
GalilTools

Galil Motion Control, Inc
270 Technology Way
Rocklin, California, 95765
Phone: (916) 626-0101
Fax: (916) 626-0102
Technical: support@galilmc.com
Docs: documentation@galilmc.com
Web: www.galil.com

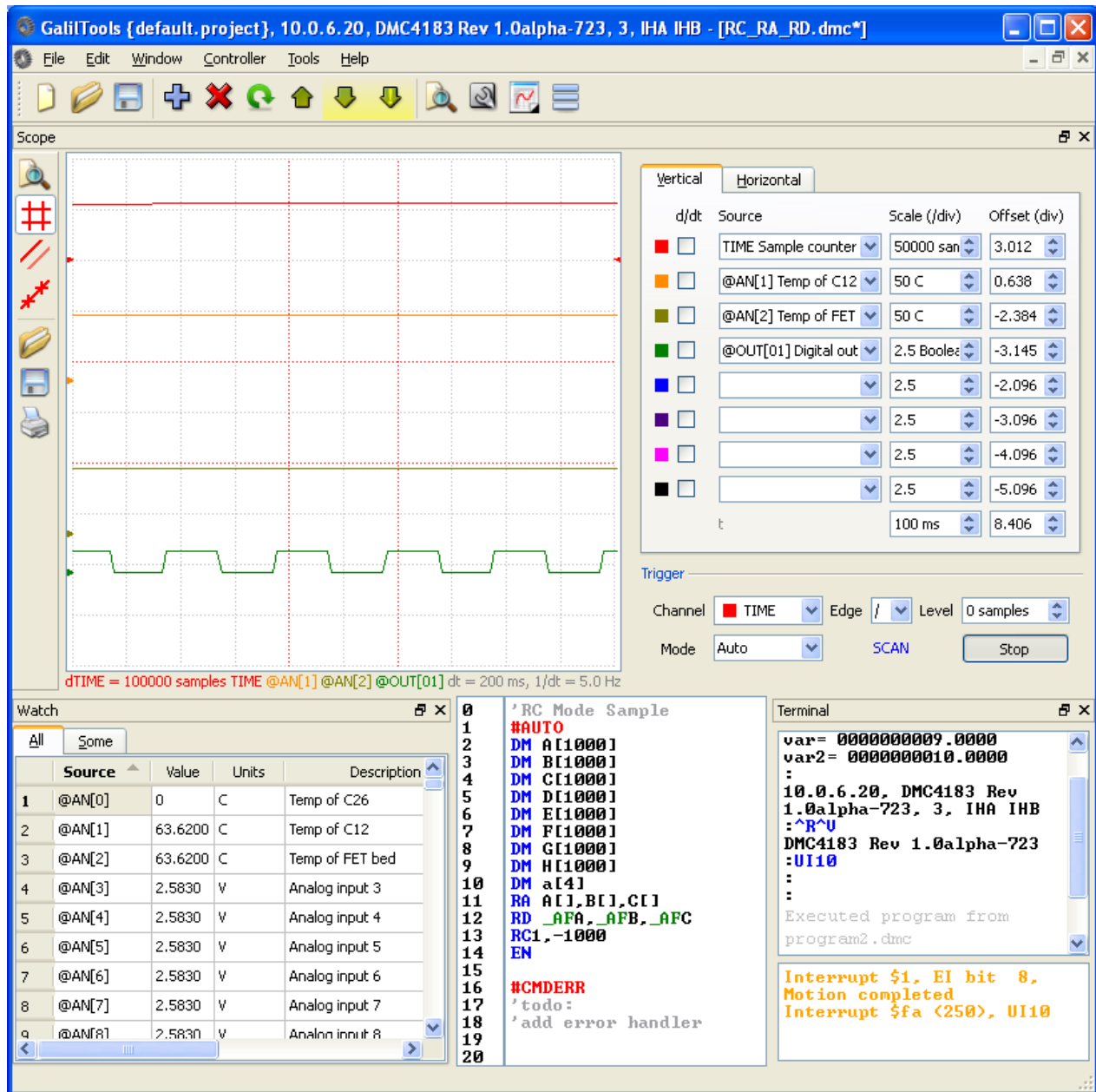
Table of Content

Table of Content	2
Introduction	3
Installing GalilTools 1.6.3 on Red Hat 5.8	7
Installing GalilTools 1.6.3 on Ubuntu 12.04	12
Galil Projects	15
Advanced Settings	16
Connections	17
Terminal	21
Editor	24
Watch	26
Graphical Scope (Full Edition Only)	28
Servo Tuner (Full Edition Only)	38
Debugging	42
Comma-Separated Values (CSV Files)	47
Menus and Hot Keys	50
Ethernet Network Guide for Connecting Galil Controllers	52
PCI on Windows (DMC-18x2/6)	55
Communication Library Reference	59
Hello	78

Introduction

Overview

[GaliTools](#) allows the user to set up, configure, tune, test and analyze motion on a [Gali](#) controller. [GaliTools-Full Edition](#) has a command [Terminal](#) with auto-repeat feature and smart coloring, program [Editor](#) with syntax highlighting, a data record [Watch](#) window including a user-configurable dashboard, an 8-channel real-time [Scope](#), and an automatic axis [Tuner](#) ([GaliTools-Lite](#) excludes the Scope and Tuner). [GaliTools](#) has been written on a cross-platform library, allowing the suite to be deployed on a variety of popular operating systems such as [Windows](#) XP, XP x64, Vista, Vista x64, and various [Linux](#) distributions on x86 and x64 (Fedora Core 8 examples in these documents). Contact [Gali](#) for other operating systems.



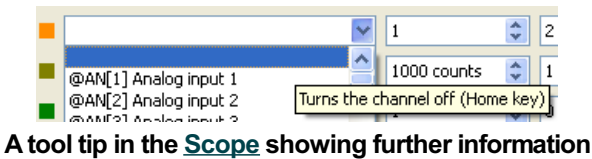
GaliTools-Full Edition

Various areas of [GaliTools](#) have a yellow highlighted color and indicate operations or fields that can potentially cause axis motion. These areas include the Download and Execute buttons, the [Terminal](#) input textbox, and the [Tuner](#) settings, Autotune button and sliders. Care should be taken to ensure potential movement will not damage equipment nor cause injury.

When running [GaliTools](#) in conjunction with a stand-alone [Ethernet](#) controller, it is recommended that an [Ethernet](#) connection be used

because an [RS-232](#) connection does not offer the optimum communication speed required for the [Scope](#). [GaliTools](#) works with the DMC-40x0, 18x6, 41x3, 21x3, 18x2 board revision D and up, and [RIO](#). New firmware (1.0b or greater) is required to use Watch on the [DMC-18x6](#) series. Minimal [DMC-1417](#) support was added in version 1.4.4 (no Watch, no Scope, no Tuner tools).

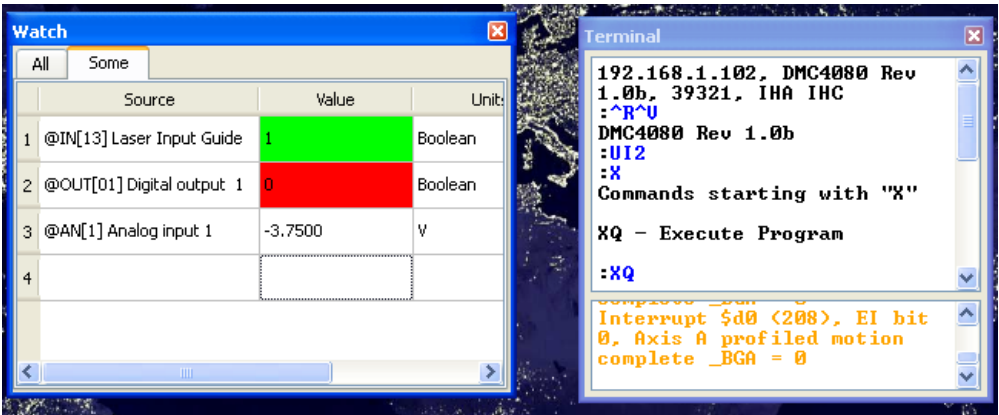
Hint: Many features in [GaliTools](#) have tool tips; further information about the feature may be obtained by hovering over a window element.



The Lite Edition of [GaliTools](#) comes packaged with a communications [Terminal](#), program [Editors](#), and a [Watch](#) window. The Full Edition adds the multi-channel [Scope](#) and [Tuner](#).

The environment contains program [Editors](#) in the center surrounded by the [Terminal](#), [Watch](#), [Scope](#), and [Tuner](#) dock windows, which are easily accessible from the tool bar. [Editors](#) represent [Gali](#)-language programs and multiple [Editors](#) can be open at the same time. The other tools ([Terminal](#), [Watch](#), [Scope](#), and [Tuner](#)) can be docked to any of the four main environment walls, stacked on top of each other to form a tabbed stack, or floated outside of the environment.

From the Controller menu, the user can Connect, Download/Upload [Arrays](#), and Download Firmware (which requires an [RS-232](#) connection on 21x3 controllers).

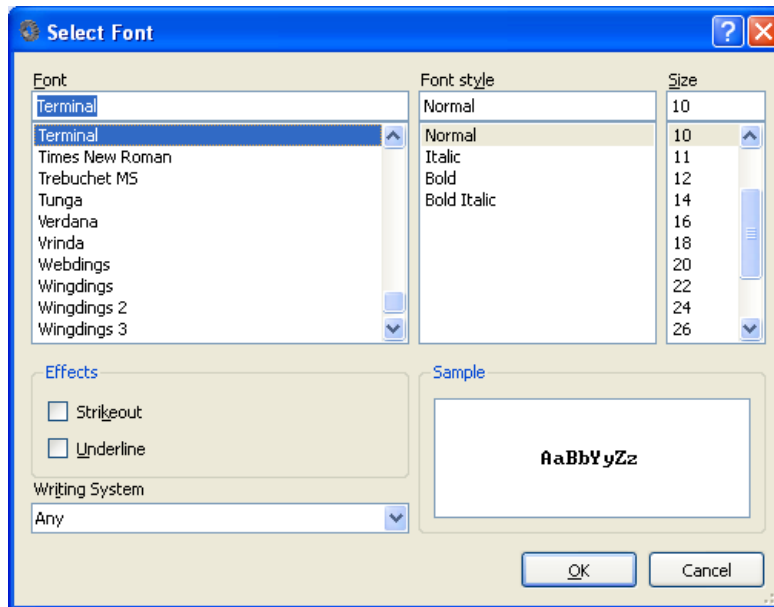


Floating the [GaliTools](#) modules outside of the environment over the [Windows XP](#) desktop.

See the individual Tool descriptions in the [table of contents](#) for more tool-specific information.

GaliTools Font

GaliTools allows the font of the terminal and editor to be customized by the user. Choose *Edit | Font* to bring up the font choser.



Users can choose a preferred font for working in GalilTools.

Windows Installation

Open the [GalilTools](#) for [Windows](#) installation executable and proceed with the setup. The installation will create a desktop shortcut and a program group in the start menu.

PCI (DMC-18x2/6)

During the installation, a prompt will ask to install the [PCI driver](#).



PCI Installation Prompt (Optional)

If [GalilTools](#) is going to be used in conjunction with a [PCI](#) controller (DMC-18x2 or 18x6), the [PCI driver](#) must be installed. Otherwise, the user can choose to skip this step of the installation by clicking Cancel. If using a [PCI](#) controller, once the installation is done, follow the onscreen directions to restart the computer.

If [GalilTools](#) is installed on a computer with legacy [Galil](#) software such as [WSDK](#) or [SmartTerm](#), it may be necessary to manually change the [PCI driver](#) to connect to [PCI](#) controllers. See the chapter [PCI on Windows \(DMC-18x2/6\)](#) in this manual should your [PCI](#) controller not be listed in the [Available](#) tab of the [GalilTools Connections](#) dialog.

Note: Contact Galil for signed drivers if a PCI based controller is to be used on a Vista x64 operating system.

[Contents](#)

Installing GalilTools 1.6.3 on Red Hat 5.8

Install GalilTools Software Package

This section covers the installation of the GalilTools 64 bit software package. If the target operating system is a 32 bit operating system, replace "x86_64" with "i386" in all of the commands seen below. After installing the GalilTools software package, additional configuration is necessary. Any previous installations of GalilTools must be removed before installing the latest version.

- Before using a PCI based controller, the GalilTools PCI driver must be installed and user permissions must be modified.
- Before using a USB or RS232 based controllers, user permissions must be modified.
- Before using an Ethernet based controller, the firewall must be modified.

This document will outline the complete setup of GalilTools for all controller communication interfaces.

1. Open a terminal by clicking Applications>Accessories>Terminal.
2. Get the GalilTools package and install it with the package manager.

```
$ wget http://www.galilmc.com/support/downloads/software/galiltools/linux/galiltools.x86_64.rpm
$ su -c 'rpm -i galiltools.x86_64.rpm'
Password:
```

3. GalilTools can be launched from the terminal with the command "galiltools" or from the system menu at Applications>Programming>GalilTools.

Install the GalilTools PCI Driver

If GalilTools is to be used with a PCI based controller, the GalilTools PCI driver must be installed. If GalilTools will not be used with a PCI based controller, this section can be skipped. If the GalilSuite PCI driver has been previously installed, it must be removed before installing the GalilTools PCI driver. While there is limited support for the GalilSuite PCI driver in GalilTools, it is strongly recommended that the GalilTools PCI driver is used with GalilTools.

1. Open a terminal by clicking Applications>Accessories>Terminal.
2. Get the Galil public key and import it.

```
$ wget http://www.galilmc.com/support/downloads/software/galilsuite/linux/galil_public_key.asc
$ su -c 'rpm --import galil_public_key.asc'
Password:
```

3. Download the required packages for the build process.

```
$ su -c 'yum install rpm-build kernel-devel kernel-headers kernel-xen gcc glib'
Password:
```

4. Prepare the build environment.

```
$ mkdir -p ~/rpmbuild/{BUILD,RPMS,SOURCES,SPECS,SRPMS}
$ echo '%_topdir %(echo $HOME)/rpmbuild' > ~/.rpmmacros
```

5. Download the Linux PCI driver from the Galil website.

```
$ wget http://www.galilmc.com/support/downloads/software/galiltools/linux/galiltools-pci.src.rpm
```

6. Rebuild the GalilTools PCI driver using rpmbuild.

```
$ rpmbuild --rebuild galiltools-pci.src.rpm
```

7. Move the newly built package into the working directory.

```
$ mv ~/rpmbuild/RPMS/x86_64/galiltools-pci*.x86_64.rpm galiltools-pci.x86_64.rpm
```

8. Install the newly built package.

```
$ su -c 'rpm -i galiltools-pci.x86_64.rpm'
```

Set User Permissions

To connect to Galil controllers over the USB, RS232 or PCI interfaces, a user must have the proper permissions to access the device. If a standard user intends to connect to a Galil controller over USB, RS232, or PCI, the following instructions must be followed. If a standard user wishes only to connect to Ethernet based controllers, this section can be skipped.

Red Hat Method 1: Using the command line

1. Open a terminal by clicking Applications>Accessories>Terminal.
2. Issue the following command to add users to the uucp group. The uucp group members have permission to use serial ports. Such permission is necessary for communication with Galil controllers over RS232 or USB. Make sure to replace "username" with the desired username.

```
$ su -c '/usr/sbin/usermod -a -G uucp username'  
Password:
```

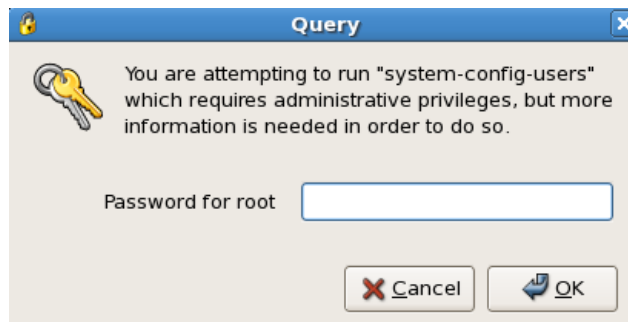
3. Issue the following command to add users to the galil group. The galil group members have permission to use Galil PCI based controllers. Make sure to replace username with the desired username. Note that the galil group will not exist unless a Galil PCI driver has been previously installed.

```
$ su -c '/usr/sbin/usermod -a -G galil username'  
Password:
```

4. The process may be repeated to modify other users.
5. For the updated permissions to take effect, the modified user must logout then log back in.

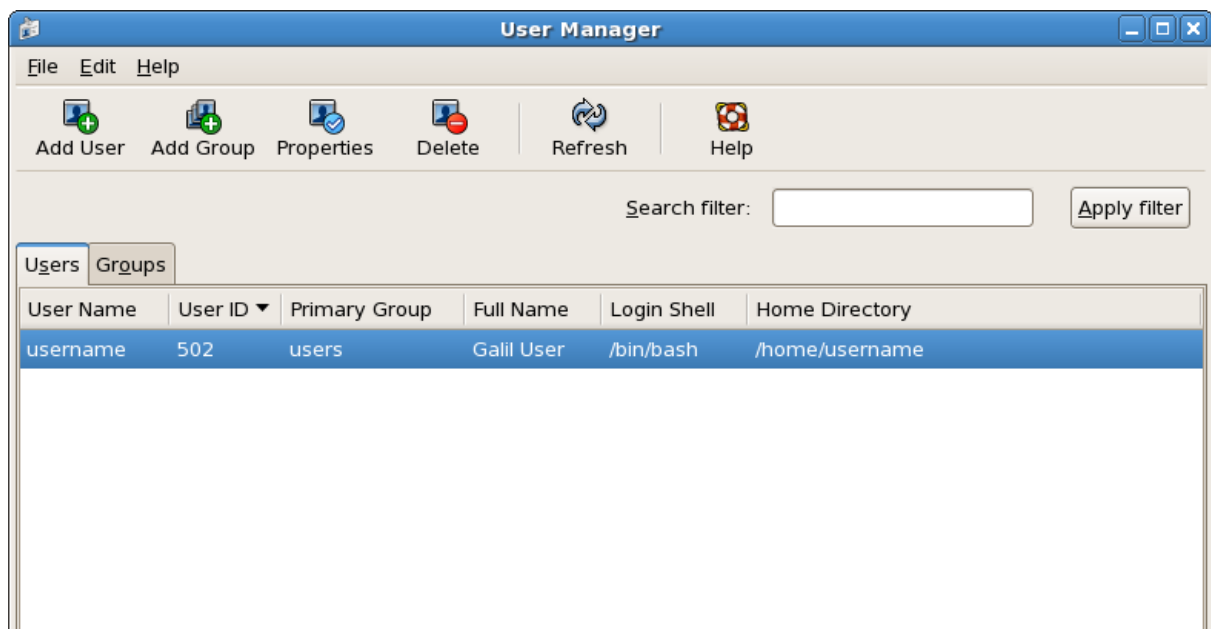
Red Hat Method 2: Using the Graphical Administration Tools

1. Open Users and Groups by clicking System>Administration>Users and Groups.
2. Input the root password into the Root Password Prompt.



Root Password Prompt

3. The Users and Groups Window should now appear.





Users and Groups Window

4. Select the user that will be modified to communicate with Galil controllers over USB, RS232 or PCI by clicking on the username in the Users and Groups Window.
5. Click the "Properties" button in the Users and Groups Window toolbar. The User Properties Window will open.

The 'User Properties' window is shown with the 'User Data' tab selected. It contains several text input fields: 'User Name' (username), 'Full Name' (Galil User), 'Password' (masked with ****), 'Confirm Password' (masked with ****), 'Home Directory' (/home/username), and 'Login Shell' (/bin/bash). At the bottom right are 'Cancel' and 'OK' buttons.

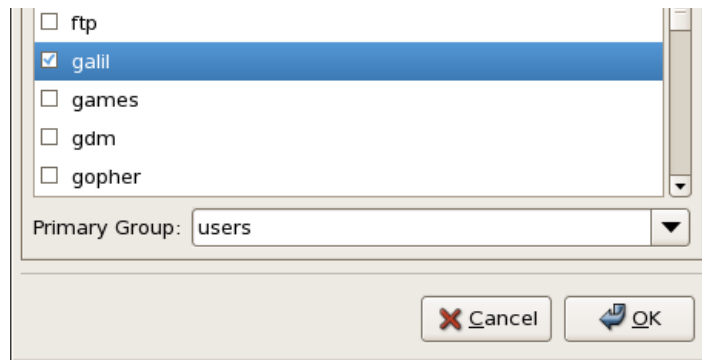
User Properties Window

6. Select the Groups tab in the User Properties Window. The User Properties Group Tab should now be visible.

The 'User Properties' window is shown with the 'Groups' tab selected. It displays a list of groups with checkboxes: utempter, utmp, uucp (checked), vboxsf, vcsa, wheel, and xfs. Below the list is a 'Primary Group' dropdown menu set to 'users'. 'Cancel' and 'OK' buttons are at the bottom right.

User Properties Group Tab showing the uucp group check box

The 'User Properties' window is shown with the 'Groups' tab selected, displaying a scrollable list of groups. The visible groups are 'disk' and 'floppy', both with unchecked checkboxes.



User Properties Group Tab showing the galil group check box

7. In the User Properties Group Tab, tick the check boxes next to uucp to give the user access to serial ports. This is necessary for communications with Galil controllers over RS232 or USB.
8. In the User Properties Group Tab, tick the check marks next to galil to give the user permissions to communicate with Galil PCI based controllers. Note that the galil group will not exist unless a Galil PCI driver has been previously installed.
9. Click the OK button. The process may be repeated to modify other users. After all desired modifications have been made, close out of the User Management tool.
10. For the updated permissions to take effect, modified users must logout then log back in.

Configure the Firewall

By default, during installation, Red Hat installs and enables a firewall. Furthermore, on Linux machines, standard users cannot typically listen on any port less than 1024. This makes a redirect necessary.

1. Accepting UDP traffic on port 50000 allows for the connection to Galil controllers over Ethernet
2. Accepting UDP traffic on port 60001 through 60007 allows the host to receive Data Record and unsolicited messages from the controller.
3. The port redirection of port 67 to port 1067 allows standard users to assign IP addresses to controllers. Note that, as a result, port 1067 must accept udp traffic.

If no firewall is in use, setting firewall exceptions for ports 1067, 50000, and 60001 through 60007 are not necessary. If standard users require permission to assign IP addresses to controllers, the port redirection must still be applied.

The following changes allow GalilTools to communicate with controllers while a firewall is enabled and allow standard users to assign IP addresses to controllers.

1. Open a terminal by clicking Applications>Accessories>Terminal.
2. Assuming that the firewall is enabled, a file similar to the following can be found at "/etc/sysconfig/iptables". Open this file for editing.

```
$ su -c 'gedit /etc/sysconfig/iptables'
Password:
```

3. If the file opened by the previous command is blank, you must re-enable the Red Hat firewall or build a custom set of rules. Modifications to the iptables file may have been previously made for services like windows file sharing. The unmodified version is given only for demonstration purposes.

```
$ su -c 'gedit /etc/sysconfig/iptables'
Password:
# Firewall configuration written by system-config-securitylevel
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:RH-Firewall-1-INPUT - [0:0]
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
-A RH-Firewall-1-INPUT -p icmp --icmp-type any -j ACCEPT
-A RH-Firewall-1-INPUT -p 50 -j ACCEPT
-A RH-Firewall-1-INPUT -p 51 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 5353 -d 224.0.0.251 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

```
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

4. Modify the "/etc/sysconfig/iptables" file to include the lines that are highlighted in the listing below. These lines include exceptions that allow for the connection to Galil controllers over Ethernet and the ability to find and assign IP addresses to Ethernet based controllers.

```
# Firewall configuration written by system-config-securitylevel
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:RH-Firewall-1-INPUT - [0:0]
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
-A RH-Firewall-1-INPUT -p icmp --icmp-type any -j ACCEPT
-A RH-Firewall-1-INPUT -p 50 -j ACCEPT
-A RH-Firewall-1-INPUT -p 51 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 5353 -d 224.0.0.251 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 1067 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 50000 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 60001:60007 -j ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
*nat
:PREROUTING ACCEPT [0:0]
-A PREROUTING -p udp --src 0.0.0.0 --dport 67 -j DNAT --to 0.0.0.0:1067
COMMIT
```

5. The addition of the NAT module will require restarting the iptables service.

```
$ su -c '/sbin/service iptables restart'
Password:
```

If standard users do not need to assign IP addresses to controllers, the redirection of port 67 to port 1067 is not necessary but the "-bootps 67 -bootpc 68" command line switch must be used to tell GalilTools to listen on port 67 instead of the default 1067. The following is an example of how to launch GalilTools from the terminal to allow a privileged user to assign an IP address to a controller without setting up the operating system to redirect port 67 to 1067.

```
$ su -c 'galiltools -bootps 67 -bootpc 68'
Password:
```

[Contents](#)

Installing GalilTools 1.6.3 on Ubuntu 12.04

Install GalilTools Software Package

This section covers the installation of the GalilTools 64 bit software package. After installing the GalilTools software package, additional configuration is necessary. Any previous installations of GalilTools must be removed before installing the latest version.

- Before using a PCI based controller, the GalilTools PCI driver must be installed and user permissions must be modified.
- Before using a USB or RS232 based controllers, user permissions must be modified.
- Before using an Ethernet based controller, the firewall must be configured.

This document will outline the complete setup of GalilTools for all controller communication interfaces.

1. Open a terminal by opening the Dash Home, typing terminal, then clicking the icon. Alternatively the <ctrl> + <alt> + T keyboard shortcut can be used.
2. Get the GalilTools package and install it with the package manager.

```
$ wget http://www.galilmc.com/support/downloads/software/galiltools/linux/galiltools_1.6.3_amd64.deb
$ sudo dpkg -i galiltools_1.6.3_amd64.deb
[sudo] password for username:
```

3. GalilTools can be launched from the terminal with the command "galiltools" or from the Dash Home by searching for "galil".

Install the GalilTools PCI Driver

If GalilTools is to be used with a PCI based controller, the GalilTools PCI driver must be installed. If GalilTools will not be used with a PCI based controller, this section can be skipped. If the GalilSuite PCI driver has been previously installed, it must be removed before installing the GalilTools PCI driver. While there is limited support for the GalilSuite PCI driver in GalilTools, it is strongly recommended that the GalilTools PCI driver is used with GalilTools.

1. Open a terminal by opening the Dash Home, typing terminal, then clicking the icon. Alternatively the <ctrl> + <alt> + T keyboard shortcut can be used.
2. Get the Galil public key and import it.

```
$ wget http://www.galilmc.com/support/downloads/software/galilsuite/linux/galil_public_key.asc
$ gpg --no-default-keyring --keyring trustedkeys.gpg --import galil_public_key.asc
```

3. Download the required packages for the build process.

```
$ sudo apt-get install build-essential autoconf automake autotools-dev dh-make debhelper devscripts fakeroot xutils lintian pbuilder
[sudo] password for username:
```

4. Download the Linux PCI driver Debian source control file and source file from the Galil website.

```
$ wget http://www.galilmc.com/support/downloads/software/galiltools/linux/galiltools-pci_1.6.3.dsc
$ wget http://www.galilmc.com/support/downloads/software/galiltools/linux/galiltools-pci_1.6.3.tar.gz
```

5. Extract the source and verify integrity of source package.

```
$ dpkg-source -x galiltools-pci_1.6.3.dsc
```

6. Move the newly extracted source directory and build the driver.

```
$ cd galiltools-pci-1.6.3
$ dpkg-buildpackage -uc -b
```

7. Return to the parent directory where the deb installer file is located and install the deb installer file.

```
$ cd ..
$ sudo dpkg -i galiltools-pci_1.6.3_amd64.deb
```

Set User Permissions

To connect to Galil controllers over the USB, RS232 or PCI interfaces, a user must have the proper permissions to access the device. If a standard user intends to connect to a Galil controller over USB, RS232, or PCI, the following instructions must be followed. If a standard user wishes only to connect to Ethernet based controllers, this section can be skipped.

1. Open a terminal by opening the Dash Home, typing terminal, then clicking the icon. Alternatively the <ctrl> + <alt> + T keyboard shortcut can be used.
2. Issue the following command to add users to the dialout group. The dialout group members have permission to use serial ports. Such permission is necessary for communication with Galil controllers over RS232 or USB. Make sure to replace "username" with the desired username.

```
$ sudo usermod -a -G dialout username
[sudo] password for username:
```

3. Issue the following command to add users to the galil group. The galil group members have permission to use Galil PCI based controllers. Make sure to replace username with the desired username. Note that the galil group will not exist unless a Galil PCI driver has been previously installed.

```
$ sudo usermod -a -G galil username
[sudo] password for username:
```

4. The process may be repeated to modify other users.
5. For the updated permissions to take effect, the modified user must logout then log back in.

Configure the Firewall

If a firewall is in place, two exceptions must be included in the firewall settings. Furthermore, on Linux machines, standard users cannot typically listen on any port less than 1024. This makes a redirect necessary.

1. Accepting UDP traffic on port 50000 allows for the connection to Galil controllers over Ethernet.
2. Accepting UDP traffic on port 60001 through 60007 allows the host to receive Data Record and unsolicited messages from the controller.
3. The port redirection of port 67 to port 1067 allows standard users to assign IP addresses to controllers. Note that as a result port 1067 must accept udp traffic.

Three possible firewall configurations will be outlined below. Please review each one to see which is right for the target system.

Standard User without Uncomplicated Firewall (UFW)

If no firewall is in use, setting firewall exceptions for ports 1067, 50000, and 60001 through 60007 are not necessary. Ubuntu does not install with a set of firewall rules so only the port forward will be necessary here. This example will only setup the port forward so standard users can assign IP addresses to controllers.

The following changes allow standard users to assign IP addresses to controllers using GalilTools.

1. Open a terminal by clicking Applications>Accessories>Terminal.
2. First a configuration file will be created at /etc/iptables.rules. This will be loaded by iptables-restore on the network interface pre-up to load the required iptables rules automatically.

```
$ sudo gedit /etc/iptables.rules
[sudo] password for username:
```

3. Add the following lines to the newly created /etc/iptables.rules file.

```
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A PREROUTING -s 0.0.0.0/32 -p udp -m udp --dport 67 -j DNAT --to-destination 0.0.0.0:1067
COMMIT
```

4. Next the /etc/network/interfaces file will be modified.

```
$ sudo gedit /etc/network/interfaces
[sudo] password for username:
```

5. At the end of the network related lines for the interface that the controller will utilize, add the line that is highlighted in green in the

example below.

```
auto lo
iface lo inet loopback
pre-up iptables-restore < /etc/iptables.rules
```

6. The firewall settings from /etc/iptables.rules will now be applied each time the interface comes up. To apply the new settings now, reset the interface or reboot the system.

Standard User with Uncomplicated Firewall (UFW)

For Ubuntu system that use UFW, the default firewall configuration tool for Ubuntu, this example will demonstrate how to create the required exceptions and setup the port forward so standard users can assign IP addresses to controllers. It will be assumed that UFW is running with the default set of rules (profile).

1. Open a terminal by clicking Applications>Accessories>Terminal.
2. First the ufw configuration file will be modified.

```
$ sudo gedit /etc/ufw/before.rules
[sudo] password for username:
```

3. Insert the following text at the bottom of the file.

```
# GalilTools firewall rules
*filter
-A ufw-before-input -p udp -m udp --dport 1067 -j ACCEPT
-A ufw-before-input -p udp -m udp --dport 50000 -j ACCEPT
-A ufw-before-input -p udp -m udp --dport 60001:60007 -j ACCEPT
COMMIT
*nat
:PREROUTING ACCEPT [0:0]
-A PREROUTING -p udp --src 0.0.0.0 --dport 67 -j DNAT --to 0.0.0.0:1067
COMMIT
```

4. To apply the new settings now restart ufw or reboot the machine.

```
$ sudo ufw disable
[sudo] password for username:
$ sudo ufw enable
[sudo] password for username:
```

Root User without Uncomplicated Firewall (UFW)

If standard users do not need to assign IP addresses to controllers, the redirection of port 67 to port 1067 is not necessary but the "-bootps 67 -bootpc 68" command line switch must be used to tell GalilTools to listen on port 67 instead of the default 1067. The following is an example of how to launch GalilTools from the terminal to allow a privileged user to assign an IP address to a controller without setting up the operating system to redirect port 67 to 1067.

```
$ sudo galiltools -bootps 67 -bootpc 68
[sudo] password for username:
```

[Contents](#)

Galil Projects

GalilTools Projects

Projects allow for the GalilTools environment to remember the user's last configuration. It also allows such configurations to be grouped for logical purposes, e.g. tuning, motion development, and productions with DMC application code.

Projects will manage the following in GalilTools:

1. Scope state: channel sources, horizontal and vertical state, trigger configuration, running state, and analysis options.
2. Watch state: "All" verses "Some" tab, contents, and the special data record fields(see [watch](#)).
3. All tools' Location, size, docked status,and the main GalilTools environment geometry
4. Open DMC files in editor
5. Controller connection

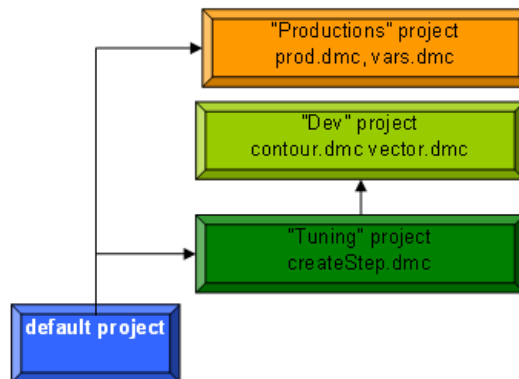
A project is a grouped set of DMC files and environment configuration designed to provide a consistent work environment throughout the design process. A user can have multiple projects relating to a particular application. For example, a user may have a "tuning" project with the tuning window, scope window, and watch-some window configured a certain way, optimal for tuning. Various DMC files such as sample motion profiles and impulse responses, may also be a part of a tuning project. A second project might be "development". This is where the motion characteristics of an application can be developed, and tools will remember the optimum setup for motion tasks. Finally, a third project could be for "production", used for production programming, basic diagnostics or other tasks.

GalilTools will launch after the first instance in the "default" project. The tool states when the user closes GalilTools will be persisted to the next launch of the default project.

To save more than one working environment, choose either:

File | New | Project... This will allow a new project to be created. Hardcoded defaults for environment will form the starting point for creating a customized layout.

File | Save As | Project... This allows a new project to be created with the current project as a starting point for environment layout.



An example of creating projects. DMC files left open will be reopened when a project is opened.

To save the current project, choose *File | Save | Project*. Note, changes to the project will be automatically saved when GalilTools is exited.

To set the project back to it's original state, choose *File | Revert | Project*. This is recommended if the environment has been changed temporarily.

To open a saved project choose *File | Open | Project...* and choose a valid project file.

Opening a project file from the operating system shell will cause GalilTools to launch and load the indicated project.

Connections will also be persisted in projects. The last connected controller will be opened when a project is opened.

[Contents](#)

Advanced Settings

Overview

GalilTools has several advanced options which are available from a software configuration file.

O.S.	Applies to	Configuration File Path
Windows	All GalilTools Projects	C:\Documents and Settings\username\Application Data\Galil\GalilTools.ini
Windows	Project projName only	C:\Documents and Settings\username\Application Data\Galil\projName.ini
Linux	All GalilTools Projects	# \$HOME/.config/Galil/GalilTools.conf
Linux	Project projName only	# \$HOME/.config/Galil/projName.conf

Configuration File Options

Note: Although the backslash (\) is used as the directory and filename delimiter in Windows, paths in the GalilTools configuration file must use a forward slash (/). See examples below.

Note: Close all instances of GalilTools before editing the configuration file

The configuration files contain the following options and defaults (Paths for Windows shown).

GalilTools.ini

File Contents and Default values	Description
<code>[GalilTools]</code>	[] specify a GalilTools module name for which the following options are mutable
<code> LastConnectionFile=/connection.last</code>	File containing the connection information for the last controller connection
<code> ShowUpdates=true</code>	On launch, check for updates. If updates exist, display info.
<code>[Terminal]</code>	Module Name
<code> CommandDictionaryDir=C:/Program Files/Galil/GalilTools/doc</code>	The path to the directory containing the terminal's command libraries

projName.ini (one for each project)

File Contents and Default values	Description
<code>[GalilTools]</code>	Module Name
<code> AutoConnect=true</code>	If true, when this project is opened, its previous connection will be re-established and files left open in the editor last time will be reopened.
<code> WindowPosition=@Point(integers)</code>	GUI state, do not hand edit.
<code> WindowSize=@Size(integers)</code>	GUI state, do not hand edit.
<code> WidgetState=@ByteArray(array of bytes)</code>	GUI state, do not hand edit.
<code>[Terminal]</code>	Module Name
<code> ShowUnsolicitedTimeStamps=false</code>	Unsolicited data will be stamped with the system time when displayed
<code> TerminalFirst=true</code>	Specifies Terminal location precedence with respect to the unsolicited messages window
<code> AutoRepeatRateInMs=1000</code>	The time interval in ms between repeats when auto-repeat is active
<code> SplitterState=@ByteArray(array of bytes)</code>	GUI state, do not hand edit.

[Table of Contents](#)

Connections

A connection to a controller in [GaliITools](#) requires an appropriate Address. An Address is a simple text string which indicates which controller and communication method to use when establishing a connection. The Addresses are in a predefined format, and can be [Saved](#) in a .con file of any name (see below). Once connected to a controller, the current GaliITools project will remember the connection for next time. This can easily be changed simply by connecting to a different controller and saving the project or exiting GaliITools. Addresses replace the legacy software requirement of registering a controller in the [Windows registry](#) prior to connection. The following are examples of valid Addresses:

Address	Connects to
GALILPC1	PCI GaliI Controller (DMC-18x6/18x2) (Windows)
/dev/galilpci0	PCI GaliI Controller (DMC-18x6/18x2) (Linux)
192.168.1.4	Ethernet Controller
RIO47100-13	DNS name for supported controller with DHCP (RIO/DMC-4000)
COM1 19200	RS-232 Port 1 at 19.2 kbaud (Windows)
COM2 115200	RS-232 Port 2 at 115.2 kbaud (Windows)
/dev/ttyS0 19200	RS-232 Port 1 at 19.2 kbaud (Linux)
OFFLINE	Opens GaliITools with no connection. The Editor functions are still available.

There are several optional address-line switches which will affect controller behaviour. Most, but not all, affect standalone controllers only.

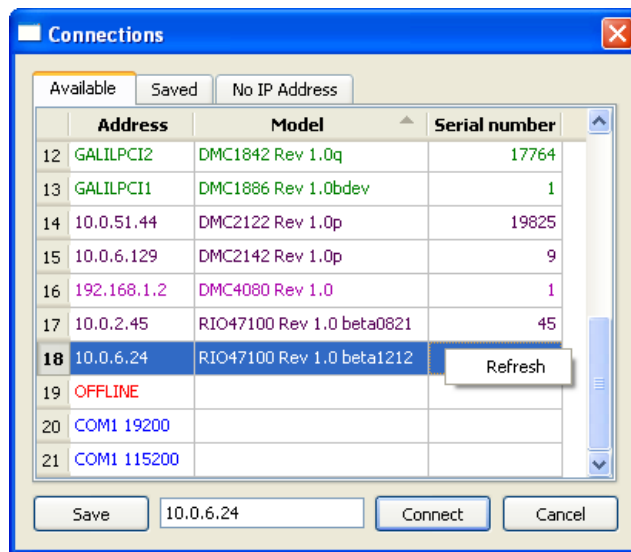
Option	Syntax	Default (if omitted)	Communication Bus	Example
Primary connection (Commands) port	-p1 n	-p1 23	Ethernet	192.168.1.101 -p1 1001
Primary connection protocol	-udp	TCP	Ethernet	192.168.1.101 -udp
Secondary connection (E/UI/DR/MG) port	-p2 n	-p2 60007	Ethernet	192.168.1.101 -p2 2005
Do not subscribe to UDP Interrupts (E/UI)	-ei 0	send EI,,h	Ethernet, DMC-40x0 rev 1.0b or higher	192.168.1.101 -ei 0
Do not subscribe to messages (MG)	-mg 0	send CFh	Ethernet, RS-232	192.168.1.101 -mg 0
Do not subscribe to data records (DR)	-dr 0	send DR,h	Ethernet	192.168.1.101 -dr 0
Baud rate	n	115200	RS-232	COM5 19200
Timeout in milliseconds for commands (timeout_ms)	-t n	-t 500	Ethernet RS-232 PCI	192.168.1.101 -t 5000
Timeout in milliseconds for longer commands (BP,BV,RS,^R^S,DL,QD)	-l n	-l 10000	Ethernet RS-232 PCI	GALILPC1 -l 8000
Connect Silently (note, watch and scope will not function)	-s	commands sent by driver upon connect	Ethernet RS-232 PCI	COM1 19200 -s
Send debug/diagnostic info to file (1 stdout, 2 stderr, or file path)	-d file	no debug output	Ethernet RS232 PCI	192.168.1.101 -d debug.log

Multiple instances of [GaliITools](#) can be opened to one or multiple [Ethernet](#) controllers, but only one connection to a particular [RS-232](#) port or [PCI](#) controller is allowed.

After launching [GaliITools](#) or clicking Connect, the Connections dialog will be displayed, which provides three tabs: [Available](#), [Saved](#), and [No IP Address](#). Note that the Address in the textbox at the bottom of the dialog indicates which connection will be opened once Connect is clicked. This can be manually entered if desired or selected from the [Available](#) or [Saved](#) tabs.

Available

The Available tab displays controllers that are accessible to connect to as well as available [RS-232](#) ports along with popular baud rates. The model, firmware revision, and serial number of [Ethernet](#) controllers with [IP addresses](#) and [PCI](#) controllers are also displayed.



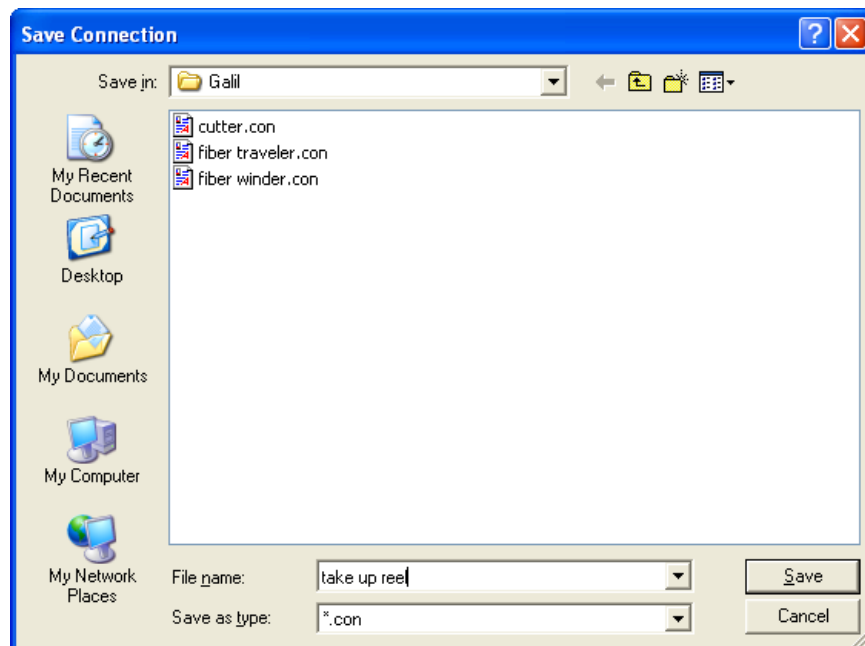
**Connections dialog as seen when starting GalilTools for the first time.
Note: Controllers shown are examples.**

Hint: Right click the connections list to allow a refresh of the Available connections. This is useful for Ethernet controllers that may have just come online.

If a desired baud rate is not present in the list, it can be manually typed in the [Connections](#) textbox. Highlighting a connection places that connection's string in the textbox and clicking Connect will attempt a connection. Although all network-connected controllers are listed, only those within the host's subnet mask can be connected to. For [RS-232](#), it is up to the user to select a port and baud rate where a controller is actually connected. If an [RS-232](#) port or [PCI](#) controller is being used by another application, or an [Ethernet](#) controller is using all [Ethernet](#) handles, it will not be listed in the [Available](#) connections. For users with several connections, groups of [PCI](#), [RS-232](#), and separate network card connections have unique font colors.

Saved

Connections that will be used repeatedly can be [Saved](#) with a user-chosen name. By clicking the Save button, the Address in the textbox can be saved to a connection (.con) file. Clicking the Save button will request a path for the .con file and the user can select an appropriate name based upon the controller's application.



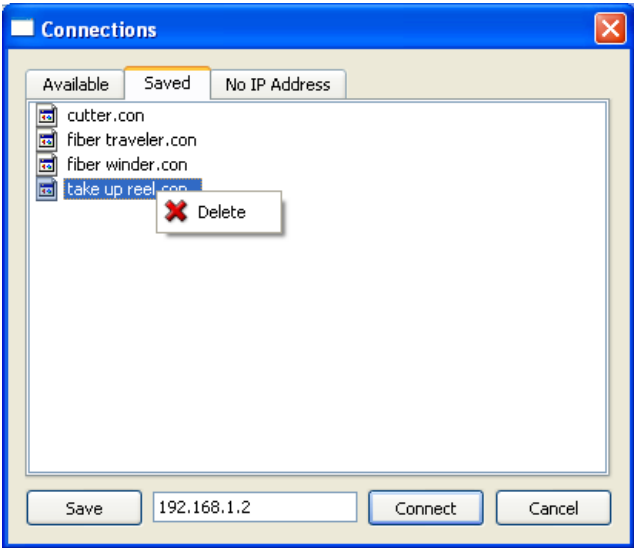
Saving a Connection

While in the Save Connection dialog or the operating system's file explorer a .con file can be renamed or deleted by right clicking on the desired file. Right clicking a [Saved](#) connection on the Saved tab of the Connections dialog also allows the user to delete it.

Connections can be saved to any writable directory (e.g. the desktop) and serve as "shortcuts" to the [Saved](#) connection. Opening the .con file by double clicking from the operating system's file explorer will open [GalilTools](#) and attempt a connection to the .con file's saved string, bypassing the [Connections](#) dialog box. If the .con file does not open into [GalilTools](#), reset its file association by right clicking on the .con file from the file explorer, choose "Open with...", and choose the [GalilTools](#) executable as the assigned application for the .con extension.

If connections are [Saved](#) to the path suggested by [GalilTools](#), it will, in future launches, present the list of [Saved](#) connections there. Only .con files in this special directory will be displayed upon [GalilTools](#) launch. This location is the "Galil" folder in the user's home folder:

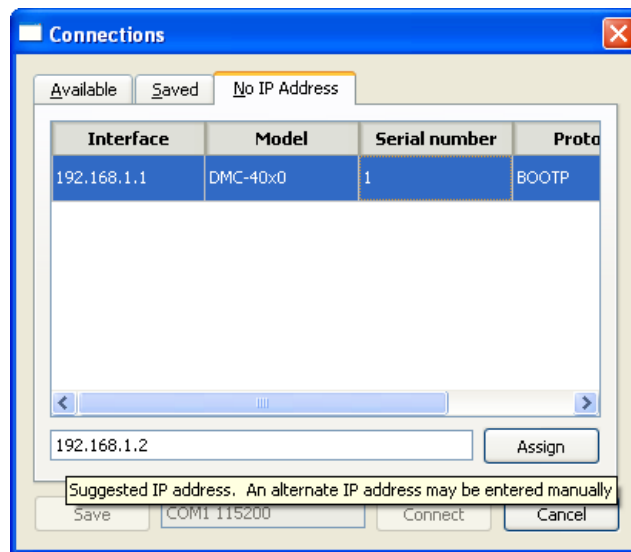
Operating System	Home Folder Location
Windows XP	C:\Documents and Settings
Windows Vista	C:\Users
Linux	/home/username/.config/Galil



[GalilTools](#) prompts the user to choose a [Saved](#) connection

No IP Address

The tab labeled No IP Address is where [Ethernet](#) controllers are listed that do not yet have an [IP address](#) assigned.



List of detected controllers without [IP addresses](#)

It may take up to 5 seconds for the list to fully populate (the time interval at which the controller broadcasts [BOOTP](#) or [DHCP](#) requests). [GaliTools](#) will suggest an [IP address](#), which can be modified in the text box; however, it must be Assigned to the controller for it to take effect. Ensure that the [IP address](#) is appropriate BEFORE assigning and contact your network administrator if necessary. To Assign the [IP address](#), highlight a controller and click the Assign button. Double clicking the controller will auto-assign the suggested IP address. The controller will disappear from the [No IP Address](#) tab and will appear under the [Available](#) tab. At this point, the [IP address](#) has been assigned and burned into the controller's [EEPROM](#). Clicking the [Available](#) tab, selecting the newly assigned controller, and clicking Connect will create a connection to the new [IP address](#).

Hint: If problems are encountered during [Ethernet](#) connection or [IP address](#) assignment, disable all network [firewalls](#) and security tools and try again.

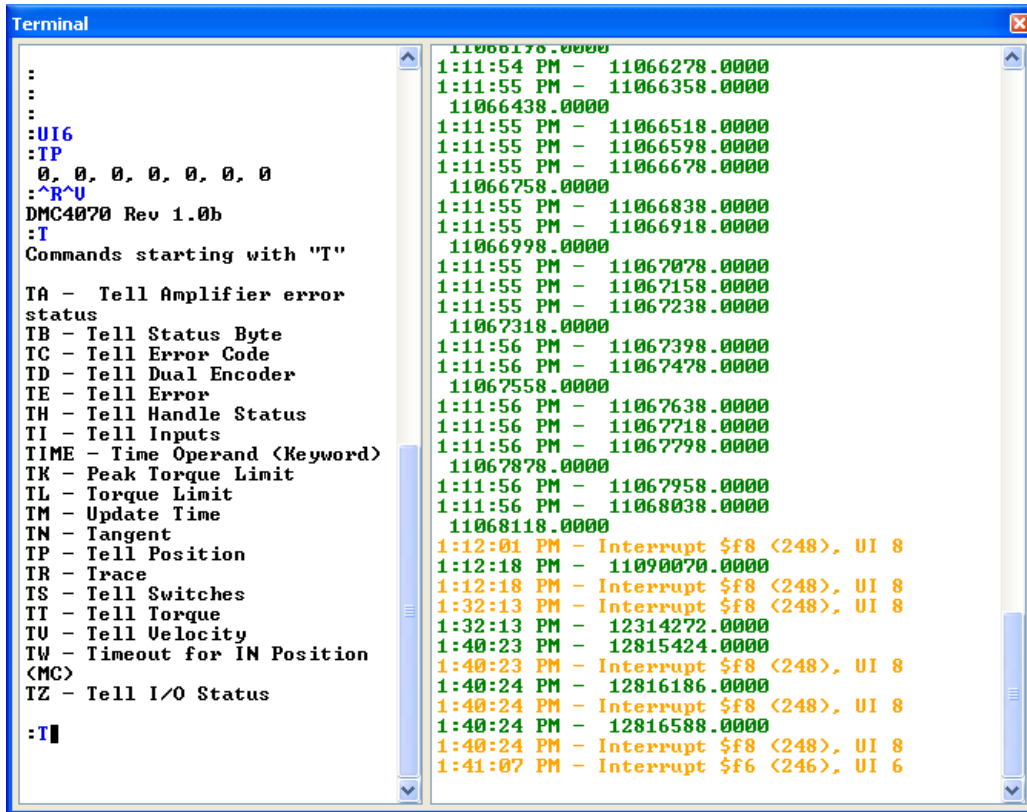
[Contents](#)

Terminal

Overview

The Terminal allows manual and auto repeat command-and-response communication with the controller, monitoring of unsolicited traffic from the controller, and is the most fundamental tool in [GaliITools](#).

The terminal also references a command-set library to provide syntax help while typing commands to the controller.



```
Terminal
:
:
:
:~R^U
DMC4070 Rev 1.0b
:T
Commands starting with "T"
TA - Tell Amplifier error
status
TB - Tell Status Byte
TC - Tell Error Code
TD - Tell Dual Encoder
TE - Tell Error
TH - Tell Handle Status
TI - Tell Inputs
TIME - Time Operand (Keyword)
TK - Peak Torque Limit
TL - Torque Limit
TM - Update Time
TN - Tangent
TP - Tell Position
TR - Trace
TS - Tell Switches
TT - Tell Torque
TV - Tell Velocity
TW - Timeout for IN Position
(MC)
TZ - Tell I/O Status

:T

11066178.0000
1:11:54 PM - 11066278.0000
1:11:55 PM - 11066358.0000
11066438.0000
1:11:55 PM - 11066518.0000
1:11:55 PM - 11066598.0000
1:11:55 PM - 11066678.0000
11066758.0000
1:11:55 PM - 11066838.0000
1:11:55 PM - 11066918.0000
11066998.0000
1:11:55 PM - 11067078.0000
1:11:55 PM - 11067158.0000
1:11:55 PM - 11067238.0000
11067318.0000
1:11:56 PM - 11067398.0000
1:11:56 PM - 11067478.0000
11067558.0000
1:11:56 PM - 11067638.0000
1:11:56 PM - 11067718.0000
1:11:56 PM - 11067798.0000
11067878.0000
1:11:56 PM - 11067958.0000
1:11:56 PM - 11068038.0000
11068118.0000
1:12:01 PM - Interrupt $f8 <248>, UI 8
1:12:18 PM - 11090070.0000
1:12:18 PM - Interrupt $f8 <248>, UI 8
1:32:13 PM - Interrupt $f8 <248>, UI 8
1:32:13 PM - 12314272.0000
1:40:23 PM - 12815424.0000
1:40:23 PM - Interrupt $f8 <248>, UI 8
1:40:24 PM - 12816186.0000
1:40:24 PM - Interrupt $f8 <248>, UI 8
1:40:24 PM - 12816588.0000
1:40:24 PM - Interrupt $f8 <248>, UI 8
1:41:07 PM - Interrupt $f6 <246>, UI 6
```

The GaliITools terminal

Command and Response, Repeat, Auto Repeat, and Syntax Help

- Commands are typed into the command window and the responses from the controller are printed in a standard terminal format.
- Pushing the up arrow key will recall previous commands similar to the Linux Bash shell or Windows doskey.
- While typing a command, typing tab-tab will list all available commands which match the user typed pattern

```

:
:
: @
Commands starting with "@"

@FRAC[n] - Fractional part
@IN[n] - Read digital input
@ABS[n] - Absolute value
@INT[n] - Integer part
@ACOS[n] - Inverse cosine
@OUT[n] - Read digital output
@RND[n] - Round
@SIN[n] - Sine
@SQR[n] - Square Root
@TAN[n] - Tangent
@AN[n] - Read analog input
@ASIN[n] - Inverse sine
@ATAN[n] - Inverse tangent
@COM[n] - Bitwise complement
@COS[n] - Cosine

: @

```

All supported commands matching the substring @

- If there is only one matching command, then a detailed listing of the command's syntax is brought up with tab-tab. The detailed help will be found on the root command only (e.g. TP) and not on compounds (e.g. TPA).

```

Terminal

@SQR[n]

FUNCTION: Square Root

DESCRIPTION:
Takes the square root of the given
number.If the number is negative, the
absolute value is taken first.

ARGUMENTS: @SQR[n] where
n is a signed number in the range
-2147483648 to 2147483647.

USAGE:
While MovingYesDefault Value-

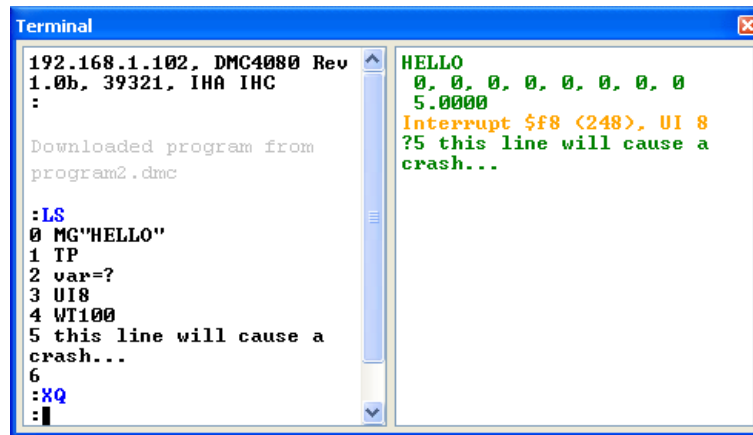
```

Type Q or Esc to exit command lookup

- Typing Alt+R will repeat the previously typed command.
- Alt+shift+R or choosing the Auto Repeat option from the right click menu will begin the auto repeat mode. The command will be continuously sent at the interval specified in the [advanced settings](#) file. Typing Esc will stop the auto mode.
- The command window right-click menu contains options to copy, paste, select all, clear and auto-repeat.

Unsolicited Messages and Interrupts

- Unsolicited messages from the controller are displayed in the window next to the command-and-response window. Unsolicited messages consist of:
 1. Messages from DMC code, e.g. MG"Hello"
 2. Command output from commands executed within DMC code, e.g. TP, var=?
 3. Interrupts, where available
 4. Error messages from the controller

A terminal window titled "Terminal" with a blue title bar. It is split into two panes by a vertical divider. The left pane shows command input: "192.168.1.102, DMC4080 Rev 1.0b, 39321, IHA IHC :", "Downloaded program from program2.dmc", ":LS", "0 MG\"HELLO\"", "1 IP", "2 var=?", "3 UI8", "4 WT100", "5 this line will cause a crash...", "6", ":XQ", and ":|". The right pane shows the output: "HELLO", "0, 0, 0, 0, 0, 0, 0", "5.0000", "Interrupt \$f8 <248>, UI 8", "?5 this line will cause a crash...", and "5.0000".

```
192.168.1.102, DMC4080 Rev
1.0b, 39321, IHA IHC
:

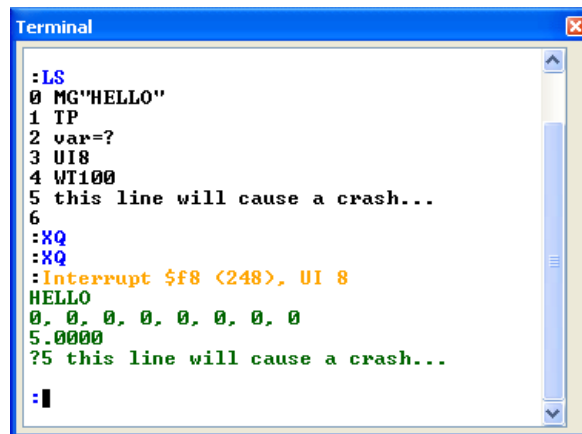
Downloaded program from
program2.dmc

:LS
0 MG"HELLO"
1 IP
2 var=?
3 UI8
4 WT100
5 this line will cause a
crash...
6
:XQ
:|
```

```
HELLO
0, 0, 0, 0, 0, 0, 0
5.0000
Interrupt $f8 <248>, UI 8
?5 this line will cause a
crash...
5.0000
```

Output from the DMC code displayed in the messages window (right)

- The divider between the windows can be dragged to change the area of the command-and-response area and the messages area. If the divider is moved to shrink the messages area to zero, messages will be ported to the command-and-response window
- The divider orientation, vertical or horizontal, is dependent upon docking within the GUI environment and will change automatically to provide optimum aspect ratio.
- The ordering of the windows can be swapped in the [advanced settings](#) file.

A terminal window titled "Terminal" with a blue title bar. The vertical divider has been moved to the right, effectively collapsing the message window. All the output from the previous window is now visible in the single command window pane.

```
:LS
0 MG"HELLO"
1 IP
2 var=?
3 UI8
4 WT100
5 this line will cause a crash...
6
:XQ
:XQ
:Interrupt $f8 <248>, UI 8
HELLO
0, 0, 0, 0, 0, 0, 0
5.0000
?5 this line will cause a crash...
5.0000
:|
```

Output from the DMC code is displayed in the command window if the message window is dragged to zero

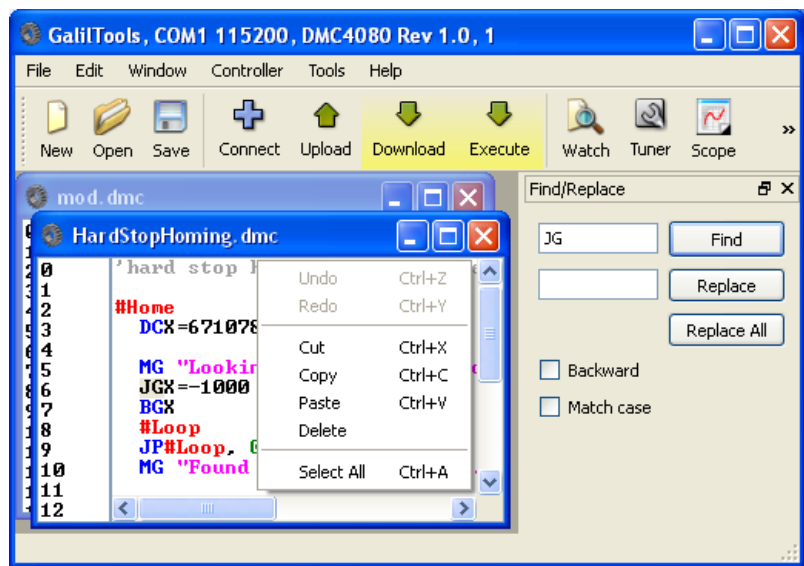
- The message window right-click menu contains options to copy, select all, and clear.
- Messages can be date-stamped by editing the [advanced settings](#) file.

[Table of Contents](#)

Editor

Overview

The program [Editor](#) allows the user to create (New), Open, edit, Save, Upload, Download, and Execute .dmc programs as well as Find/Replace within them.



The [GalilTools](#) multi-file [Editor](#) and Find/Replace utility

Hint: [Galil](#) programs are saved in text files with the extension ".dmc". Refer to your controller's [command reference](#) for valid commands and [user manual](#) for sample programs. Also see Galil's [sample dmc code library](#) for ready-to-use, downloadable code.

- The **File**, **Edit**, and **Window** menus and toolbar have [Editor](#)-related functions which operate on the currently-selected [Editor](#) window.
- **Upload** copies the program on the controller to an [Editor](#) window on the PC, and **Download** copies the selected program in GalilTools to the controller.
 - **Execute** downloads the program from the [Editor](#) and then runs it. [GalilTools](#) will stop (ST) any motion or program before downloading.
 - **New** opens a blank [Editor](#) window, **Open** provides a file selection dialog for opening a saved .dmc file, and **Save** allows saving of the currently-selected [Editor](#).

Any .dmc file opened by double clicking from the desktop or file explorer will open into an [Editor](#) window in [GalilTools](#) and the connection will be determined by the project state. (click [Connect](#) to change connection). If .dmc files are not associated with [GalilTools](#), right click on the .dmc file from the file explorer, choose "Open with...", and choose the [GalilTools](#) executable.

The [Editor](#) displays line numbers in the left column and the font size can be temporarily changed by holding down the Ctrl key and scrolling the mouse wheel and permanently changed by choosing *Edit | Font...*

Color Coding

If a line is over 80 characters, the overflow will be colored red and this designates lines that are too long to download to a controller without compression (40 characters for the RIO-471x0). The [Editor](#) syntax-highlights different text corresponding to its purpose according to the rules below:

Magenta	Text (e.g. MG)
Blue	Galil Commands
Black	Variables, Arrays , Numbers, Axes, and Operators (+, -, ...)
Red	Program Labels
Grey	Commented Text (' or NO)

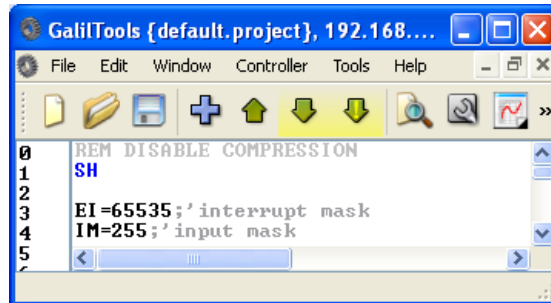
Light Grey	REM Text (not downloaded to controller)
Green	Operands (e.g. _TPA, TIME)

For very long programs which are larger than the controller's program space (e.g. 80 characters by 2000 lines on DMC-40x0), Download and Execute will attempt to compress the program before downloading. This is done by removing white space and concatenating multiple commands onto each line of code. A warning in the terminal and in a dialog will display if compression was necessary. If the user is actively debugging code, listing code (LS) or uploading compressed code to the editor allows the user to see the code as it appears on the controller. Note, compression does not modify the original source file.



A dialog or terminal message will display if file compression was necessary to fit the program on the controller

Compression can be disabled for a particular DMC file by including *REM DISABLE COMPRESSION* in line zero of the DMC code. If compression is disabled, but required to fit the code on the controller, an error will be thrown and the download will fail.



Disabling the compression option in DMC code.

If comments must be preserved, even in instances of compression, use "NO" instead of " " as the comment indicator.

[Table of Contents](#)

Watch

Overview

Watch allows viewing hundreds of Boolean, integer, and analog controller parameters (Sources) such as I/O, position, and torque (all Sources are also available in the [Scope](#)). Boolean values are colored uniquely: a 1 is **green** and a 0 is **red**. By double-clicking any of the digital outputs (@OUT), the output point can be toggled. The [Watch](#) window is divided into the tabs All and Some, which both contain dynamically updated status from the controller data record (QR).

Hint: Use the refresh button (F5) in the toolbar to refresh the connection if you change the configuration of AQ,DQ,TM,CO, and BA.

All

	Source	Value	Units	Description	Scale	Offset
1	@AN[0]	0	V	Analog input 0	6552	0
2	@AN[1]	0	Newtons	Force Sensor	600000	0
3	@AN[2]	0	Fahrenheit	Thermo Couple	6000	0
4	@AN[3]	0.0037	V	Analog input 3	6552	0
5	@AN[4]	0	V	Analog input 4	6552	0
6	@AN[5]	0	V	Analog input 5	6552	0
7	@AN[6]	0.0073	V	Analog input 6	6552	0
8	@AN[7]	0.0024	V	Analog input 7	6552	0
9	@AO[0]	0	V	Analog output 0	13107	0
10	@AO[1]	0	V	Analog output 1	13107	0
11	@AO[2]	0	V	Analog output 2	13107	0
12	@AO[3]	0	V	Analog output 3	13107	0
13	@AO[4]	0	V	Analog output 4	13107	0
14	@AO[5]	0	V	Analog output 5	13107	0
15	@AO[6]	0	V	Analog output 6	13107	0
16	@AO[7]	0	V	Analog output 7	13107	0
17	@IN[00]	1	Boolean	Digital input 0	1	0
18	@IN[01]	1	Active	User Prox Switch	1	0
19	@IN[02]	1	10x	Feed Rate Increase	1	0
20	@IN[03]	1	Alert True	Machine Alert	1	0
21	@IN[04]	1	Pause	User Pause	1	0

The data record [Watch](#) window. The All tab displays the entire controller Data Record, and allows access to customizing the data record to suit an application

On the All tab, each column can be sorted by clicking on the title header. For example, sorting by Description groups axis data together (_TPA, _RPA) and sorting by Source groups similar data together (_TPA, _TPB). When keyboard focus is within the Source column, entries can be looked up by typing the first few characters of their Source name.

Some

To monitor a subset of the All tab, select the desired Sources under the Some tab.

	Source	Value	Units
1	@AN[1] Force Sensor	0	Newtons
2	@AN[2] Thermo Couple	0	Fahrenheit
3	@IN[01] User Prox Switch	1	Active
4	@IN[06] Laser Pulse	1	PulseTrueFalse
5	@IN[05] Press Engage	1	EngageNow
6	@IN[02] Feed Rate Increase	1	10x
7	@AN[3] Analog input 3	0.0122	V
8	TIME Sample counter	7770	samples
9	▼		

Monitoring a subset of the data record with [Watch Some](#)

Clicking in a row's Source cell provides access to a drop down list of all data record Sources. Choose one from the last row to create a new row. Right clicking in the value cell of any row allows that row to be Removed or another row to be Inserted.

The rows selected in the Watch Some are remembered in the GalilTools Project for the next launch. The active tab, All or Some, is also remember along with window position, size, etc.

User Units, Description, Scale and Offset

Watch ALL is a listing of the controller's data record. GalilTools represents the data record as "sources". Sources are used in both the Watch and the Scope to provide information to the user regarding the controllers current state. A source has the following components.

- Source - The static Galil name for the data, e.g. @IN[01]
- Value - The current value of the source from the controller, post-processed by Scale and Offset
- Units - The engineering units of the source. Galil recommends a units value, but the user can specify this by typing a new unit string in the units cell
- Description - The human-readable description of the source. Galil recommends a description, but the user can change it by typing a new description string in the description cell
- Scale - Used to scale the value returned from the controller. The scale is in effect a divisor. Raw values are divided by the scale value to provide post-processing for user display. A new value can be set in the value cell to provide application-specific scaling.
- Offset - The offset bias to add to the raw value from the controller to provide application specific post-processing of data

Scale and Offset operate on the controller's raw data record in the form of:

$$\text{Source Value} = \frac{\text{Raw Value}}{\text{Scale}} + \text{Offset}$$

If Units, Description, Scale, or Offset are changed, these changes are also reflected in Watch Some and the Scope operation. Changes are persisted by the Galil Project through launches. To create a factory default version of the Galil data record sources, create a new project.

[Table of Contents](#)

Graphical Scope (Full Edition Only)

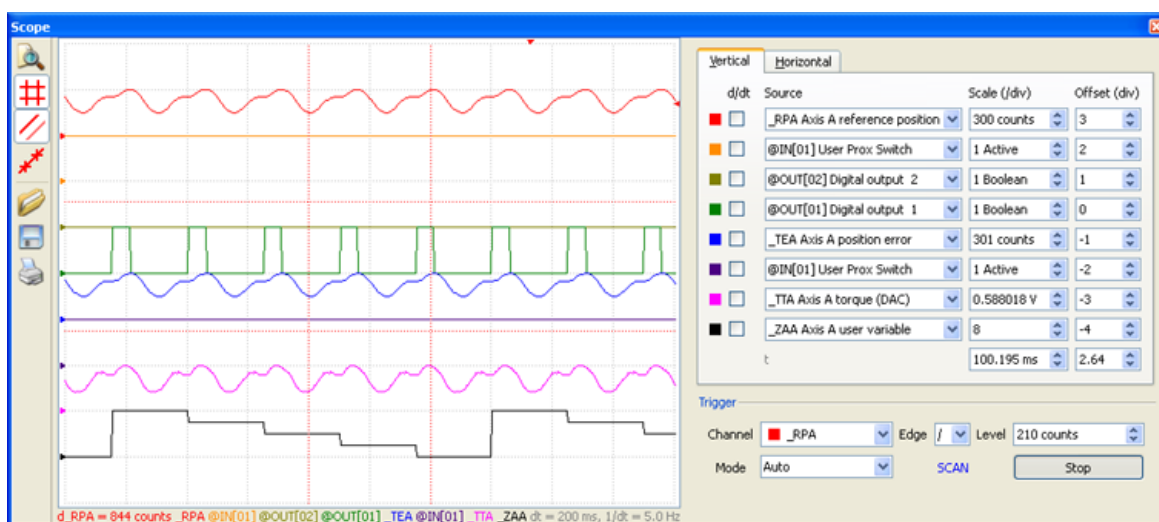
Overview

The 8-channel [Scope](#) provides real-time visual feedback from the controller in the form of an [oscilloscope](#). Any data Source from [Watch](#) may be chosen to view on the [Scope](#) and up to 8 different Sources can be displayed simultaneously. This not only allows the user to see analog values such as torque or velocity, but also Boolean values such as digital inputs and outputs (set Scale near 1 when scoping Booleans).

There are many configuration options for the Scope tool (outlined below), and the Galil Project will automatically remember the settings from one GalilTools launch to the next.

Hint: One very important distinction is that although only up to 8 channels can be displayed on the [Scope](#) at one time, all of the data record (typically hundreds of Sources) is recorded. This means that for a particular capture, if a desired Source is not present on the [Scope](#), the user need only select that Source for one of the channels and the recorded data will then be displayed (see critically damped example below).

Hint: Use the refresh button (F5) in the toolbar to refresh the connection if you change the configuration of AQ,DQ,TM,CO, and BA.



8 Channel Scope

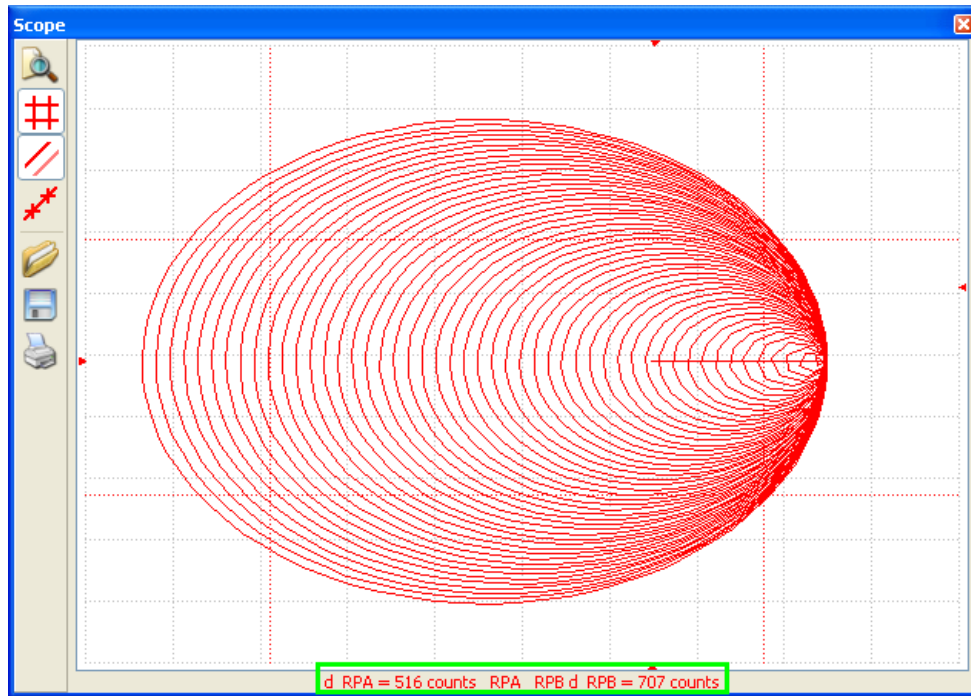
The Scope is divided into two sections: the toolbar, grid and traces on the left and the Sources, Scales, Offsets, and Trigger condition on the right. The distance between two grid lines is a division (div) and a channel's Scale represents the range of one division. The Offset arrows on the left represent the channels' zero ("ground") locations and can be adjusted to overlay or stack traces. The Scales and Offsets can be adjusted to provide the desired zoom and pan.

Vertical Tab

In normal operation, the 8 Sources are graphed against time. In this mode, the Horizontal tab is unused, and the horizontal domain is always time, scaled by a single "t" combo-box at the bottom of the sources list. Time can also be offset with the single time Offset combo box to the right of the time scale. All sources use the same time characteristics.

Horizontal Tab, Four Channel "X-Y" Mode

To graph a source against another source, the horizontal tab is employed. Changing the horizontal domain allows for up to 4 source-source graphs which are useful for X-Y cartesian planes such as stages, and for two dimensional phase diagrams such as dual-axis error monitoring.



The GalilTools scope allow X-Y plots. Here, the reference position of axis A is graphed verses axis B

Note, by adjusting the time scale in the vertical tab, the length of time that is plotted can be changed. As for the vertical source, scale and offset of the horizontal source can be adjusted, under the horizontal tab.

Data Analysis

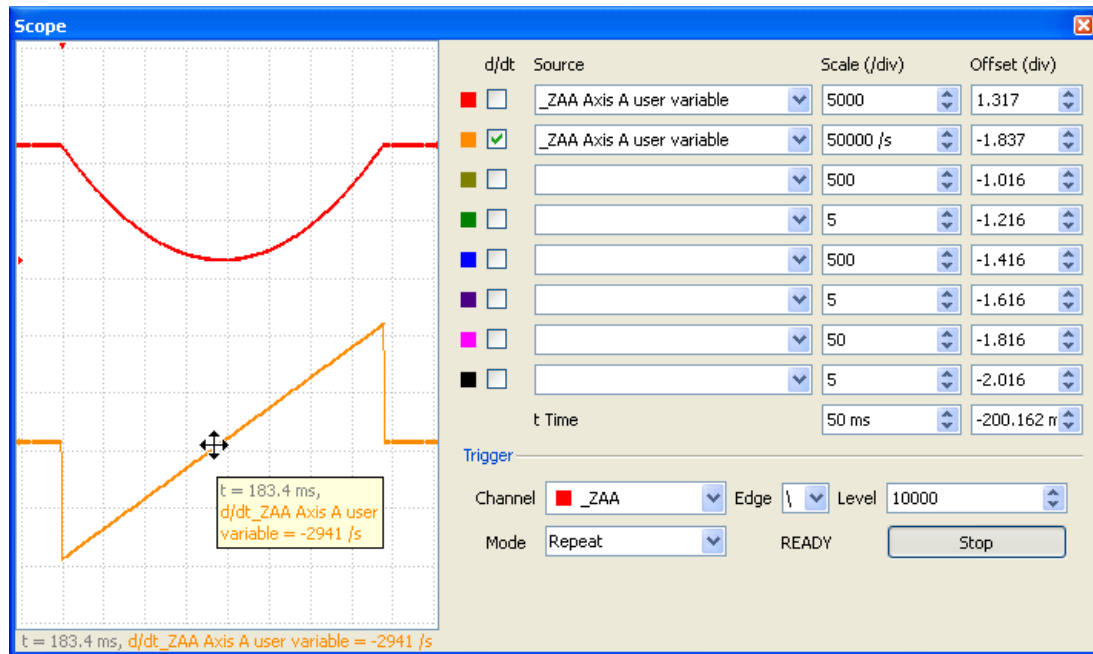
The scope contains a wide variety of analytical tools. Tools can be found in the left-side toolbar, the right click menu in the scope grid, and in the Vertical tab. The vertical-horizontal data entry section and the toolbar can be made smaller and/or hidden by dragging the horizontal boundaries of the scope grid. The X-Y plot above shows the horizontal-vertical tabs hidden via this method to increase the rendered scope area.

Derivative (d/dt)

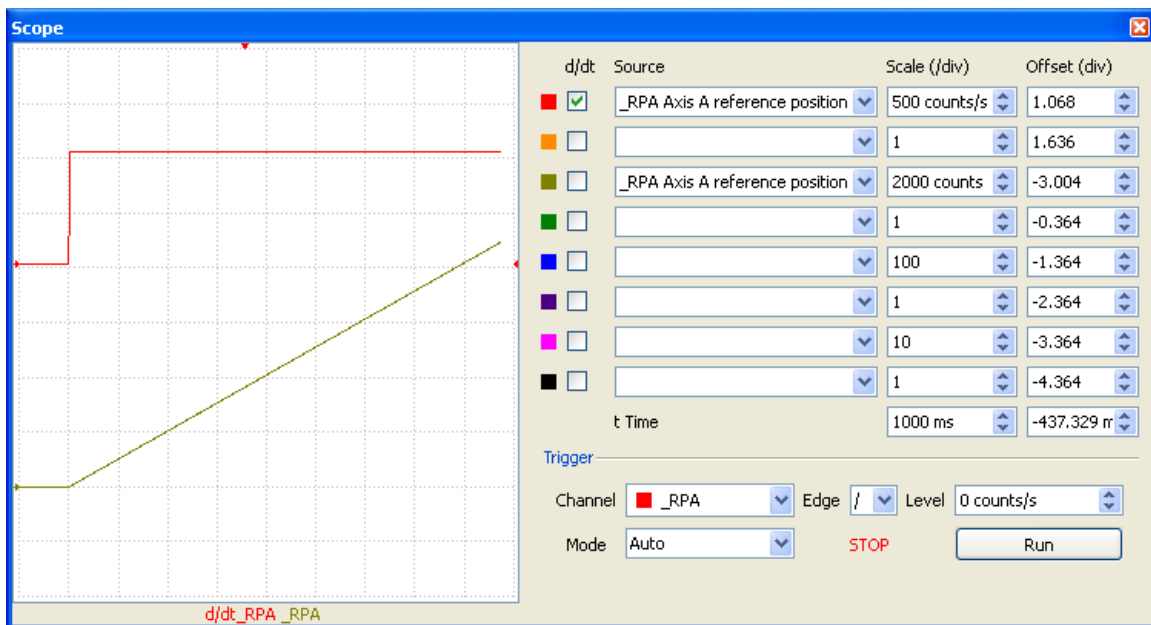
By selecting the d/dt checkbox for a source, the discrete derivative is performed. This is a one dimensional derivative of the form:

$$T*(f(t) - f(t-1))$$

Where T is sampling frequency in Hertz and t is the sample.

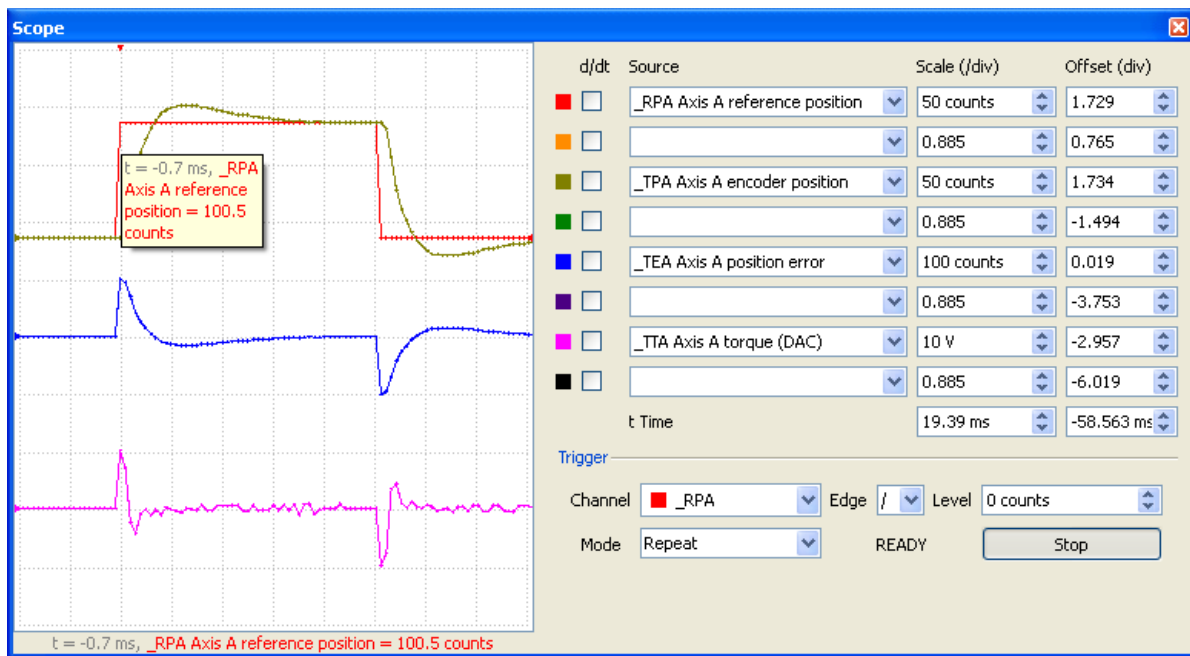


The user variable ZAA forms a parabolic form over time [$f(x)=x^2$]. The derivative of this is a sloped line. [$f'(x)=2x$]. Note, the derivative is in units per second.



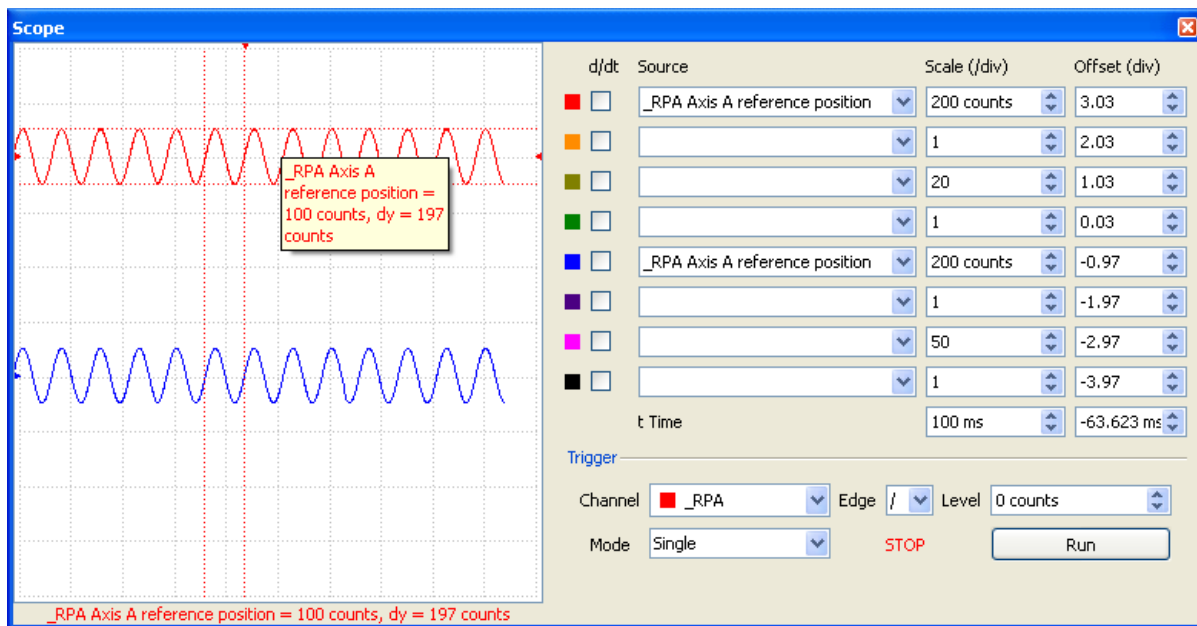
A jog (JG) move of 1024 c/s. The derivative shows a constant speed.

Viewing Data: Status Bar and Tooltip



Mousing over a cursor, a scope trace or an overlay trace will display pertinent data in the scope status bar and a popup tooltip.

Cursors



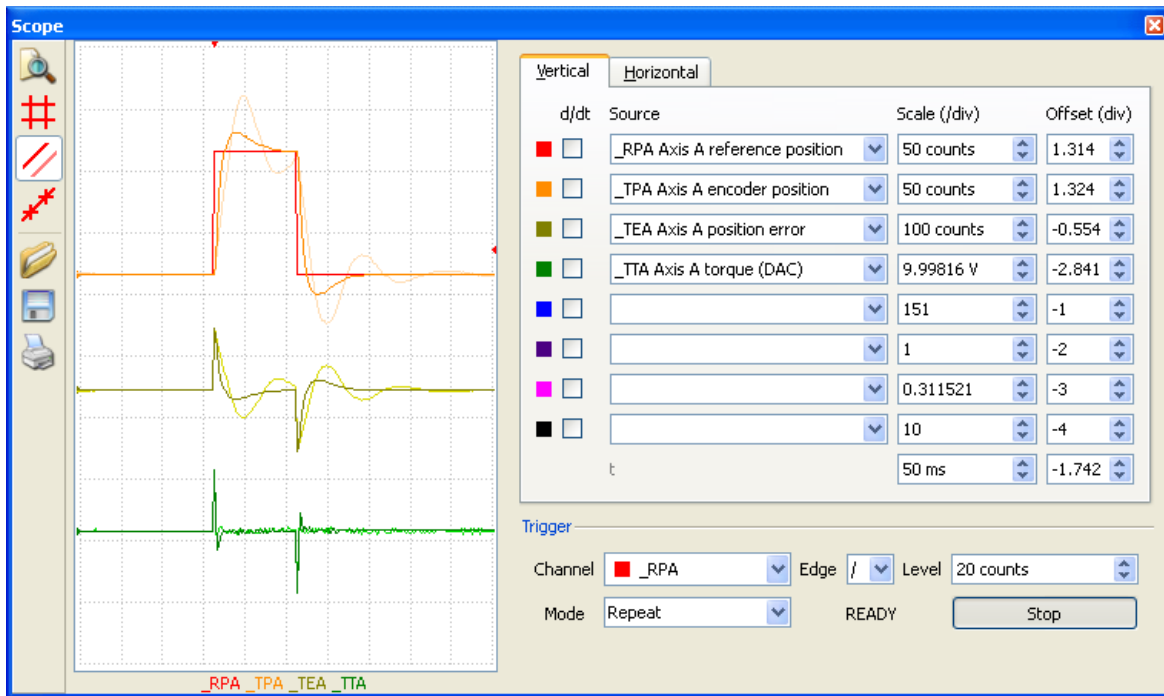
Mousing over a cursor displays cursor-specific data in a tooltip and the statusbar.

The cursor references the active trace, selected by left clicking the trace in the scope field. The cursor color will reflect the active trace color. Trace location and delta values between traces are displayed in the status bar and tooltip.

Move cursors by clicking and dragging.

Overlay

Scope overlay allows for traces from a previous scope capture to be compared to current waveforms. By clicking the overlay icon in the toolbar, the current scope traces are rendered in a lighter color in the scope background. Clicking the icon again will turn off the overlay, and the older data is lost.



Older data can be overlayed to compare with subsequent scope data.

Store, Recall data

The **Save** icon in the toolbar exports the current scope data to a csv file.

The screenshot shows a Microsoft Excel spreadsheet titled "RIO.Scope.csv". The data is organized into columns for time and various analog inputs.

	A	B	C	D	E
	t Time (ms)	@AN[0] Analog in	@AN[1] Analog in	@AN[2] Analog	@AN[3] Analog input 3 (V)
232	414.062	0.001221	0	0	0.00976801
233	416.016	0	0.002442	0.001221	0.00854701
234	417.969	0.001221	0.001221	0	0.00732601
235	419.922	0.002442	0.002442	0.002442	0.00976801
236	421.875	0.001221	0	0.001221	0.00732601
237	423.828	0	0	0	0.002442
238	425.781	0.002442	0.002442	0.001221	0.00732601
239	427.734	0.001221	0	0	0.010989
240	429.688	0.001221	0	0	0.00854701
241	431.641	0.001221	0	0	0.010989
242	433.594	0.001221	0	0	0.00854701
243	435.547	0.001221	0	0	0.001221
244	437.5	0	0	0	0.00854701
245	439.453	0	0	0	0.00976801

Scope data can be saved to CSV from the toolbar.
CSV can be easily imported into many third party softwares.

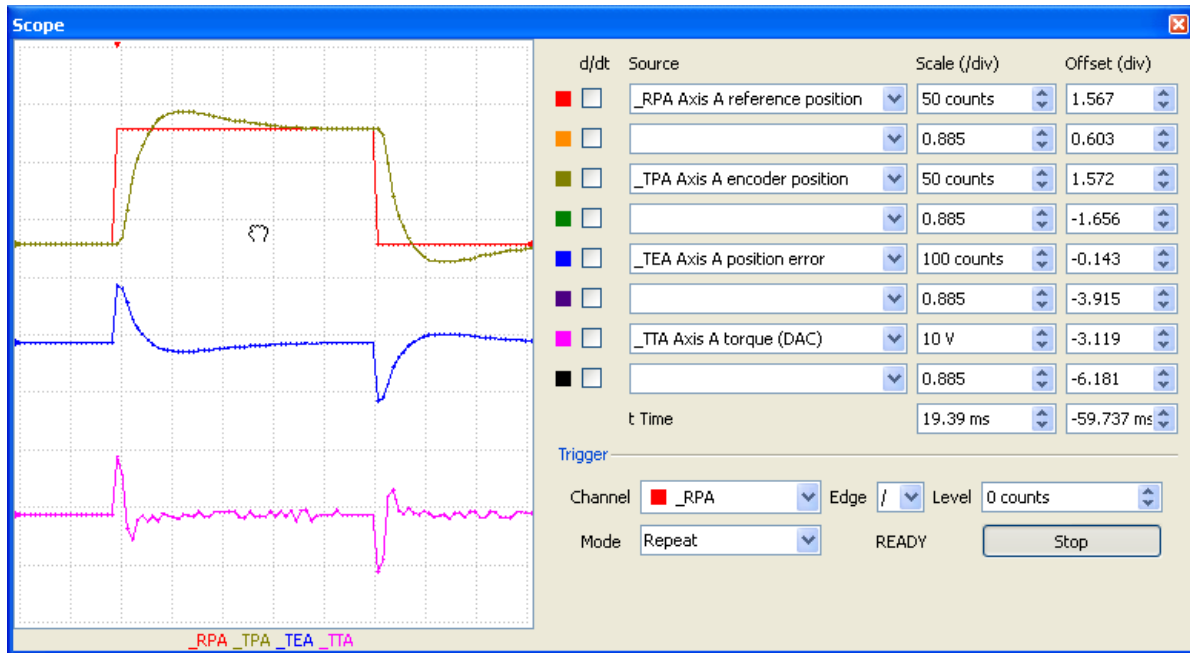
The **Open** icon opens scope data into the scope as overlay data (background). This allows for previously saved data to be analyzed with the Scope's analytical tools and allows for comparison to current waveforms (foreground).

Print

Clicking the toolbar print button will send the scope window to a printer.

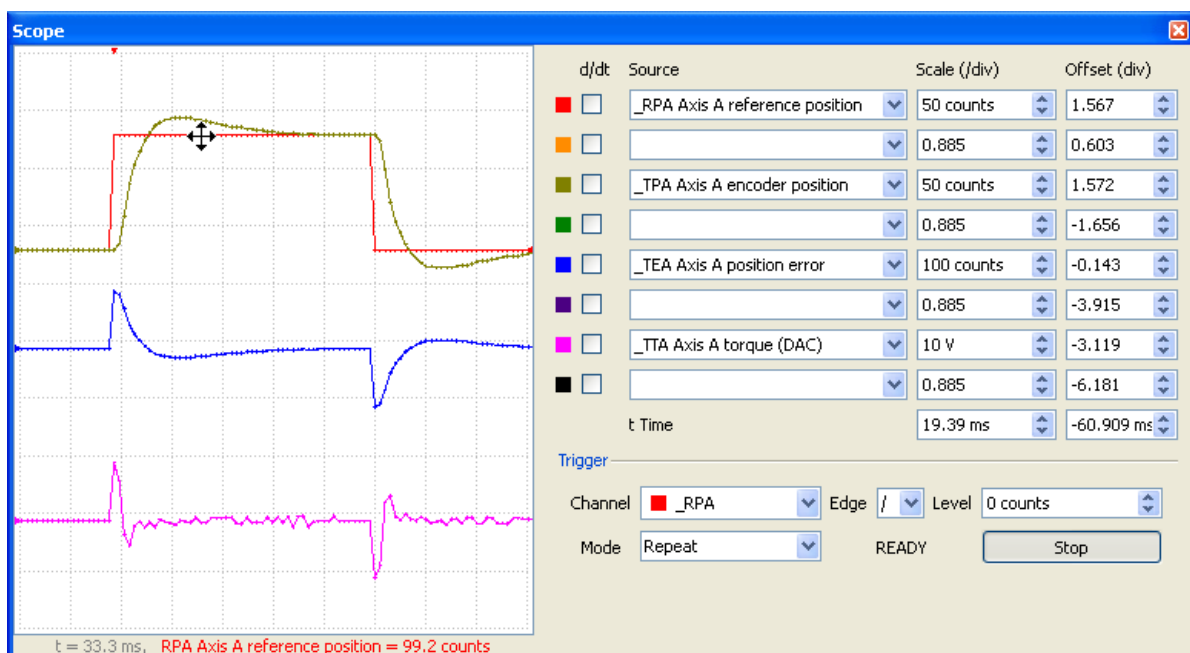
Scope View

Panning view



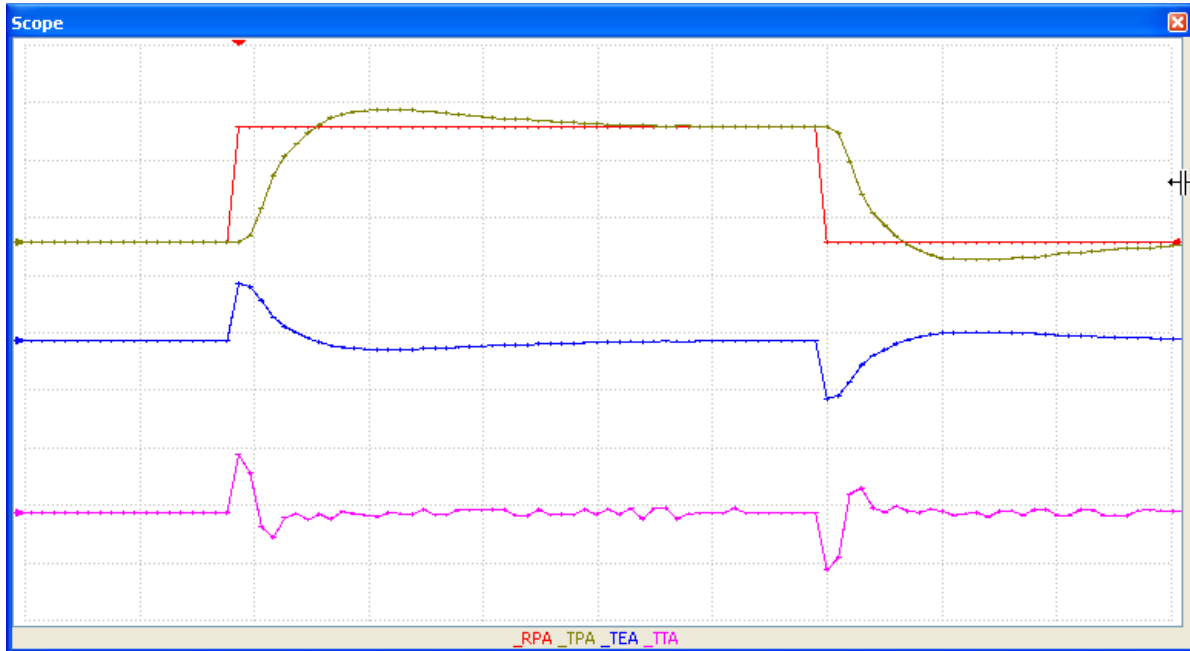
Left clicking on the scope background allows two dimensional pan.

Move trace verticle



Left clicking on a trace allows two dimensional pan: trace-only vertical, all traces horizontal.

Hide Controls



Clicking and dragging the toolbar's or control panel's scope boundary allows the scope grid to fill a larger section of the full scope window.

Zooming

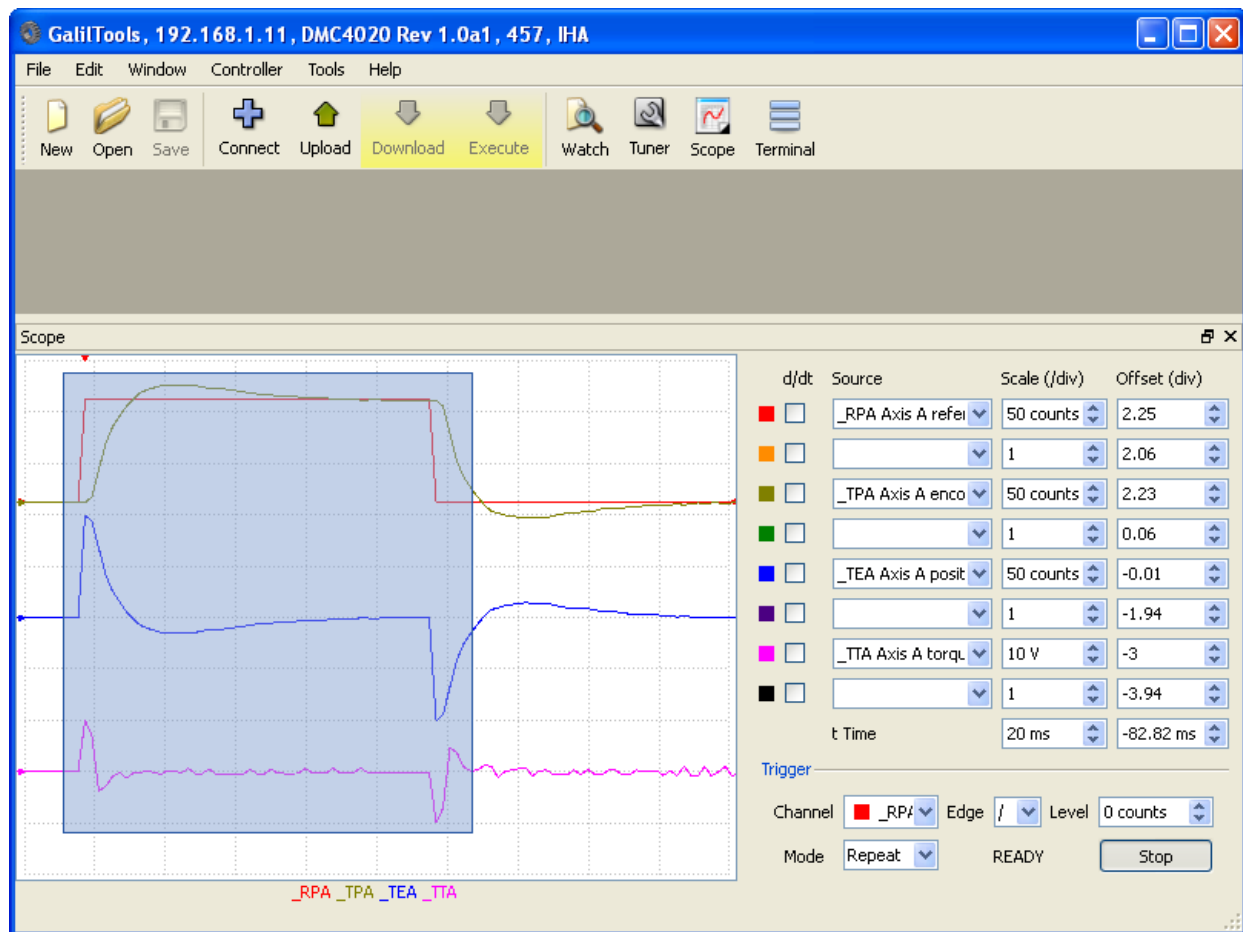
Scroll Wheel All (Zoom Camera)

By spinning the mouse scroll wheel when the mouse is over the scope background, all traces will be zoomed in or out. Both vertical scales and horizontal scales will be adjusted so that the zoom appears to be moving the viewer's perspective in or out (similar to moving the view camera in or out). Rolling the mouse wheel forward, towards the display, will zoom the view in. Moving away from the screen will zoom the view out.

Scroll Wheel One (Zoom Trace)

When over a trace, the scrollwheel will zoom just that trace's scale. The time scaling is unmodified. This is useful for resizing one trace relative to another. Rolling forward zooms in, rolling away zooms out.

Zoom All Box



Ctrl + Left-Click allows a box region to be defined which will be zoomed upon click release. **Esc** escapes the box without zooming.

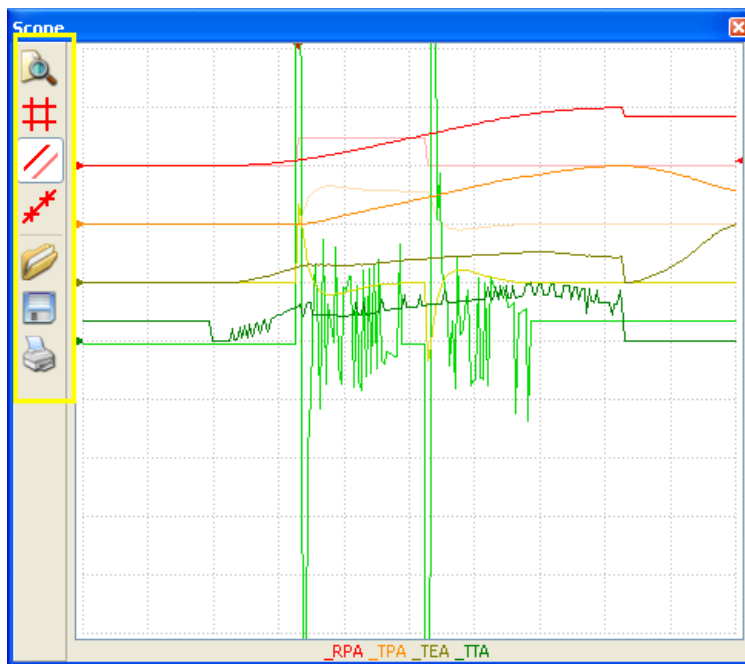
Multiple zoom box operations can be undone with **Ctrl + Right-Click**.

Zoom-box operations are also in the right click scope grid menu.

Autoscale

Clicking the Autoscale icon in the toolbar will put the scope in Autoscale mode. In this mode, scales and offsets will be adjusted dynamically to keep the data on the scope, in-boundary and stacked. Click again to disable.

Toolbar



The Toolbar provides several scope-related options.

Icon (top to bottom)	Description
Autoscale	Dynamically keeps waveforms onscreen
Cursors	Turns on or off the cursors
Overlay	Turns on or off the Overlay feature
Points	Displays actual points used to render the waveforms
Open	Open dataset from CSV. Waveforms are rendered as overlays, in background.
Save	Saves current scope data to CSV. Saves data for all sources, not just displayed.
Print	Print the Scope Window

Triggering

The Trigger menu at the bottom of the vertical tab can be applied to one of the 8 [Scope](#) channels on either a Rising or Falling Edge and allows the user to choose between three types of scanning modes common in oscilloscopes: Auto, Repeat, or Single. Auto will constantly display live data in the [Scope](#) window (useful to set the Trigger Level). Repeat will draw only after a Trigger and align the Trigger edge with the horizontal-offset arrow at the top (useful to catch the *last* of a series of triggers). Single will only capture the first triggered edge and will not refresh until the user clicks the Run button again (useful to catch the *first* of a series of triggers).

The scope has a status label to the left of the Run/Stop button which relays relevant information regarding trigger status. The following table describes the various states.

Status	Meaning
STOP	Scope is stopped. If data has been captured, the data will stay drawn and static. Changing the Source in a channel will draw that data from memory.
ARMING...	The Scope is preparing to arm for Single or Repeat mode. No triggers will be detected. Data left of the Time offset is being buffered.
READY	The Scope is ready to detect a trigger condition.
TRIGGERED	The Scope has detected a trigger condition and is gathering data to fill the scope space right of the Time offset.
SCAN	When the Mode is Auto and the Time Scale is $\geq 100\text{ms}$, the Scope is in Scan mode. Data is drawn continuously and triggers are ignored.

Communication

For the following [Ethernet](#) and [PCI](#) controllers, the [Scope](#) sample period can be set with the DR command from the [Terminal](#) (see your controller's [command reference](#)): DMC-18x6, 21x3, 40x0, and RIO-471x0. GalilTools will automatically choose the fastest sampling period (DR) possible. For the DMC-18x2 QR is polled as quickly as possible, and the resulting sample rate may be uneven and depends on the controller and PC workload. For stand-alone controllers connected over RS-232, the scope uses the QR mode at a much slower sampling

rate than DR over Ethernet. **For increased scope performance on standalone controllers, Ethernet is recommended.**

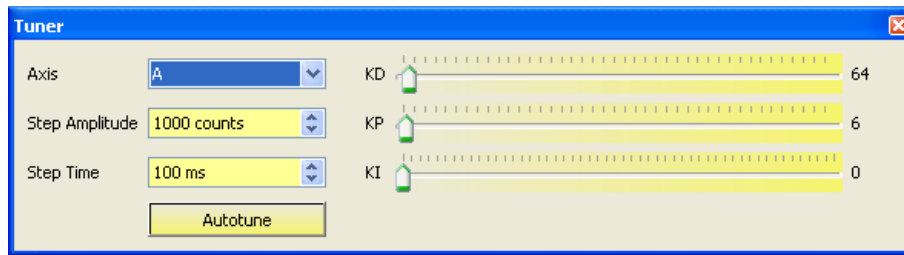
[Table of Contents](#)

Servo Tuner (Full Edition Only)

Overview

Autotune

The Autotune button automatically chooses optimal [PID](#) values by putting the selected Axis through a series of predetermined movements (which may cause the motor to buzz) and finally shows a Step response of the specified Amplitude and Time. Autotune can provide accurate results; however, manual tuning is often also required.

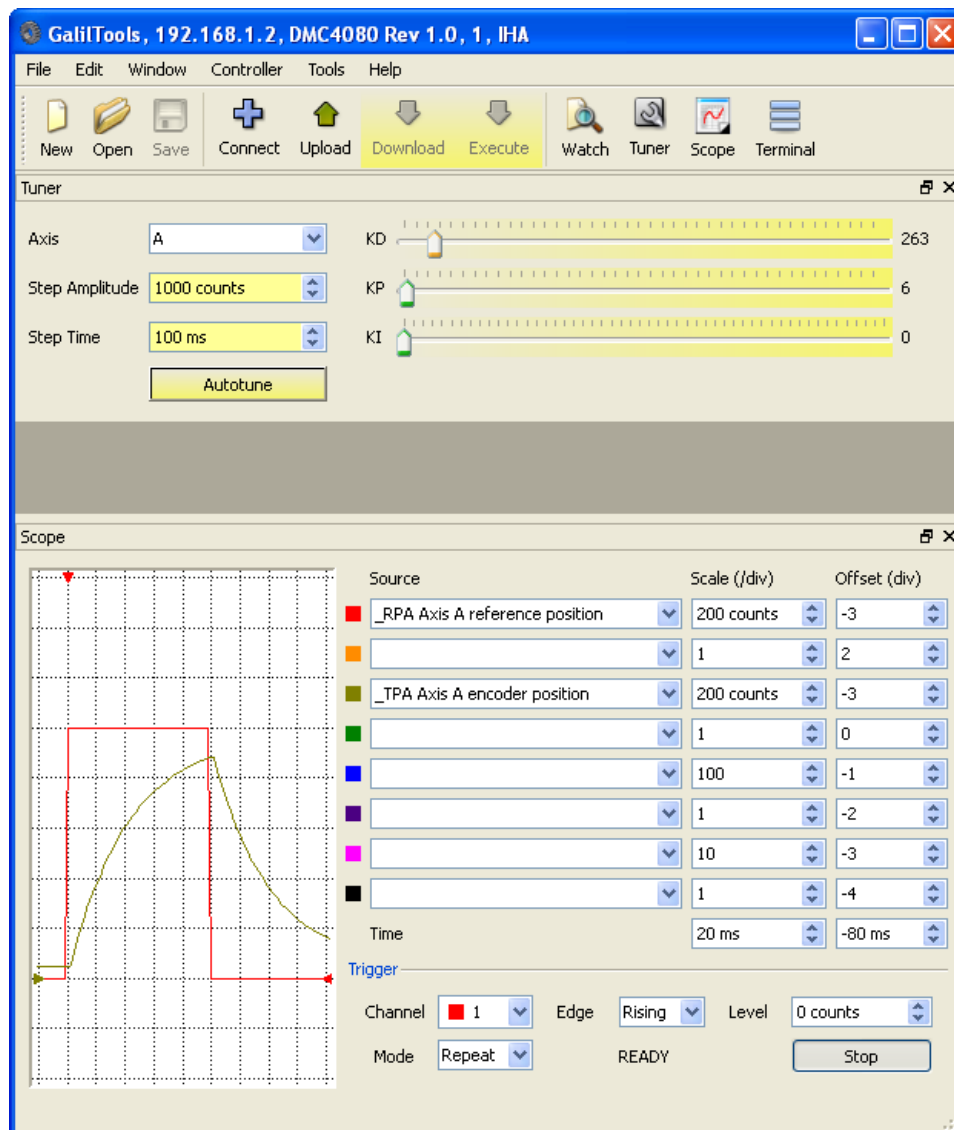


Servo Tuning Utility

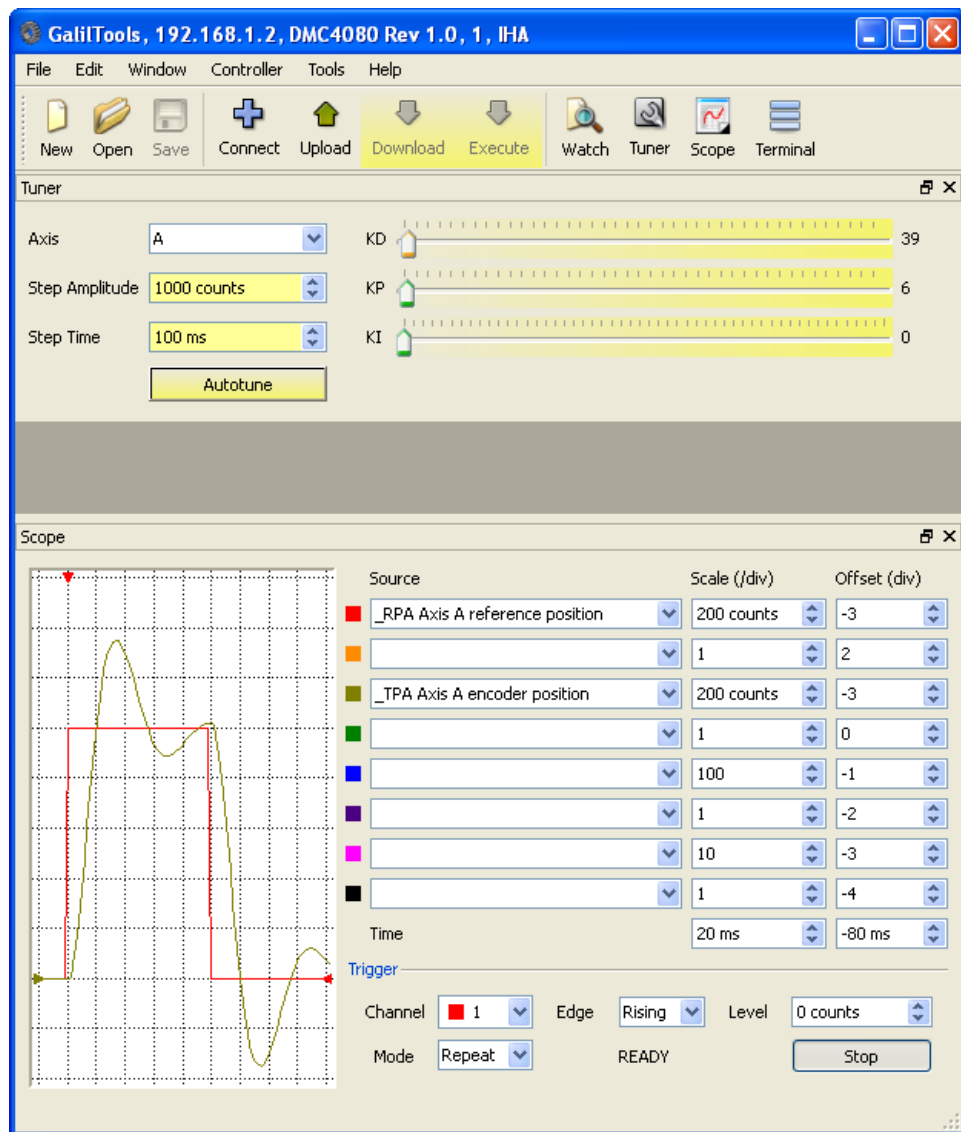
Manual

Manual tuning can be accomplished on the selected Axis by clicking on the [PID](#) sliders, which causes a step response whose Amplitude and Time (pulse width) can be adjusted to suit mechanics. If Step Amplitude is set to zero or a program is running in thread 0, the sliders will adjust the [PID](#) values without performing a step response, allowing a user program to be run. To manually tune an Axis, place the [Scope](#) in Repeat Mode, select desired [Scope](#) Sources, and slide the [PID](#) sliders to see the change in response.

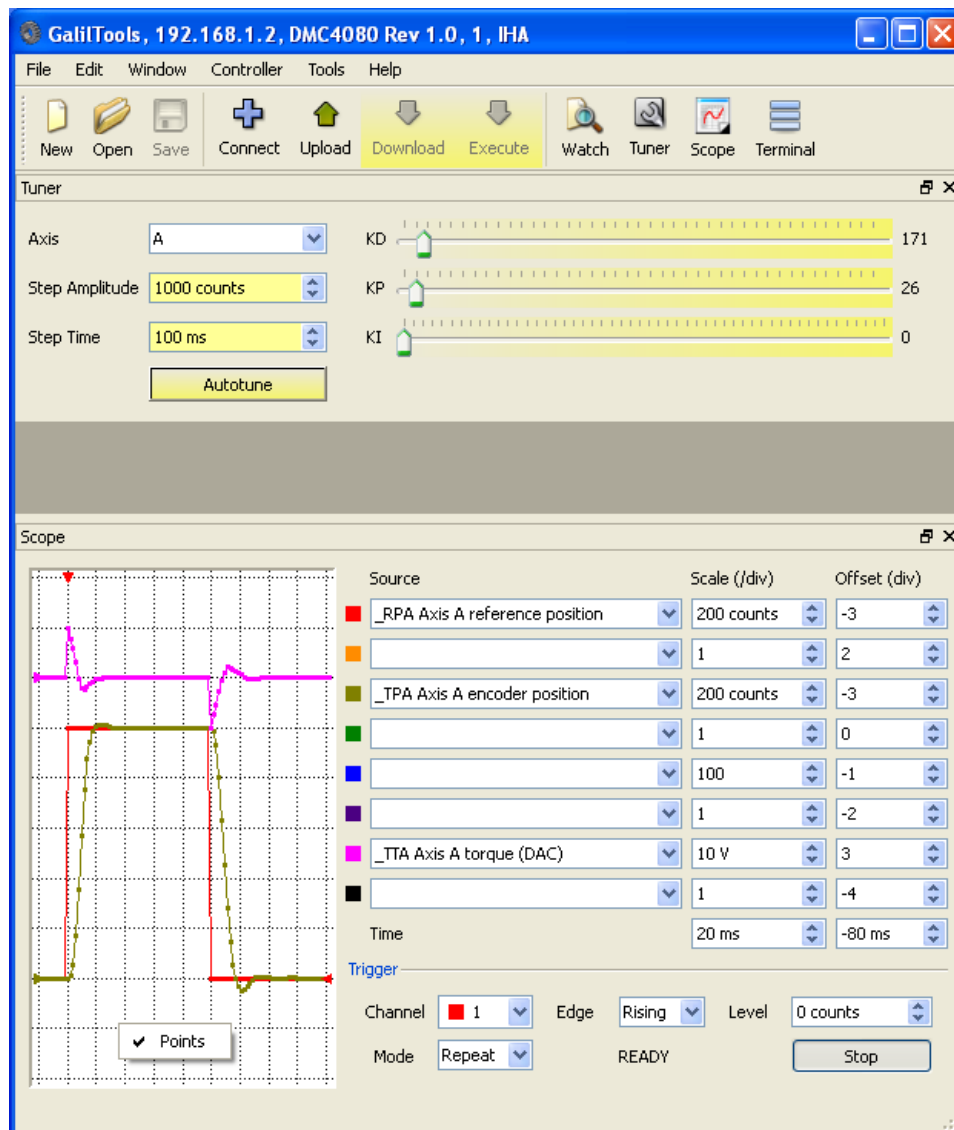
The following are screenshots of using the [Scope](#) and [Tuner](#) to manually tune a servo.



Overdamped System (poorly tuned)



An underdamped system with ringing (poorly tuned)



A critically damped, well tuned, system with minimal overshoot. Note the _TTA channel was added *after* the step response generation (all [Scope](#) Sources are recorded and available for later analysis).

[Table of Contents](#)

Debugging

Commands

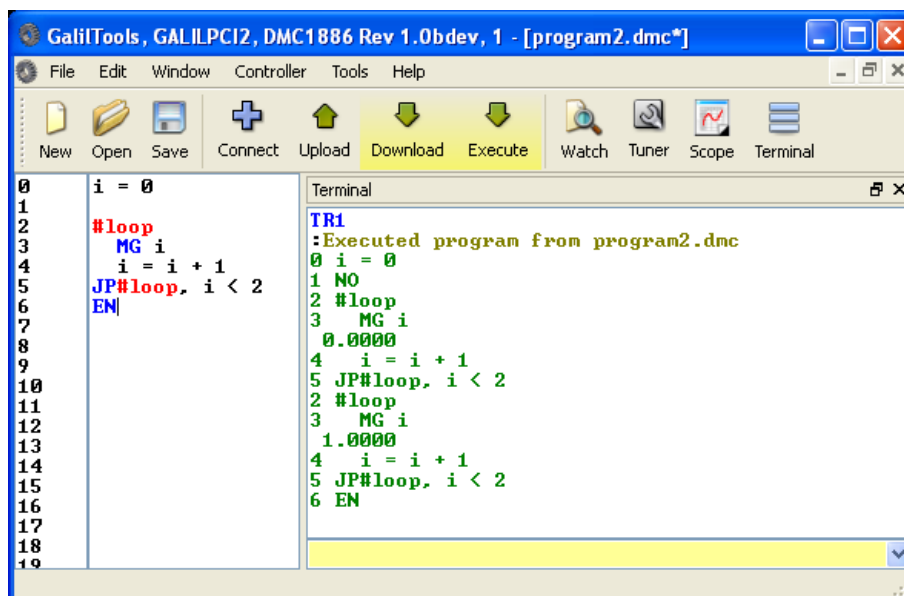
Debugging [Galil](#) code with [GalilTools](#) is accomplished with a few basic commands:

Command	Description
TR1	Trace. Prints each command as it's executed.
MG	Print message strings, variables, and controller parameters.
MG_XQn	Print current line number that is executing in thread n (or -1 if thread is not running)
LV	List all declared user variables and their values
LS	List the controller program
LS first line, last line	List a program section
LS_XQn, XQn	List current running line of code
QU array[]	Print the contents of a given array
SC	Stop code for axis
TC1	Tell error code and brief description (reason for controller returning a "?")
XQ	Begin Program
ST	Stop Program
BK line	Set breakpoint at line
BK	Resume from breakpoint
SL	Single Step

Messages (MG) can be placed in key points or subroutines to determine whether the program has executed that particular line. Stop codes (SC) will return the cause of a stopped axis and are helpful in determining motion issues. TC1 will return any error codes generated in the controller program; however, [GalilTools](#) will automatically return a **COMMAND ERROR in red** if one is generated from the [Terminal](#). Once [GalilTools](#) displays the error code, it will be cleared from controller memory (TC returns 0). When the error code appears in the [Terminal](#) window, please refer to [application note 2443](#) for further information.

Trace (TR)

A useful debugging tool is the trace (TR) function. By enabling trace, all commands in the controller program will be printed as they are executed, allowing for program flow analysis and simple debugging. The following shows a simple program and the [Terminal](#) output of the trace function (note that each program line is preceded by its line number):



The screenshot shows the GalilTools interface for a GALILPC12, DMC1886 Rev 1.0bdev, 1 - [program2.dmc*]. The main window displays a program with the following code:

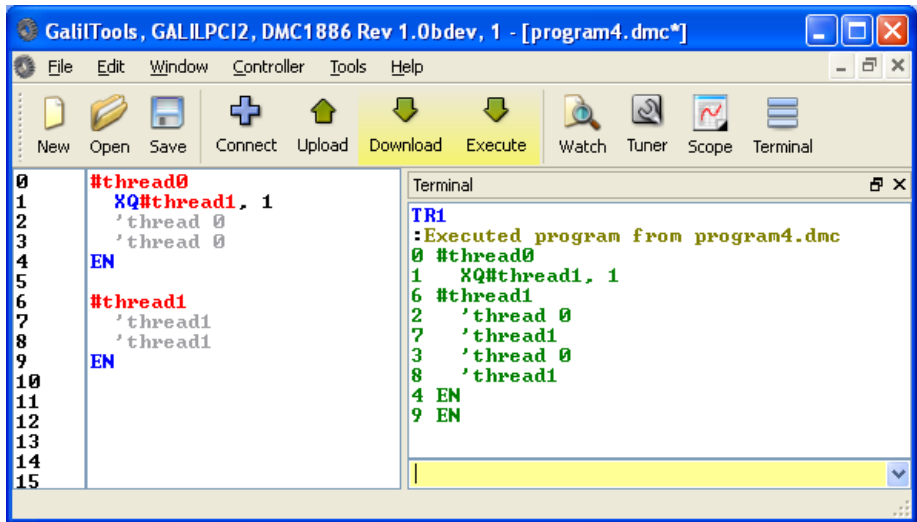
```
0 i = 0
1
2 #loop
3 MG i
4 i = i + 1
5 JP#loop, i < 2
6 EN
7
8
9
10
11
12
13
14
15
16
17
18
19
```

The Terminal window shows the trace output:

```
TR1
:Executed program from program2.dmc
0 i = 0
1 NO
2 #loop
3 MG i
0.0000
4 i = i + 1
5 JP#loop, i < 2
2 #loop
3 MG i
1.0000
4 i = i + 1
5 JP#loop, i < 2
6 EN
```

Example of the trace function

If multiple [threads](#) are executed with the TR command enabled, the user will see the executing lines interleaved (e.g. [thread](#) 0 line 0, [thread](#) 1 line 0, [thread](#) 0 line 1, [thread](#) 1 line 1, etc.). Note: comments preceded with a single quote ' or NO are evaluated as no operation commands.

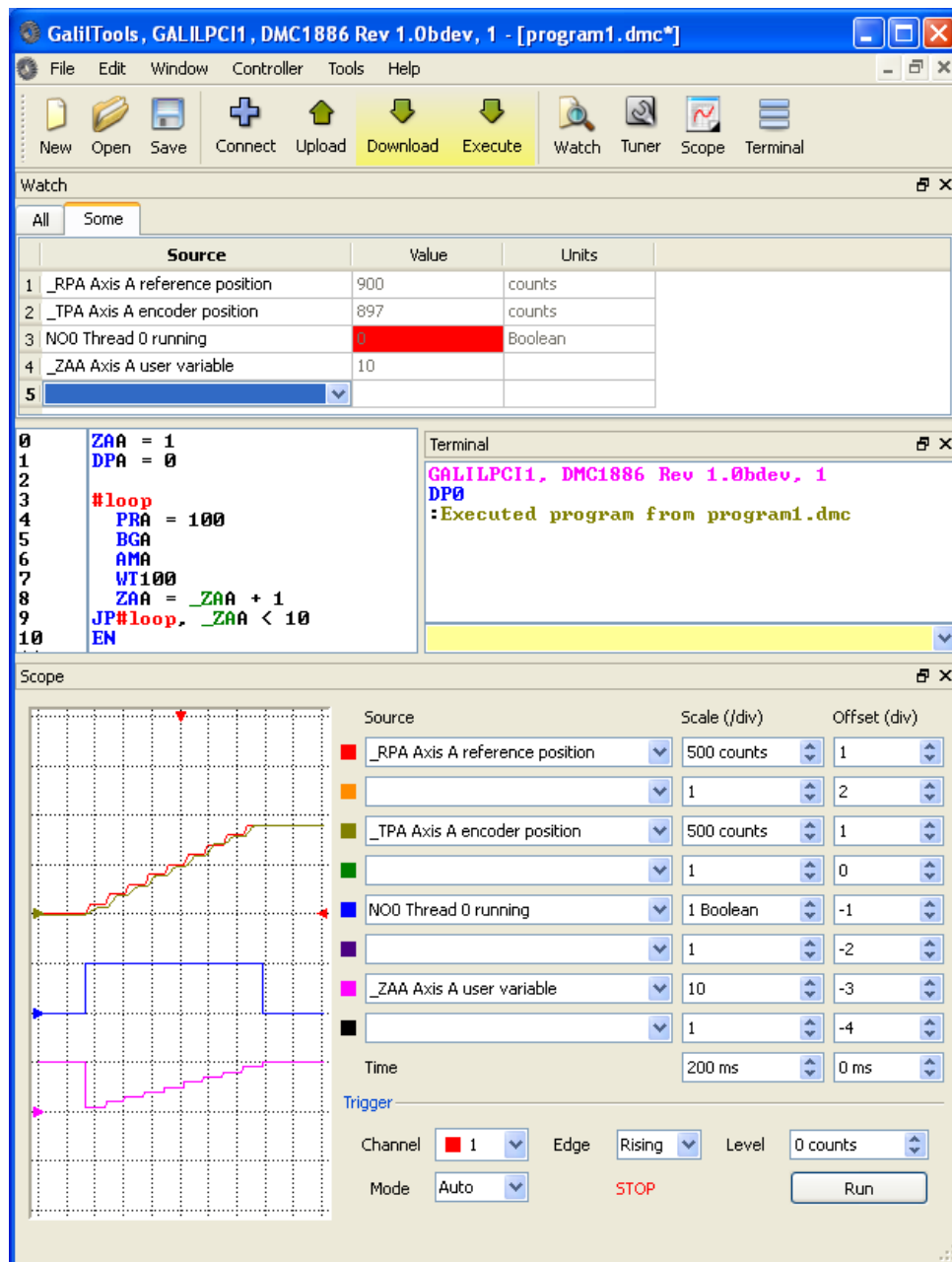


Executing [thread](#) lines are interleaved in the trace output

The Accelera series controllers have a [thread](#)-selectable trace function, which allows only executing lines of specified [threads](#) to print (see TR in your Accelera [command reference](#)).

Breakpoint (BK) and Single Step (SL)

[Breakpoint](#) (BK) and Single Step (SL) allow the user to control the flow of an executing program. Below is a [GalilTools](#) setup to debug some example code. In this example, the motion is supposed to profile 1000 counts, but a bug in the exit condition of the iterative loop has caused a "fencepost" error and only 900 counts are profiling.



Debugging Example. Target stops 100 counts before target.

If it were unclear why only 9 movements have been made, a [breakpoint](#) could be set at BGA on line 5 with BK5 to break at each individual move and turning on TR gives further information.

```

Terminal
Downloaded program from program2.dmc
TR1
: BK5
: XQ
: 0 ZAA = 1
1 DPA = 0
2 NO
3 #loop
4 PRA = 100
SL
: 5 BGA
MG _RPA, _ZAA
100.0000 1.0000
:

```

After issuing BK5 and XQ, the execution pauses before line 5. SL single steps into line 5 and interrogating _RPA and counter variable _ZAA shows all is in order so far.

```
Terminal
: BK5
:6  AMA
7  WT100
8  ZAA = _ZAA + 1
9  JP#loop, _ZAA < 10
3  #loop
4  PRA = 100
SL
:5  BGA
MG _RPA, _ZAA
200.0000 2.0000
:
```

Issuing BK5 again allows the execution to continue through one loop iteration and pauses again at line 5. Single stepping and interrogating shows as expected.

```
Terminal
BK5
:6  AMA
7  WT100
8  ZAA = _ZAA + 1
9  JP#loop, _ZAA < 10
3  #loop
4  PRA = 100
SL
:5  BGA
MG _RPA, _ZAA
900.0000 9.0000
:
```

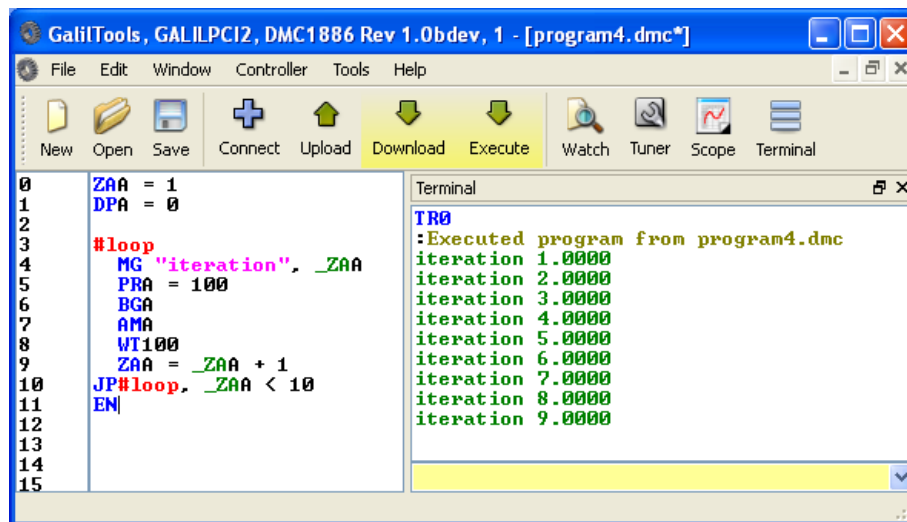
Continuing in the fashion brings us to the 9th iteration. The position is 900 and the counter is 9.

```
Terminal
: BK5
:6  AMA
7  WT100
8  ZAA = _ZAA + 1
9  JP#loop, _ZAA < 10
10 EN
```

Expecting another iteration, we issue BK5. The code continues and exits without iterating again and the early exit condition becomes clear (line 0 should read ZAA=0).

Printing Messages (MG)

[Breakpoint](#) (BK) and single step (SL) can be used to great utility to diagnose difficult bugs; however, printing messages with the MG command often works for quick checks:



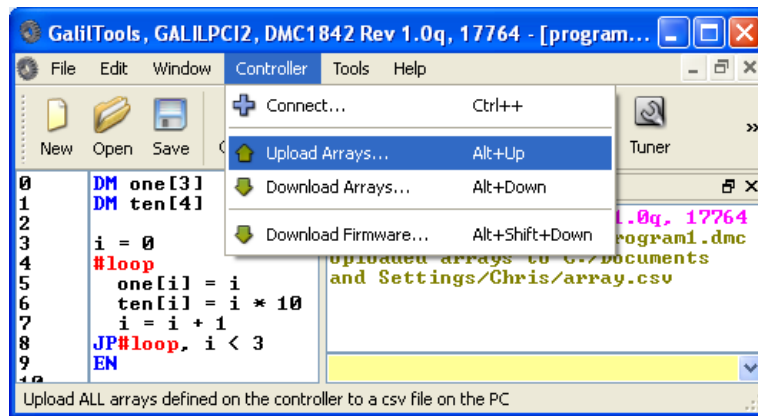
Line 4 has been added to print a message on each loop and the early exit is also evident.

[Table of Contents](#)

Comma-Separated Values (CSV Files)

Arrays

[GaliTools](#) uses Comma-Separated Values ([CSV](#)) format for representing array data.



GaliTools Upload Arrays...

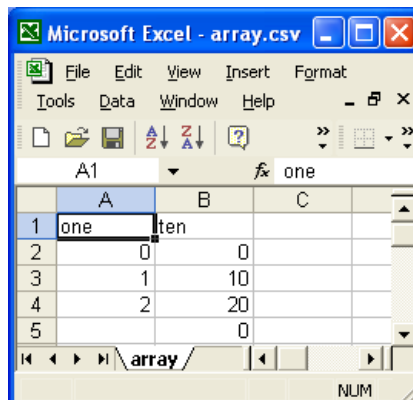
Uploading arrays from a [Gali](#) controller to the host PC will save the arrays as a text file in [CSV](#) format. When Downloading arrays from the host PC to the controller, [GaliTools](#) expects an input file in [CSV](#) format. The following is an example [CSV](#) file containing 2 arrays where "one" has 3 elements and "ten" has 4 elements:

one,ten
0,0
1,10
2,20
,0

The name of each array is in the first row of the data, separated by commas, and the columns from left to right are saved by [GaliTools](#) in alphabetical order based upon the array name. Element zero of each array is on the following line, followed by the rest of the array table data. Note that the shorter array "one" includes a place-holder comma

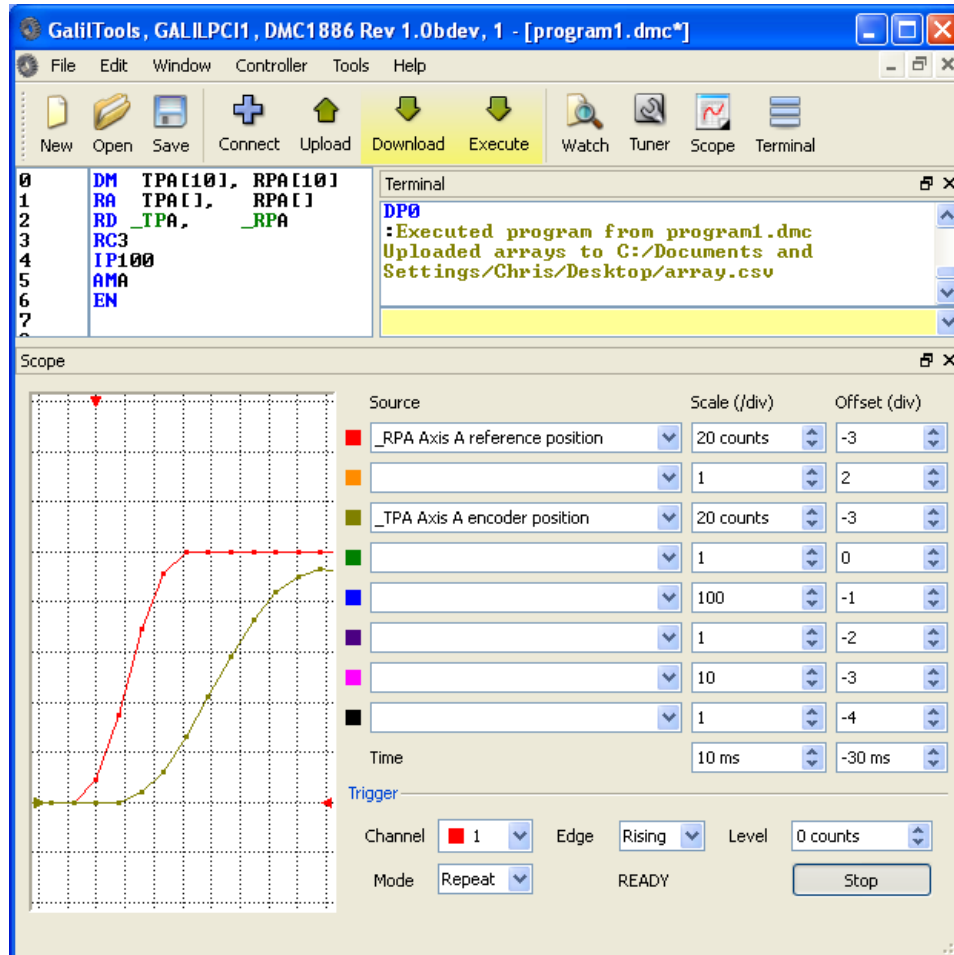
Hint: When Uploading array data from the controller to the PC, all arrays defined on the controller are saved. When Downloading array data to the controller, the CSV data is "appended" to the arrays on the controller (arrays with the same name on the controller as in the CSV file will be overwritten on the controller). See DA in the [command reference](#) to remove arrays from the controller.

CSV is widely supported and can be opened in any text editor such as Notepad and spreadsheets such as Microsoft Excel or [OpenOffice.org](#) Calc.

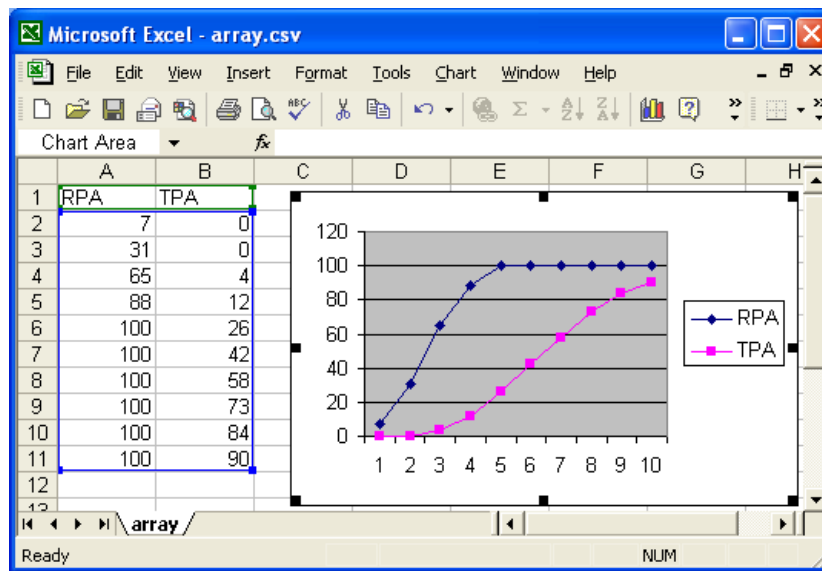


Axis and I/O Data

Below is an example of how to import axis data stored in controller arrays to a spreadsheet application such as Microsoft Excel. In this example, the Record Array (RA) feature of the controller is used to cache data during a move while the [GalilTools Scope](#) graphs the data in real time. Once the arrays are filled, the data can be Uploaded from the controller and graphed in a spreadsheet application.



The Galil code profiles movement and stores pertinent data to internal arrays while the [GalilTools Scope](#) plots the data in real time.



After the motion completes, the array data can be Uploaded and graphed in a spreadsheet. Note the similarities between the [GaliITools Scope](#) and Excel graph.

[Table of Contents](#)

Menus and Hot Keys

Many [GalilTools](#) features can be opened or manipulated from the application and right-click menus and many of these have associated [hot keys](#) for ease of use. Below is a table summarizing the available menu items and [hot keys](#):

MENU	ITEM	HOT KEY	DESCRIPTION
File	New	Ctrl+N	Open a blank Editor window
	Open...	Ctrl+Shift+O	Open a .dmc file from a saved location
	Save	Ctrl+Shift+S	Save the currently-active Editor window
	Save As...		Asks for a filename and path to save the currently-active Editor window
	Print...		Prints the currently-active Editor window
	Upload	Ctrl+Up	Upload .dmc file from controller to a GalilTools Editor window
	Download	Ctrl+Down	Download currently-active Editor contents to controller
	Execute	Ctrl+Shift+Down	Download and Execute currently-active Editor contents
	Exit	Alt+F4	Exit GalilTools application
Edit	Cut	Ctrl+X	Cut the currently-selected text to clipboard
	Copy	Ctrl+C	Copy the currently-selected text to clipboard
	Paste	Ctrl+V	Paste the contents of the clipboard (text only) to the current cursor location
	Find/Replace...	Ctrl+F	Open the Editor Find/Replace utility
	Find Again...	F3	Find the text in the Find/Replace
Window	Close		Close currently-selected Editor window
	Close All		Close all Editors
	Tile		Tile the open Editor windows for concurrent editing
	Cascade		Stack the open Editor windows for individual editing
	Next	Ctrl+Tab	Bring next Editor window to the front
	Previous <i>program.dmc</i>	Ctrl+Shift+Tab	Bring previous Editor window to the front Bring the indicated Editor to the front
Controller	Connect...	Ctrl++	Open Connections dialog
	Offline	Ctrl+-	Disconnect current connection
	Refresh	F5	Refresh current connection
	Upload Arrays...	Alt+up	Upload all arrays from controller to CSV file on PC
	Download Arrays...	Alt+down	Download array(s) from CSV file on PC to controller Note: Do not Download Arrays while a Record Array Mode is running on the controller (RC,RD,RA)
	Download Firmware...	Alt+Shift+Down	Ask the user for a .hex file for upgrading the controller firmware
Tools	Watch	Ctrl+Alt+W	Opens and/or places keyboard focus on the Watch window
	Tuner	Ctrl+Alt+U	Opens and/or places keyboard focus on the Tuner
	Scope	Ctrl+Alt+S	Opens and/or places keyboard focus on the Scope
	Terminal	Ctrl+Alt+T	Opens and/or places keyboard focus on the Terminal
Help	Contents	F1	Opens this help utility
	About		Opens application About dialog, containing versions and links
Connections	Refresh		Refresh Available connections
	Delete		Delete a Saved connection
Terminal Output	Copy		Copy the currently-selected text to clipboard
	Select All		Select all text in buffer

	Repeat	Alt+R	Repeat last command
	Clear		Clear Terminal output buffer
		Ctrl+wheel	Change font size
Terminal Input, Editor	Undo	Ctrl+Z	Step back in typed history
	Redo	Ctrl+Y	Step forward in typed history
	Cut	Ctrl+X	Cut the currently-selected text to clipboard
	Copy	Ctrl+C	Copy the currently-selected text to clipboard
	Paste	Ctrl+V	Paste the contents of the clipboard (text only) to the current cursor location
	Delete	Delete	Delete selected text
	Select All	Ctrl+A	Select all text in buffer
		Ctrl+wheel	Change font size (Editor only)
Terminal Input	Repeat	Alt+Shift+R	Repeat last command (toggle)
	Repeat	Alt+R	Repeat last command (momentary)
	Clear History		Erase the command history
Watch	Insert		Insert a row on Some tab
	Remove		Remove a row on Some tab
Scope	Points		Turn on/off individual point display

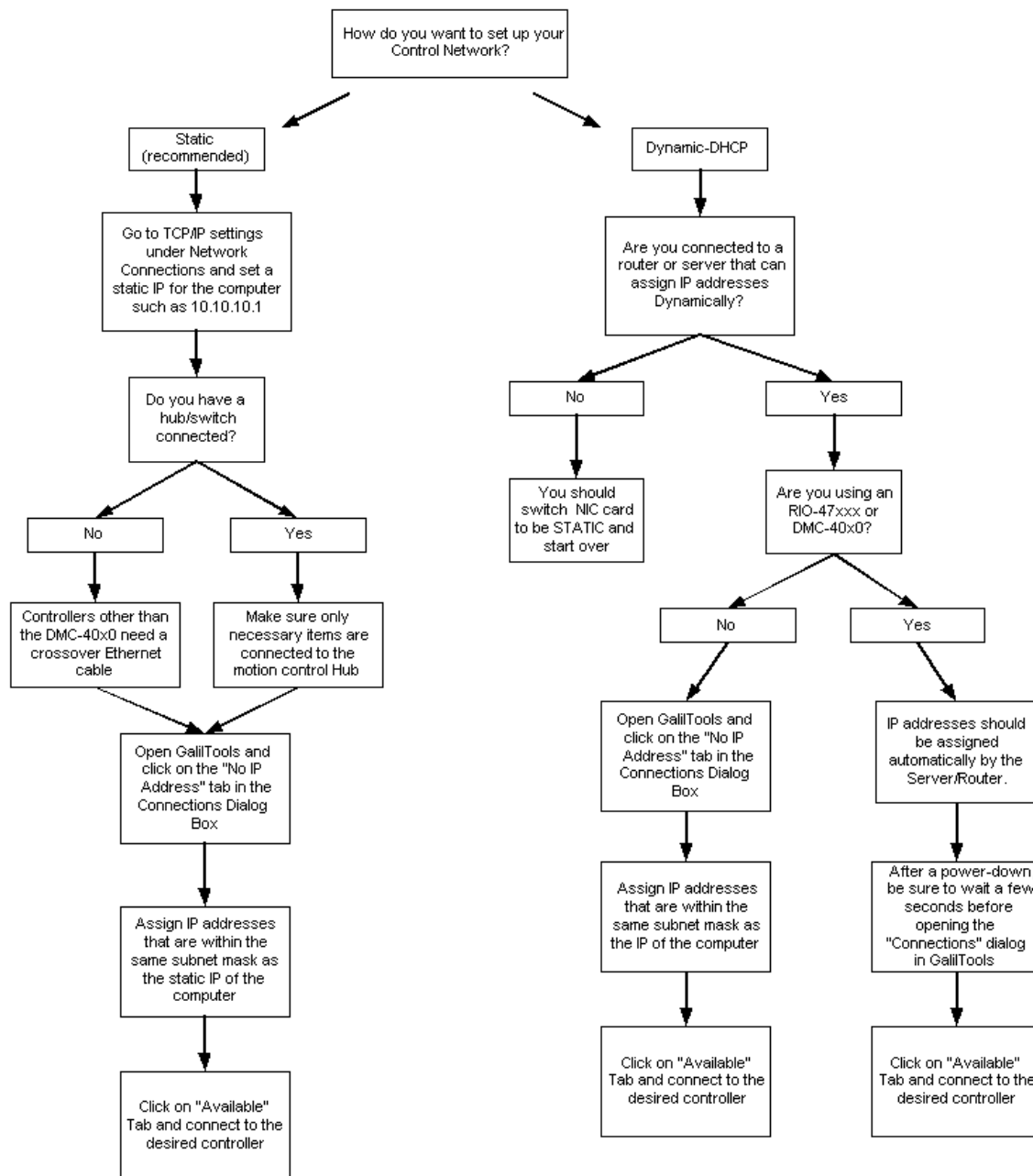
[Table of Contents](#)

Ethernet Network Guide for Connecting Galil Controllers

Many Galil controllers use Ethernet as a high performance communication bus. This chapter discusses the possible Ethernet network configurations that come up when installing Galil controllers. The first section shows a block diagram of connecting a [static network versus a dynamic network](#). The second section shows examples of [typical network configurations](#).

Overview

Setting a Controller IP Address



Block diagram that walks the user through connecting to a controller via a static versus a dynamic network

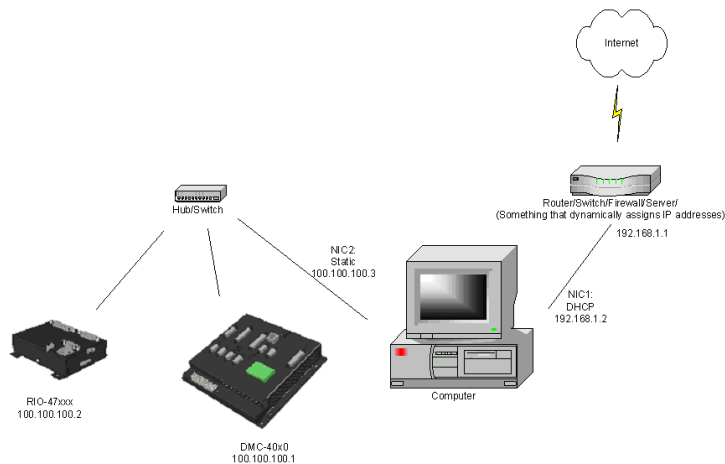
The instructions above reference the GalilTools [Connections](#) dialog and the [No IP](#) tab.

Typical Network Configurations

The following are examples of possible network configurations when using a Galil controller. The IP addresses shown are for example purposes only and can/will be different for actual installations. In all cases where there are two Network interfaces such as a static NIC and a wireless NIC - make sure that the two NIC interfaces are using IP addresses that are on different subnets. For example, a static network card cannot have the IP address of 192.168.1.5 if the wireless NIC has an IP address of 192.168.1.4. Instead, the static NIC should have something like 10.10.10.1 in order to make sure it is on a different subnet*.

Network Configuration #1

This first network configuration is Galil's recommended way of having a computer connected to both the outside world (Internet or company network) at the same time as connecting to an internal controller network. In the diagram, it shows that the first network card labeled NIC1 has an IP address of 192.168.1.2 and is dynamically assigned an IP address from the Router (via DHCP). The second network card is labeled NIC2 and should be set up as a static IP address via the computers Network card settings. In this scenario, the two controllers on the network must be assigned an IP address via Ethernet using Galil software such as GalilTools, SmartTerm or WSDK or via an RS-232 serial connection (not shown) using the IA command and then using BN to burn it in to non-volatile memory. (When using the Galil software to assign the controller IP over Ethernet, the BN command is automatically issued after the IP assignment.) If Internet is not needed, then this same configuration is valid by removing NIC1 from the diagram.

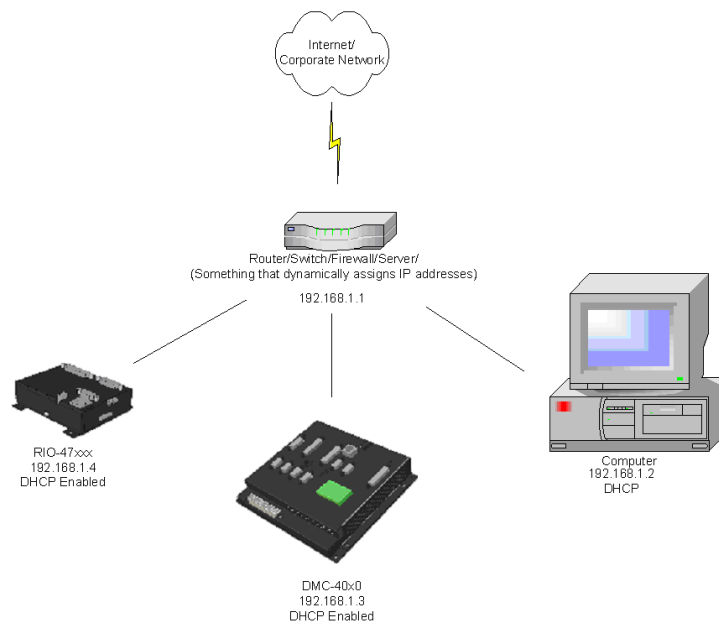


Network Configuration #1

Network Configuration #2

The third configuration is for users that would like to attach an auxiliary device to the controller via Ethernet such as an RIO to get more I/O points but who do not want to use a Hub/Switch to connect them. A Crossover Ethernet cable can be used to connect from the controller to the RIO. In the case of the DMC-40×0, a crossover Ethernet cable is not necessary as it will automatically configure itself based on the cable being used. For more information on connecting a RIO-47xxx as external I/O, go to:

<http://www.galilmc.com/support/appnotes/accelera/note2512.pdf>



Network Configuration #2

Footnotes

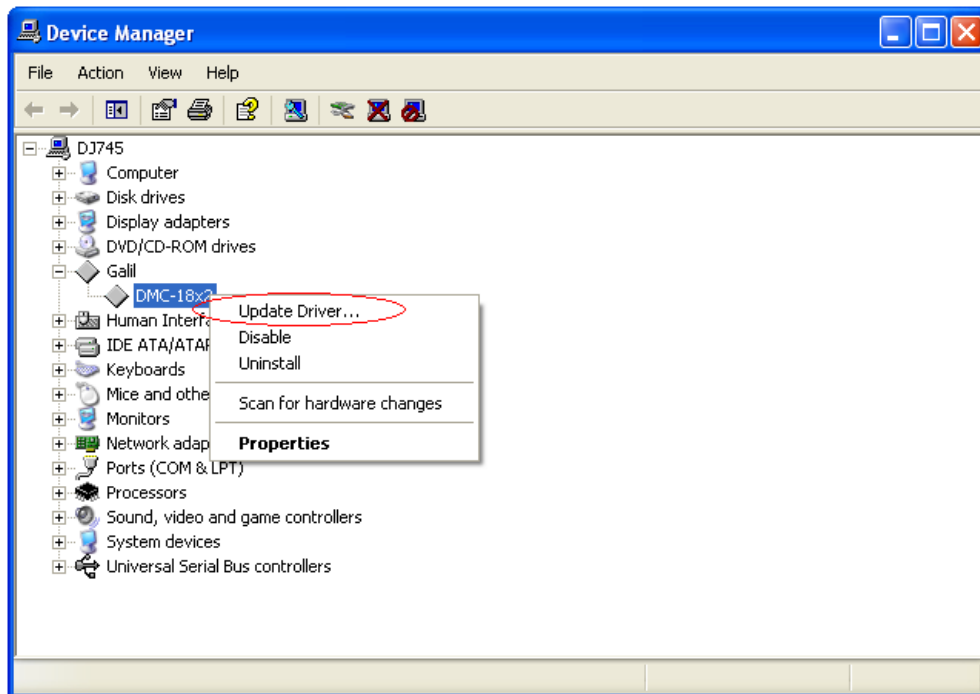
*Subnet - a subnet is a group of IP addresses that are grouped together as a “sub-network” in order to be used for a specific purpose. IP addresses outside of the “subnet mask” are not allowed to communicate to IP addresses inside the subnet. A subnet mask is generally specified by using a value of 255 designating which parts of the IP address are common to that subnet. For example, a subnet mask of 255.255.0.0 specifies that the first two bytes of the IP address are restricted to communication to other devices with the same values for those first two bytes (ie: a device with an IP address of 192.168.x.x can communicate to any other device with an IP address of 192.168.x.x)

[Table of Contents](#)

PCI on Windows (DMC-18x2/6)

GaliTools uses a different PCI driver than legacy Gali software such as SmartTerm or WSDK. As a result, it is not possible to switch between GaliTools and legacy applications without reconfiguring Windows to select the appropriate PCI driver. If both applications are not required, Gali recommends uninstalling SmartTerm/WSDK. This chapter details the steps required to switch between PCI drivers, which can be done as often as desired to accommodate both old and new software.

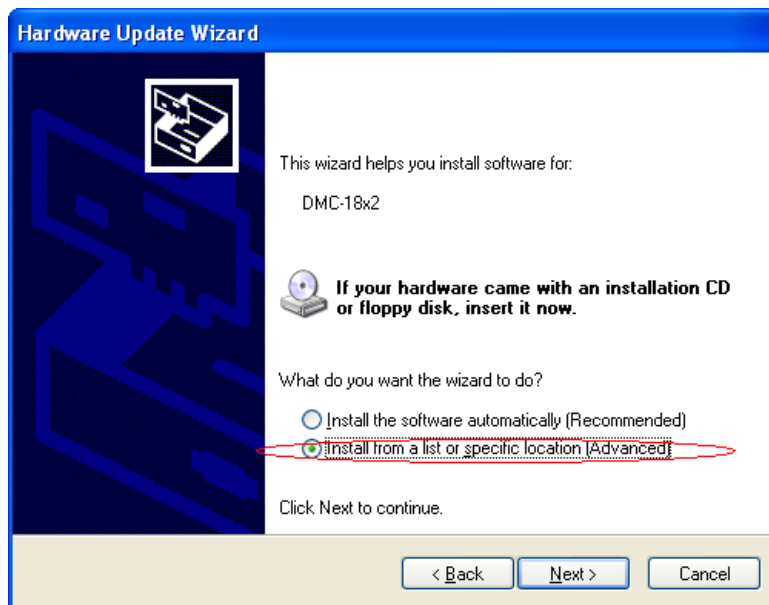
Switching between drivers is done through the Update Driver... option in the Device Manager. To access the Device Manager, go to the Start Menu and select "Run...". In the dialog, type `devmgmt.msc` and click OK (the Device Manager can also be navigated to from Control Panel | System | Hardware). Once in the Device Manager, select the DMC-18x2 or 18x6 device under Gali, right click, and select "Update Driver...".



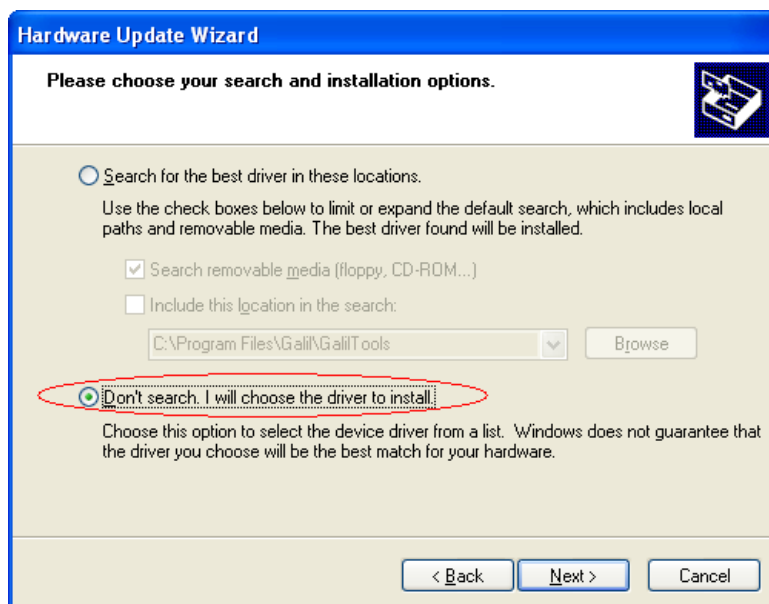
Changing the driver associated with a PCI controller. Select *Update Driver...*



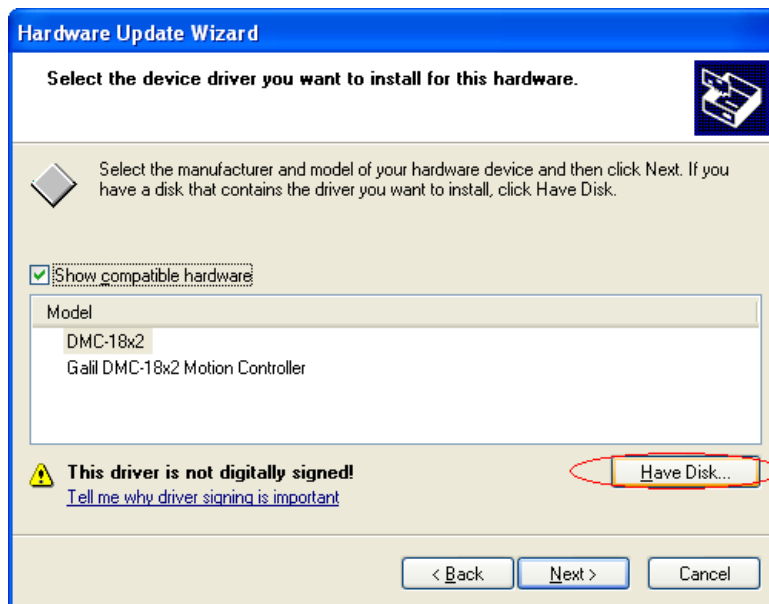
Do not permit Windows to look online



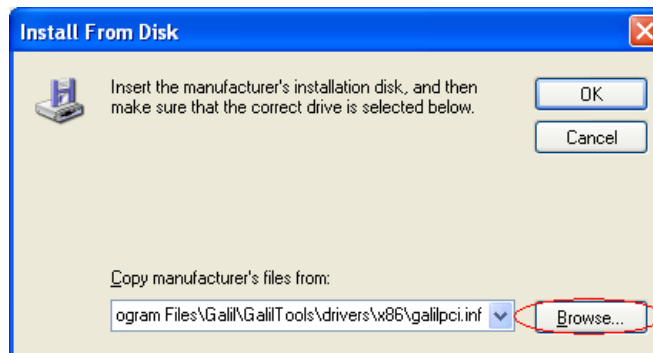
Choose to install from a specific location



Choose Don't Search



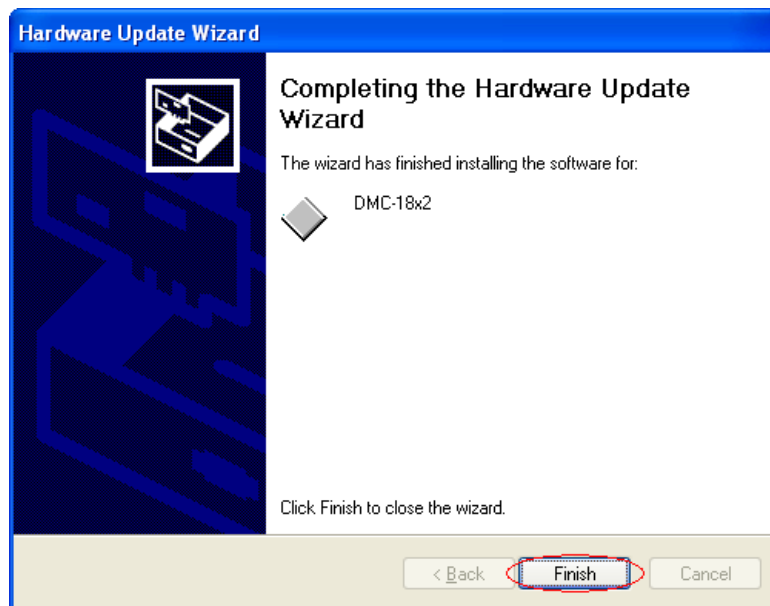
Choose *Have Disk...*



Navigate to the appropriate driver by clicking **Browse** or typing the absolute path

	<u>GalilTools driver</u> path:	Legacy <u>driver</u> path:
x86	C:\Program Files\Galil\GalilTools\drivers\galilpci.inf	C:\Program Files\Galil Common\DevInstall\32Bit\glwdmpci.inf
x64	C:\Program Files\Galil\GalilTools\drivers\galilpci.inf	C:\Program Files\Galil Common\DevInstall\64Bit\glwdmpci.inf

Click OK and then Next. Windows will show progress as the driver updates.



The driver has now been updated

[Table of Contents](#)

Communication Library Reference

The GalilTools Communication Library (Galil class) provides methods for communication with a Galil motion controller over Ethernet, RS-232 or PCI buses. It consists of a native C++ Library and a similar COM interface which extends compatibility to Windows programming languages (e.g. VB, C#, etc). [More...](#)

Public Members

• **Connecting and Disconnecting**

- [Galil](#)(string address = "") [address property](#)
- [~Galil](#)()
- vector<string>[addresses](#)()
- string [connection](#)()
- int [timeout_ms](#)

• **Basic Communication**

- string [command](#)(string command = "MG TIME", string terminator = "\r", string ack = ":", bool trim = true)
- double [commandValue](#)(string command = "MG TIME")
- string [message](#)(int timeout = 500) [onMessage Event](#)

• **Programs**

- void [programDownload](#)(string program = "MG TIME\rEN")
- void [programDownloadFile](#)(string file = "program.dmc")
- string [programUpload](#)()
- void [programUploadFile](#)(string file = "program.dmc")

• **Arrays**

- void [arrayDownload](#)(vector<double> array, string name = "array")
- void [arrayDownloadFile](#)(string file = "arrays.csv")
- vector<double> [arrayUpload](#)(string name = "array")
- void [arrayUploadFile](#)(string file, string names = "")

• **Advanced**

- void [firmwareDownloadFile](#)(string file = "firmware.hex")
- string [read](#)()
- int [write](#)(string bytes = "\r")
- int [interrupt](#)(int timeout_ms = 500) [onInterrupt Event](#)
- static string [libraryVersion](#)()

• **Data Record**

- vector<string> [sources](#)()
- void [recordsStart](#)(double period_ms = -1)
- vector<char>[record](#)(string method = "QR") [onRecord Event](#)
- double [sourceValue](#)(vector<char> record, string source = "TIME")
- string [source](#)(string field = "Description", string source = "TIME")
- void [setSource](#)(string field = "Description", string source="TIME", string to= "Sample counter")

Detailed Description

The GalilTools Communication Library (Galil class) provides methods for communication with a Galil motion controller over Ethernet, RS-232 or PCI buses. It consists of a native C++ Library and a similar COM interface which extends compatibility to Windows programming languages (e.g. VB, C#, etc).

A Galil object (usually referred to in sample code as "g") represents a connection to a Galil controller. For Ethernet controllers, which support more than one connection, multiple objects may be used to communicate with the controller.

The library is conceptually divided into six categories:

1. Connecting and Disconnecting - functions to establish and discontinue communication with a controller.
2. Basic Communication - The most heavily used functions for command-and-response and unsolicited messages.
3. Programs - Downloading and uploading embedded programs.
4. Arrays - Downloading and uploading array data.
5. Advanced - Lesser-used calls.
6. Data Record - Access to the data record in both synchronous and asynchronous modes.

C++ Library (Windows and Linux)

Both Full and Lite versions of GalilTools ship with a native C++ communication library. The Linux version (libGalil.so) is compatible with g++ and the Windows version (Galil1.dll) with Visual C++ 2008. Contact Galil if another version of the C++ library is required. See the [getting started guide](#) and the hello.cpp example in /lib.

COM (Windows)

To further extend the language compatibility on Windows, a COM (Component Object Model) class built on top of the C++ library is also provided with Windows releases. This COM wrapper can be used in any language and IDE supporting COM (Visual Studio 2005, 2008, etc). The COM wrapper includes all of the functionality of the base C++ class. See the [getting started guide](#) and the hello.* examples in /lib for more info.

The Data Record

The data record is a Galil controller feature that is ideal for data collection and controller monitoring. It is a binary data structure generated by the controller's firmware either in a periodic and asynchronous manner (DR) or via an interrogated, synchronous command (QR). The transmitted data contains a variety of controller information: encoder positions, reference positions, profiler information, I/O, program information, user variables on the Accelera family, and much more.

The Watch All window in GalilTools displays the entire data record for a particular controller and is an excellent way to conceptualize the data record operation.

Graphical depiction of the data record operation.

Watch

All Some

	Source	Value	Units	Description
1	@AN[1]	1.7871	V	Analog input 1
2	@AN[2]	1.7871	V	Analog input 2
3	@IN[01]	1	Boolean	Digital input 1
4	@IN[02]	1	Boolean	Digital input 2
5	@IN[03]	1	Boolean	Digital input 3
6	@IN[04]	1	Boolean	Digital input 4
7	@IN[05]	1	Boolean	Digital input 5
8	@IN[06]	1	Boolean	Digital input 6
9	@IN[07]	1	Boolean	Digital input 7
10	@IN[08]	1	Boolean	Digital input 8
11	@IN[09]	0	Boolean	Digital input 9
12	@IN[10]	1	Boolean	Digital input 10
13	@IN[11]	0	Boolean	Digital input 11
14	@IN[12]	0	Boolean	Digital input 12

Annotations:

- `r = g.record("QR")`
Stuffs r with a full data record in binary corresponding to the current values of each of the sources. QR or DR can be arguments to record(). The record corresponds to the dynamic data in the Value column of the Watch All window. Each call to record() "refreshes" the data.
- `s = g.sources()`
Returns the entire source column from Watch All.
Individual source strings, e.g. @IN[08], are used as arguments to the sourceValue(), sourceUnits(), and sourceDescription() functions.
- `myValue = g.sourceValue(r, "@IN[08]")`
Passed a data record and a valid source string, sourceValue() returns the value of the source.
- `myUnits = g.sourceUnits("Units", "@IN[08]")`
Given a valid source string s, source("Units", s) returns the units of that source. Note the units are static by connection.
- `myDesc = g.source("Description", "@IN[08]")`
Given a valid source string s, source("Description", s) returns the human readable description of that source. Note the description is static by connection.

As in GalilTools, the data record Units, Description, Scaling, and Offset can all be set with the [setSource\(\)](#) function. See [setSource\(\)](#) and [Watch](#) for more information.

QR vs. DR

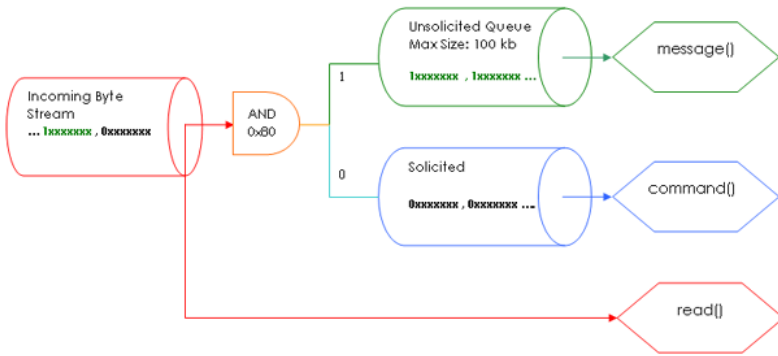
There are two methods for data record acquisition:

- **QR.** By sending the command "QR" from the host, the controller responds back with a Data Record. This is a command-and-response acquisition. GalilTools Watch uses QR for low frequency, non-periodic data records.
- **DR.** By using [recordsStart\(n\)](#), where n is a sample period in milliseconds, the controller is configured in asynchronous mode. Data will be sent by the controller periodically and without action from the host. The GalilTools Scope uses the DR method where possible for high-frequency, periodic data records.

The QR and DR details are abstracted by the data record API. Consult the controller command reference for more information regarding QR and DR.

Flow of data from controller.

The most significant bit in received bytes is used to determine unsolicited versus solicited data during a synchronous (solicited) transaction (See CW in controller command reference). If data is received by the library NOT during a synchronous transaction (e.g. command()) the data will automatically be put in the unsolicited queue regardless of the most significant bit.



Errors

The exceptions thrown by the Galil class were designed to allow for easy debugging by providing human-readable information. No error-code table needs to be referenced by the programmer as the error information is included in the error. **All thrown objects are strings.** String search functions or error codes can be used to determine the nature of an error.

"TC1" is sent to the controller automatically by the drivers in the event of a fault. This controller-generated error text is included in the string thrown by the drivers.

Example 1. Using string functions to programmatically handle thrown errors (below is a partial Try/Catch handler)

```

VB
Catch exception As System.Runtime.InteropServices.COMException
    Console.WriteLine(exception) 'print error message

    If exception.Message.Contains("COMMAND ERROR") Then
        Console.WriteLine("a command error occurred") 'special processing for command errors
    End If
End Try
  
```

For programmers wishing to use a traditional return code, the first 4 characters of every error message contains a unique error code.

The first character of the return code contains the broad category of error:	The next two digits contain a number specifying the function which generated the error. Note there is a private function at x01x that may throw in other functions.	
1xxx TIMEOUT 2xxx COMMAND 3xxx MONITOR 4xxx FILE 5xxx OPEN 6xxx WRONG BUS 7xxx INVALID 8xxx WARNING 9xxx OFFLINE (COM only)	x00x Galil() x02x command() x03x message() x04x interrupt() x05x programUpload() x06x programDownload() x07x programUploadFile() x08x programDownloadFile() x09x arrayUpload() x10x arrayDownload() x11x arrayUploadFile() x12x arrayDownloadFile() x13x firmwareDownloadFile() x14x recordsStart() x15x record() x16x sourceValue() x17x source() x18x setSource() x19x sources() x20x write() x21x read() x22x commandValue() x23x connection() x24x timeout_ms	The last digit (xxx0) is a serial number (0-9) to provide uniqueness among very similar errors.

Example 2. Using the first 4 characters of a thrown string as a return code.

```

C++
catch(string e) {
    int t = atoi( (e.substr(0,1)).c_str() ); // get int for type, first digit of code

    int f = atoi( (e.substr(1,2)).c_str() ); // int for function, middle two digits of code

    int u = atoi( (e.substr(3,1)).c_str() ); // int for unique serial, last digit of code

    //ints can now be used in switch statements to handle all variations of error
}
  
```

Member Documentation

C++ Galil::Galil(string address = "")

VB address As String

C# string address

Galil() is the constructor for the Galil C++ class and invoking it will create a new instance and open the specified connection. There is one argument to the constructor class which indicates the connection to open. Set the *address* property when using the COM class on Windows.

C++ Type	Argument	Example	Description
string	address	"GALILPCI1"	Connection string for controller (see examples of valid strings below). If "" (null), display the Connections dialog to allow the user to choose a controller.

Example 1. null string

With a null string passed to the Galil constructor, the [Connections](#) dialog will be presented for the user to choose the connection to open.

```
VB Dim g As New Galil.Galil 'Dimension g variable for instantiated object
g.address = "" 'Specify null address, Connections dialog will display
C++ Galil g(""); //prompt user for connection
C++ Galil g; //uses default "" from header
```

Example 2. Valid addresses (Note: strings in quotes are valid for all programming languages)

```
C++ Galil g("COM1 19200"); //RS-232 port 1, 19200 baud, Windows (DMC-21x3/40x0)
Galil g("/dev/ttyS0 19200"); //RS-232 port 1, 19200 baud, Linux (DMC-21x3/40x0)
Galil g("192.168.1.105"); //IP Address (TCP) (DMC-21x3/40x0/RIO)
Galil g("RIO47100-13"); //DNS name for supported controller (DHCP support) (RIO/DMC-4000)
Galil g("GALILPCI1"); //Galil PCI (DMC-18x6/18x2), Windows (DMC-18x2/18x6)
Galil g("/dev/galilpci0"); //Galil PCI (DMC-18x6/18x2), Linux (DMC-18x2/18x6)
Galil g("OFFLINE"); //Offline connection.
```

Note: The connection behaviour can be modified by appending address line switches to the connection address string. See the [connection address options table](#) in the connections chapter for a complete list.

Example 3. Path to Connection file

If the address ends in the substring ".con", it refers to a con file as saved by the [Connections dialog](#).

```
C++ Galil g("connection.con"); //Connect to string in con file
Galil g("c:/documents and settings/John Doe/Galil/myconnection.con"); //con file at different path than executable
C# Galil.Galil g = new Galil.Galil(); //Dimension g variable for instantiated object
g.address = "myController.con"; //connect to contents of con file
```

A .con file is simply a text file containing a valid address.

A connection can be changed simply by modifying the object's address.

Example 4. Changing connection of Galil object

```
VB For i As Integer = 1 To 5 '5 iterations
g.address = "" 'display connection chooser to determine connection
Label1.Text = g.connection 'display connection on Form label
Next i
```

Upon invoking the constructor, several values are interrogated: controller revision information, presence of analog inputs, serial number, connected handle, etc., and the following state changes are made:

Command Sent	Purpose	Consequences
CFI	Set the current connected handle as the handle used for unsolicited messages. (Ethernet Only)	CFI will "steal" the unsolicited handle away from any other connection currently using it.
EO0	Disable Echo on RS-232 connections.	Commands typed will not be echoed back by the controller. This is important so that echoed commands are not interpreted as command responses.

CW1	Mark Unsolicited Bytes.	All unsolicited messages will have the most significant bit (extended ASCII) set. This allows the library to differentiate between solicited (zero in the highest bit) and unsolicited (highest bit 1) bytes. See message() for more info.
-----	-------------------------	--

Throws:

```
5000 OPEN ERROR. User cancelled connection from dialog in Galil::Galil()
4000 FILE ERROR. Galil::Galil() failed to open file...
5001 OPEN ERROR. OFFLINE specified to Galil::Galil()
5002 OPEN ERROR. Galil::Galil() failed to open PCI device...
5003 OPEN ERROR. Galil::Galil() failed to open RS-232 port...
5004 OPEN ERROR. Galil::Galil() failed to open Ethernet host...
1010 TIMEOUT ERROR. Galil::command(...) took longer than ... ms to write
1011 TIMEOUT ERROR. Galil::command(...) took longer than ... ms to read : response...
```

[top](#)

C++ Galil::~Galil()

~Galil is the destructor for a Galil object and is automatically invoked when a Galil object's scope is ended (or when delete is called in C++). When the destructor runs, the connection to the controller is closed.

Example 1. Destructor gets called when scope of object ends.

```
C++ { // scope for g
    Galil g("192.168.1.100"); //Constructor -- Connect to controller
} //destructor invoked on g -- Connection is closed
```

Example 2. Disconnecting from a controller (handling OFFLINE error).

```
VB Try 'main Try Catch block
    g = New Galil.Galil 'create a new Galil object

    g.address = "10.0.6.134" 'specify address
    MsgBox("Connected to " + g.connection) 'display connection information

    Try 'nested Try catch block
        g.address = "OFFLINE" 'OFFLINE causes the destructor to run, disconnecting from the controller
        Catch ex As Exception 'OFFLINE throws an error so catch and ignore it

        End Try
        MsgBox("Now Disconnected") 'code keeps running
        Catch ex As Exception 'catch for the main Try. This would catch normal errors, such as connection timeout
    End Try
```

If the DR interval is 0 upon instantiation and recordsStart() is called during the extent of a Galil object, the destructor will issue a DR0 to terminate the data record automatic dispatch. If the DR interval was nonzero at connection or if recordsStart() was not called, the destructor does not change the DR state.

TCP connections will be closed, however UDP resources, being connectionless, are not released. If the user desires to close UDP resources, the IH command may be used prior to destructor. See the controller's command reference under IH.

Throws: none

[top](#)

C++ vector<string> Galil::addresses()

Returns a list of available addresses to connect to (e.g. "1.2.3.4"). Each item in the list may be fed to the constructor Galil(). This is a dynamically generated list with the same contents as the [Available Tab](#) in the GalilTools connections dialog. Each time this function is called is like clicking refresh in the [Available Tab](#)

C++ Type	Argument	Example	Description
vector<string>	return	"192.168.1.100 ; 192.168.1.101 ; 192.168.1.102"	A list of all available controllers by address

Although not necessarily addresses of controllers, available RS-232 ports will also be listed.

Example 1. Printing available addresses

```
VB Dim g As New Galil.Galil 'note, no connection needed to invoke addresses()
    For Each address As String In g.addresses()
        TextBox1.AppendText(address + vbCrLf)
    Next
```

Throws: none

C++ string Galil::connection()

Returns a string containing connection-related information. This information includes the address, controller firmware revision information, serial number, and Ethernet handle (where applicable). The following are return examples.

GALILPC11, DMC1886 Rev 1.0b, 36
COM2 19200, DMC2182 Rev 1.0p, 202
192.168.1.2, DMC4020 Rev 1.0a, 189, IHD

C++ Type	Argument	Example	Description
string	return	"GALILPC11, DMC1886 Rev 1.0b, 36"	Controller and connection information

Example 1. Printing the connection information

```
VB MsgBox(g.connection, MsgBoxStyle.DefaultButton1, "Galil Controller Info") 'display message box with connection info
C# Console.WriteLine(g.connection()); //write connection information to the console
C++ cout << g.connection() << endl; //print connection info to screen
```

Throws:
 9230 UNINITIALIZED OBJECT ERROR. Galil::connection() called without Galil::address set

C++ int Galil::timeout_ms

Specifies the general purpose timeout in milliseconds for Galil(), command(), commandValue(), programUpload(), programUploadFile(), arrayUpload(), arrayUploadFile(), and record(). It defaults to 500ms.

Example 1. Specifying the timeout for a connection

```
VB g.timeout = 2000 'set timeout to two seconds
```

Throws (COM only):
 9240 UNINITIALIZED OBJECT ERROR. Galil::timeout_ms set without Galil::address set
 9241 UNINITIALIZED OBJECT ERROR. Galil::timeout_ms read without Galil::address set

C++ string Galil::command(string command = "MG TIME", string terminator = "\r", string ack = ":", bool trim = true)

Sends a command to the Galil controller and returns the response from the controller if ack is received within [timeout_ms](#). If an unexpected response is returned (usually a "?") or the timeout is exceeded, the function will throw an error.

C++ Type	Argument	Example	Description
string	return	"123"	The function returns a string containing the response from the controller.
string	command	"TPA"	A valid Galil command to send to the controller.
string	terminator	"\r"	A string to append to the end of command. In almost all cases a carriage return should be used. NOTE: Do not append a \r in the "command" string.
string	ack	":"	The expected last character acknowledged in a good response. In almost all cases a colon is expected.
bool	trim	true	If true, the leading space and the trailing carriage return, line feed and colon will be stripped from the response. False is useful for terminal applications.

Example 1. Basic command() use:

```
VB Button1.Text = g.command("MG@IN[1]") 'update Button1 text with the state of controller digital input 1
C# button1.Text = g.command("TI", "\r", ":", true); //update button1 text with input byte 0 (Tell Input, TI). Note, C# requires all command() args present
```

Example 2. Using the extra arguments (advanced). Specify a different terminator to allow program download with overwrite (See DL in controller command reference).

```
C++ g.programDownload("MG\Start\`r#HERE`rMG`Overwrite This`\rEN\r"); //download code
g.command("DL#HERE`rMG`Overwritten`\rEN\r",""); //overwrite the program at label #HERE. Note the backslash terminator.
```

Example 3. Using the extra arguments (advanced). Specify a different terminator to allow array download with overwrite (See QD in controller command reference).

```
VB Dim arrayData = New Integer() {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} 'create array
g.arrayDownload(arrayData, "A") 'download the array
g.command("QD A[,8,9" + vbCr + "100" + vbCr + "200", "\", ":", True) 'overwrite the last two elements of array. Note the backslash terminator.
```

Example 4. Binary command() use (advanced)

In almost all applications, use of binary commands is not necessary; however, for those applications requiring it, the command function can be used.

```
C++ g.command(string("\xA0\x00\x00\x01",4), ""); //binary for BGA. Note no carriage return in terminator.
```

Throws:
7020 INVALID COMMAND ERROR. DL, UL, ED, and QD are not allowed from Galil::command()
1010 TIMEOUT ERROR. Galil::command("...") took longer than ... ms to write
1011 TIMEOUT ERROR. Galil::command("...") took longer than ... ms to read : response...
2010 COMMAND ERROR. Galil::command("...") got ? instead of : response...
3010 MONITOR ERROR. Galil::command("...") got > instead of : response. Got...
9020 UNINITIALIZED OBJECT ERROR. Galil::command() called without Galil::address set

[top](#)

```
C++ double Galil::commandValue(string command = "MG TIME")
```

Convenience function for interrogating a single integer ("MG_TPA"), real number ("MG@AN[1]"), or boolean ("MG_BGA") value from the controller.

C++ Type	Argument	Example	Description
double	return	2.3524	Response from controller converted to floating point
string	command	"MG@AN[1]"	The command to send to the controller. NOTE: Do not append a \r in the "command" string.

Example 1. Interrogating Position

```
C# double positionA = g.commandValue("TPA");
```

Note: For commands that don't return a number (e.g. XQ, ST, BG), a zero will be returned.

Throws:
7020 INVALID COMMAND ERROR. DL, UL, ED, and QD are not allowed from Galil::command()
1010 TIMEOUT ERROR. Galil::command("...") took longer than ... ms to write
1011 TIMEOUT ERROR. Galil::command("...") took longer than ... ms to read : response...
2010 COMMAND ERROR. Galil::command("...") got ? instead of : response...
3010 MONITOR ERROR. Galil::command("...") got > instead of : response. Got...
9220 UNINITIALIZED OBJECT ERROR. Galil::commandValue() called without Galil::address set

[top](#)

```
C++ string Galil::message(int timeout_ms = 500)
```

```
VB Event Sub onMessage(ByVal message As String)
```

```
C# Event void onMessage(string message)
```

Returns unsolicited messages (MGs) sent from the controller. Traffic transmitted by the controller to the host can be categorized into two primary groups: solicited and unsolicited. Solicited bytes are responses from commands sent by the host. For example, a solicited message is an encoder position string returned from *g.command("TPX")*. Unsolicited messages are transmitted asynchronously and could be output from MG commands from a running program aboard the controller, asynchronous error messages, trace (TR1) output, or responses from commands in a controller-side DMC program (TP, RP, etc).

For Ethernet connections, the library will automatically open two Ethernet handles on the controller. One TCP handle for command-and-response traffic, and one UDP handle for asynchronous data, including messages (MGs).

C++: If the host-side message queue is empty, message() will wait up to timeout_ms for a message. If none occur within timeout_ms, a timeout error will be thrown. If there are already messages in the queue when message() is called, the function will immediately return the contents. If a zero timeout is specified, no errors will be thrown; message() will simply return the waiting queue (even if it is empty, ""). A -1 timeout will cause message() to block until a message is received.

COM: There are no timeouts for the onMessage event.

C++ Type	Argument	Example	Description
string	return	"Hello\rMy TP Value-123.0000"	Returned contents of the unsolicited messages buffer
int	timeout_ms	500	The time in milliseconds to wait for the existence of data in the message queue

Example 1. Printing unsolicited messages

VB	<pre>Private Sub g_onMessage (ByVal message As String) Handles g.onMessage 'event runs when message received messagesTextBox.AppendText (message) 'print to textbox End Sub</pre>
C#	<pre>//in initialization code block g.onMessage += new Galil.Events_onMessageEventHandler (g_onMessage); //hook up to the onMessage Event //event subroutine void g_onMessage (string message) { //handler for the onMessage event messagesTextBox.AppendText (message); }</pre>
C++	<pre>cout << g.message (0); //Print current message buffer to console. No waiting or throws.</pre>
Throws: 1030 TIMEOUT ERROR. Galil::message() took longer than ... ms to read MG	

[top](#)

C++ void Galil::programDownload(program = "MG TIME\rEN")

Downloads a program from a host buffer to the Galil controller. For very long programs which are larger than the controller's program space (e.g. 80 characters by 2000 lines on DMC-40x0), the function will attempt to compress the program before downloading. This is done by removing white space and concatenating multiple commands onto each line of code. A warning (8060) will be thrown signaling that compression was required and a try-catch block included in every call to this function is recommended to handle this possible warning condition.

If line zero of the buffer contains *REM DISABLE COMPRESSION*, code will not be compressed and downloading otherwise compressible code will throw an error and fail.

See write() for examples of downloading code with insertion and while motion is profiling.

C++ Type	Argument	Example	Description
string	program	"MG TIME\rEN"	String containing the program to download with each line separated by a carriage return.

Example 1. Download a program to controller from buffer

C++	<pre>g.programDownload("#A\rMGTIME\rWT500\rJP#A\rEN"); //Download a program to controller</pre>
------------	---

Example 2. Download program from textbox and then execute

VB	<pre>g.programDownload (ProgramTextBox.Text) 'Download the contents of the textbox g.command ("XQ") 'Send execute command</pre>
-----------	---

Throws:
7060 INVALID CHARACTER ERROR. Galil::programDownload() can't download program with backslash \ character. Use {^92} in MG commands
7061 INVALID LENGTH ERROR. Galil::programDownload() can't compress line "..." of ... columns or more
7062 INVALID LENGTH ERROR. Galil::programDownload() can't download compressed program with more than ... lines by ... columns.
Contact Galil for special memory expansion firmware: www.galilmc.com/products/accessories/upgd_options.html#expanded_memory
7063 INVALID LENGTH ERROR. Galil::programDownload() can't download program with more than ... lines by ... columns because compression is disabled.
8060 COMPRESSION WARNING. Galil::programDownload() modified program to fit in ... lines by ... columns (check LS)
1010 TIMEOUT ERROR. Galil::command("...") took longer than ... ms to write
1011 TIMEOUT ERROR. Galil::command("...") took longer than ... ms to read : response...
2010 COMMAND ERROR. Galil::command("...") got ? instead of : response...
3010 MONITOR ERROR. Galil::command("...") got > instead of : response. Got...

[top](#)**C++** **void Galil::programDownloadFile(string file = "program.dmc")**

Downloads a program file to the Galil controller. The program text is read from a file located on the host file system. For very long programs which are larger than the controller's program space (e.g. 80 characters by 2000 lines on DMC-40x0), the function will attempt to compress the program before downloading. This is done by removing white space and concatenating multiple commands onto each line of code. A warning (8060) will be thrown signaling that compression was required and a try-catch block included in every call to this function is recommended to handle this possible warning condition.

If line zero of the file contains *REM DISABLE COMPRESSION*, code will not be compressed and downloading otherwise compressible code will throw an error and fail.

See write() for examples of downloading code with insertion and while motion is profiling.

C++ Type	Argument	Example	Description
string	program	"galilProgram.dmc"	String containing path and filename of dmc file. If only the filename is specified, the executable's directory will be used as the path.

Example 1. Display file chooser and download selected file

```

VB Dim chooseFile As New OpenFileDialog() 'dimension file chooser object
chooseFile.Filter = "DMC Files (*.dmc)|*.dmc" 'set filter to dmc files
chooseFile.ShowDialog() 'display chooser to user
g.programDownloadFile(chooseFile.FileName) 'download selected path (note, no error checking included)

```

Example 2. Download a program to controller from file

```

C++ g.programDownloadFile("myProgram.dmc"); //Download a program to controller

```

Throws:

```

4080 FILE ERROR. Galil::programDownloadFile() failed to open file...
7060 INVALID CHARACTER ERROR. Galil::programDownload() can't download program with backslash \ character. Use (^92) in MG commands
7061 INVALID LENGTH ERROR. Galil::programDownload() can't compress line "..." of ... columns or more
7062 INVALID LENGTH ERROR. Galil::programDownload() can't download compressed program with more than ... lines by ... columns.
Contact Galil for special memory expansion firmware: www.galilmc.com/products/accessories/upgd_options.html#expanded_memory
7063 INVALID LENGTH ERROR. Galil::programDownload() can't download program with more than ... lines by ... columns because compression is disabled.
8060 COMPRESSION WARNING. Galil::programDownload() modified program to fit in ... lines by ... columns (check LS)
1010 TIMEOUT ERROR. Galil::command("...") took longer than ... ms to write
1011 TIMEOUT ERROR. Galil::command("...") took longer than ... ms to read : response...
2010 COMMAND ERROR. Galil::command("...") got ? instead of : response...
3010 MONITOR ERROR. Galil::command("...") got > instead of : response. Got...
9080 UNINITIALIZED OBJECT ERROR. Galil::programDownloadFile() called without Galil::address set

```

[top](#)**C++** **string Galil::programUpload()**

Uploads the controller's program to a buffer on the host.

C++ Type	Argument	Example	Description
string	return	"#A\r\nMGTIME\r\nWT500\r\nJP#A\r\nEN"	Uploaded controller program with lines separated by carriage return, new line.

Example 1. Upload program and print to screen

```

C# OutputTextBox.AppendText(g.programUpload()); //Append controller program buffer to form textbox (VB syntax similar)
C++ string buffer = g.programUpload(); // declare string and stuff with uploaded program
cout << buffer; // print buffer to console

```

Throws:

```

1010 TIMEOUT ERROR. Galil::command("...") took longer than ... ms to write
1011 TIMEOUT ERROR. Galil::command("...") took longer than ... ms to read : response...
2010 COMMAND ERROR. Galil::command("...") got ? instead of : response...
3010 MONITOR ERROR. Galil::command("...") got > instead of : response. Got...
9050 UNINITIALIZED OBJECT ERROR. Galil::programUpload() called without Galil::address set

```

C++ void Galil::programUploadFile(string file = "program.dmc")

Accepts a string argument containing the path to a local file where the uploaded program is to be saved. If no path is specified, the executable's directory is used. If a file with the same name and path exists, it will be overwritten. If no matching file exists, a new file will be created.

C++ Type	Argument	Example	Description
string	file	"uploadedProgram.dmc"	Desired filename where the controller program will be saved

Example 1. Upload program and save to file

```
C++ g.programUploadFile("Galil.dmc"); // upload program and save to file
```

Example 2. Display save file chooser and save controller program buffer to location user specifies

```
VB Dim saveFile As New SaveFileDialog() 'dimension file chooser object
saveFile.Filter = "DMC Files (*.dmc)|*.dmc" 'set filter to dmc files
saveFile.ShowDialog() 'display chooser to user
g.programUploadFile(saveFile.FileName) 'save controller buffer to file
```

Throws:

```
4070 FILE ERROR. Galil::programUploadFile() failed to open file...
1010 TIMEOUT ERROR. Galil::command(...) took longer than ... ms to write
1011 TIMEOUT ERROR. Galil::command(...) took longer than ... ms to read : response...
2010 COMMAND ERROR. Galil::command(...) got ? instead of : response...
3010 MONITOR ERROR. Galil::command(...) got > instead of : response. Got...
9070 UNINITIALIZED OBJECT ERROR. Galil::programUploadFile() called without Galil::address set
```

C++ void Galil::arrayDownload(vector<double> array , string name = "array")

Downloads a single, one dimensional array to the controller. If the array name already exists on the controller, the existing array will be deallocated and replaced with the new array (this allows for smaller or larger arrays than the existing array to be downloaded).

Do not execute this API while a Record Array Mode is running on the controller (RC,RD,RA). _RC can be interrogated to ensure the Record Array Mode is idle before downloading to the array table.

C++ Type	Argument	Example	Description
vector<double>	array	(1,2,3,4)	Data on host for download.
string	name	"A"	Name of array on controller where data will be saved. Array does not need to preexist.

Example 1. Downloading an array of numbers to the controller

```
VB Dim myArray() As Integer = {1, 2, 3, 4, 5} 'create an array of information to download to the controller
g.arrayDownload(myArray, "array") 'Create an array on the controller named "array" and load the information from myarray into it
vector<double> arraydata; //declare vector to hold array
C++ for(int i = 0; i < 1000 ; i++)
arraydata.push_back(3.1416); //fill array with 1000 values of pi
g.arrayDownload(arraydata,"A"); //download array to controller array "A"
```

Throws:

```
1010 TIMEOUT ERROR. Galil::command(...) took longer than ... ms to write
1011 TIMEOUT ERROR. Galil::command(...) took longer than ... ms to read : response...
2010 COMMAND ERROR. Galil::command(...) got ? instead of : response...
3010 MONITOR ERROR. Galil::command(...) got > instead of : response. Got...
7101 INVALID ARRAY DOWNLOAD ERROR. Galil::arrayDownload() does not modify ... while RC is running.
9100 UNINITIALIZED OBJECT ERROR. Galil::arrayDownload() called without Galil::address set
```

C++ void Galil::arrayDownloadFile(string file = "arrays.csv")

Allows a group of arrays to be downloaded to the controller from a [csv](#) file.

If there are existing array names on the controller which match array names in the specified csv file, the controller arrays are overwritten. If there are existing array names on the controller not in the csv file, the non-matching arrays are untouched. This means that the csv download function will append to the array table where possible, and overwrite where necessary.

Do not execute this API while a Record Array Mode is running on the controller (RC,RD,RA). `_RC` can be interrogated to ensure the Record Array Mode is idle before downloading to the array table.

C++ Type	Argument	Example	Description
string	file	"dataTable.csv"	Path to csv file on host containing one or more arrays to write to the controller's array table. Note the array names are the first row of the csv.

Example 1. Downloading multiple arrays from a csv file to the controller.

```
VB g.arrayDownloadFile("arrays.csv") 'download the contents of arrays.csv to the controller array table
Label1.Text = g.command("LA") 'write the names and sizes of the controller's arrays (LA) to a form label

C++ g.arrayDownloadFile("array.csv"); //download the contents of array.csv to the controller array table
```

Throws:

```
4112 FILE ERROR. Galil::arrayDownloadFile() failed to open file...
7121 INVALID ARRAY DOWNLOAD ERROR. Galil::arrayDownloadFile() does not modify ... while RC is running.
1010 TIMEOUT ERROR. Galil::command(...) took longer than ... ms to write
1011 TIMEOUT ERROR. Galil::command(...) took longer than ... ms to read : response...
2010 COMMAND ERROR. Galil::command(...) got ? instead of : response...
3010 MONITOR ERROR. Galil::command(...) got > instead of : response. Got...
9120 UNINITIALIZED OBJECT ERROR. Galil::arrayDownloadFile() called without Galil::address set
```

[top](#)

C++ **vector<double> Galil::arrayUpload(string name = "array")**

Uploads a particular array from the controller to a host buffer.

C++ Type	Argument	Example	Description
vector<double>	return	(1.234, 2.456, ...)	Contents of the uploaded array
string	name	"array"	Name of controller-side array to be uploaded. Array names can be interrogated with the "LA" (List Arrays) command.

Example 1. Uploading array data from controller

```
C++ vector<double> arrayUp; //create vector for array
arrayUp = g.arrayUpload("A"); //upload array from controller
```

Example 2. Uploading and printing array data from controller

```
Array arrayUp = (Array)g.arrayUpload("a"); //upload controller array "a" to local host array.

C# for (int i = 0; i < arrayUp.Length; i++)
    OutputTextBox.AppendText(i.ToString() + ": " + arrayUp.GetValue(i) + "\r\n"); //print the uploaded array
```

Throws:

```
1010 TIMEOUT ERROR. Galil::command(...) took longer than ... ms to write
1011 TIMEOUT ERROR. Galil::command(...) took longer than ... ms to read : response...
2010 COMMAND ERROR. Galil::command(...) got ? instead of : response...
3010 MONITOR ERROR. Galil::command(...) got > instead of : response. Got...
9090 UNINITIALIZED OBJECT ERROR. Galil::arrayUpload() called without Galil::address set
```

[top](#)

C++ **void Galil::arrayUploadFile(string file, string names = "")**

Uploads the entire controller array table or a subset thereof and saves it as a [csv](#) file specified by the user.

C++ Type	Argument	Example	Description
----------	----------	---------	-------------

string	file	"c:\\Documents and Settings\\JohnDoe\\array\\array.csv"	The path to the file to which the array data should be saved
string	names	"array1 posX A B"	A space delimited string containing the array names that should be uploaded. A null string, "", uploads all arrays.

Example 1. Uploading a controller array subset to a csv file

```
C++ g.arrayUploadFile("arraysUp.csv", "A B C"); //upload arrays A, B, and C from controller to arraysUp.csv
```

Example 2. Uploading the entire array table to a user-specified path and filename, and then opening that file in Microsoft Excel

```
VB Dim saveFile As New SaveFileDialog() 'dimension file chooser object

saveFile.Filter = "CSV Files (*.csv)|*.csv" 'set filter to csv files
saveFile.ShowDialog() 'display chooser to user
g.arrayUploadFile(saveFile.FileName) 'save controller array table to csv file

'Shell() is used to execute external applications in the Windows shell. The path to Excel is version-specific
Shell("C:\\Program Files\\Microsoft Office\\Office10\\EXCEL.EXE """" + saveFile.FileName + """"", AppWinStyle.MaximizedFocus, False)
```

Throws:

```
4110 FILE ERROR. Galil::arrayUploadFile() failed to open file...
1010 TIMEOUT ERROR. Galil::command(...) took longer than ... ms to write
1011 TIMEOUT ERROR. Galil::command(...) took longer than ... ms to read : response...
2010 COMMAND ERROR. Galil::command(...) got ? instead of : response...
3010 MONITOR ERROR. Galil::command(...) got > instead of : response. Got...
9110 UNINITIALIZED OBJECT ERROR. Galil::arrayUploadFile() called without Galil::address set
```

[top](#)

C++ void Galil::firmwareDownloadFile(string file = "firmware.hex")

Allows controller firmware to be upgraded. Note that Galil distributes firmware files in a zip archive which must be decompressed prior to loading. Firmware files end in a .hex extension. Ensure the correct firmware version/model before downloading (loading an incompatible firmware will require a reloading of the correct firmware).

DMC-21x3: this function is only allowed via RS-232.

Upon invoking firmwareDownloadFile() a GUI progress bar dialog will display.

C++ Type	Argument	Example	Description
string	file	"c:\\Documents and Settings\\JohnDoe\\Galil Firmware\\d400rcur.hex"	The path to the firmware hex file to download. If no path is specified, the executable directory is used.

Example 1. Ask user for firmware file path and update controller firmware

```
VB Dim chooseFile As New OpenFileDialog() 'dimension file chooser object

chooseFile.Filter = "Hex Files (*.hex)|*.hex" 'set filter to hex files
chooseFile.ShowDialog() 'display chooser to user
g.firmwareDownloadFile(chooseFile.FileName) 'download firmware

C++ string filename; //create string for filename
cout << "Enter firmware path to download:" << endl; //prompt user

getline(cin, filename); //get input from the console (hex file expected) -- getline requires #include <fstream>
g.firmwareDownloadFile(filename); //update controller firmware
```

Throws:

```
4130 FILE ERROR. Galil::firmwareDownloadFile() failed to open file...
6130 WRONG BUS ERROR. Galil::firmwareDownloadFile() isn't allowed via Ethernet. Use RS-232
1010 TIMEOUT ERROR. Galil::command(...) took longer than ... ms to write
1011 TIMEOUT ERROR. Galil::command(...) took longer than ... ms to read : response...
2010 COMMAND ERROR. Galil::command(...) got ? instead of : response...
3010 MONITOR ERROR. Galil::command(...) got > instead of : response. Got...
2010 COMMAND ERROR. Galil::command(...) got ? instead of > response...
9130 UNINITIALIZED OBJECT ERROR. Galil::firmwareDownloadFile() called without Galil::address set
```

[top](#)

C++ string Galil::read()

Allows for direct read access to the input buffer from the controller. This function is not intended for normal use, as the other APIs provide a higher level of functionality. This function should be used in cases of firmware specials which break the standard Galil communication standards, or for users who wish to implement their own lower-level data-parsing algorithms. See [Detailed Description](#) for a graphical overview of how the input buffer is parsed.

C++ Type	Argument	Example	Description
string	return	"123\r\n"	Actual bytes read

Example 1. Read generic data from controller

C++	<pre> string r; g.command("CW2"); //controller won't set MSB for MGs. Note Galil software usually expects CW1 while (!) { r = g.read(); //read any available data if (r.size() != 0) //if non-empty cout << r; //print } //while </pre>
VB	<pre> Dim r As String g.command("CW2") 'controller won't set MSB for MGs. Note Galil software usually expects CW1 While (Not Console.KeyAvailable) 'no key pressed r = g.read() 'read any available data If (r.Length <> 0) Then 'if non-empty Console.WriteLine(r) 'print End If End While </pre>

Note: The status of the data adjustment bit (CW) will influence how unsolicited messages are returned from the controller.

Throws:
9210 UNINITIALIZED OBJECT ERROR. Galil::read() called without Galil::address set

[top](#)

C++ int Galil::write(string bytes = "r")

Allows for direct writes to the controller. This function is not intended for normal use, rather for special applications where command() can not be used. Note: command() can be used for the majority of even special applications.

C++ Type	Argument	Example	Description
int	return	18	The actual number of bytes written (which can be less than the length of the "bytes" argument below)
string	bytes	"MGTIME\rTPX\rMGTIME\r"	Buffer to write to the controller. Length must be less than or equal to 511 bytes.

Example 1. write/read transaction

C++	<pre> string buffer = ""; //buffer to hold read data g.write("\x12\x16\r"); //write <ctrl>R<ctrl>V to the controller while (buffer == "") buffer = g.read(); //while read returns nothing keep reading cout << "read() returned " << buffer.size() << " bytes. It said " << buffer << endl; //print response </pre>
VB	<pre> Dim buffer As String = "" 'buffer to hold read data g.write(Chr(18) + Chr(22) + vbCr) 'write <ctrl>R<ctrl>V to controller (Chr takes ascii decimal, not hex) While (buffer.Length = 0) 'while read returns nothing buffer = g.read() 'keep reading End While Console.WriteLine("Read() returned " + buffer.Length.ToString + " bytes. It said " + buffer + vbCrLf) 'print response </pre>

Example 2. Program download with insertion, or while motion is profiling

C++	<pre> /* programDownload() and programDownloadFile() do not support download with insertion or download while motion is profiling Assume file 'dmcFile' is open and 'lines' contains the number of program lines in the file (less REM lines) 'lineCap' contains the max number of lines downloadable Program line capacity: DMC-40x0, 18x6, 41x3 = 2000 RIO-47xx0 = 200 </pre>
-----	---

```

RIO-47xx2 = 400
Others = 1000
'charsCap' contains the number of characters per line
RIO-xxxx = 40
Others = 80

'location' is a std::string containing the insertion point
""
download at beginning of program buffer
#Label Begins download at line following #Label (more lines checking required)
# Begins download at end of program in RAM (more lines checking required)
*/
C++ if (lines <= lineCap)
{
    g.write("DL"+location+"\r"); //start the download

    while (!dmcFile.eof())
    {
        getline(dmcFile, line);
        if (line.substr(0,3) != "REM") //filter out REM lines
        {
            if (line.size() > charsCap)
            {
                cout << "Line too long: " << line << endl;
                g.write("\n"); //close write session on controller
                return 1;
            }
            g.write(line + "\r");
        }
    } //while
    g.write("\n"); //close write session on controller
}
else
    cout << "Too many lines: " << lines << endl;

```

Throws:
9200 UNINITIALIZED OBJECT ERROR. Galil::write() called without Galil::address set

[top](#)

C++	int Galil::interrupt(int timeout_ms = 500)
VB Event	Sub onInterrupt(ByVal status As Integer)
C# Event	void onInterrupt(int status)

Provides access to the PCI and Ethernet interrupt status byte for both the EI and UI conditions. DMC-18x2/6 and DMC-40x0 Rev 1.0b only.

C++: When invoked, this function will sleep until an interrupt is received (or timeout expires) and returns the controller's status byte. A timeout of zero will check for an interrupt condition but will not throw an error. A timeout of -1 will block until an interrupt occurs.

COM: The onInterrupt event does not have a timeout.

C++ Type	Argument	Example	Description
int	return	0xf0	The returned status byte from an interrupt event. See chapter 4 in PCI controller manual for a reference table. See EI in the DMC-40x0 command reference for further Ethernet information.
int	timeout_ms	100	time in milliseconds to wait for interrupt to occur

Example 1. Returning status byte from an interrupt.

```

VB Private Sub g_onInterrupt(ByVal status As Integer) Handles g.onInterrupt 'subroutine runs when Interrupt event occurs
    interruptsTextBox.AppendText(status.ToString + vbCrLf) 'print status byte to text box
End Sub

C# //in initialization code block
g.onInterrupt += new Galil.Events_onInterruptEventHandler(g_onInterrupt); //hook up to the onInterrupt Event

//event subroutine
void g_onInterrupt(int status){ //handler for the Interrupt event
    interruptsTextBox.AppendText(status.ToString() + "\r\n"); }

C++ int status = g.interrupt(100); //sleep for interrupt 100ms

```

Throws:
1040 TIMEOUT ERROR. Galil::interrupt() took longer than ... ms to read status byte
6040 WRONG BUS ERROR. Galil::interrupt() not supported. Use Galil::message()

C++ static string Galil::libraryVersion()

Returns a string containing the library version. This is a static function and does not require a Galil object to be instantiated (C++) or address property to be set (Com). The string returned will resemble: *1.0.0.0 Jan 30 2008 14:32:55 libGalil.so*

C++ Type	Argument	Example	Description
string	return	"0.1.0.4 Aug 20 2008 18:28:39 GalilClass0.dll 1.1.0.4 Aug 20 2008 18:28:19 Galil1.dll"	The library version. COM version (line 1) is also included when using it.

Example 1. Printing the library version

```
C++ cout << Galil::libraryVersion() << endl; //print library version information to console
VB statusTextBox.AppendText(g.libraryVersion) 'Printing library version info to text box
```

Throws: none

C++ vector<string> Galil::sources()

Returns an array of strings corresponding to the names of all available data record sources. A source (e.g. _TPA) is used as a key into sourceValue() to return a source's value from a given record. It is also the key into sourceUnits() and sourceDescription() which return static but useful information about a source's unit of measure and description.

The returned sources array is static for a particular controller and so therefore need only be retrieved once per connection.

The array is returned in alphabetic order.

See the [detailed description](#) for more information.

C++ Type	Argument	Example	Description
vector<string>	return	("@AN[1]","@AN[2]","@IN[01]","@IN[02]",...,"_TVB","_ZAA","_ZAB")	An array of strings containing source names in the controller data record.

Example 1. Retrieving sources and assigning a source name to a string variable.

```
VB Dim s As Object = g.sources() 'create an object to hold all available sources
Dim firstSource As String = s(0) 'create a string to hold the first source name
MsgBox("The first available source is " + firstSource) 'print the first source name in the data record

C# Array s = (Array)g.sources(); //get all sources available on this controller
String firstSource = s.GetValue(0).ToString(); //get the first source name for example
label1.Text = label1.Text + "The first available source is : " + firstSource + "\r\n";

C++ const vector<string> s = g.sources(); //create a vector variable to hold all available sources
const string firstSource = s[0]; //create a string to hold the name of the first source
cout << "The first available source is " << firstSource << endl;
```

Example 2. Printing all available sources

```
VB Dim s As Object = g.sources() 'create an object to hold all available sources
For Each element As String In s 'step through sources and print every source name
    watchTextBox.AppendText(element + vbCrLf)
Next

C++ vector<string> s = g.sources(); //create a vector to hold all available sources
for (int i = 0; i < s.size(); i++) //step through sources and print every source name
```

```
cout << s[i] << endl;
```

Throws:

9190 UNINITIALIZED OBJECT ERROR. Galil::sources() called without Galil::address set

[top](#)

C++ void Galil::recordsStart(double period_ms = -1)

Issues a DRn command to the controller to begin the asynchronous DR method of data record acquisition. The actual argument sent to the controller depends on the controller, but the fastest supported DR rate is configured by default. If period_ms > 0, the library will attempt to set DR to period_ms milliseconds based upon the controller's servo update rate (TM). If period_ms is unattainable, recordsStart() will set DR to an approximate value, throw an error, or set DR to zero (depending on the value of period_ms).

The standard 18x2 firmware does not support the DR data record method (use "QR" in record()). For stand-alone controllers, DR is valid when connected over Ethernet, but not over RS-232.

A UDP handle will be opened on Ethernet controllers to support the DR data.

Consult the controller command reference under DR for more information.

C++ Type	Argument	Example	Description
double	period_ms	32	The desired period in true milliseconds of the asynchronous data dispatch or -1 for as fast as possible

Example 1. Starting the data record with DR and checking the current DR value.

```
VB
g.recordsStart() 'start the records via DR mode
MsgBox("Started records in DR mode. DR argument is " + g.command("MG_DR")) 'check sample rate

C#
g.recordsStart(-1); //start data records in DR mode
label1.text = g.command("MG_DR"); //check sampling rate

C++
g.recordsStart(); //start the records via DR mode
cout << "Started records in DR mode. DR argument is " << g.command("MG_DR") << endl; //check current sample rate
```

Throws:

6140 WRONG BUS ERROR. Galil::recordsStart() isn't allowed via RS-232. Use Ethernet
5140 OPEN ERROR. Galil::recordsStart() failed to open UDP handle on port...
1010 TIMEOUT ERROR. Galil::command(...) took longer than ... ms to write
1011 TIMEOUT ERROR. Galil::command(...) took longer than ... ms to read : response...
2010 COMMAND ERROR. Galil::command(...) got ? instead of : response...
3010 MONITOR ERROR. Galil::command(...) got > instead of : response. Got...
9140 UNINITIALIZED OBJECT ERROR. Galil::recordsStart() called without Galil::address set

[top](#)

C++ (QR & DR) vector<char> Galil::record(string method = "QR")

VB (QR) Function record(Optional ByVal method As String = "QR") As Object

VB Event (DR) Sub onRecord(ByVal record As Object)

C# (QR) public virtual object record(string method)

C# Event (DR) void onRecord(object record)

Returns the controller's data record. If the QR method is specified, this function will perform a synchronous record acquisition by sending QR and waiting for the response (somewhat like command()). If the method is DR, this function will wait for the asynchronous transmission from the DR mode. Note, recordsStart() must be called prior to record() if the DR method is to be used. If the record is not returned within [timeout_ms](#), the function will throw a timeout error.

COM: the onRecord event will occur when a DR record is received. The record() function is also available for the QR method.

C++ Type	Argument	Example	Description
		binary	

vector<char>	return	<i>unitary data</i>	record() will return this vector filled with the data record bytes.
string	method	"QR"	A string (either "QR" or "DR") specifying the data record acquisition method to be used.

Example 1. Getting a record() using QR.

```
VB Dim r As Object = g.record() 'get a record from the controller using QR (default) method
C# Object r = g.record("QR"); //get a record using QR
C++ vector<char> r = g.record(); //get a record from the controller using QR method
```

Example 2. Getting a record() using DR.

```
VB Public Class Form1
    Dim WithEvents g As Galil.Galil 'dimension Galil object WithEvents enabled

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Try
            g = New Galil.Galil
            g.address = "192.168.1.11" 'connect to Ethernet controller
            Me.Text = g.connection() 'print connection information to form titlebar

            g.recordsStart(100) 'start records at 100 ms period
            Catch ex As System.Runtime.InteropServices.COMException
                TextBox1().AppendText(ex.Message)
            End Try
        End Sub
    Private Sub g_onRecord(ByVal record As Object) Handles g.onRecord 'event to handle record

        TextBox1.AppendText(g.sourceValue(record, "TIME").ToString + vbCrLf) 'print TIME source for each record

    End Sub
End Class

C# Galil.Galil g; //galil object variable

private void Form1_Load(object sender, EventArgs e)
{
    try
    {
        g = new Galil.Galil(); //create a new object

        g.address = "192.168.1.11"; //connect to controller
        this.Text = g.connection(); //print connection in form titlebar

        g.onRecord += new Galil.Events_onRecordEventHandler(g_onRecord); //hook up to event

        g.recordsStart(100); //start records at 100 ms period
    } catch (System.Runtime.InteropServices.COMException ex) { //catch any errors

        textBox1.AppendText(ex.Message);
    } //try
} //Form1_Load
void g_onRecord(object record) //event runs when record received

{
    textBox1.AppendText(g.sourceValue(record, "TIME").ToString() + "\r\n"); //print time
}

C++ g.recordsStart(); //start the asynchronous DR mode
vector<char> record = g.record("DR"); //get a record via DR
```

Throws:

```
6150 WRONG BUS ERROR. Galil::record("DR") not supported on RS-232. Use Ethernet or Galil::record("QR")
1150 TIMEOUT ERROR. Galil::record("DR") took longer than ... ms to read ... bytes
1151 TIMEOUT ERROR. Galil::record("DR") took longer than ... ms to read ... bytes
1152 TIMEOUT ERROR. Galil::record("QR") took longer than ... ms to write
1153 TIMEOUT ERROR. Galil::record("QR") took longer than ... ms to read ... bytes. Got ... bytes
9150 UNINITIALIZED OBJECT ERROR. Galil::record() called without Galil::address set
```

[top](#)

```
C++ double Galil::sourceValue(vector<char> record, string source = "TIME")
```

Maps a data record source name to its value. Given a data record returned from record() and a valid source name as a key, this function returns the appropriate value.

If an invalid source name is specified to sourceValue(), the value of TIME is returned.

C++ Type	Argument	Example	Description
double	return	6.4306	The value returned corresponding to source
<code>vector<char></code>	record	<i>binary data</i>	A vector (as returned from record()) containing one valid controller data record
string	source	"@AN[1]"	A valid string as found in the sources() array specifying the desired data

Example 1. Returning the TIME value from a data record buffer.

```
VB Dim r As Object = g.record("QR") 'Get a record. In this example, the QR method is used
Label3.Text = g.sourceValue(r, "TIME").ToString 'Print the current TIME value into a form label
```

Example 2. Display the value of the first source of the data record.

```
C# label1.Text = label1.Text + firstSource + " value is: " + g.sourceValue(r, firstSource) + "\r\n";
C++ cout << firstSource << " value is: " << g.sourceValue(r, firstSource) << endl; //display value
```

Throws:
9160 UNINITIALIZED OBJECT ERROR. Galil::sourceValue() called without Galil::address set

[top](#)

C++ string Galil::source(string field = "Description", string source = "TIME")

Provides access to source related fields. By specifying a valid source field name and source, the field's value is returned.

C++ Type	Argument	Example	Description
string	return	"Sample counter"	The value returned corresponding to the specified source's specified field
string	field	"Description"	The field of interest for a specified source. Valid fields are "Description", "Units", "Scale", & "Offset"
string	source	"TIME"	A valid source as found in the sources() array specifying the source in which to lookup field

Example 1. Printing all fields of a source.

```
VB TextBox1.AppendText(g.source("Description", "@AN[1]") + vbCrLf)
TextBox1.AppendText(g.source("Units", "@AN[1]") + vbCrLf)
TextBox1.AppendText(g.source("Scale", "@AN[1]") + vbCrLf)
TextBox1.AppendText(g.source("Offset", "@AN[1]") + vbCrLf)
'Prints:
'Analog input 1
'v
'3276.8
'0
```

Throws:
9170 OFFLINE ERROR. Galil::source() called without Galil::address set

[top](#)

C++ void Galil::setSource(string field = "Description", string source="TIME", string to= "Sample counter")

Used to set the various fields of a data record source to custom values instead of Galil factory defaults.

C++ Type	Argument	Example	Description
string	field	"Description"	The field to set for a specified source. Valid fields are "Description", "Units", "Scale", & "Offset" This argument is overloaded to take a file path to a GalilTools project file. This allows loading data record configurations created from the GalilTools Watch-All tab. Leave the rest of the arguments blank.
string	source	"TIME"	A valid source as found in the sources() array specifying the source in which to

string	source	FIELD	change field
string	to	"16 bit upcounter"	The value to assign to the specified field in the specified source

Changing the Scale and Offset fields will affect the values returned from `sourceValue()`. These fields are used for pre-processing the raw controller data record before being passed up to the API. Set `scale=1` and `offset=0` to receive the raw controller data record values from `sourceValue()`. See the [Watch](#) chapter for more information.

Example 1. Modifying fields of a source.

```

'print default values
TextBox1.AppendText("@AN[1]=" + g.sourceValue(g.record("QR"), "@AN[1]").ToString)
TextBox1.AppendText(" " + g.source("Units", "@AN[1]"))
TextBox1.AppendText(" (" + g.source("Description", "@AN[1]"))
TextBox1.AppendText(" with scale " + g.source("Scale", "@AN[1]"))
TextBox1.AppendText(" and offset " + g.source("Offset", "@AN[1]") + ")" + vbCrLf)
'change fields
g.setSource("Description", "@AN[1]", "RAW ADC Counts")
g.setSource("Units", "@AN[1]", "counts(signed 16bit)")
g.setSource("Scale", "@AN[1]", "1")
'print new values
TextBox1.AppendText("@AN[1]=" + g.sourceValue(g.record("QR"), "@AN[1]").ToString)
TextBox1.AppendText(" " + g.source("Units", "@AN[1]"))
TextBox1.AppendText(" (" + g.source("Description", "@AN[1]"))
TextBox1.AppendText(" with scale " + g.source("Scale", "@AN[1]"))
TextBox1.AppendText(" and offset " + g.source("Offset", "@AN[1]") + ")"
'prints, depending on voltage on analog input one
'@AN[1]=-9.248046875 V (Analog input 1 with scale 3276.8 and offset 0)
'@AN[1]=-30256 counts(signed 16bit) (RAW ADC Counts with scale 1 and offset 0)

```

Example 2. Loading a GalilTools project file.

```

VB g.setSource("C:\Documents and Settings\user\Galil\default.project")

```

Throws:
9150 OFFLINE ERROR. Galil::record() called without Galil::address set

[Contents](#)

Hello

This section details how to get started using the [GalilTools Communication Library](#) with step-by-step examples that show how to print a controller's connection information (e.g. "GALILPC11, DMC1886 Rev 1.0b, 36") using popular programming languages. See the [Hello Galil](#) project on Galil's website for a complete list of examples.

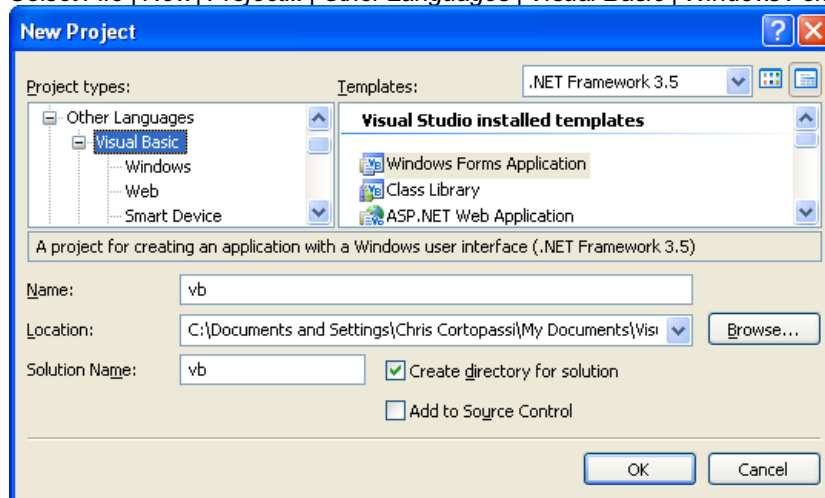
- [Visual Basic](#)
- [C#](#)
- [C++/CLI](#)
- [Visual C++ 2008](#)
- [g++ on Linux](#)
- [LabVIEW](#)

Make sure GalilTools is [installed](#) and a [connection](#) to the controller can be established from that application. A try-catch error handler is required for most Galil functions, but is left out here for brevity. Please see the [Library Reference](#).

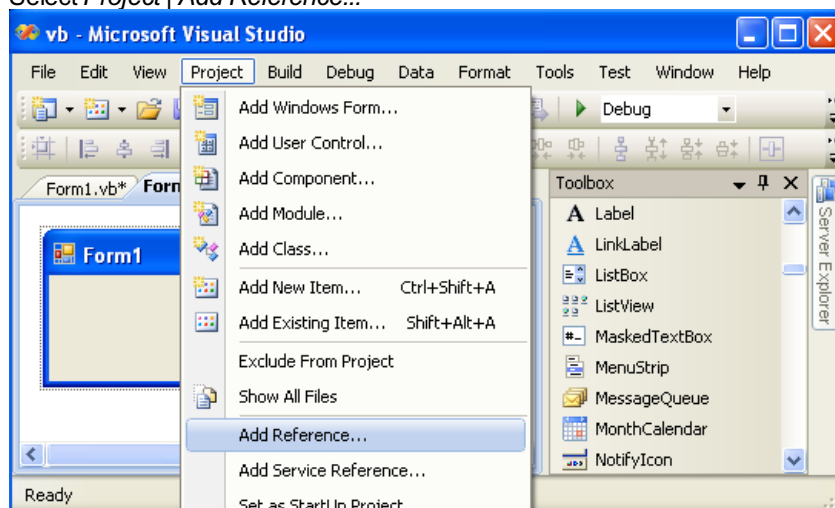
Visual Basic

[Visual Studio](#) or the free-of-charge [Express](#) edition may be used (below configures Visual Studio 2008, but versions back to 6 may be used)

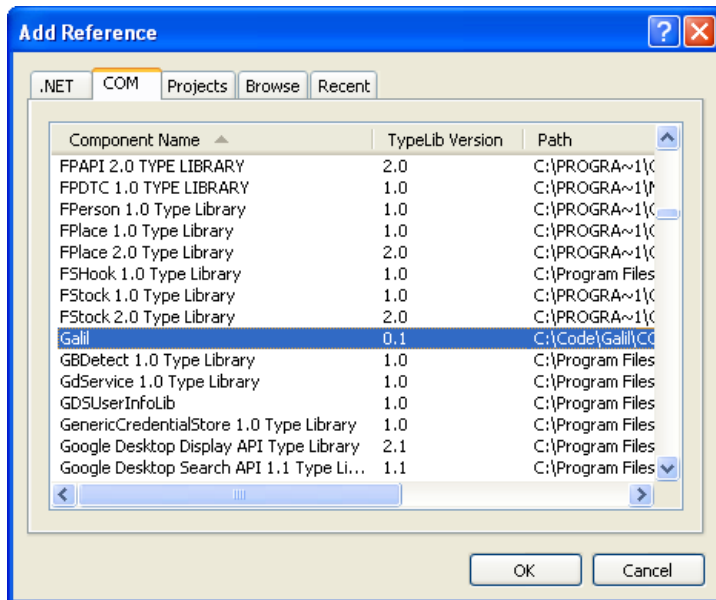
1. Select **Start | All Programs | Microsoft Visual Studio 2008 | Microsoft Visual Studio 2008**
2. Select **File | New | Project... | Other Languages | Visual Basic | Windows Forms Application** and click **OK**



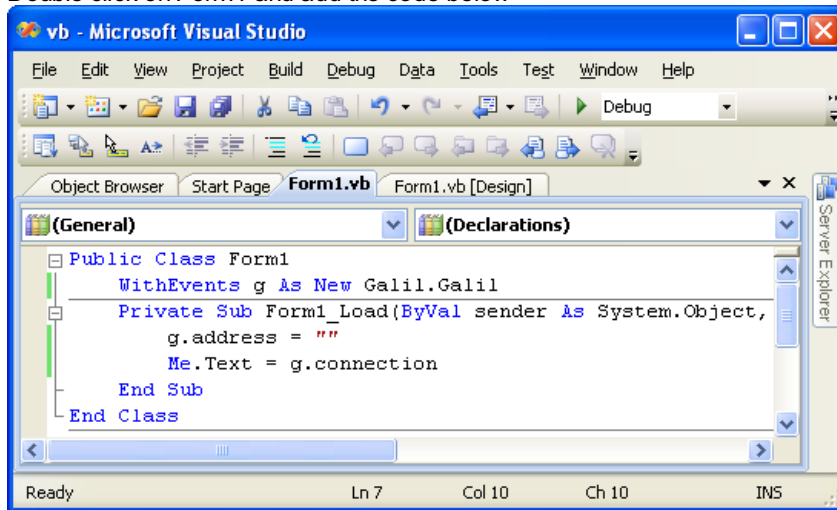
3. Select **Project | Add Reference...**



4. Choose the **COM** tab, select **Galil**, and click **OK**



- Double click on *Form1* and add the code below



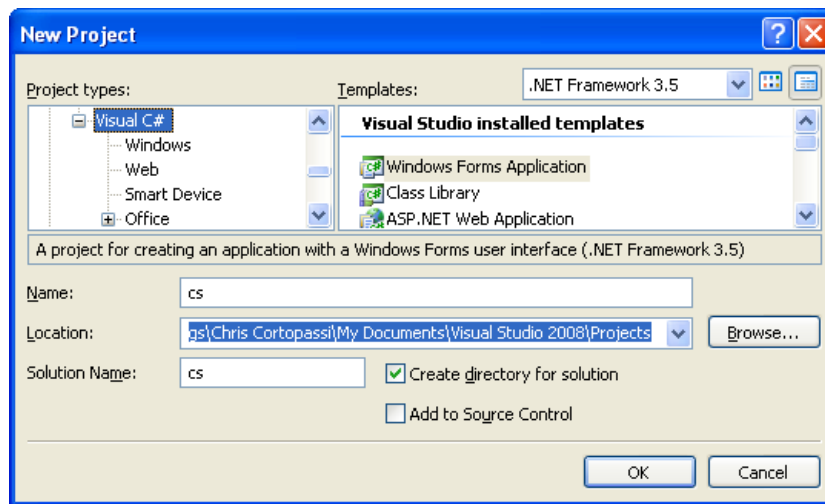
- Hit **F5** to run the program



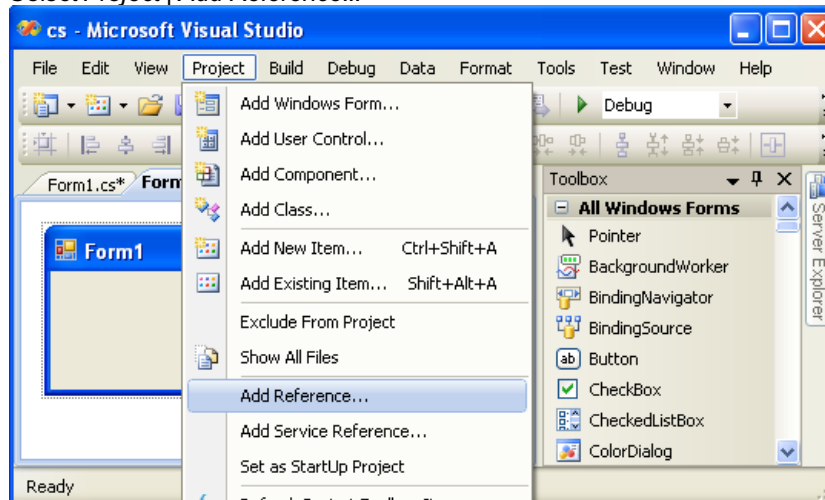
C#

[Visual Studio](#) or the free-of-charge [Express](#) edition may be used (below configures Visual Studio 2008, but versions back to 2002 may be used)

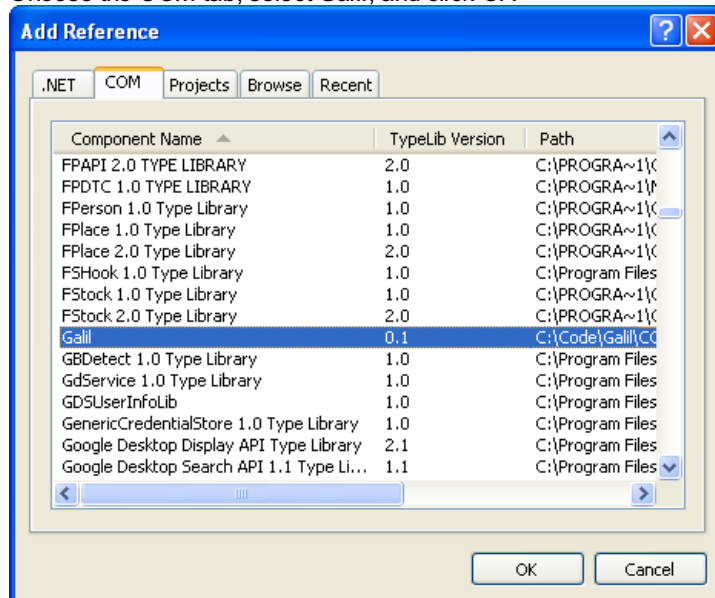
- Select *Start | All Programs | Microsoft Visual Studio 2008 | Microsoft Visual Studio 2008*
- Select *File | New | Project... | Other Languages | Visual C# | Windows Forms Application* and click *OK*



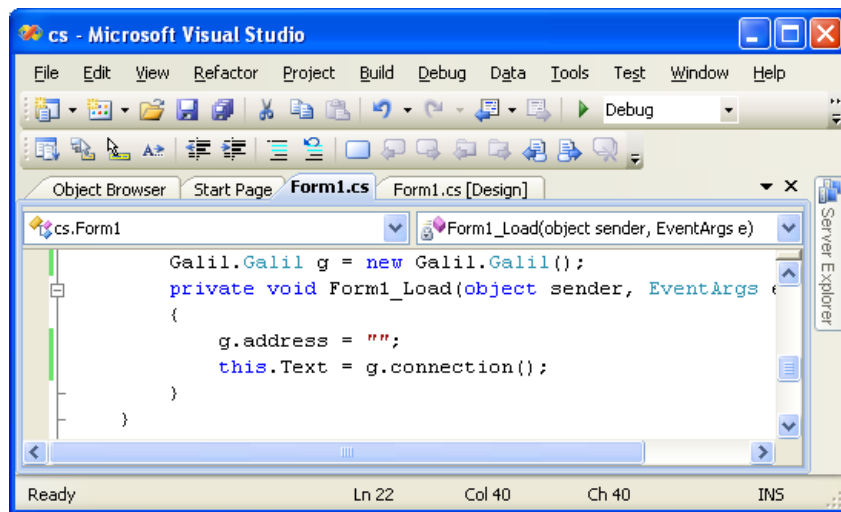
3. Select *Project* | *Add Reference...*



4. Choose the *COM* tab, select *Galil*, and click *OK*



5. Double click on *Form1* and add the code below



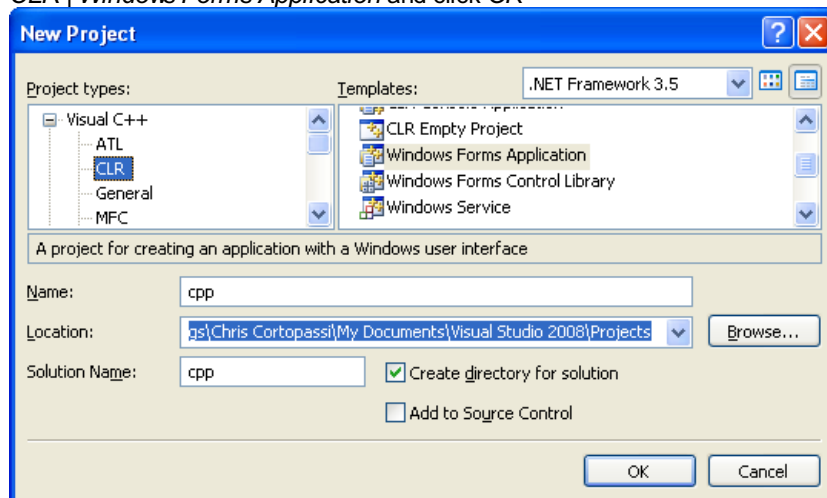
6. Hit **F5** to run the program



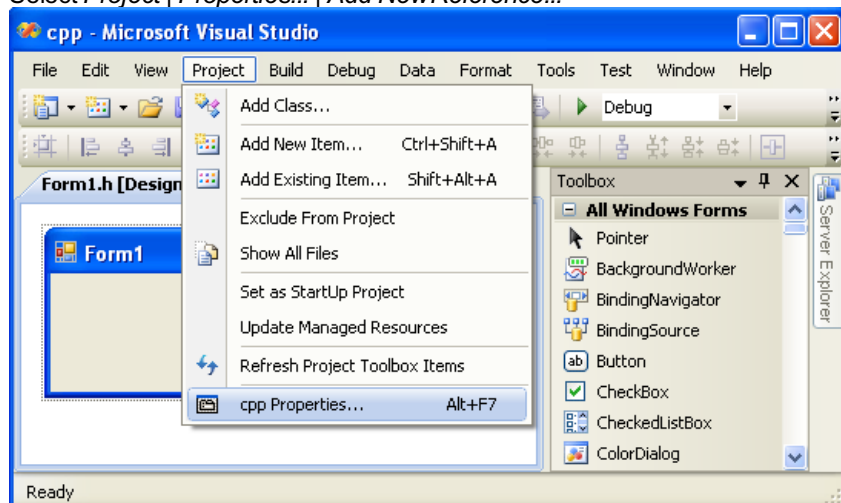
C++/CLI

[Visual Studio](#) or the free-of-charge [Express](#) edition may be used (below configures Visual Studio 2008, but 2005 may also be used). This section covers C++/CLI (Common Language Infrastructure) targeting the CLR (Common Language Runtime). For native (unmanaged) C++, see [Visual C++ 2008](#)

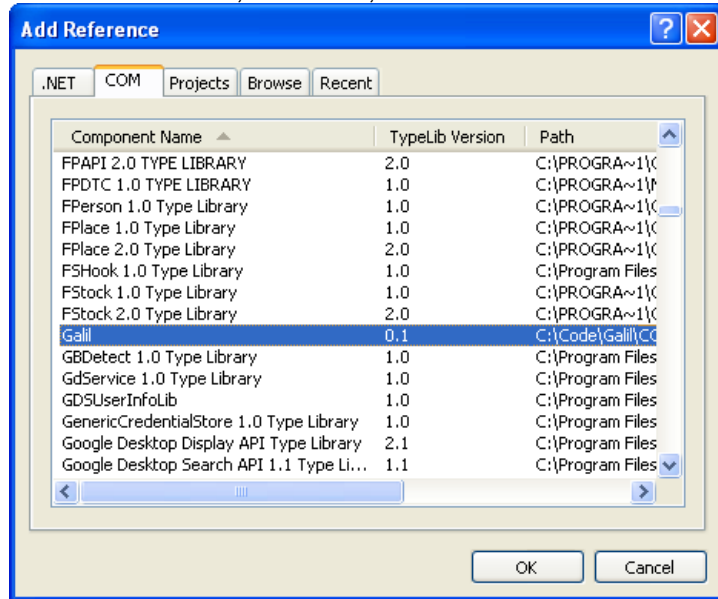
1. Select *Start | All Programs | Microsoft Visual Studio 2008 | Microsoft Visual Studio 2008 | File | New | Project... | Visual C++ | CLR | Windows Forms Application* and click **OK**



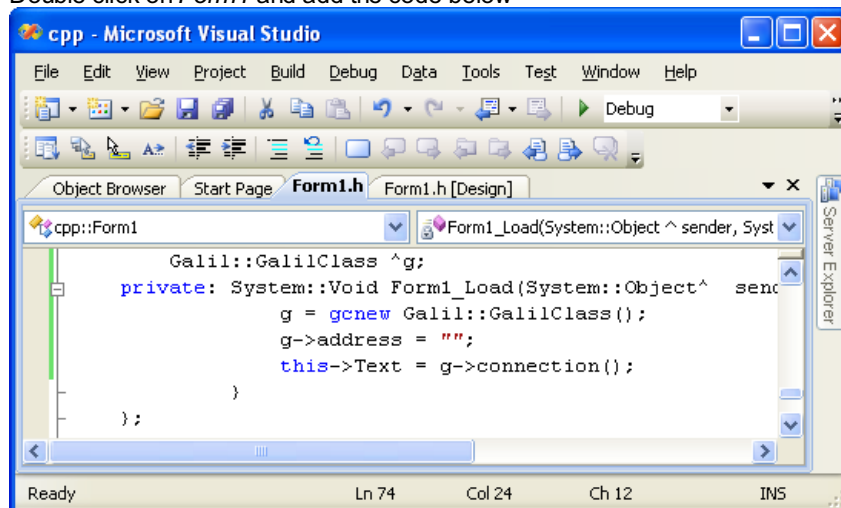
2. Select *Project | Properties... | Add NewReference...*



3. Choose the *COM* tab, select *Galil*, and click *OK*



4. Double click on *Form1* and add the code below



5. Hit **F5** to run the program



Visual C++ 2008

[Visual Studio](#) or the free-of-charge [Express](#) edition may be used (contact Galil if another C++ compiler is required). This section covers native C++. For managed C++, see [C++/CLI](#)

1. Open a compiler command prompt

Start | All Programs | Microsoft Visual Studio 2008 | Visual Studio Tools | Visual Studio 2008 Command Prompt

2. Change to the directory containing the example C++ program (if you're not an Administrator, you'll need to copy the lib folder to %HOMEPATH% and work there)

```
C:\Program Files\Microsoft Visual Studio 9.0\VC>cd %ProgramFiles%\Galil\GalilTools\lib
```

3. Compile

```

C:\Program Files\Galil\GalilTools\lib>cl hello.cpp Galil1.lib /EHsc /MD
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 15.00.21022.08 for 80x86
Copyright (C) Microsoft Corporation. All rights reserved.

hello.cpp
Microsoft (R) Incremental Linker Version 9.00.21022.08

```

```
/out:hello.exe
hello.obj
Galil1.lib
```

4. Tell Windows where to find Galil1.dll

```
C:\Program Files\Galil\GalilTools\bin>set path=%path%;%ProgramFiles%\Galil\GalilTools\bin
```

5. Run

```
C:\Program Files\Galil\GalilTools\lib>hello
1.1.0.4 Aug 29 2008 14:15:14 Galil1.dll
GALILPCI2, DMC1886 Rev 1.0b, 988
...
```

6. If you want to use the GUI debugger (devenv.exe or vcexpress.exe), you'll need to

- add %ProgramFiles%\Galil\GalilTools\bin to your PATH (*Start | Control Panel | System | Advanced | Environment Variables*)
- File | New | Project... | Visual C++ | General | Empty Project*
- In the *Solution Explorer*, right click *Source Files | Add | Existing Item... | C:\Program Files\Galil\GalilTools\lib\hello.cpp*
- Set *Project | Properties... | Configuration Properties | C/C++ | General | Additional Include Directories* to "**C:\Program Files\Galil\GalilTools\lib**" (QUOTES IMPORTANT!)
- Set *Project | Properties... | Configuration Properties | C/C++ | Code Generation | Runtime Library* to *Multi-threaded DLL (/MD)*
- Set *Project | Properties... | Configuration Properties | Linker | Input | Additional Dependencies* to "**C:\Program Files\Galil\GalilTools\lib\Galil1.lib**" (QUOTES IMPORTANT!)

Note: GalilTools ships with release libraries (/MD switch above). In order to use debug (/MDd) the user **MUST** download the debug version of the Galil lib.

Debug libraries and libraries for other versions of Visual Studio are available in the GalilTools archive:

<http://www.galilmc.com/support/downloads/software/galiltools/windows/>

g++ on Linux

1. Change to the lib directory

```
[dj@localhost Galil]$ cd lib
```

2. Compile

```
[dj@localhost lib]$ g++ hello.cpp -L. libGalil.so
```

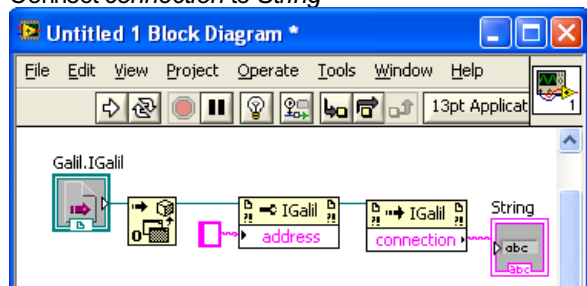
3. Run

```
[dj@localhost lib]$ ./a.out
1.1.0.4 Aug 29 2008 14:15:14 libGalil.so
/dev/galilpci0, DMC1886 Rev 1.0b, 988
...
```

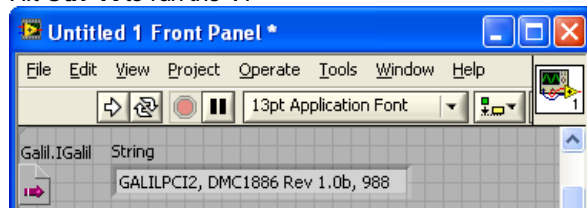
LabVIEW

- Open [National Instruments LabVIEW](#) and select *Blank VI*
- On the *Front Panel*, select *View | Controls Palette | Search*, type **String Indicator**, and drop it onto the *Front Panel*
- Choose *Window | Show Block Diagram*
- Choose *View | Functions Palette | Search*, type **Automation Open**, and drop it onto the *Block Diagram*
- Right-click on the *Automation Open* block and choose *Select ActiveX Class | Browse...*
- Choose the *Galil Version x.x Type Library*, click the *Galil Object*, and click *OK*
- Right-click on the *Automation Open* block, choose *ActiveX Palette | Property Node*, and drop it onto the *Block Diagram*
- Connect the *Automation Refnum* terminal of the *Automation Open* block to the *reference* terminal on the *Property Node*
- Left-click on the *Property* element of the *Property Node* and choose *address*
- Choose *View | Functions Palette*, type **String Constant**, drop it onto the *Block Diagram*, and connect it to *address*
- Right-click on the *Automation Open* block, choose *ActiveX Palette | Invoke Node*, and drop it onto the *Block Diagram*
- Connect the *Galil.IGalil* (reference output) terminal of the *Property Node* to the *reference* terminal on the *Invoke Node*

13. Left-click on the *Method* element of the *Invoke Node* and choose *connection*
14. Connect *connection* to *String*



15. Hit **Ctrl+R** to run the VI



[Contents](#)