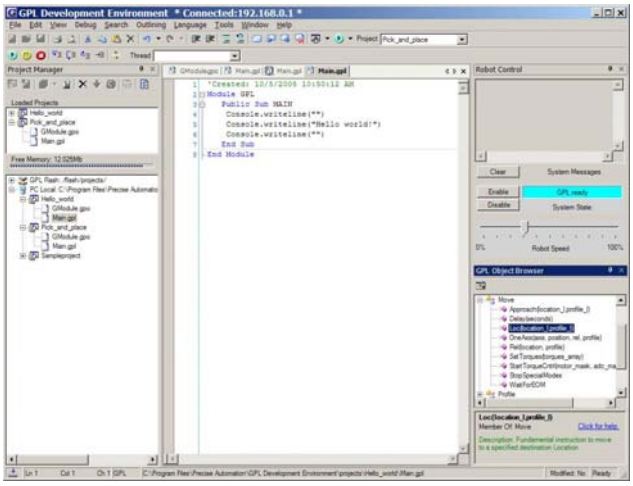
PRECISE AUTOMATION

Guidance Development Environment



Introduction and Reference Manual

Version 3.0.0, May 18, 2010
P/N: GDE0-DI-00010

Precise Automation Inc., 727 Filip Road, Los Altos, California 94024
www.preciseautomation.com

Document Content

Trademarks

Disclaimer

Precise Automation Inc.
727 Filip Road
Los Altos, California 94024
U.S.A.
www.preciseautomation.com

Warning Labels

The following warning and caution labels are utilized throughout this manual to convey critical information required for the safe and proper operation of the hardware and software. It is extremely important that all such labels are carefully read and complied with in full to prevent personal injury and damage to the equipment.

There are four levels of special alert notation used in this manual. In descending order of importance, they are:

DANGER: This indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.

WARNING: This indicates a potentially hazardous situation, which, if not avoided, could result in serious injury or major damage to the equipment.

CAUTION: This indicates a situation, which, if not avoided, could result in minor injury or damage to the equipment.

NOTE: This provides supplementary information, emphasizes a point or procedure, or gives a tip for easier operation

Table Of Contents

Guidance Development Environment

GDE Introduction

The Guidance Development Environment (GDE) is a software package that allows you to develop and debug Guidance Programming Language (GPL) software projects. GDE runs on a Windows PC. To execute and debug programs, the PC must be connected to the controller via Ethernet either locally or remotely. However, for offline development, the PC need not be connected to a Guidance Controller.

In this document, instructions for installing and executing GDE are provided. This is followed by a description of each of the components of GDE, their basic operation, and the functions that they provide. Finally, the step-by-step process of developing and executing the traditional "Hello World" program is presented along with a simple robot pick-and-place application.

In order to execute the examples in this document, the following are required:

- A 500 MHz or faster PC running Windows 2000 or Windows XP
- Microsoft Internet Explorer version 6.x or later
- A 10/100 Ethernet card for the PC
- At least 20 MB of space on the PC's disk
- A 2x CD-ROM drive interfaced to the PC
- The Guidance Development Suite distribution disk (P/N: PDS0-DA-00010)
- A Guidance Controller interfaced to a robot, e.g. a PrecisePlace 2300 or 2400

WARNING: Before proceeding with this Guide, please ensure that the following steps have already been performed:

- The robot has been properly mounted, all required safety interlocks have been installed and tested, and power has been connected. For the PrecisePlace robots, this information is provided in the *"PrecisePlace 2300/2400 Robot, Hardware Introduction and Reference Manual"*.
- If you are integrating the Guidance Controller to a new mechanical system for the first time, please see the section on "Setting Up a Controller" in the *Precise Documentation Library* for instructions on configuring the controller.

Prior to reading this document, it is highly recommended that you first do the following:

- Perform the exercises in the *"Guidance Controller Setup and Operation, Quick Start Guide"* to familiarize yourself with the operation of the controller and to verify that the controller has been properly interfaced to the PC.

- Read the *“Guidance Programming Language, Introduction to GPL”* to gain a basic understanding of the functions available in GPL and their associated syntax. For the examples in this document, a thorough understanding of GPL is not necessary. However, when you wish to learn more, all of the GPL statements and classes with their methods and properties are described in the *"Guidance Programming Language, GPL Dictionary Pages"*.

Once you have completed these steps, it’s time to get started.

Installation and Startup

Installing GDE/GDS on a PC

GDE is distributed as a component within the Guidance Development Suite of programs (GDS). GDS includes GDE plus other useful PC applications such as the Guidance Datalogger Viewer, which graphs the results of controller data collection sessions. To install GDE, you must install GDS.

If you previously installed GDS on your computer, you should un-install the old version by performing the steps below. If you are installing GDS for the first time, you can skip the first set of instructions.

» Shut down all programs that are running including virus protection programs.

» Bring up the Window's Control Panel by clicking "***Start > Settings > Control Panel***".

» Double click on the "***Add or Remove Programs***" selection.

» In the "Add or Remove Programs" popup window, scroll down and click on "***Guidance Development Suite***".

» Click the "***Remove***" button and click on "***Yes***" to confirm the action.

» Close all of the windows that you opened.

To install GDS on your computer, perform the following steps:

» Shut down all programs that are running including virus protection programs.

» Insert the Guidance Development Suite CD-ROM into your computer's CD-ROM drive. A panel should popup that welcomes you to the GDS Setup Wizard. If the installer does not automatically start, click "***Start > Run***", type in "***D:\setup.msi***" (where D is the CD-ROM drive), and click "***Ok***".

» Follow the instructions in the Setup Wizard to install GDS. Note, GDS relies upon the Microsoft .NET Framework in order to operate. This is a standard module that Microsoft provides free of charge. If the Setup Wizard detects that this module is not available, you will be asked if you wish to install the Framework now. You should respond "***Yes***". This will launch a browser to take you to the download site for the required software.

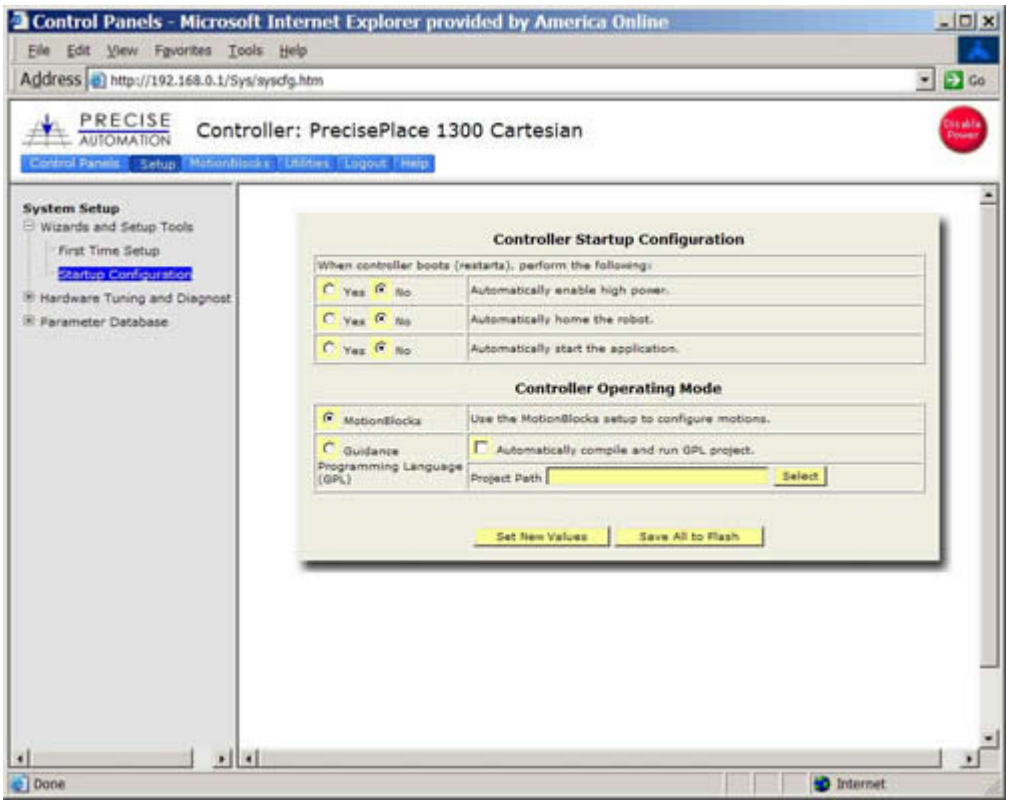
At the conclusion of this process, GDE will be installed on your PC along with the other components of the Guidance Development Suite. You can now begin to use GDE. However, please note that in order to continue to use GDE and other elements of GDS, **you must register this product with Precise Automation.** After you start GDE, go to the "Register product" item under the "Help" top-level menu item for information on the registration process.

Configuring the Controller for GPL Execution

In order to execute a GPL program that moves the robot, the Guidance controller must be configured for GPL execution instead of MotionBlocks execution. Once this configuration is stored in the controller's flash disk, this configuration will be preserved even if the controller is turned off and restarted. To set this configuration, perform the following steps.

» Please see the *"Guidance System Setup and Operation, Quick Start Guide"* and follow the instructions for bringing up the web based Guidance Operator Interface for the controller.

» In the web interface, open the Controller Startup Configuration page, ***"Setup > Wizards and Setup Tools> Startup Configuration***". The web page should look as follows:



» Press the red "***Disable Power***" button at the top of the page. This is required because the changes that you will make are not permitted when power to the robot is enabled.

» In the "Controller Operating Mode" panel, click on the "***Guidance Programming Language (GPL)*** " radio button

» Press the "***Set New Values***" button to store this setting into memory

» Press the "***Save All to Flash***" button to store this setting in the flash disk. This ensures that this setting will remain in effect if the controller is restarted. The button will flash for 10-30 seconds as the data is being written. **DO NOT TURN OFF YOUR CONTROLLER WHILE THE BUTTON IS BLINKING SINCE THIS MAY CORRUPT THE FLASH DISK.**

You should note that when you selected GPL execution mode, the ***"Automatically start the operating mode"*** option was also set to ***Yes***. With these two settings, each time the robot's power is enabled, the controller is put into a state where it is ready to accept a **Robot.Attached** command to permit a GPL program to take control of a robot.

The controller is now ready to execute a GPL program that includes motion instructions.

Activating GDS Components

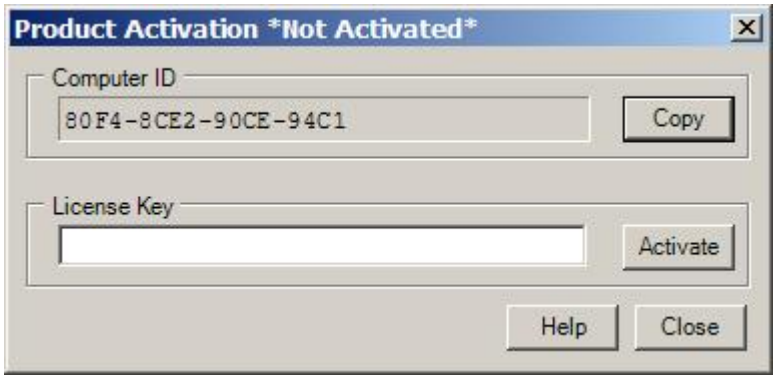
To begin using the Guidance Development Environment, you must start the application on the PC.

» To launch GDE, click on ***"Start > Programs > Precise Automation > GDS xxx > Guidance Development Environment".*** The first time GDE begins execution, you will see the following popup.

The license that you purchased for GDS permits you to use GDE to develop software for an unlimited number of controllers but the package itself is only licensed to execute on a single PC. To complete the software activation process, you must send information concerning the PC to Precise Automation. To allow time for sending the information and receiving a reply, all of the applications within GDS are fully functional for 30 days without being activated. You only need to execute the activation process in one component of GDS in order to activate all of the components.

If for some reason, you do not wish to initiate the activation process now, this popup will be displayed each time you start an un-activated component of GDS. In addition, in any GDS component, you can click on ***"Help > Product Activation***" to display the activation popup.

» To initiate the activation process, click ***"Yes"*** in the popup above. This will display the following activation popup.



This popup displays the "Computer ID" that you need to submit to Precise as part of the activation process.

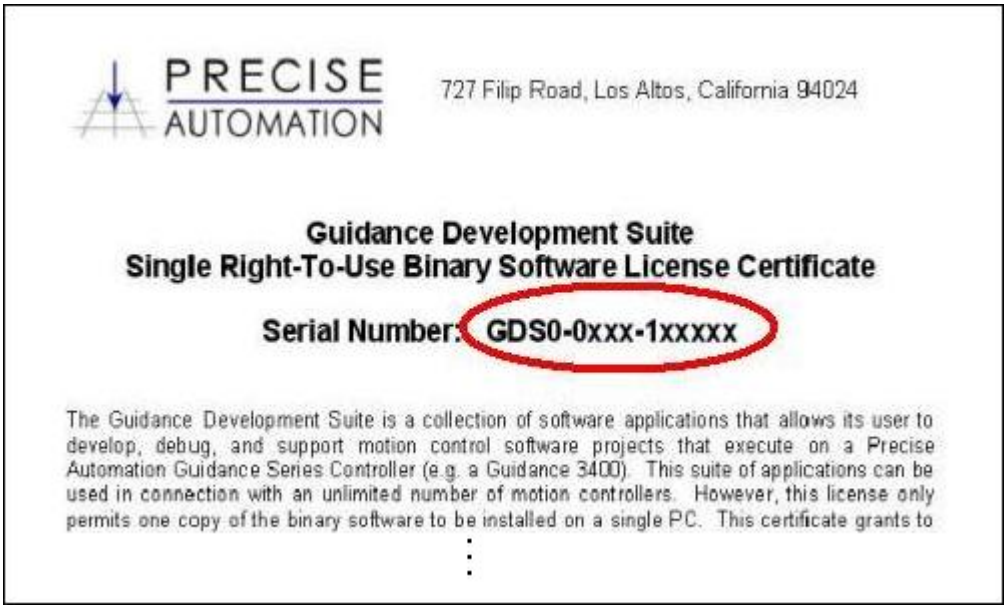
» To obtain a license key to activate GDS, please send an email to Precise Automation that contains the following information:

To: sales@preciseautomation.com
Subject: GDS License Key Request

Customer Name: <your name here, optional>
Customer Company: <your company name here>
Telephone Number: <your phone number here, optional>

GDS Serial Number: GDS0-0xxx-xxxxxx
Computer ID Number: xxxx-xxxx-xxxx-xxxx

» You can obtain the GDS Serial Number from the GDS License Certificate that you received with your order. Below is a picture of the top portion of a typical GDS License with the serial number circled in red.



» The Computer ID number should be copied from the Product Activation popup.

» As a convenience, if you are viewing this help in the *Precise Documentation Library*, you can click on the following link to get an email template: sales@preciseautomation.com.

» Alternatively, if you do not have email access, please Fax this information to 408-516-8348 together with your own fax number.

Your name and your phone number are optional. However, we recommend that you provide this information in case there is a problem with your license and we need to get in touch with you. A sample email should look as follows.

In response to your request, you will receive a license key.

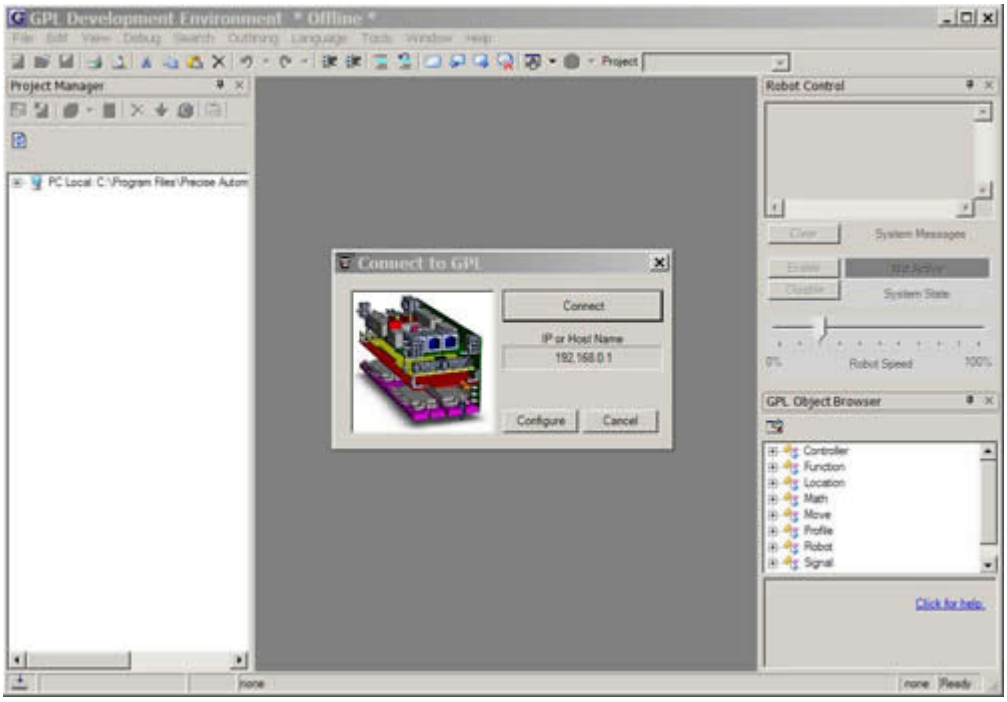
» When you receive your license key, access the Product Activation popup by restarting GDE or clicking in the GDE menu item ***"Help > Product Activation"***.

» Enter the license key in the ***"License Key"*** box and press ***"Activate"***.

All of your GDS components are now activated.

Connecting to the Controller

Once you launch the Guidance Development Environment, you will be presented with the following screen and a pop-up for connecting to the Guidance controller via Ethernet.



If you only intended to develop software off-line, you could press the "**Cancel**" button and skip the process of connecting to the controller. However, since we wish to execute our code on the controller, you need to establish an Ethernet connection. In the box labeled "IP or Host Name", verify that the IP address for the controller is correct. By default, Guidance controllers are configured to respond to 192.168.0.1. If the IP address is correct, you can skip the next instructions.

» To set a new IP address for the controller, press the "***Configure***" button.

» In the Preference popup window, click on the "***Connection***" tab, enter the new IP address, and click "***Update***".

With the correct controller IP address set, you can now connect GDE to the Guidance controller.

» Click the "***Connect***" button on the connection popup window. The PC should connect to the controller in just a few seconds.

Note, if you do not wish to be automatically prompted to connect each time that GDE is launched, you can go to ***File > Preferences > Connection*** to disable this feature.

GDE Windows and Tools

GPL Projects and Libraries Overview

Before we examine the specific elements of GDE, it is important to take a moment to understand GPL "Projects". In GPL, rather than executing a "program", a "Project" is the basic executable entity.

A Project consists of two or more text files that are stored within a single disk folder (i.e. directory). Each file is a standard human-readable ASCII file. The folder name and the Project name are synonymous. Project names conform to the standard GPL symbol name convention and therefore must start with an alphabetic character or "_" and cannot be a single "_". The first character can be followed by any combination of alphanumeric characters and "_". Since Project folders can be stored on the flash disk, Project names are limited to 43 characters in length. Also, since flash disk names are case sensitive, the first alphabetic character in the Project name is always upper case and all other alphabetic characters are lower case, e.g. "Test_project". Specific operations within GPL and GDE are provided for loading, compiling, and executing a Project.

The file "Project.gpr" must always be present in each project folder and is referred to as the "Project File". This file contains information on the other files within the Project. For example, the Project File stores the name of the procedure that is invoked when the Project begins execution. GDE automatically manages the contents of this file and so it is normally hidden within GDE.

There can be multiple GPL source files within a Project. Each source file has a "gpl" extension. These files each can contain one or more program modules, which in-turn can contain multiple variable declarations and procedures.

In addition, a Project can contain one, several or no files with a "gpo" extension. This type of file stores a global module that is used to defined global **Location** and **Profile** objects. This file is convenient for saving taught robot **Locations** and general motion **Profiles** that are accessible by all procedures within the Project.

Almost all of the work done within GDE involves the creation, debugging, and management of the '*.gpl" and '*.gpo" files for a given Project.

Since a Project consists of a collection of files within a disk folder, loading a Project into memory or copying a Project from memory or between disk units is equivalent to copying a file folder and all of its contents. So, Projects can be managed by graphically dragging and dropping their associated folder onto the desired destination device. Although only one Project can be executed at any time, multiple Projects can be concurrently loaded into memory.

GPL Project Libraries

As a convenience when developing large software projects or for sharing software modules, GDS supports GPL Project Libraries. This feature permits any Project to reference another GPL Project and utilize its public routines and data.

No special operations must be performed to convert a Project into a Library. Any Project can be utilized as a Library. To reference a Project as a Library, the main Project must be modified to add the name of the Library into its Project File using the Project Manager facility within GDS. A main Project can reference multiple Libraries and Libraries can reference other Libraries.

When the main Project is compiled, all the files in its referenced Libraries are logically included into the main Project. If two different main Projects refer to the same Library, the Library files are compiled separately into each main Project. This means:

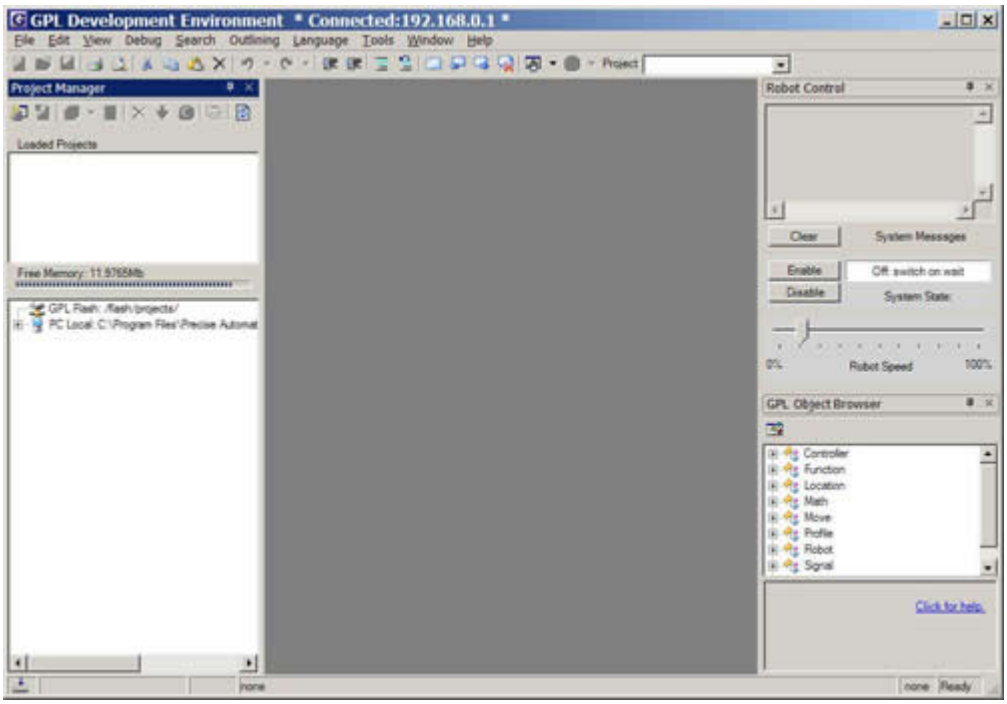
- The use of shared Library Projects does not save memory.
- Global variables defined in the Library Projects are allocated separately for each main Project, so multiple main Projects cannot share data using global variables in the Libraries.

When GDE loads a main Project from flash, it automatically loads any referenced Library Projects from flash. If the Libraries are already loaded, the loaded Libraries are used. If you use GDE to load Libraries from the PC during development, the corresponding Libraries on flash are ignored. When you unload a main Project, the referenced Libraries are not unloaded.

If you use GDE to load a main Project from the PC, you must also manually load the Libraries from the PC or from flash.

GDE Screen Layout

After you have successfully connected to GDE for the first time, you will be presented with the following initial application layout.



The GDE layout contains a title bar that displays the IP address of the connected controller, a top menu bar, a tool bar, the main editor/debugger area and a variety of (dockable) windows. Each of these dockable windows can be displayed or hidden, resized and repositioned into the arrangement that is most efficient for your use. Any space not occupied by a dockable window is utilized by the editor/debugger.

To reposition a window, you simply click in its title bar and drag it to its new location. If you drag a window on to another window, they can split the space or share the space using tab controls. If you click on the close icon (x) in the top right of a window's title bar, you can use the "View" top-level menu to redisplay the window. If you click on the "push pin" in the top right of a window's title bar, you will either "pin" a window and fix its location or "un-pin" a window so it can share its space with another window.

Windows can be resized by grabbing a border and stretching or shrinking it to the desired dimension.

To restore the original positions and sizes of all of the dockable windows, in the main menu, select ***View > Window Layout > Load Default Layout***.

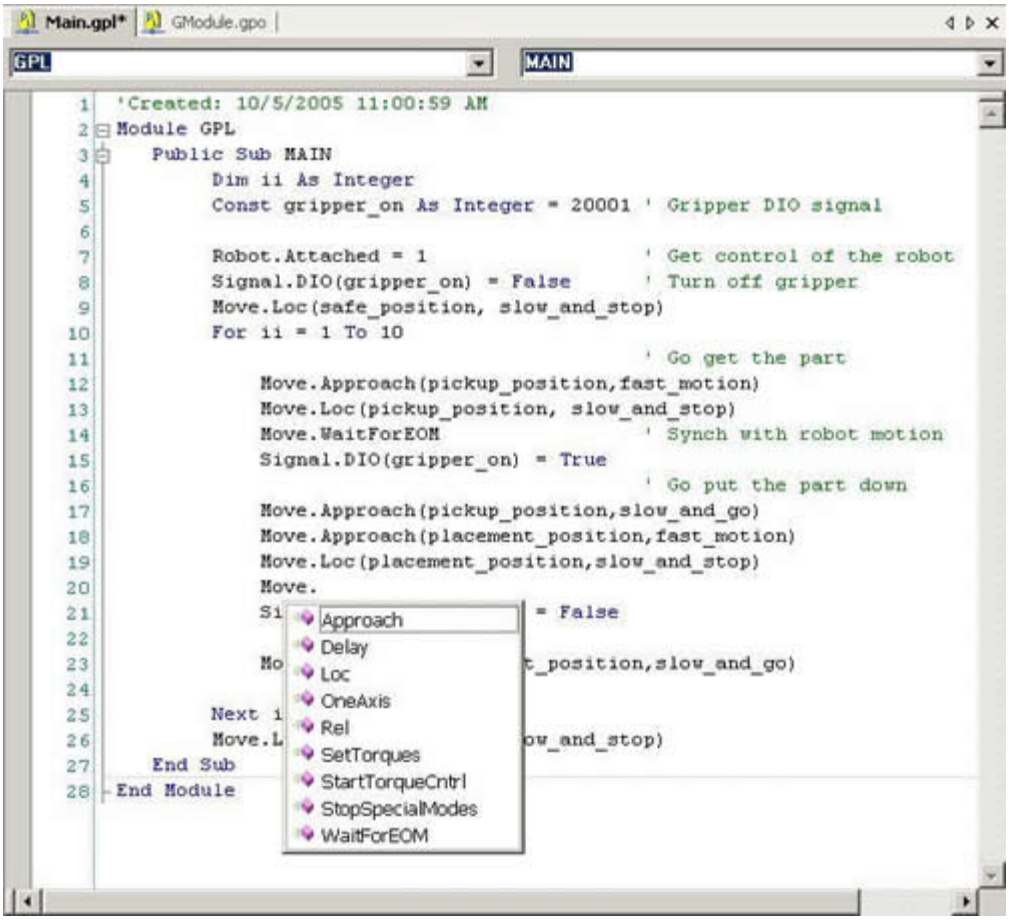
In the following sections, the functions available from each of the major components of GDE (i.e. main menu bar, tool bars, and windows) will be described. The components are presented in order of importance rather than alphabetically. The following table briefly summarizes each of the components.

Component	Description
Editor/Debugger Window	Main window for editing and debugging procedures and global motion data.
Main Menu Bar	Main menu bar that provides access to most of the functions provided by GDE.
Main Toolbar	Provides quick access to common editing and other functions.
Debug Toolbar	Provides debug functions for single stepping, pausing and stopping threads.
Project Manager Window	Displays and manages Projects that are resident in the controller's memory and in the flash disk and the PC's hard drive in the standard GPL Project areas.
Object Browser Window	Provides a list of the methods and properties of all GPL Classes. Automatically displays abbreviated help information as text is entered.
Robot Control Window	Displays the controller system messages and state. Allows robot power to be enabled and disabled and the robot speed to be reduced for testing.
GPL Output Window	Displays all output generated by the controller in connection with the execution of GPL Projects.
Threads Window	Displays execution status and the last procedure executed in each GPL thread.
Program Stack Window	Displays a list of procedures that are on the execution stack for a given thread.
File Manager Window	Displays and manages all folders and files on the controller's flash disk.
Console Window	Provides access to the controller's console. Allows GPL Console Commands to be entered and executed.

Editor and Debugger Window

The Editor and Debugger Window is the primary focus of GDE and occupies all of the space not utilized by the displayed dockable windows. This window allows you to create and modify GPL source files and global GPL motion variables, and to debug GPL procedures by single stepping, setting breakpoints, displaying variable values, etc.

In its normal editing mode, this window will look as follows.



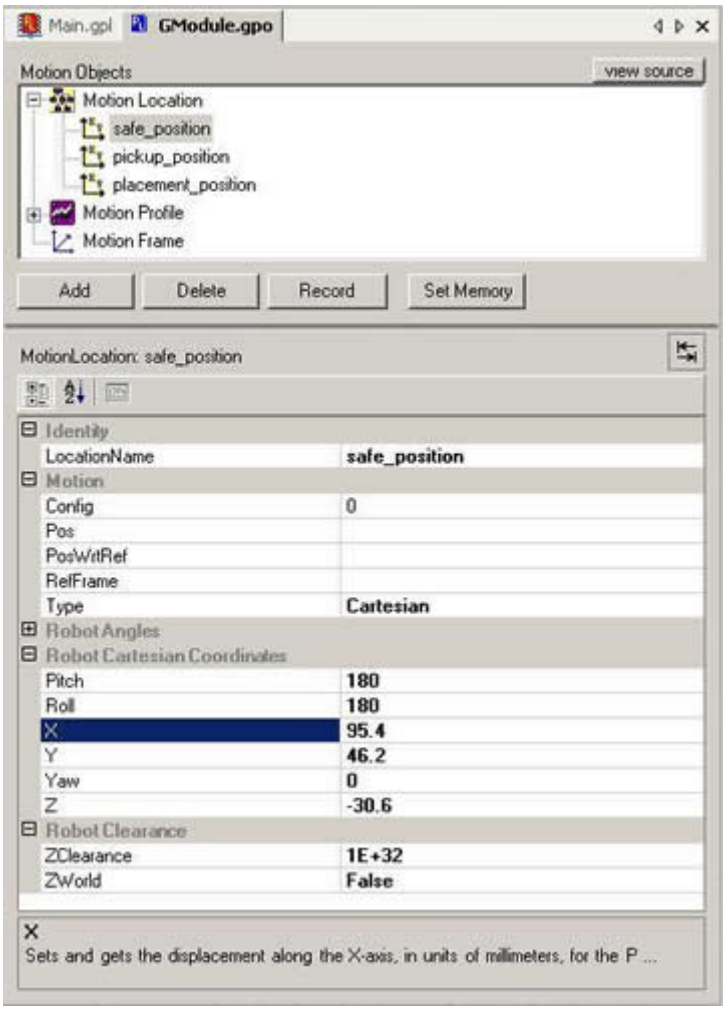
To open a file for editing, you can either double click on the file name in the Project Manager Window or right click on the file name and choose "Edit" in the pop-up menu.

As each file is opened, at the top of the editor window, tabs with the names of opened files are dynamically created to allow you easily switch between files. You can edit files located in the controller's memory, in the controller's flash disk or in the PC's hard drive. Each tab icon is color coded to indicate where the file is stored: in memory (red), the flash (yellow) or the PC (blue). Any files that have been modified and not saved are indicated with an "*" following the name.

Just below the file tabs, two pull-down menus are available for quickly positioning the cursor at a specific module and procedure within a file. For example, in the picture above, the editor is currently position within the "GPL" module and in the "Main" procedure.

The editor operates in the typical manner for inserting, deleting, cutting, and pasting text. In addition, context sensitive help is always active. For example, in the screen shot above, a popup box is automatically displayed as soon as the editor recognizes that you wish to enter a property or method for the GPL Move class. When you select a property or method, the GDE Object Browser will display the syntax for this item as well as a short description. Also, the editor automatically capitalizes keywords and built-in system classes, methods, and properties and color codes the text for greater readability.

If you edit a global modules file that contains motion objects, i.e. a *.gpo file, the editor screen will appear as follows.



This editor display allows you to create new global Motion **Location, Profile** and **RefFrame** objects (***"Add"***), delete objects (***"Delete"***), set the recorded position of a **Location** equal to the current location of the robot (***"Record"***), or modify values in a GPL project as it is being executed on a controller (***"Set Memory"***). The "Set Memory" mode is especially convenient for tuning a robot cycle by changing the properties of a **Profile** or tweaking the components of a **Location** as a project is continuously being executed.

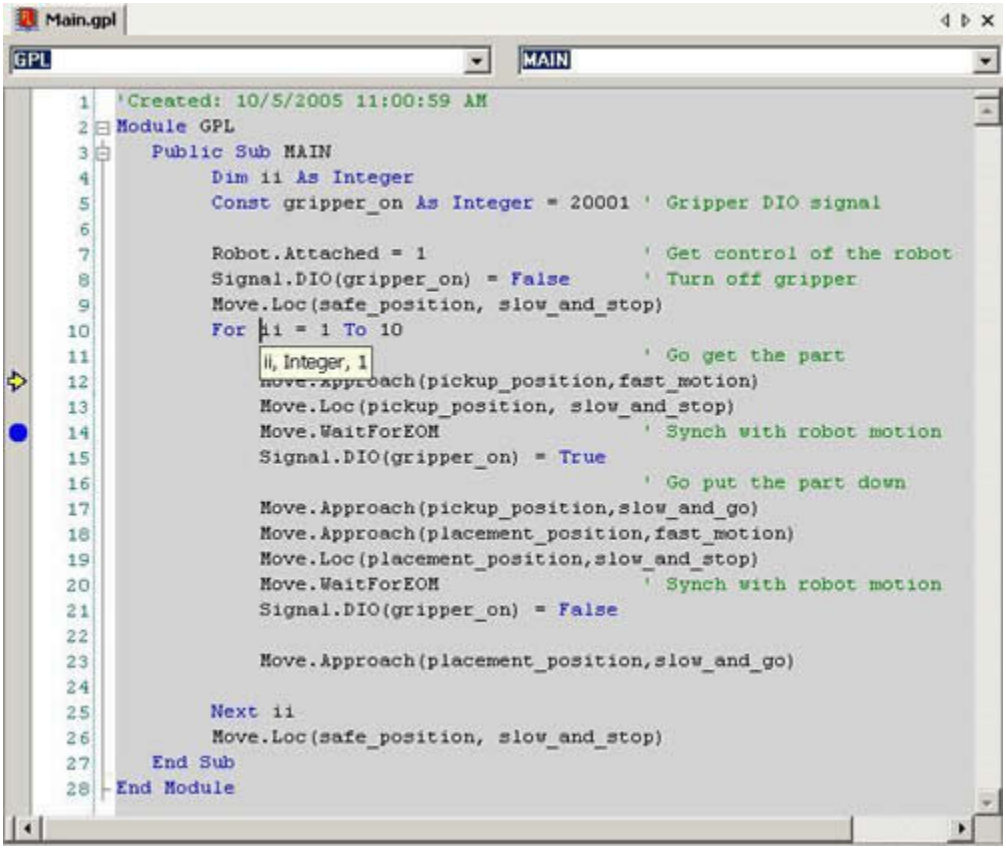
If you select an object in the top treeview panel, all of the properties of the object are displayed below in the properties window. If you select a specific property, a short description is displayed in the bottom panel.

Any objects defined within a .gpo file are globally available to all procedures within the Project. This editing window provides a very simple means of creating, managing, and teaching key application positions and generic motion **Profiles**.

If you would prefer to display the treeview panel and the property panel to the right and left of each other rather than vertically stacked, press the layout icon that is just above the property window and to the right.

To display the underlying text that is generated by this global object editor, press on the "View Source" button.

Once you start program execution, if you pause stepping by hitting a breakpoint or other means, you are automatically placed into the debugger. Files being controlled by the debugger appear on their own tabs in the same manner as files being edited.



As a visual cue, files that are being debugged are displayed with a gray background. Also, in the left margin, the next step to be executed is indicated by a yellow arrow. Any break points that are set are indicated by a blue dot as in line 14 above. Break points can be set or cleared by clicking in the left margin in addition to using the selection in the top menu and toolbar. If you hover the cursor over a variable, the variables name, type, and current value are displayed for the current context. This is shown for the integer "ii". To display other variables, the right-click menu in debug mode can open the "Debug Show Variable " pop-up window. This window permits values not currently displayed to be evaluated in the current execution context.

Both the standard source code editor and the debugger can have their screens split. So, multiple editors and debuggers can be simultaneously active.

For information on all of the debugging aids, please see the section on the Debug Toolbar.

Main Menu Bar

The Main Menu Bar allows you to execute the majority of the functions available within GDE. Many of the more common functions are also provided via the Toolbars or within the dockable windows.

The following tables describe the operation of each of the selections within the pull-down menus and are organized by the name of the menu.

File	Description
New Project	A new Project is created with the name specified in a subsequent popup dialog box. That popup also permits you to select whether you wish to create the Project in the controller's memory, in the flash disk, or in the PC's hard drive. The new Project will have one empty source code file and one empty global module file assigned to it. This operation is equivalent to the "Create a new project file" operation in the Project Manager.
Save	Saves the file being edited back into its Project folder. The folder can be in memory, in the flash, or on the PC hard drive. Changes to files in memory do not take effect until the file is saved back to memory.
Connect	Displays the "Connect to GPL" popup window that allows you to connect GDE to a Guidance controller.
Disconnect	Terminates the connection between the controller and GDE.
Preferences	Displays the GDE Preferences window. This window allows you to change the IP address of the controller, the folder on the PC for storing GPL projects, the text font size, and other settings.
Save as HTML	Standard functions for saving the current editing window in either HTML or RTF format.
Save as RTF	
Print...	Standard functions for printing or previewing the text displayed in the active editor window.
Page Setup...	
Print Preview...	
Exit	Terminates execution of GDE.

Edit	Description
Undo	Standard undo and redo functions that reverse the effect of the previous editing operations or re-instate the effect of operations that were undone.
Redo	
Cut	Standard cut, copy, paste, and delete functions that operate on the selected text within the GPL editor.
Copy	
Paste	
Delete	
Indent	For the selected GPL statements in the editor, either moves all of the text left or right by one tab (4 character positions).
Outdent	
Comment Selection	For the selected GPL statements in the editor, either inserts or

Uncomment Selection	deletes a comment character in the first column of each line to "comment out" or revert the code.
Make Uppercase	For the selected GPL statements in the editor, either converts all of the text to upper or lower case.
Make Lowercase	
Delete Horizontal Whitespace	For the selected GPL statements in the editor, deletes all leading space and tab characters.
Toggle Line Numbering	Enables or disables displaying the line numbers in the editor and debugger windows.
Mark Line Modifications	If enabled, draws a yellow bar in the left margin of all lines that have been modified but not yet saved to the Project file.
Bookmarks	Displays a submenu of functions that set, clear, and return to bookmarks in files being edited. When you set a bookmark on a line in a file, the editor remembers the line number and file name. You can then quickly scroll through all of the bookmark'ed sections of code by using the "Previous Bookmark" and "Next Bookmark" selections.

View	Description
Window Layout	Provides "Load Default", "Load Layout" and "Save Layout" selections for permitting custom desktop layouts to be preserved, reloaded or set back to the normal default.
Project Manager, File Manager, ...	Contains a pull down menu selection for each of the dockable windows. Permits windows to be re-opened if they have been closed.

Debug	Description
Start with Break	Compiles and starts execution of the Project specified in the "Project" box in the main toolbar. However, execution is paused at the Project's first statement to allow you to utilize the debugging facilities
Start	Compiles and starts execution of the Project specified in the "Project" box in the main toolbar.
Compile only	Compiles the Project specified in the "Project" box in the main toolbar.
Step Into	Executes the next sequential statement in the current thread after which execution is paused once again. If the statement is a procedure call, execution is paused inside of the procedure before its first statement.
Step Over	Executes the next sequential statement in the current thread. However, if the statement is a call to a procedure, the entire procedure is executed and execution is paused at the first statement following the procedure call.
Step Out	Executes all remaining statements in the current procedure and pauses execution at the first statement following the call to the current procedure.
Stop All	Halts execution of all threads. After execution has been halted, execution can no longer be continued and must be restarted.
Toggle Break Points	Sets or clears a break point on the current line of a procedure in

	memory. When execution of a procedure encounters a line with a break point set, execution is paused.
Clear All Break Points	Clears all of the break points in the file currently displayed in the editor/debugger.
Open Debug Show Variable	Pops up a window that permits values not currently displayed to be evaluated in the current execution context.

Search	Description
Find/Replace...	Displays a standard popup window for finding or replacing text within the editing window and provides shortcuts for repeating the operation either going forward or backwards in the file.
Incremental Search	
Reverse Incremental Search	
Go To Line...	Moves the cursor to the specified line in the editor or debugger window.

Outlining	Description
Toggle Outlining Expansion	Outlining allows you to display all text lines or collapse to their first line procedures or modules within a file. This function is convenient since most files have multiple procedures and modules. Outlining allows you to display just those elements that are currently of interest to you.
Toggle All Outlining	
Stop Outlining	

Tools	Description
Web Interface	Brings up a web browser and connects it to the Operator Interface for the controller. The browser window is treated just like another opened file within the GDE editor and is displayed in its owned tab'ed window.

Window	Description
Split Horizontally	Divides the current editor or debugger window into 1, 2 or 4 separate panels. Each panel can be individually scrolled to view different sections of a file.
Split Vertically	
Split Four-Way	
No Splits	

Help	Description
Contents	Opens the *Precise Documentation Library* in an independent window.
About GDE	Generates a popup window that displays the GDE version and ID

	information along with build information for key components of GDE.

Main Toolbar

The Main Toolbar provides quick access to a number of commonly used functions, particularly text editing operations that are also provided in the pull-down list of the Main Menu.

The Main Toolbar, which has been split into two for easier viewing, is shown below.

The following table describes the operations available via this toolbar.

Icon	Tool Tip Title	Description
	New Project	A new Project is created with the name specified in a subsequent popup dialog box. That popup also permits you to select whether you wish to create the Project in the controller's memory, in the flash disk, or in the PC's hard drive. The new Project will have one empty source code file and one empty global module file assigned to it. This operation is equivalent to the "Create a new project file" operation in the Project Manager.
	Save Document	Saves the file being edited back into its Project folder. The folder can be in memory, in the flash, or on the PC hard drive. Changes to files in memory do not take effect until the file is saved back to memory.
	Print	Standard functions for printing or previewing the text displayed in the selected editor window.
	Print Preview	
	Cut	Standard cut, copy, paste, and delete functions that operate on the selected text within the GPL editor.
	Copy	
	Paste	
	Delete	
	Undo	Standard undo and redo functions that reverse the effect of the previous editing operations or re-instate the effect of operations that were undone.
	Redo	
	Outdent	For the selected GPL statements in the editor, either moves all of the text left or right by one tab (4 character positions).
	Indent	
	Comment Selection	For the selected GPL statements in the editor, either inserts or deletes a comment character in the first column of each line to "comment out" or revert the code.
	Uncomment Selection	
	Toggle Bookmark	Sets, clears, and returns to bookmarks in files being

	Previous Bookmark	edited. When you set a bookmark on a line in a file, the editor remembers the line number and file name. You can then quickly scroll through all of the bookmark'ed sections of code by using the "Previous Bookmark" and "Next Bookmark" buttons.
	Next Bookmark	
	Clear Bookmarks	
	Disable Robot Power	Immediately disables power to the robot's motors. This is equivalent to the "Disable" button in the Robot Control Window.
	Break Points	Clicking on this icon or selecting "Toggle Break Point" in the pull down list, sets or clears a break point on the current line of a procedure in memory. When execution of a procedure encounters a line with a break point set, execution is paused. A second selection in the pull down list, "Clear All Break Points", clears all of the break points in the file currently displayed in the editor/debugger.
	Compile and Run	Clicking on this icon or selecting "Start" in the pull down list compiles and starts execution of the Project specified in the "Project" box to the right. If you select "Start with Break" in the pull down list, the Project is compiled and started, but execution is paused before the first statement to allow you to utilize the debugging facilities. The "Compile Only" option is a convenience to allow you to just compile your Project to check for syntax errors. The "Stop All" selection is equivalent to the "Stop All" button in the debugger panel. This function halts execution of all active Projects. After being halted, execution cannot be continued and must be restarted.

Debug Toolbar

Whenever a Project is executing or has been paused, the Debug Toolbar is automatically displayed below the Main Toolbar. The Debug Toolbar provides easy access to the functions available for continuing, pausing, stopping and single stepping the execution of a specified thread.

The Debug Toolbar is shown below.

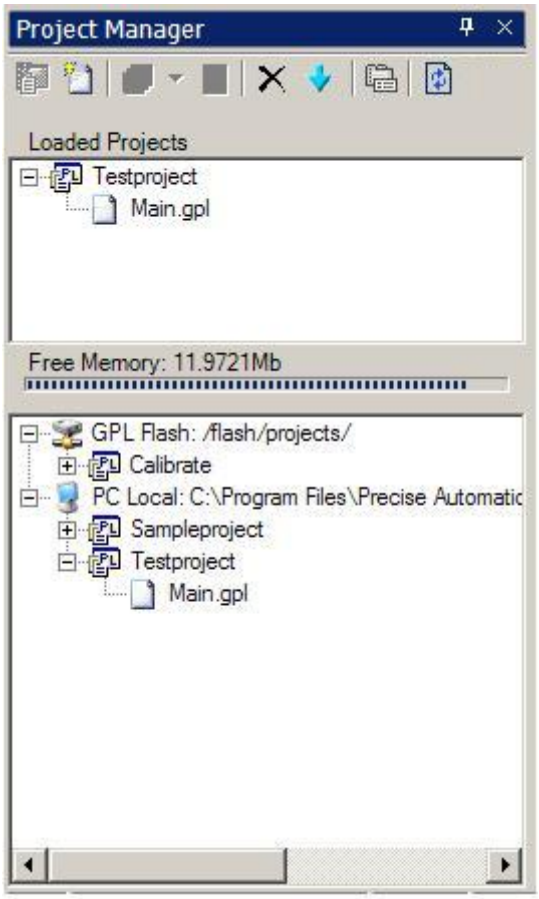
The following table describes the operations available via this toolbar.

Icon	Tool Tip Title	Description
	Continue Execution	Continues execution of the thread specified in the "Thread" box to the right after the thread has been paused due to an error, a break point, or a Break command.

	Break	Pauses the execution of the thread specified in the "Thread" box to the right. This is equivalent to hitting a breakpoint in the specified thread.
	Stop All	Halts execution of all threads. After execution has been halted, execution can no longer be "Continued" and must be restarted.
	Step Into	Executes the next sequential statement in the specified thread after which execution is paused once again. If the statement is a procedure call, execution is paused inside of the procedure before its first statement.
	Step Over	Executes the next sequential statement in the specified thread. However, if the statement is a call to a procedure, the entire procedure is executed and execution is paused at the first statement following the procedure call.
	Step Out	Executes all remaining statements in the current procedure and pauses execution before the first statement following the call to the current procedure.
	Goto Line	Moves the next step pointer to another line within the same procedure that is currently paused. When execution is continued, it will be from the specified line.

Project Manager Window

The Project Manager Window displays Projects that are loaded in the controller's memory as well as Projects stored in the controller's flash disk and in the PC's hard drive. This window allows Projects to be created, edited, selected for execution, deleted, and transferred between the controller's memory and the disk areas. This dockable window typically looks as follows:



The upper window indicates all Projects loaded into the controller's memory. Once loaded into memory, these Projects are eligible to be executed. In this example, a copy of "Testproject" is loaded into memory. Just below this window, the amount of the controller's memory still available for use is displayed.

To select a Project for execution or debugging, double click on the Project in the Loaded Projects window or right click on the Project and choose "***Select***" in the pop-up menu. Alternatively, Project selection can be done on the top GDE menu bar.

The lower window indicates the Projects stored in the controller's flash disk (/flash/projects/) directory and Projects stored in GPL Projects areas on the PC's hard drive. In this example, a Project named "Calibrate" is stored in the flash while two Projects (Sampleproject and Testproject) are stored on the hard drive.

The PC hard drive folder that contains GPL projects can be changed by accessing the Preferences pop-up panel (File > Preferences > Project). The Preferences panel actually permits multiple hard drive folders to be specified. Each specified folder will appear as a top-level node in the lower Project Manager window. This permits multiple hard drive folders to be simultaneously displayed and accessed.

To edit a file within a Project, expand the contents of the Project and double click on the file of interest. You can edit Projects stored in memory, in the flash, or on the PC's hard drive.

To copy a Project between memory, the flash, or the hard drive, simply drag-and-drop the desired project. Dragging-and-dropping a Project to memory is equivalent to loading a Project in preparation for

execution. Projects that have been modified while in memory must be dragged-and-dropped to flash or the hard drive if you wish to preserve the changes in the event that the controller is powered down.

The following table describes the operations available via the Project Manager tool bar.

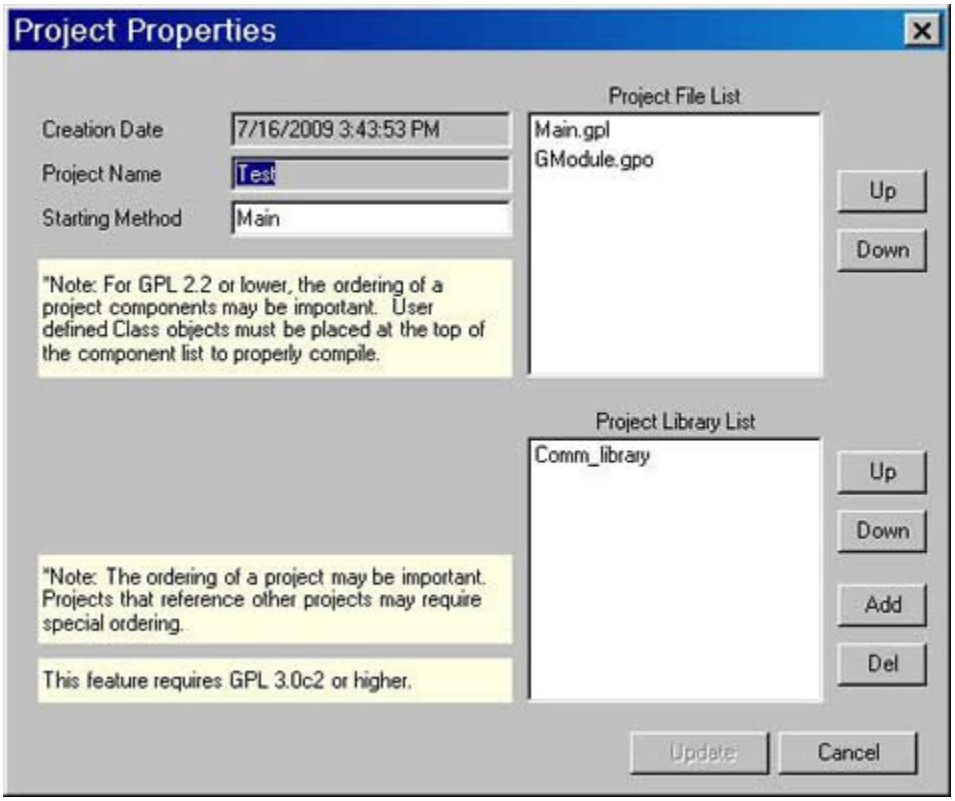
Icon	Tool Tip Title	Description
	New Project	A new Project is created with the name specified in a subsequent popup dialog box. This popup also permits you to select whether you wish to create the Project in the controller's memory, in the flash disk, or in the PC's hard drive. The new Project will have one empty source code file and one empty global module file assigned to it. This operation is equivalent to the "New Project" operation in the top GDE menu bar.
	Add Item to Project	For the selected Project, adds either an additional source code file or a global module file with the name specified in a subsequent popup dialog box.
	Save Memory To:	For a Project stored in memory, allows the operator to: "Delete from memory", "Save to flash", or "Save to PC"
	Edit File	Opens the selected component in the GDE editor. This is equivalent to double clicking on the file.
	Delete Project Or Component	After confirmation, deletes the selected Project or selected file within a Project.
	Load Project	Copies a Project stored in flash or on the PC's hard drive to the controller's memory in preparation for execution.
	Project Properties	Displays information from the Project File including the Project name and list of components. Allows the procedure, which is called when the Project is executed, to be changed. (See below)
	Refresh View	Updates the Project and component displayed for the selected device.

The following table describes the operations available by right-clicking in the Project Manager window.

Right-click	Description
Select Project	Selects a Project that is loaded into the controller's memory for execution or debugging. The Project name will be displayed in the pull-down window on the main toolbar.
Project Properties	Displays information from the Project File including the Project name and list of components. Allows the procedure, which is called when the Project is executed, to be changed. (See below)
Set PC Project Path	These items allow the path to a PC hard drive GDE project area to be easily modified or restored. When this path is modified, the Projects in the new path are automatically displayed in the Project Manager's lower window. When GDE is restarted, the last selected path is remember and again put into effect.
Recent Project Paths	
Import Project	Permits a Project to be copied between a folder in the PC's file

Export Project	system and the controller's memory, the controller's flash drive, or the hard drive GDE project area. These functions simplify sharing Projects in a common network drive and are convenient for exchanging Projects via email. In addition, Projects can be dragged from the PC's file system and dropped into the GDE project area. However, the reverse process is not currently supported.
Edit	Same as the "Edit File" toolbar selection.
New Project	Same as the "New Project" toolbar selection.
Add New Item	Same as the "Add Item to Project" toolbar selection.
Copy Project	Copies or duplicates the selected Project. A pop-up window is displayed that permits the destination and new name of the copy of the project to be specified.
Delete	Same as the "Delete Project or Component" toolbar selection.
Rename	Renames a single project file. This function can be operate on files stored in Flash, Memory and on the PC. This operation does not support renaming an entire Project.
Protect	When you wish to prevent other users from seeing or modifying your GPL source code or other data, you can encrypt ("protect") a file within a project. Protected files can be executed by anyone but cannot be edited. The protection requires a password that is embedded into the encrypted file. Selecting this operation will prompt the user for a password and a one line description that is shown if the file is opened in the editor. **WARNING** Once a file is protected, it can only be unprotected using the same password. Please be sure to keep a secure backup of the unprotected version of the file in case you forget the password. This process can only be performed on individual files that are part of a project that is stored on the PC, not an entire project. Once a file is protected within a project, the project can then be transferred to a controller's Flash or Memory. Attempt to edit a protected file will invoke a read only editor with the description provided when the file was protected.
Unprotect	Allows a developer to unprotect a previously protected file that is part of a project. The unprotect function can only operate on files that are stored on a PC and requires the password originally used to protect the file. If a file resides on a controller, it must first be transferred to the PC and then unprotected. This operation cannot be performed on an entire project, only individual files of a project.
Load Project to Controller Memory	Same as the "Load Project" toolbar selection.
Compile Project	Same as the "Compile Only" main toolbar selection. This is a convenience feature to allow you to just compile your Project to check for syntax errors.
Refresh	Same as the "Refresh View" toolbar selection.

When "Project Properties" is selected, the following pop-up window is displayed that contains information on the files the are contained within the selected Project.



The top right panel displays and manages the files that are contained within the Project. In GPL 3.0 and later, the ordering of files is less critical, but buttons are provided to change the ordering of the files nonetheless. The "Starting Method" specifies the name of the GPL program that is first executed when the Project execution begins.

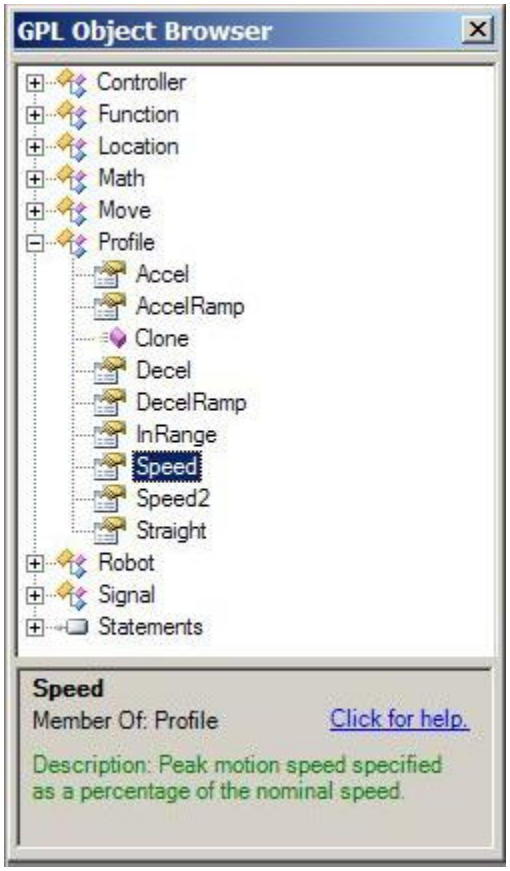
The bottom Project Library List contains the names of other GPL Projects whose programs and data are referenced from within the specified Project. This allows you to develop libraries of GPL routines, Class definitions and Global data that can be utilized by multiple other Projects.

The files contained within each Project Library are compiled into the main Project as though they were explicitly named in the Project File List.

If a GPL Project is loaded from the Flash Disk to a controller's memory, any referenced Libraries are automatically loaded. If a GPL Project is loaded from the PC, any referenced Libraries must be manually loaded into a controller's memory.

Object Browser Window

The Object Browser Window displays syntax and help information for all of the GPL statements and class methods and properties. This dockable window operates in a fashion similar to the .NET Object Browser and typically looks as follows:



You can browse the treeview in the upper panel of the Object Browser for syntax information on specific statements, methods or properties. An icon to the left of each line provides a quick visual queue to indicate the type of the language element. When you select an item in the top panel, a short description of the language element is displayed in the lower panel. If you wish to access the complete GPL dictionary page for the item, just click on the "***Click for help***" link or double click the icon in the upper panel. This will open the *Precise Documentation Library* at the dictionary page for the selected item.

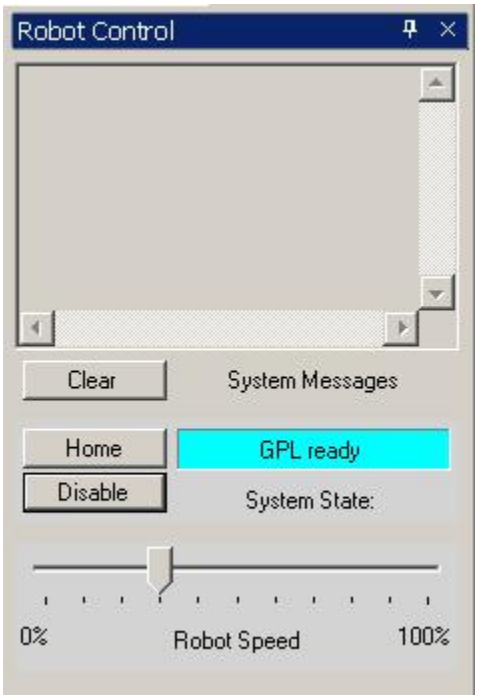
As you use the GDE editor to create new program steps or modify existing steps, the Object Browser automatically updates the treeview in the top panel and the brief description in the lower panel to display the information for the language element that you are entering.

The Object Browser is an information window and source of help information only, so there are no tools associated with this window.

Robot Control Window

The Robot Control Window provides the minimum functions necessary to operate and monitor the status of the robot. With this window, you can enable and disable power to the robot, home the robot, view any error and system messages that have been generated, see the current system execution state, and reduce the overall speed of the robot for testing purposes. These functions are the necessary subset of the operations available in the web based Operator Control Panel and Virtual Manual Control Pendant.

A sample of the Robot Control Window is shown below.



The top panel displays all recent error and system messages that have been generated by the controller. For example, when the robot is stopped due to an error condition, a message is displayed in the top panel that indicates the nature of the error. At any time, you can clear the message buffer by pressing the "***Clear***" button.

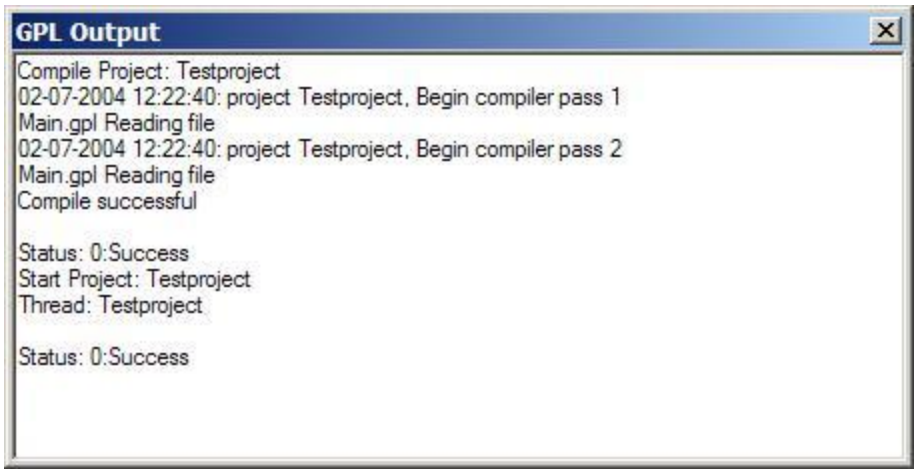
The "***Enable***/***Disable***" button turns the power to the robot's motors on and off. The ***"Home"*** button executes the robot's homing sequence to establish the zero position of each axis when the controller is restarted. The text box to the right of these buttons displays the current execution state of the system. For example, it indicates if robot power is currently enabled and whether a hardware E-Stop is being asserted. Prior to executing a GPL program that moves the robot, the robot power must be enabled, the robot must be homed, and the System State must indicate "GPL ready". If you are not familiar with the procedures that are necessary for initializing the controller and robot, please see the *"Guidance System Setup and Operation, Quick Start Guide"*.

The bottom "***Robot Speed***" slider is equivalent to the speed control on the web Operator Control Panel. This allows you to proportionally slow down the overall speed of the robot while you are debugging a new Project. Changes to the slider take effect immediately even during the middle of a motion.

GPL Output Window

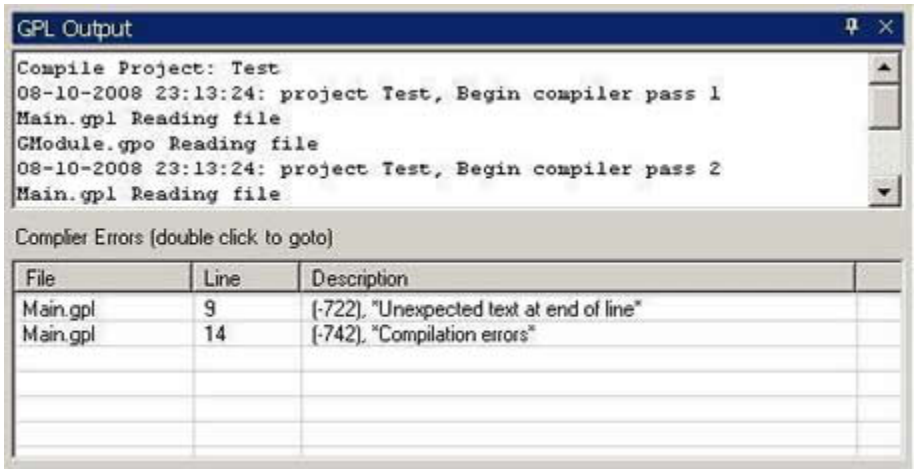
The GPL Output Window displays all output from the controller that is generated in connection with executing a GPL program. For example, when you compile a GPL program, the output of the compiler including any error messages will be displayed in this window. In addition, if your GPL program generates any text output, e.g. by executing a "Console.WriteLine" method, this output will also be displayed in the GPL Output window. If the GPL Output window is closed when you begin execution of a Project, this window will automatically be opened.

A sample of the GPL Output window is shown below after the "Testproject" has been successfully compiled and executed.



If you want to ***"Clear"*** the contents of this window or ***"Copy"*** the contents to a file, right click anywhere in the window to get a popup menu to execute these operations. For the copy function, any text that is currently selected in the window will be copied to the Windows copy/cut/paste buffer.

As a debugging convenience, if you compile a program and any compiler errors occur, the GPL Output window will split and the compiler errors will be listed in a tabular "Goto" list.

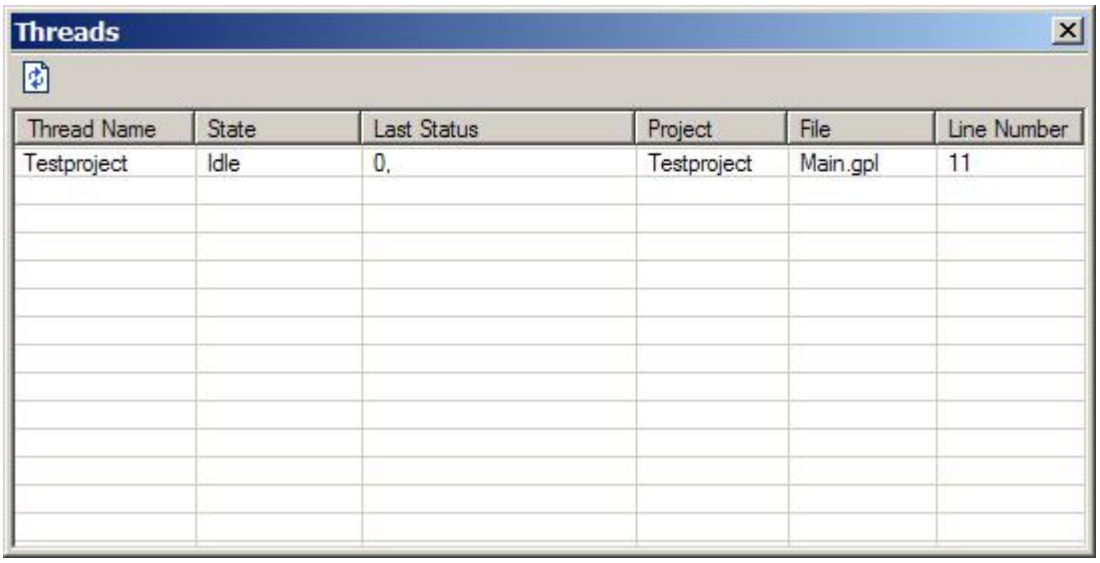


You can then double click on any line in the Goto list and the cursor in the Editor Window will jump to the text line that generated the error.

Threads Window

The Threads Window displays status information for each active execution thread in the controller. The main procedures for your Project will always run in their own thread. In addition, more complex applications may initiate additional threads to allow independent execution of selected code segments.

A sample of the Threads Window is shown below where each line displays the information for a different execution thread.



The first column specifies the name of the thread. The thread name is normally the same as the Project name. The "State" indicates if the thread is running or has ceased execution for some reason (e.g. paused due to a breakpoint or error). The "Last Status" displays any error message that was generated when the thread ceased execution. The "Project" displays the name of the Project running in the thread. If the thread has ceased execution, the "File" and "Line Number" indicate the name of the file that contained the last procedure executed and the number of the last step executed relative to the start of the file.

For more information on the interpretation of these values, please see the documentation for the "Show Threads" Console Command.

The following table describes the operations available via the Threads Window tool bar.

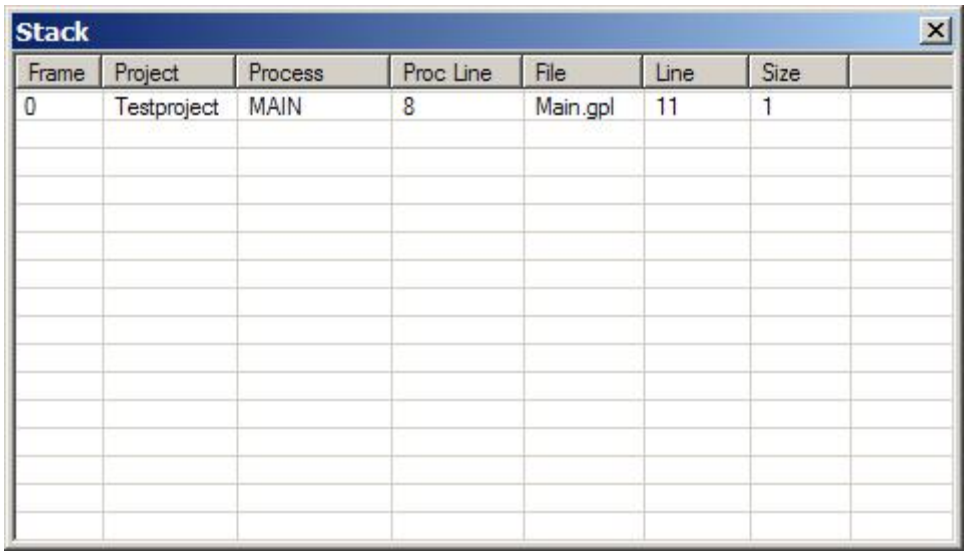
Icon	Tool Tip Title	Description
	Refresh	Updates the displayed thread information.

Program Stack Window

The Program Stack Window displays the list of procedures that are currently on the execution stack for the thread specified in the Debug Toolbar "Thread" box.

When a procedure is running, information on the current statement being executed is saved on the execution stack. When a procedure calls another procedure, information about the current statement in the calling procedure is preserved on the execution stack and a new "frame" is created on the stack to store the step information for the called procedure.

A sample of the Program Stack Window is shown below where each line displays the information for a single stack frame. So, the number of lines indicates the depth of procedure calls currently in effect. Note, this window will only display information when the referenced thread is active but not running.



The first column specifies the frame number where 0 indicates the frame currently being executed, i.e. the top of the stack. The "Project" is the name of the Project being executed and "Process" is the name of the procedure being executed. The "Proc Line" is the number of the statement being executed relative to the start of the procedure. The "File" is the name of the source code file that contains the procedure. The "Line" column indicates the number of the statement being executed relative to the start of the file. Finally, the "Size" displays the size of the stack frame in kbytes.

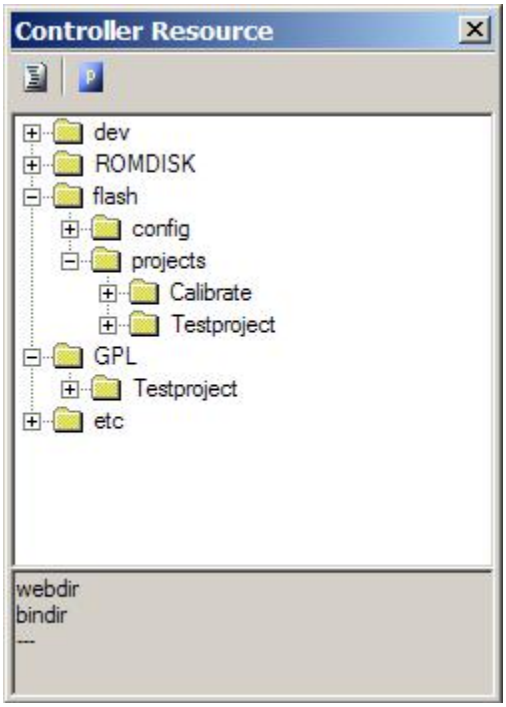
For more information on the interpretation of these values, please see the documentation for the "Show Stack" Console Command.

File Manager Window

The File Manager provides access to the entire file structure of the controller. This is wider access than is available via the Project Manager, which only displays the Project area of the flash disk and the controller's memory. The entire file structure includes: all folders on the flash, such as the Parameter Database PAC files in the "/config" folder; the ROMDISK (i.e. the in-memory simulated disk) that contains the Operator Interface web pages; and the GPL memory image that displays the Projects loaded in the controller's memory.

The File Manager is provided as a convenience for displaying files not accessible using the Project Manager such as the Project.gpr files. Should you wish even greater read/write/drag-and-drop access to the controller's file system, you should utilize the File Manager that is available as part of the Backup and Restore function of the web Operator Interface.

A sample of the File Manager Window is shown below. In this example, the contents of the flash's project folder and the GPL folder correspond to the contents displayed in the Project Manager Window example in this document.



The following table describes the operations available via the File Manager tool bar.

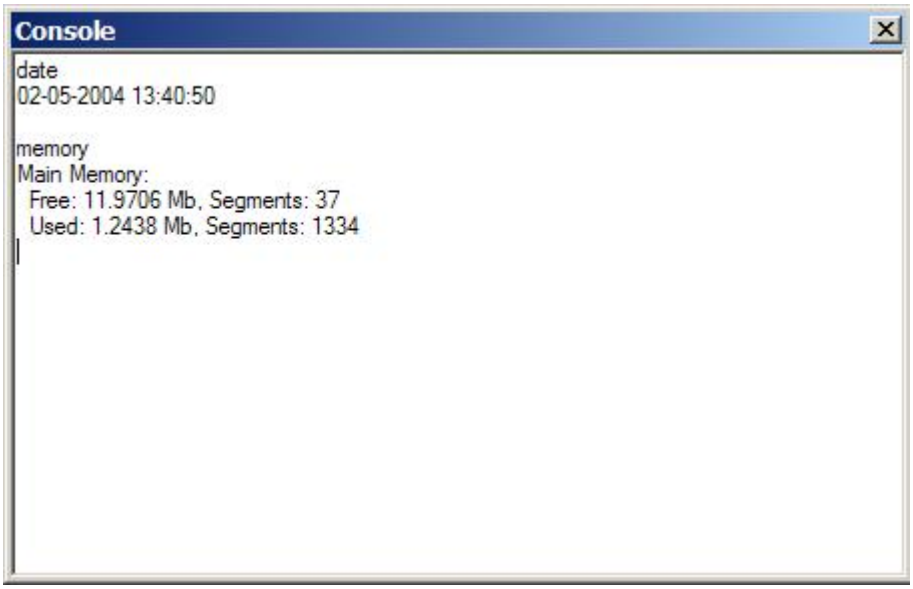
Icon	Tool Tip Title	Description
	Edit File	Opens the selected file in the GDE editor. This is equivalent to double clicking on the file.
	Preview File	Opens the selected file in a read-only text display window. This can be used for viewing any file.

Console Window

The Console Window provides access to the GPL Console Commands. This window is equivalent to connecting to the serial port of the controller. The Console Commands are simple, non-graphic text commands that perform rudimentary operations such as displaying the current memory utilization.

During normal operation and software development, you should not need to issue Console Commands since their functionality plus more is provided by the web Operator Interface and GDE. However, the Console Window is provided in GDE for completeness.

A sample of the Console Window is shown below where two typical commands have been issued.



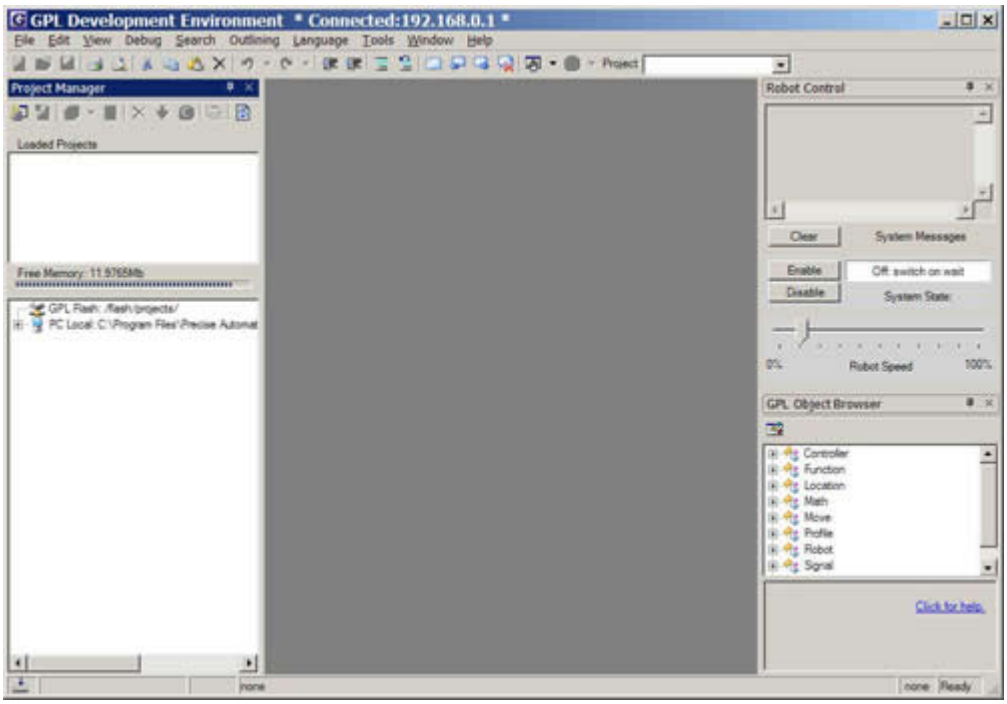
For more information on the GPL Console Commands, please see the Software Reference section of the *Precise Documentation Library*.

GDE Programming Examples

Hello World Example

Now that you have an understanding of the features of the Guidance Development Environment, we want to put that knowledge to work by developing and executing the traditional "Hello World" computer program. In this exercise, you will learn how to create a Project, write a simple procedure that outputs the text "Hello World", load the Project into the controller, and then execute it.

At this time, your GDE environment should be connected to your controller and show look approximately like the following. Don't worry if you have re-arranged any of the windows or added any projects to the system. Also, since we will not be moving the robot, the power to the robot need not be enabled.



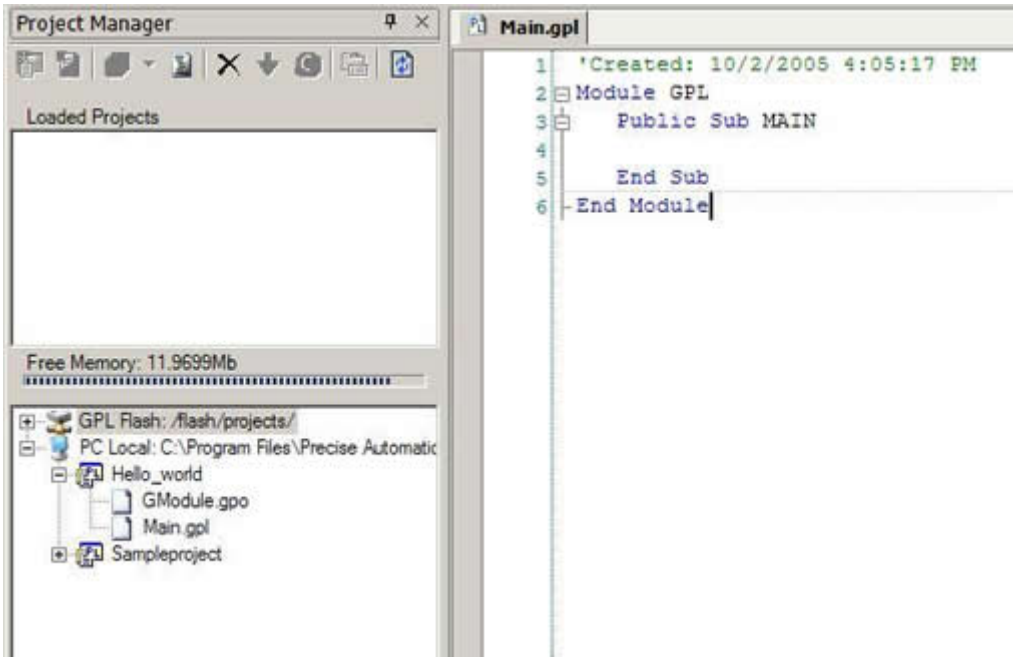
The first step is to create a new Project. For this exercise, you will create this Project on your PC's hard drive although the Project could just as easily be created on the controller's flash disk.

» To create a new project, click on ***"File > New Project***".

» In the***"Add New Project"*** popup box, enter "Hello_world" for the Project name, ensure "***PC hard drive***" is selected, and press the ***"OK"*** button.

» To bring up the main source code file for editing, in the Project Manager, expand the new "***Hello_world***" Project folder and double click on the ***"Main.gpl"*** file.

A zoomed view of the Project Manager and the GDE editor will now look like the following.



When you created the Hello_world Project, you saw that GDE automatically adds a main source code file and a global modules file to new Projects as a convenience. Also, the "Main.gpl" file already contains the definition for the "MAIN" public procedure. By default, this is the procedure that will be started when we execute your project, although this can be easily changed by accessing the Project properties.

Next, we want to edit the MAIN procedure and add statements to output the desired text. Then, we want to load the Project into the controller in preparation for execution.

» In the editor window, below "Public Sub Main", insert the following lines of text.

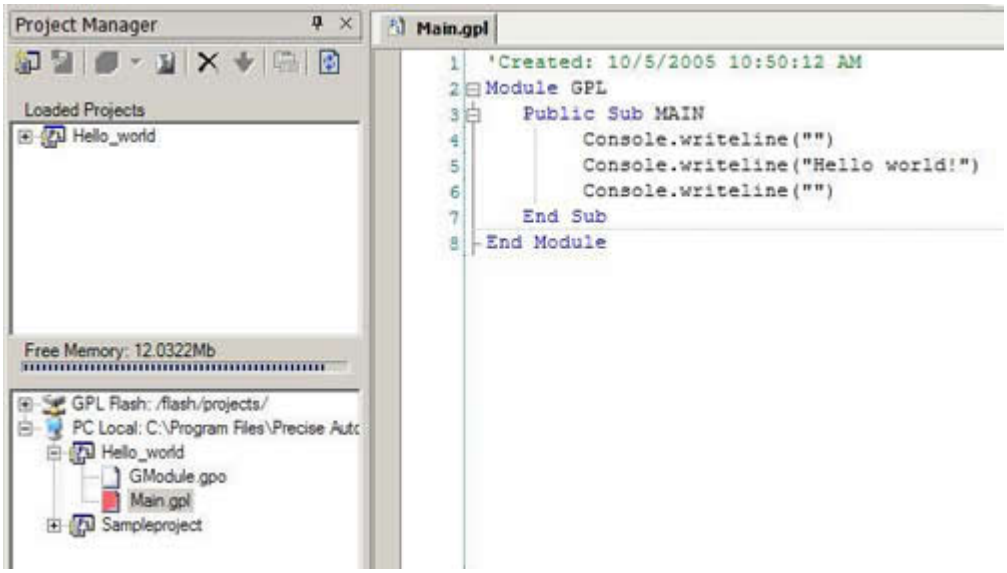
```
console.writeline("")
console.writeline("Hello world!")
console.writeline("")
```

» Press the ***"Save Document"*** icon on the main tool bar to save your changes.

» In the Project Manager, drag the ***"Hello_world"*** Project from the PC to the top ***"Loaded Projects"*** panel and drop it.

As you were entering the text, you might have noticed some aids that are built into GDE. First, after you typed "console." the system detected that you were referring to the standard Console class and displayed a popup that you could use to automatically select "Writeline". Secondly, even though you typed "console" in lower case, the system automatically changed the case of this word. Finally, as you typed, the Object Browser was changing its contents to highlight the syntax for "writeline" as well as showing a brief description of this method.

At this point, your new Project is loaded into the controller and is ready to execute. An expanded view of the Project Manage and editor should now look as follows:

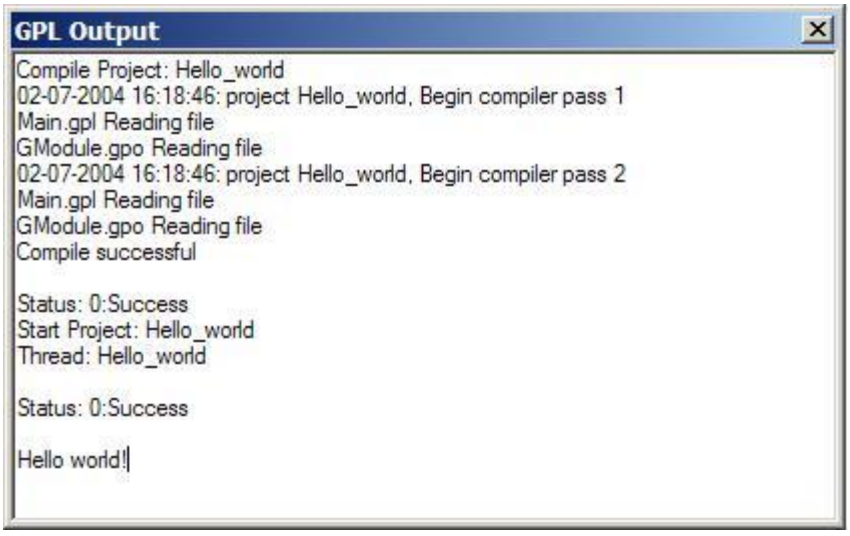


You are now ready to execute your Project.

» Either double click on the ***"Hello_world"*** Project in the Loaded Projects panel of the Project Manager to select the project for execution or select the Project name in the top tool bar in the ***Project*** box.

» In the top tool bar, press the ***"Compile and Run"*** button (i.e. the green button with the right arrow).

At this point, the GPL Output window should display text similar to that illustrated below with your "Hello world!" message shown at the bottom.



Congratulations! You have now successfully created and executed your first GPL Project.

If you copy this Project to the flash disk, you could now utilize the web Operator Control Panel to load and execute your new Project entirely from the web interface without the use of the PC. For instructions on this procedure, please see the *"Guidance System Setup and Operation, Quick Start Guide"*.

Pick and Place Example

In this exercise, you will develop a GPL Project that performs a simple simulated pick-and-place operation. A pick-and-place operation moves the robot's tool tip to a position to pick up a part and then moves to a second position to drop off the part. In order to clear any possible obstacles and to avoid dragging the part, the tool tip is retracted after picking up the part and then approaches the place location slightly over the placement position.

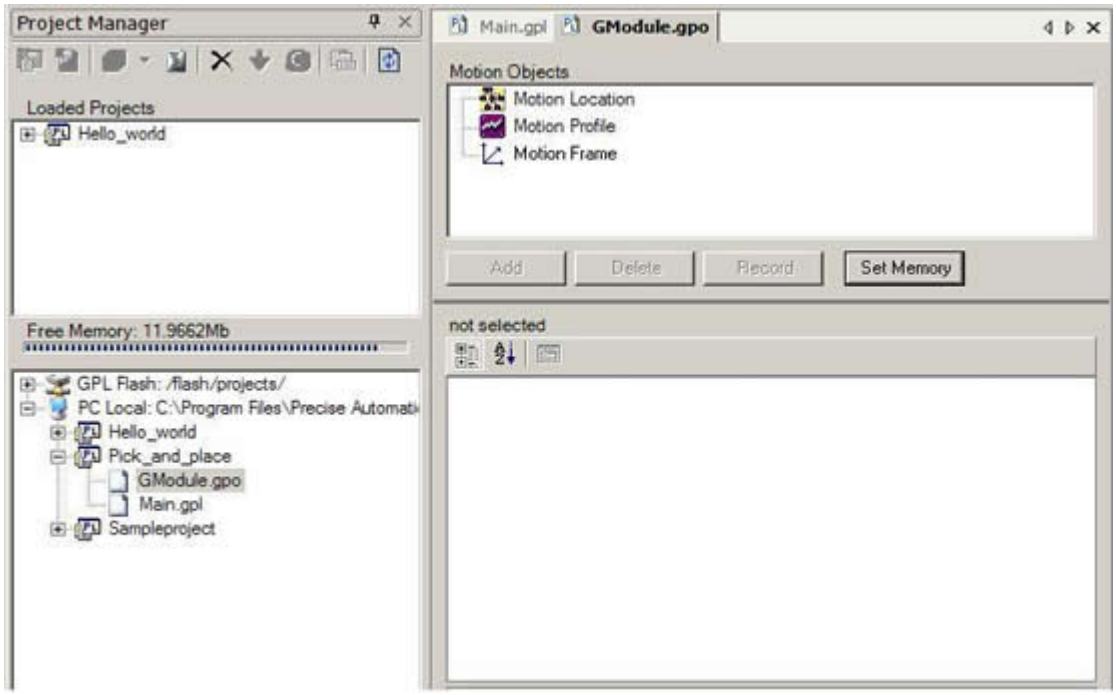
For this application, the robot power must be enabled and the robot must be homed. In addition, the web Guidance Operator Interface will be used to manually move the robot to teach the pick and place locations. As in the Hello World example, the starting point for this exercise is to create a new Project. However, the second step in this process will be to define the global **Location** and **Profile** data rather than writing the program.

» To create a new project, click on ***"File > New Project***".

» In the***"Add New Project"*** popup box, enter "Pick_and_place" for the Project name, ensure "***PC hard drive***" is selected, and press the ***"OK"*** button.

» To bring up the global module file for editing, in the Project Manager, expand the new "***Pick_and_place***" Project folder and double click on the ***"GModule.gpo"*** file.

At this time, the GDE application should be displaying the panel for editing and managing global robot data. The zoomed up view of the Project Manager and editor should look as follows:



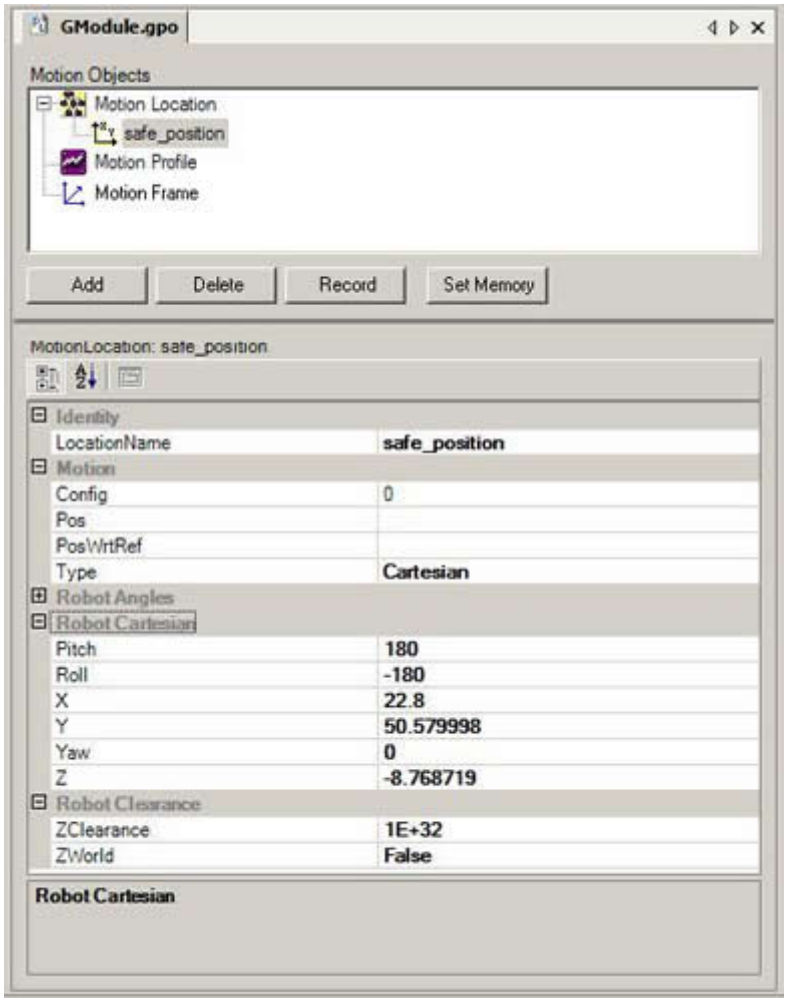
Next, we wish to define and teach three motion **Locations**: the simulated pick-up position, the placement position, and a safe position. We will start with the safe position. This should be a robot location that is above the work surface and one that we can safely reached from most positions in the work envelope.

» Using the Virtual Manual Control Pendant in the web Operator Interface, move the robot to a "safe" location that you have selected.

» To create a new global **Location** object, in the Motion Objects panel, click on the "***Motion Location***" icon and press the ***"Add"*** button. In the pop-up window that requests the **Location's** name, enter "***safe_position***" and press "***Ok***".

» To set the position of the new **Location** equal to the current position of the robot, click on ***"safe_position"*** in the Motion Objects panel and press the ***"Record"*** button at the bottom of the Motion Objects panel and click ***"Yes"*** to confirm the operation in the pop-up window.

After these steps are completed, the main editing display should look like the following.



Since new **Locations** are created with a type of "Cartesian" by default, we have collapsed the "Robot Angles" properties in the picture above as these properties are only meaningful for "Angles" **Locations**. The important information to note is that a new **Location** object is displayed in the top window and the property window shows its property values along with its object name. Your actual "Robot Cartesian

Coordinates" will be different since they will reflect your actual safe position. You should also note that the **ZClearance** value is huge since we have not defined a clearance distance for this position.

Repeating this procedure, we will now define **Location** objects for the simulated pickup and the placement positions. For these positions, you should select places in your workspace that are clear of obstacles and, just to be safe, 20-40mm above the work surface.

» Repeat the procedure above to move the robot to the pickup position, create a new global **Location** object named ***"pickup_position"***, and record the robot location.

» For this position, we want to set the **ZClearance** value to a height above the pickup location that we will allow us to move to the pickup location without hitting any obstructions. For example, enter a small clearance value, such as 25mm to 50mm for the **ZClearance** property.

» Repeat the procedure again and move the robot to the placement position, create a new global **Location** object named ***"placement_position"***, and record the robot location.

» For this position, we want to set the **ZClearance** value to a height above the placement location that we will allow us to move to the placement position without hitting any obstacles. For example, enter a small clearance value, such as 25mm to 50mm for the **ZClearance** property.

» Press the "***Save Document***" icon in the main tool bar to store the new **Location** values.

You have now finished creating and defining all of the robot position data that will be needed for this simple robot application.

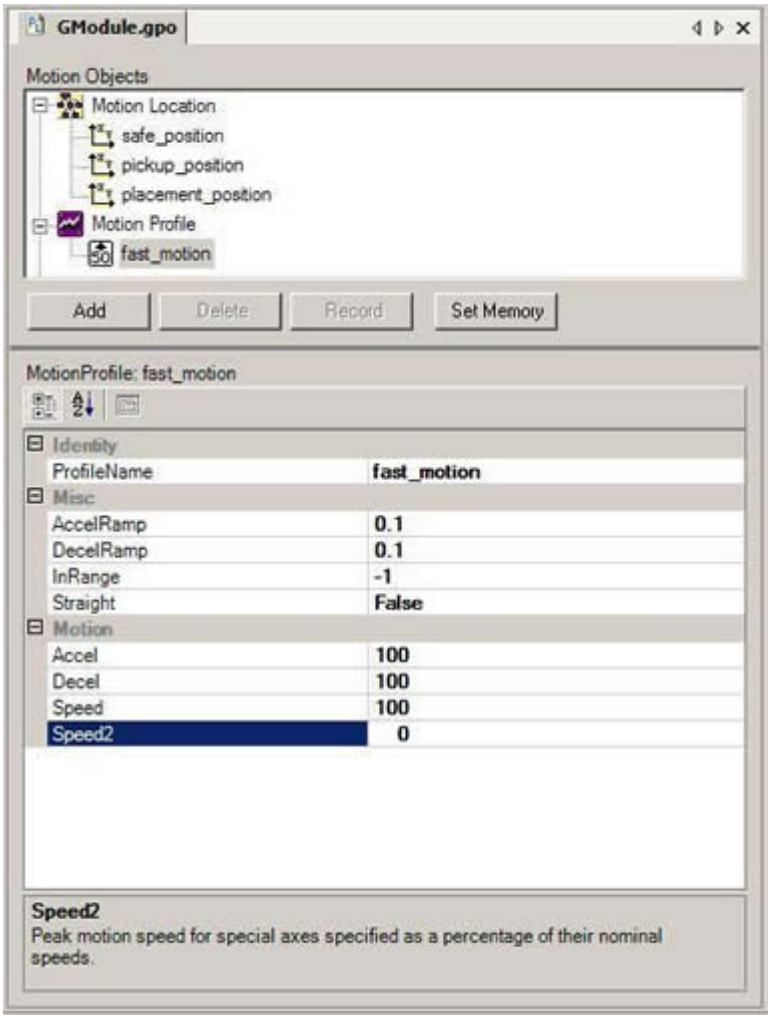
Next, we wish to define some global motion **Profile** objects. These will be used to control the speed of the robot during the various types of motions and will determine when the robot stops. You will be creating three **Profiles**: one that slews the robot at high speed, a second that moves the robot at a slower speed and stops, and a final value that moves the robot at a slow speed but doesn't stop at the end of the motion.

» To create a new global **Profile** object, in the Motion Objects panel, click on the "***Motion Profile***" icon and press the ***"Add"*** button. In the pop-up window that requests the **Profile**'s name, enter "***fast_motion***" and press "***Ok***".

» In the properties window, set the **Speed**, **Accel** and **Decel** to 100 for a high speed motion, and set **Speed2** to 0 to ignore this secondary property for now.

» Set **InRange** to -1 to indicate that the robot does not have to stop at the end of the motion and this motion should be smoothly blended with the next motion if possible.

With this data, the main editing display should look as follows:



» Repeat the process to create a second **Profile** object named "***slow_and_stop***". This should have **Speed**, **Accel** and **Decel** set to 25, **InRange** set to 10, and **Speed2** set to 0. This **InRange** setting will force the robot to stop at the destination and delay until the final position is roughly achieved.

» Repeat the process to create a third **Profile** object named "***slow_and_go***". This should have **Speed**, **Accel** and **Decel** set to 25, **InRange** set to -1, and **Speed2** set to 0. Like ***fast_motion***, the robot will not stop at the destination position.

» Press the ***"Save Document***" icon in the main tool bar to store the new **Profile** values.

This completes the generation of the global data and we can now turn our attention to writing the GPL program that will make use of this information.

» To bring up the main source code file for editing, in the Project Manager, in the "***Pick_and_place***" Project folder, double click on the ***"Main.gpl"*** file.

» In the editor window, below the "Public Sub Main" statement, insert the following lines of text. If you are reading this exercise in the *Precise Documentation Library* (the online help file),

this can be accomplished by copying and pasting if you wish to save yourself some typing.

```
Dim ii As Integer
Const gripper_on As Integer = 20001 ' Gripper DIO signal

Robot.Attached = 1                  ' Get control of the robot
Signal.DIO(gripper_on) = False      ' Turn off gripper
Move.Loc(safe_position, slow_and_stop)
For ii = 1 To 10
                                    ' Go get the part
    Move.Approach(pickup_position,fast_motion)
    Move.Loc(pickup_position, slow_and_stop)
    Move.WaitForEOM                 ' Synch with robot motion
    Signal.DIO(gripper_on) = True
                                    ' Go put the part down
    Move.Approach(pickup_position,slow_and_go)
    Move.Approach(placement_position,fast_motion)
    Move.Loc(placement_position,slow_and_stop)
    Move.WaitForEOM                 ' Synch with robot motion
    Signal.DIO(gripper_on) = False

    Move.Approach(placement_position,slow_and_go)

Next ii
Move.Loc(safe_position, slow_and_stop)
```

» Press the ***"Save Document"*** icon on the main tool bar to save your changes.

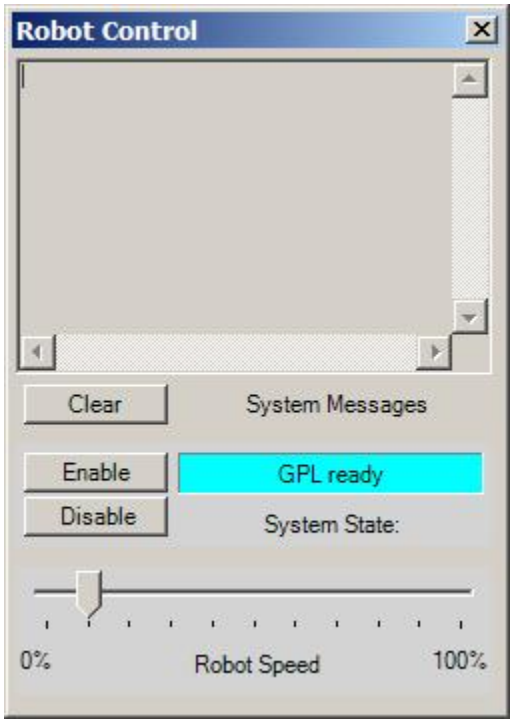
» To load your application, in the Project Manager, drag the ***"Pick_and_place"*** Project from the PC to the top ***"Loaded Projects"*** panel and drop it.

You have now completed all of the software development and teaching for your Project. It's time to execute it on the controller. Before we do that, we have to ensure that the controller is in the proper state to run an automatic GPL program.

» Verify that the robot power is on and that the robot has been homed.

» Place the Virtual Manual Control Pendant back into ***"Computer"*** mode. This allows an automatic program to take control of the robot.

At this time, the Robot Control Window should look as shown below. The System State should be "***GPL ready***" indicating that power is enabled and a GPL program can take control of the robot.



» In the Robot Control Window, set the ***Robot Speed*** to approximately 5%. This will force your Project to move the robot at 1/20th of the normal speed for this application.

» Either double click on the ***"Pick_and_place"*** Project in the Loaded Projects panel of the Project Manager to select the project for execution or select the Project name in the top tool bar in the ***Project*** box.

» **This next action will move the robot automatically, so be prepared to hit the E-Stop button or disable power if any problem occurs!** In the top tool bar, press the ***"Compile and Run"*** button (i.e. the green button with the right arrow).

Your Project should now be slowly moving the robot through its pick-and-place operation. Congratulations! You've just successfully developed and executed your first GPL robot application.

If the robot is moving safely in the workspace, you can gradually increase the overall speed in the Robot Control Window. If you wish to stop that Project at any time, you can press the yellow ***"Break"*** button in the Debug toolbar. Also, using the web interface, you can view the state of digital signal 20001 on the Soft Internal IO panel. You will see the simulated gripper signal turn on then off as the robot reaches its pickup and placement positions, respectively.

As a final step, in preparation for executing this application in a standalone controller, i.e. with the Guidance Development Environment disconnected, you should copy the Project to the controller's flash disk.

» Independent of whether the robot is moving or stopped, to copy your application to the flash, in the Project Manager, drag the ***"Pick_and_place"*** Project from the PC to over the ***"GPL Flash: /flash/projects/"*** icon in the same panel and drop it.

With the Project in flash, you can now execute your application using only the web interface. For detailed instructions on executing GPL Projects via the web Operator Interface, please see the *"Guidance System Setup and Operation, Quick Start Guide"*.

Appendix A: FAQ

Frequently Asked Questions

This section contains a compilation of frequently asked questions related to the Guidance Development Environment.

How do I transfer my GDS license to another host computer?

The Guidance Development Suite and its components (e.g. the Guidance Development Environment) can be used to develop software applications for multiple controllers but is only licensed to execute on a single PC. In general, if you need to execute GDS on additional PC's, additional licenses must be purchased.

However, if the PC on which GDS is licensed to execute is being obsoleted, you can transfer a purchased GDS license to a new PC.

Note, as soon as you execute the following procedure, GDS will no longer be permitted to execute on your old PC.

To transfer a purchased GDS license to another PC, perform the following procedure.

1. Follow the standard procedure for installing GDS on the **new** PC.
2. On the **new** PC, execute the procedure for licensing GDE to obtain the PC's ID information. This is needed to issue you a new license for this PC.
3. Execute GDE 2.0 or later on the **old** PC. If you are executing an older version of GDS/GDE, obtain a new version from the Precise Support website and install it on the **old** PC.
4. Select *"Help > Product Activation*".
5. Click the *"Uninstall"* button.
6. Follow the instruction to execute the uninstall procedure. *After you execute this procedure, you will no longer be able to execute GDS on the **old** PC.*
7. Email the following information to sales@preciseautomation.com and request that the license be transferred.
    - Your company name
    - The name of the individual that the license was issued to, if you know
    - The original license code displayed by the uninstall procedure
    - The uninstall code displayed by the uninstall procedure
    - The new PC's ID displayed by the licensing procedure
8. Once you obtain your new license key, execute the standard activation process on the **new** PC.

This completes the license transfer process.