown below in Figure 88) of the SRV02-E directly to encoder channel 0 of your DAC card.
   - Connect a cable from your DAC analog output channel 0 to the UPM **From D/A** connector.

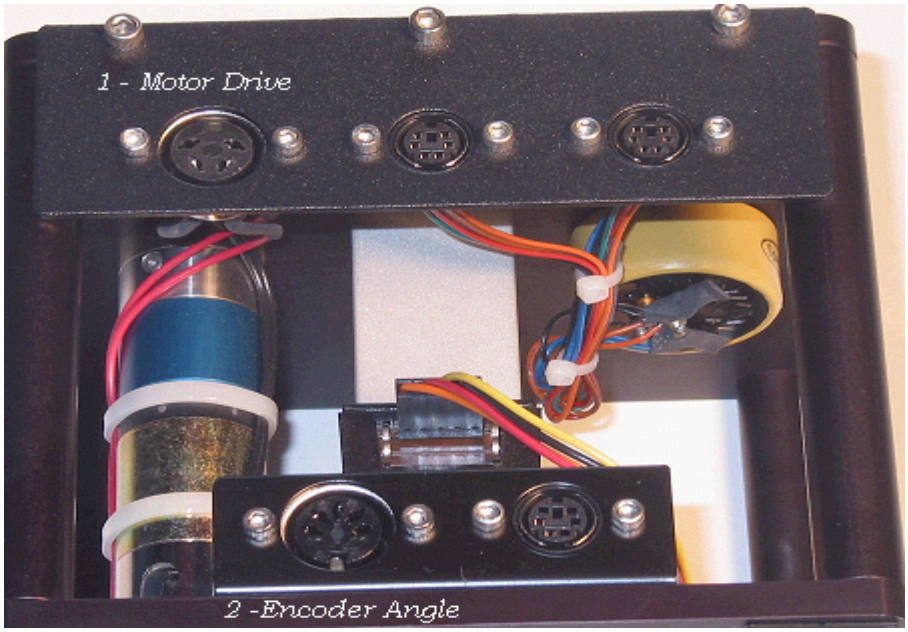


Figure 88 SRV02-E Connections

2. To open the model, follow the instructions in section WinCon Demonstrations Window, and open the model called *q_p.mdl*. Remember to choose the model that is meant for your particular board. This model incorporates a simple proportional controller as part of the closed-loop system. The purpose of this controller is to track a desired angle.

   You should notice the proportional gain block does not have a numeric value but uses a variable **Kp** instead. This feature allows the user to enter data directly from the MATLAB environment. At this point, you should go to the MATLAB command window and assign **Kp** an initial value of 0.15.

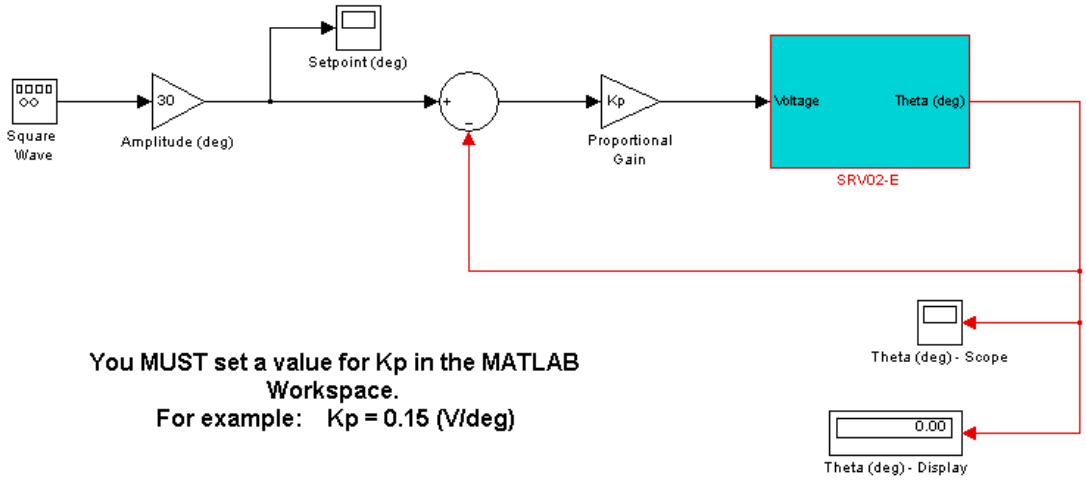


Figure 89 Proportional Controller Model in Simulink

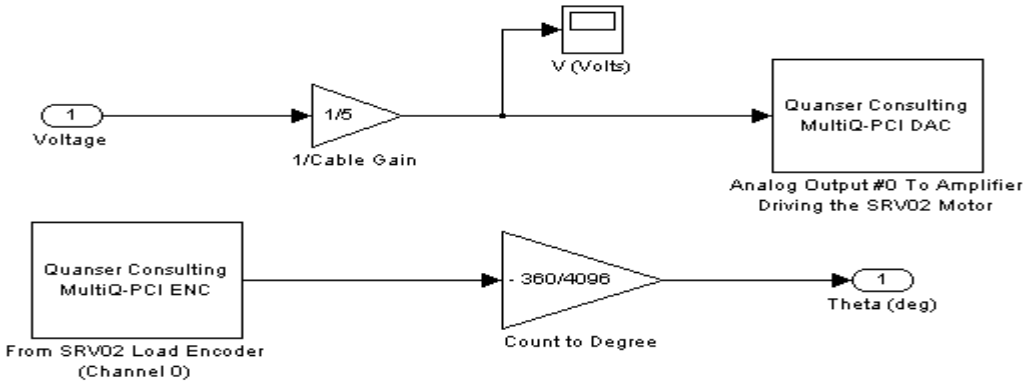


Figure 90 SRV02-E Sub-block

3. You can now **Build** the controller and start it using the WinCon Server. Open a WinCon Scope and select **Theta (deg) – Scope**. You can also go ahead and open a WinCon Digital Meter by selecting **Theta (deg) – Display** to obtain a digital read-out of the output angle. To fully gauge the performance of a controller, it is important to see the output plotted along with the desired angle. In your Scope plot window, select *File* | *Variables...* and select **Setpoint (deg)**. You should now be seeing two plots simultaneously. This plot gives provides a good indication of how your controlller is performing. You should have also noticed that you could have chosen any block to plot under the Variables menu. This flexibility comes in handy when attempting to diagnose different sections of your overall signal path.

4. With the controller still running, return to your MATLAB command window and change the value of **Kp** to 0.25, by typing `Kp = 0.25;` at the prompt. Return to your Simulink model and select *Edit* | *Update Diagram (Ctrl+D)*. This action updates all the MATLAB variables currently in your Simulink model. You should now see a different response on your Scope display. The transition of the response as Kp is varied from 0.15 to 0.25 can be seen in Figure 94 below.

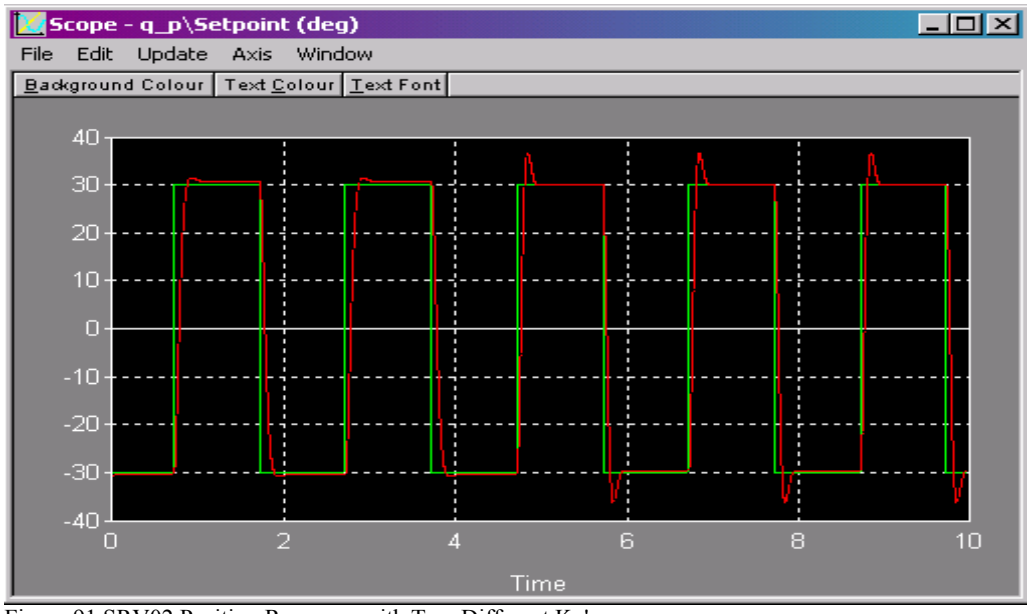


Figure 91 SRV02 Position Response with Two Different Kp's

5. With most controllers it is nescessary to ***tune*** the parameters to achieve the desired response. WinCon facilitates this need by allowing parameters to be changed via the Simulink diagram, or through WinCon Control Panels. From your WinCon Server, select *Window* | *Control Panel*. A new control panel window will appear. From the menu, select *Control* | *Insert Control*. For this application, a knob control will suffice. You will now be prompted to associate a parameter with the knob control. Select *Proportional Gain* | *Gain (1,1)*. This control is now associated with the proportional gain block in your model. From the menu, select *Control* | *Properties*, and set the maximum to a value of 0.5 while leaving the minimum at 0. Finally, de-select *Window* | *Design Mode*. Your window should now look similar to Figure 92 below:

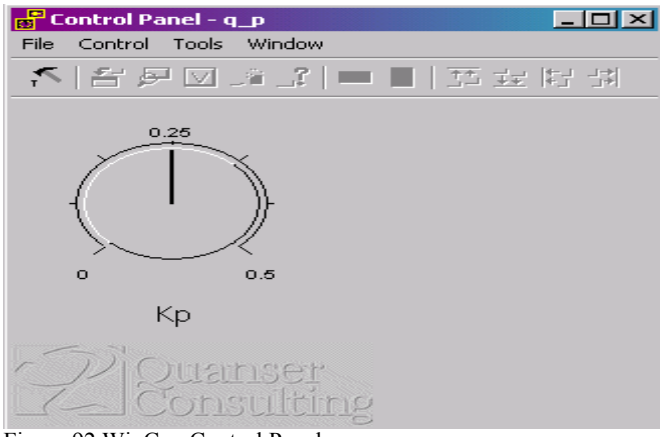


Figure 92 WinCon Control Panel

While adjusting the position of the knob, notice the change in your response (Scope-Plot). This feature gives the user a tool to fine-tune the controller. Take note how the response reacts in **real-time** to changes in the knob setting. For more information about WinCon Control Panels, refer to section WinCon Control Panel on page 84.

6. After finding a setting that achieves the desired response, return to the Simulink model and change the desired angle from 30 to 45 degrees (*Amplitude (deg)*). Take note how the controller responds to these changes in the desired angle. You can now stop the controller.

7. From your WinCon Server window, select *File* | *Save*. Choose a name and location of your WinCon Project. After saving your project, exit WinCon Server. Return to the MATLAB command window and type exit**.** MATLAB and Simulink should no longer be running. Locate your saved WinCon Project (i.e. .wcp file), and double-click to open it. You should see your Scope as well as the knob control windows pop-up. From WinCon Server, start your controller again. You should now be seeing your response through the scope and your knob will still control the proportional gain. With WinCon Projects, you can re-open the controller with all the displays and controls without requiring MATLAB or Simulink.

The Proportional Control example illustrated the following points:

- **Setting model parameters on-the-fly** through the MATLAB or Simulink environments as well as directly from WinCon, via its Control Panels.
- Using **WinCon Controls** to set model parameters.
- Using **WinCon Projects** to save and restore WinCon Server's Client connections, displays, and Control Panels. WinCon Projects also allow you to operate without MATLAB and Simulink.

Script Example: script_q_p.m

This example is used to illustrate the scripting functionality of WinCon. As you will see in this example, scripts are very useful when you would like to systematically increment a variable and record its response. This script varies the gain associated with a proportional control system similar to the one used in the previous example. The proportional control system for this example is depicted in Figure 93 below. Refer to the previous example described in section “Proportional Control Example: q_p.mdl“ for instructions on how to connect and wire the system.

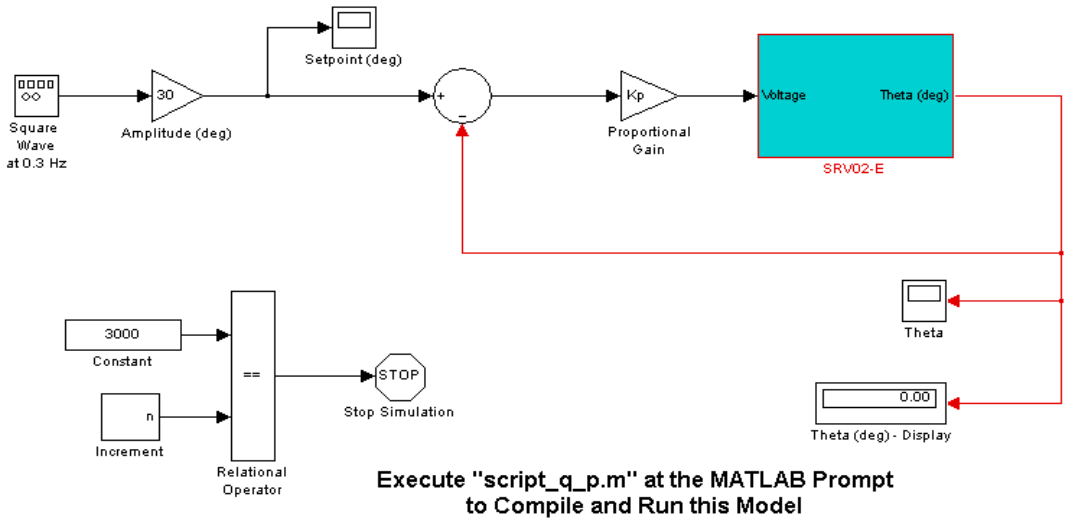


Figure 93 Proportional Controller Model used by the WinCon Script

The following script, complete with comments, demonstrates the WinCon functions available in MATLAB. For more information about writing a script, please refer to chapter WinCon Scripting Commands in MATLAB on page 139. The script below initializes Kp to 0.05, and increments it by 0.1 up to a final value of 0.45, while capturing the response for each iteration.

```
% SCRIPT_Q_P

%
% This WinCon script in Matlab collects the actual position response
% of the Quanser SRV02-E plant for increasing proportional gains Kp
% changing from 0.05 V/deg to 0.45 V/deg in increments of 0.1.
%
% Copyright (C) 2002 Quanser Consulting Inc.
% Quanser Consulting Inc.

clear all
% sets model name and path
model = 'q_p_script';
path = wc_path( model );
```

```
% set the initial proportional gain Kp
Kp = 0.05;
% set the proportional gain increment between run
Kp_incr = 0.1;

% clean compilation
wc_clean( model )
wc_setoptions
wc_build( model, path )

% open the "Theta" Scope because we will be saving the data it collects
plot_name = 'Theta';
wc_openplot( model, [model '\' plot_name] );
% also open a Digital Meter for the user to visualize the Theta angle
wc_openplot( model, [model '\Theta (deg) - Display'] );

% during the desired number of runs ("N_runs"),
% gradually increase the gain Kp,
% and collect response data each time.
N_runs = 5;
for I = 1 : N_runs
   fprintf( 'Run # %d:\n', I );
   fprintf( '\tProportional gain: Kp = %.2f V/deg\n', Kp );

   % Run the controller. Wait for it to stop.
   wc_start( model, path );
   fprintf( '\tcontroller running' )
   while wc_isrunning( model )
      fprintf( '.' );   % display progress dots
      pause( 0.5 );
   end;
   wc_stop( model, path );          % always call wc_stop to clean up properly

   % Save the data that was collected. Delete the previous file, if any, so
   % that we don't get any "File exists. Do you want to replace it?" messages.
   fname = ['res_' int2str(I) '.mat'];
   if exist( fname, 'file' )
      delete( fname ); % to make sure that it does not exist before we save it
   end;
   wc_saveplot( ['Scope - ' model '\' plot_name], fname );

   % Wait for the data to be saved.
   fprintf( '\tsaving data to ''%s'': ', fname );
   done = 0;
   while( done == 0 )
      pause( 0.5 );
      if( exist( fname,'file' ) == 2 )
       done = 1;
     end
   end
   fprintf( ' data saved.\n' )

   % Increment the proportional gain by Kp_incr
   Kp = Kp + Kp_incr;
end
% Disconnect from the real-time code
wc_disconnect( model, path );

% Load the data and plot a comparison in a MATLAB Figure
% Offset each curve of (5*I) (deg).
```

```
for I = 1 : N_runs
   fname = ['res_' int2str(I) '.mat'];
   load( fname );
   nn = size( q_p_script_Theta );
   for J = 1 : nn(1),
      out( J, I ) = q_p_script_Theta( J ) + 5 * I;
   end
end
plot( out )
title( 'Responses for Different Proportional Gains' )
xlabel( 'Sample #' )
ylabel( 'Theta [deg]')
grid on
```

Using the above script, the Matlab program starts by initializing the gain **Kp** and executing the command **wc_start,** which starts the controller. The script then waits for the termination of the controller by executing the wc_isrunning function. The controller terminates at sample number 3000 using a Stop Simulation block in the diagram. When the controller stops, a filename is created (**res_n.mat, n = 1..5**) and the contents of the plot named "**Scope - q_p_Angle**" are saved to the file.

The script then increments Kp and runs the controller again with a new value of Kp. **Note that wc_stop MUST be called for proper operation**, even though the controller has already stopped (it disconnects). This process is performed 5 times and at the end of the cycle, the files that were created are loaded into the MATLAB workspace and the plots shown below are generated. In this manner, the user can evaluate system response for different values of Kp. Note that the ***buffer length of the real-time plot should be longer than the duration of the run***. This way you ensure that you do not end up collecting only part of the data from the run.

To run the example, type *wc_examples in* the MATLAB command window. At this point you must change to the directory that matches your DAC card. If you have the MultiQ-PCI, for example, type *cd MultiQ-PCI* to change to the proper directory. Finally, type *script_q_p* to run the script. Wait for the script to fully execute. While executing, you will see a digital display showing you the current position of the plant. After 5 iterations, when the script is completed, you will be presented with a plot similar to Figure 84 below.

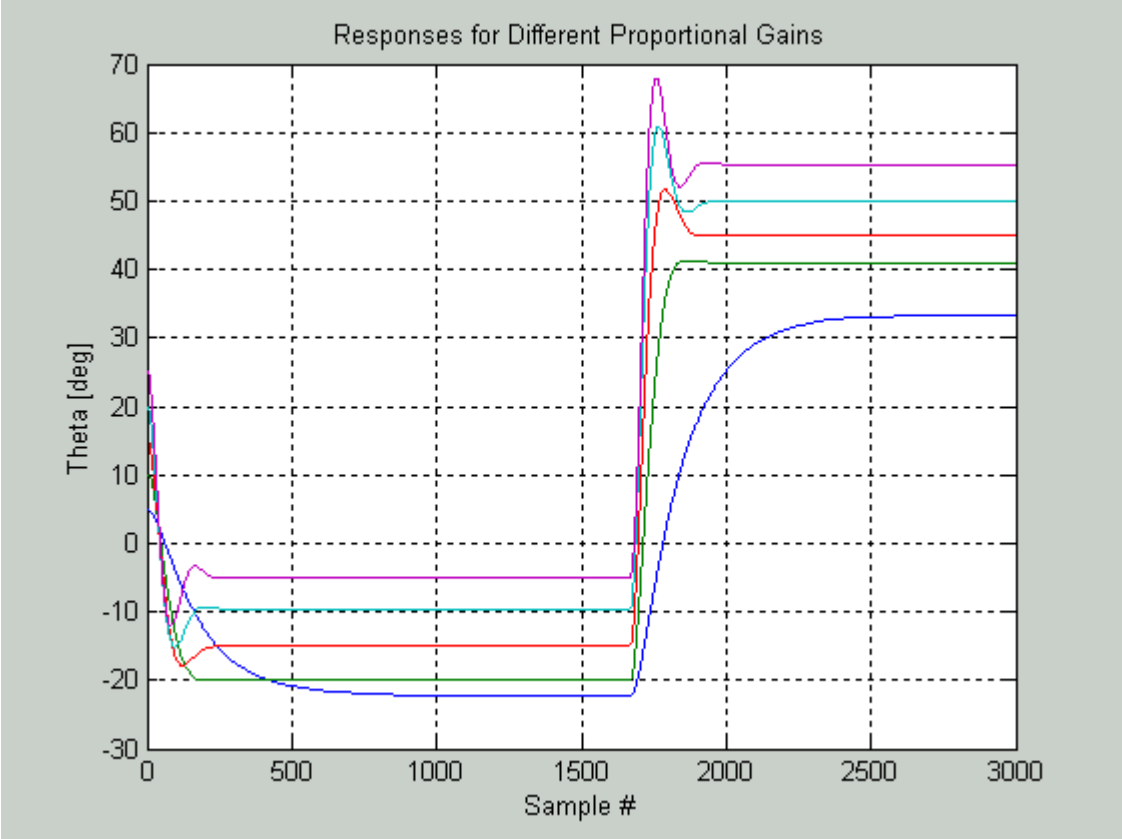


Figure 94 SRV02 Position Response To Incremental Changes of Kp

Notice that each position response is offset by 5 degrees from the previous one to better differentiate all 5 responses.

Troubleshooting

On the rare occasion, users may experience problems when trying to run WinCon. This Troubleshooting section is designed to assist the user in diagnosing some of the potential problems.

Scope Disappears When Stopping Controller

When collecting very large amounts of data (e.g., more than one million samples) in a Scope, it is possible that the Scope may close itself automatically when stopping the controller. The reason the Scope closes is due to timeouts in the communications architecture between WinCon Client and WinCon Server. Drawing millions of points can take longer than usual for a Scope to perform. If it takes too long, WinCon Client may conclude that the Scope is not responding and terminate the connection to the Scope. Closing the connection then causes the Scope window to close.

The solution is to collect less data or to use more than one Scope. For example, if several variables are being drawn on a single Scope, try splitting the variables amongst more than one Scope. Alternatively, decrease the buffer size of the plot or reduce the sampling period of the system so that fewer data points are collected.

Problems Using Visual C++ 6.0 SP5 and RTX 4.3.2.1

Some clients have experienced errors when attempting to use WinCon with RTX 4.3.2.1 and Visual C++ 6.0 Service Pack 5 installed. In this case, revert to Visual C++ 6.0 Service Pack 3 instead. Alternatively, upgrade to a newer version of RTX.

Controller Fails to Load

There are a number of reasons why a controller may fail to load into WinCon Client. For example, be sure that code has actually been generated for the model. To generate real-time code for a model, select Build from the WinCon menu of the Simulink diagram.

This problem may also arise when building a model that was used with a previous version of WinCon. Whenever WinCon has been upgraded, it is always important to clean out the temporary files for models that were built with the older version. For example, suppose the Simulink model is called q_sine. When real-time code is generated, a subdirectory called q_sine_wc_rtw is created. The object files and other intermediate files are placed in this subdirectory. WinCon uses this subdirectory to speed up subsequent builds.

When WinCon is upgraded, and q_sine rebuilt, these old object files may be reused if q_sine is built in the same location, causing the resulting code to be incorrect. To prevent this problem from occurring, always select the Clean menu item from the WinCon menu of the Simulink diagram for each model after upgrading WinCon. The Clean command removes the temporary subdirectory used by WinCon for storing object files and other intermediate files used in the build process. After invoking Clean, then rebuild the model; WinCon will then rebuild the real-time code from scratch.

Another possibility is that the wrong template makefile was used. When WinCon is installed, it automatically sets all the default options for the current operating system. Hence, models created after WinCon is installed should always have the correct options. However, if a model is copied from another computer, or was created earlier using a different version of Windows, then the real-time code generation options may not be set correctly. To be sure that the options are set correctly, invoke Set WinCon Options from the WinCon menu. This command changes all the required options to the proper settings for your WinCon installation. Then use the Clean menu item to remove temporary intermediate files, and finally rebuild the real-time code using the Build menu item.

Note also that the real-time kernel (RTX) under NT/2000/XP does not always recover properly from an attempt to load real-time code that was compiled for the wrong operating system. For example, if you are using Windows 2000 and compile using the win_msvc.tmf template makefile (designed for Windows '98), then the real-time code will be incompatible. When WinCon Client attempts to load this incompatible code, RTX may get corrupted. In this case, use the Clean menu item to clear out the generated code, use Set WinCon Options to correct the options, save the model, reboot the computer, and then rebuild the code. This time, since the correct operating system selected, the real-time code should load.

If you attempt to load real-time code that had previously worked with the same version of WinCon and it fails to load with the error “Invalid file format”, then it is possible that WinCon Service is no longer running. WinCon Client uses WinCon Service to load the real-time code. Hence, if WinCon Service is not running, WinCon Client may attribute the inability to load the real-time code to a problem with the code itself and report this error. To check whether WinCon Service is running, use the Services control panel applet, found under Control Panel. A user with Administrative privileges can restart WinCon Service if it is not running. Otherwise, contact the network administrator to get it running.

Finally, real-time code may fail to load if the hardware that it uses is not installed in your computer. For example, a Simulink diagram containing a MultiQ-PCI block will compile, but fail to load if you do not have a MultiQ-PCI card installed in your computer. When real-time code is loaded, WinCon performs some preliminary initialization of the hardware. To initialize PCI cards, WinCon first locates the card. If the card cannot be located then it returns an error and the real-time code will fail to load.

WinCon Menu Missing from Simulink Diagram

If the WinCon menu is missing from the Simulink diagrams, then WinCon Link is not running. Make sure that you rebooted the computer after installing WinCon Server. You can run WinCon Link at any time, but be sure to close any Simulink diagrams first. Then run WinCon Link and reopen the Simulink diagrams.

Simulink Diagram Fails to Compile

At times a Simulink diagram will fail to compile. If this problem occurs immediately after installing WinCon, make sure that you have run mex –setup to configure MATLAB for the version of Visual C++ that you have installed. Refer to The MathWorks documentation for more details on setting up the MATLAB environment for the mex command.

The most common cause of a Simulink diagram failing to compile is the use of custom S-functions that call functions that are illegal in a real-time kernel environment, or attempts to use code produced by the MATLAB Compiler. This problem is more common under Windows '98 because there is no support for Win32 functions and very little support for the standard C library under these operating systems. There is much more support for Win32 functions and the standard C library under the RTX real-time kernel used in Windows NT/2000/XP. Refer to section Restrictions on Real-Time Code on page 58 for more details. Quanser recommends using Windows NT, 2000 or XP (or future versions of the same line).

More detail can often be obtained by looking for errors in the output in the MATLAB Command Window. When building code, WinCon produces a list of the operations being performed in the MATLAB Command Window. Errors during the build process are inserted somewhere in this output, depending on when the error in the build process was detected.

Scope Fails to Plot Signal

WinCon Scopes only plot signals of type 'double', the default Simulink data type. However, certain blocks output a data type other than double. For example, the Serial Input block's output has the type 'uint8' (unsigned byte). To plot this signal on a WinCon Scope, insert a Data Type Conversion block from the Signals & Systems library just prior to the Scope in the Simulink diagram. Set the data type to 'double'.

MultiQ-3 TimeBase Block Doesn't Work

The MultiQ-3 data acquisition card is an ISA card and is not plug-and-play. The IRQ used by the card, and the clock employed, are set by jumpers on the card itself. The settings in

the MultiQ-3 TimeBase block *must* match the jumper settings on the card, or the TimeBase block will not work.

Also, because the IRQ level is configured by a jumper, it is important to ensure that the IRQ level selected does not conflict with other devices in the system. Use Device Manager to check whether the IRQ level chosen interferes with other devices in your system, such as the sound card. The IRQ for the MultiQ-3 cannot be shared.

See the next section for other possibilities that are not specific to a particular data acquisition card.

TimeBase Block Doesn't Work

The TimeBase blocks use hardware interrupts. Under NT/2000/XP, the RTX real-time kernel requires that there be an RTX device driver whenever interrupts are used. An RTX device driver is assigned to a device by converting the Windows driver to an RTX driver via the RTX Properties control panel. Refer to the documentation that comes with RTX for more information on this procedure. Note that TimeBase blocks are not necessary under NT/2000/XP unless you require sampling periods faster than 10kHz.

To achieve 10kHz sampling rates, set the HAL Timer Period in the RTX Properties control panel to 100ns. For this setting to take effect, you will have to reboot your machine. Alternatively, stop WinCon Service using the Services control panel applet (as Administrator). Stop RTX using the RTX Properties control panel applet. Restart RTX and then restart WinCon Service.

To convert a Windows 2000 or Windows XP device to an RTX device:

1. From the Windows Start menu, click **Programs | VenturCom | RTX | RTX Properties** to access the RTX Properties control panel.
2. From the **RTX Properties** control panel, select the **Plug and Play** tab. The window displays the devices on your system.

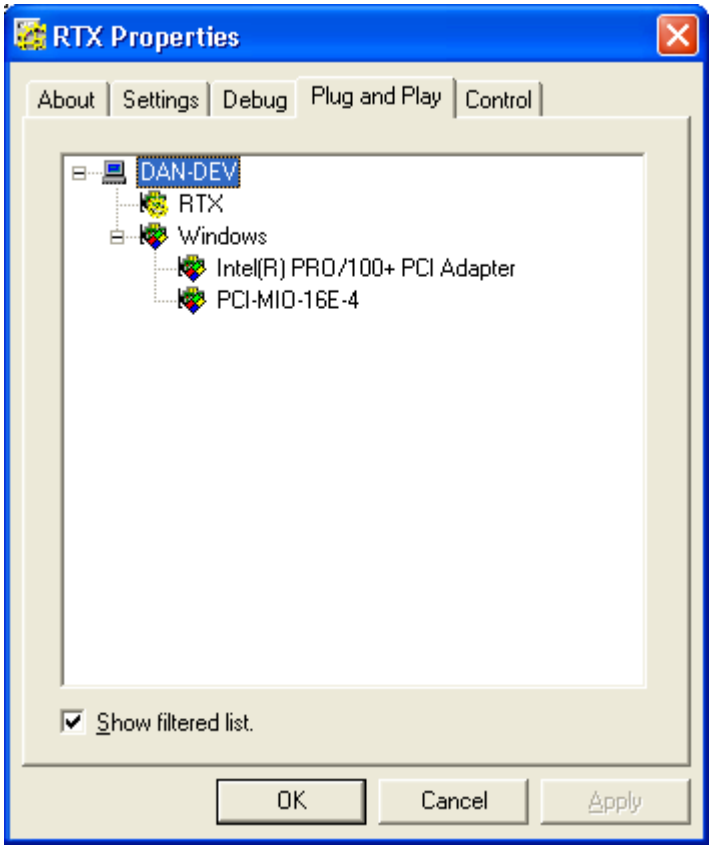


Figure 95 Converting Windows drivers to RTX

3. Right-click on the Windows device you wish to convert, then click **Convert to RTX**.
4. Click **Apply** to select another device or click **OK** to close the RTX Properties control panel.

If the RTX Properties control panel could not remove the specified device, you will receive a warning to uninstall the device and the Device Manager will be started.

To uninstall the device:

1. Select the device you are trying to convert to RTX.
2. From the **Action** menu, click **Uninstall**.
3. Restart your system.
4. To verify that the device has been installed correctly, open the RTX Properties control panel and check the Devices tab. Your device will be listed under the RTX Hardware list, along with the IRQ and Shareable flag set by default.

It may not always be possible to convert a Windows driver to RTX. Some Windows drivers appear to be incompatible with RTX. Contact VenturCom (http://www.vci.com) if problems arise trying to convert a driver to RTX.

Cannot Achieve 10kHz Sampling Rate

When using NT/2000/XP, make sure that the HAL Timer Period for RTX is set to the fastest rate of 100ns. Set the HAL Timer Period using the RTX Properties control panel applet. **The sampling rate of the controller is limited to be no faster than the HAL Timer Period when a hardware time-base is not being used.** If the HAL Timer Period is slower than the sampling period of the controller then Scopes will not display properly, the controller will not operate at the correct rate (it will be run at the HAL Timer Period instead) and the control system may go unstable!

If the HAL Timer Period is set correctly but you haven't rebooted the system since changing the HAL Timer Period, then reboot the system to ensure that the change takes effect. Also note that RTX has a bug in which the HAL Timer Period is not set correctly if RTX is configured to run at boot time instead of on demand. **Always configure RTX to run on demand.**

If the HAL Timer Period is set to 100ns then the controller may simply be too computationally expensive to achieve the desired 10kHz sampling rate. Check the WinCon Client window to see how much time is being spent doing computations (approximately). Note that analog I/O can take many microseconds to perform, depending on the data acquisition card, and thus is a limiting factor in achieving fast sampling rates. Also note that most mechanical systems can be controlled with a 1kHz sampling rate.

Obtaining Support

Note that a support contract may be required to obtain technical support. To obtain support from Quanser, go to http://www.wincon.quanser.com and click on the Tech Support link. Fill in the form with all requested version information and a description of the problem encountered. Submit the form. Be sure to include your email address and a telephone number where you can be reached. A qualified technical support person will contact you.

Index

Index