KComm

Software User's Manual

Kloehn Ltd.
10000 Banburry Cross Drive
Las Vegas, Nevada 89144
U.S.A.

Telephone: (702) 243-7727
Fax: (702) 243-6036
Web: www.kloehn.com

Part Number
Rev IR

Table of Contents

List of Figures

1.0 INTRODUCTION

The ***KComm*** software package provides a Microsoft Windows© software interface to various Kloehn Company devices such as syringe pump modules and valve drives. Using ***KComm***, a user may send commands via a serial port to one or more devices, save and recall command strings in a file, and set communications protocols.

The ***KComm*** program is designed to function as a terminal emulator that is customized to the needs of the Kloehn devices. A **Commands** drop list is provided to help the user recall specific device commands. A command string parser breaks a command string into individual commands to make a string more readable. Operation of the program depends on the ASCII command character level of program and command entry. Other Kloehn software products provide more intuitive programming environments for basic users and also drivers for embedding communications protocols into user programs. ***KComm*** is intended to give a user-friendly way of sending command strings and receiving the responses.

1.1 System Requirements

The ***KComm*** program consists of several files which reside in a single directory. The total disk space required, less the Adobe Acrobat© version of the manual, is about 210 Kbytes.

The operating system must be a 32-bit verison of Microsoft Windows©. These include Windows 95, Windows 98, Windows NT, and Windows 2000.

The program communicates with the Kloehn devices via a serial port such as Com1 or Com2. If present, Com3 or Com 4 may be used as well.

1.2 Conventions Used in This Manual

This manual uses certain text conventions to represent different things.
Bold type denotes menu names and selections.
Bold italic denotes file names.
A **bold serif** font represents text to be typed by you.
Text in <**brackets**> refers to keyboard keys.
The notation <**key1**>+<**key2**> denotes holding down key1 while pressing key2.
Text in *italics* is for comments which clarify, but are not a part of a command.
A sequence of selections is denoted by: **selection1 –> selection2**.
Text in [**brackets**] is optional, while text in {**braces**} is something which must be included in a typed line. The braces are not part of the command, but are

used to enclose a description of what is to be typed.
The phrase “right-click” means to press the right mouse button.
The phase “left-click” means to briefly press the left mouse button.

1.3 Installation

1.3.1 System Requirements

KComm runs under Windows© 95, 98, NT, ME, and 2000. The ***KComm*** program requires 4.3 MB of installation disk space with the manual. The manual uses about 4MB of this space.

1.3.2 Installation Procedure

KComm is supplied on disk as a set of files or obtained via the Internet as a compressed “zip” file. In both cases, the installed text files should not be altered, as they are essential to the operation of the program.

The installation procedure consists of placing the compressed ***Kcomm.zip*** file into a directory and decompressing it. Please follow these steps.

(1) Place the file ***kcomm.zip*** into the directory in which the **KComm** directory will be installed. This will usually be the **c:\Programs** directory.
(2) Expand the ***kcomm.zip*** file.
    (a) With mouse pointer resting on the kcomm.zip file, right-click the mouse.
    (b) Select **Extract to folder.**
    (c) When the extraction is done, the files will be installed in the ***KComm*** directory.
(3) Create a desktop shortcut to the program.
    (a) Right-click on the ***KComm.exe*** file.
    (b) Left-click on **Create Shortcut**.
    (c) Drag the new ***Shortcut to KComm.exe*** icon onto the desktop.
    (d) Rename the Icon **KComm** by right-clicking on the icon and then typing “**KComm**” into the text window.
    (e) Drag the new ***Shortcut to KComm.exe*** icon onto the desktop.
    (f) Rename the Icon **KComm** by right-clicking on the icon and then typing “**KComm**” into the text window.

1.3.3 Installed Files

The installed **KComm** files are:

(1)	***KComm.exe***	Executable program file.
(2)	***kcomm.zip***	Compressed KComm file containing all the other files.
(3)	***Mscomm32.ocx***	Run-time support file.
(4)	***50120.txt***	Intellect (50120) command syntax file
(5)	***50300.txt***	50300 pump syntax file for firmware versions 1.16 and 1.17.
(6)	***50300A.txt***	50300 pump syntax file for firmware versions 1.2x.
(7)	***50300B.txt***	50300 pump syntax file for firmware versions 1.18 and 1.19.
(8)	***50358.txt***	50357 and 50358 series pump syntax file.
(9)	***programs.txt***	This file is created automatically to store program strings saved with the ***KComm*** program.
(10)	***Kcomm.pdf***	Adobe Acrobat version of software manual.

The ***programs.txt*** file is created the first time you save a command string with a name. All saved and named program strings are saved into this one file. The syntax of the file is simple and is explained in the section on saving command strings.

1.4 Uninstalling

The KComm software installation does not create any files outside the **KComm** directory. The uninstall procedure is therefor to delete all fines in the **KComm** directory, then delete the directory itself.

2.0 GETTING STARTED

When the ***KComm*** icon is double-clicked, the ***KComm*** screen starts up. The screen appears as in Figure 1. Note there are two boxes: the **Command** box in the upper half and the **Response** box in the lower half. The **Command** box is for sending and the **Response** box is for receiving.

The **Command** box is used to configure the communications protocols, save and recall program strings, and to create and send command strings.

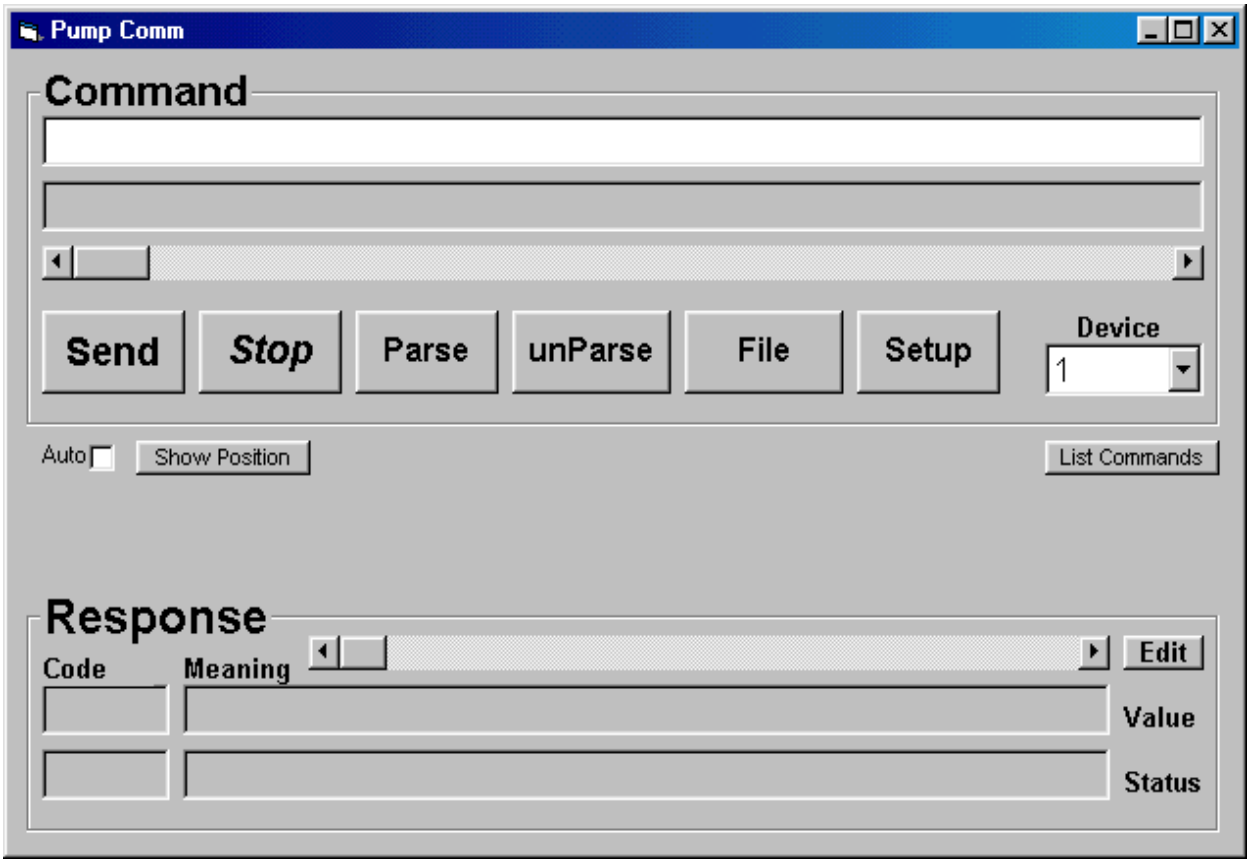


Figure 1 The KComm screen

The **Response** box is provides a display of a device's responses to commands sent to the device from the **Command** box. Responses are made of two parts: **Value** and **Status**. For each part, both the returned **Code** and the **Meaning** of the code are displayed.

2.1 Setting Up for Communications

Referring to Figure 1, two features determine the program protocol setup: the **Setup** button and the **Device** selection list. A **Device** address must be selected, and the **Setup** parameters **Baud**, **Protocol**, **Comm Port**, and **Device Type** must be selected.

2.1.1 Selecting a Device Address

Clicking on the arrow on the right side of the **Device** selection list activates a drop list which lists all possible device addresses. Since up to 15 Kloehn devices can be placed on a single communications bus, each device is distinguished from other devices by a device address. This address is set by an address switch on each device.

Individual addresses range from "1" to "9" and from "A" to "F", where "A" is device "10" and "F" is device "15". A single device is selected by left-clicking on the desired address. All devices are shipped with a default address of "1", and this is the default value of the **Device** address window. All commands sent to an individual device will respond with a **Status** report. Commands which request some specific information from the device will also report a **Value**. The **Status** and **Value** are shown in the **Response** section of the screen when they are received.

Multiple devices can be addressed simultaneously. They can be addressed two at a time, four at a time, or globally (all devices). These address modes can also be selected from the **Device** address list. When a multiple address mode is selected, the devices do not respond with any status or value reports, even when queried. If an error should occur in a device, that device will no longer respond to commands, either stored or sent, until the device's error message is received. This can only be done in the individual address mode.

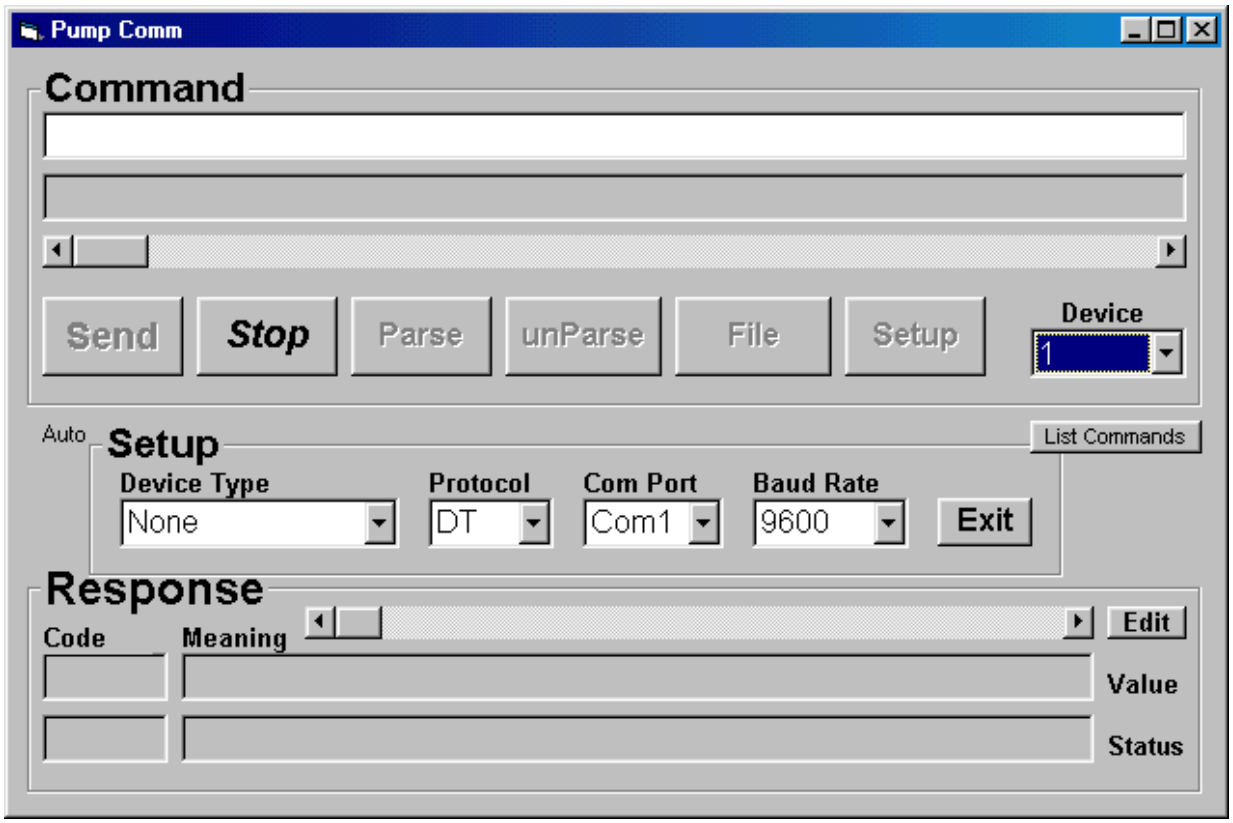
The **Device** address can be changed at any time, but it cannot be included in a command string.

2.1.2 Selecting a Device Type

When a device type is selected, the **Commands** menu list changes to reflect the legal commands for that type of device. The device type menu is accessed by left clicking on the **Setup** button. This will display a screen with several lists on it, as shown in Figure 2. To set the device type, left click on the **Device Type** box, then left click on the desired device number and revision in the list which appears. Note the 50300 has three choices: Rev 1.1x Rev 1.2x, and Rev 1.18-19. The Rev 1.1x

selection is for all 50300 firmware revisions from 1.17 and lower. The Rev 1.2x is for all revisions from 1.20 and higher. When the **Device Type** has been selected, the **Setup** screen can be exited by left clicking the **Exit** button, or other setup parameters can be selected from the remaining boxes.

Figure 2 The Setup Screen



2.1.3 Setting Baud Rate

As shown in Figure 2, the communications port baud rate is set from a list on the **Setup** screen. To set the baud rate, click on the **Baud Rate** box and then click on the desired rate. The default rate is 9600 baud. Whenever the baud rate is other than 9600, the *Default Jumper* on the device must be removed, as this jumper forces the device rate to be 9600 baud. Consult the individual device type hardware manual for the location of the *Default Jumper*. The baud rate can be changed at any time. If no other setups are required, use the **Exit** button to quit the **Setup** screen.

2.1.4 Setting the Protocol

There are two protocols supported by KComm: OEM and DT. The DT (data terminal) protocol is the most popular, as it is supported by all PC serial port communications software and is easily programmed in controller firmware. The OEM provides more secure communications because it contains error checking. The OEM protocol is not inherently supported by any standard PC software and requires special programming. Syntax checking is inherent in the pump.

To set the protocol, click on the **Setup** button to get the **Setup** screen. On the screen, click on the **Protocol** box and then click on the desired protocol. The protocol may be changed at any time, but communication with the device requires that the device protocol be set to match the selected protocol. If no other setups are required, use the **Exit** button to quit the **Setup** screen.

The default protocol is DT. Whenever the protocol is set to OEM, the *Default Jumper* on the device must be removed, as this jumper forces the protocol to be DT. Consult the individual device type hardware manual for the location of the *Default Jumper*.

2.1.5 Setting the Comm Port

All PC's today have at least one Comm port. This is the serial port located on the back of the PC. The port connector is either a 9-pin or a 25-pin connector. Many PC's also have two or even up to four Comm ports. ***KComm*** supports port selection. The ports are identified by their number, one through four. The default port is Com1.

To set the Comm port, left click on the **Setup** button to get the **Setup** screen. On the screen, left click on the **Com Port** box and then left click on the desired port number in the list. If no other setups are required, use the **Exit** button to quit the **Setup** screen.

2.2 Sending Commands

Commands are sent to a device by setting up the communications, selecting a device address, typing a command string into the **Command** text box, and clicking the **Send** button. When the **Send** button is clicked, the command string in the text box, together with the selected device address, is sent to the device via the selected Comm Port. ***KComm*** automatically receives and processes the response, if any, from the device.

2.2.1 Typing the Command

All command strings sent to the devices are sent from the white text box in the **Command** section. While entering the command string into the text box, all normal editing techniques used by typical Windows programs may be used. These include the following:

<**Delete**>	(the Delete key) will delete the character after the cursor location or any highlighted text.
<**Home**>	moves the cursor to the beginning of the text string.
<**End**>	moves the cursor to the end of the text string.
<**Backspac**e>	deletes the text preceding the cursor location.
<**Ctrl**>+<**x**>	cuts highlighted text and places it into the clipboard.
<**Ctrl**>+<**c**>	copies the highlighted text to the clipboard.
<**Ctrl**>+<**v**>	pastes the contents of the clipboard into the location beginning at the cursor location in the text box.

Any text copied to the clipboard may be put into another program such as Microsoft Word© or a text editor by changing to the destination program, clicking the mouse pointer at the desired insertion location, and pressing <**Ctrl**>+<**v**>.

Text is highlighted by left clicking the mouse pointer at the beginning of the desired section of text and, while holding the mouse button down, dragging the cursor to the end of the text portion and releasing the mouse button.

Command strings are not checked for correctness during text entry. A typing error will result in an incorrect command string. The **Parse** function checks for errors.

2.2.2 Using the Command Button

Located just below the **Command** section is a small button labeled **List Commands**. This button will display the commands available for the type of device selected in the **Setup –> Device Type** selection box. The command list may be scrolled up and down with the slider control on the right edge of the box. The **Commands** list box is shown in Figure 3.

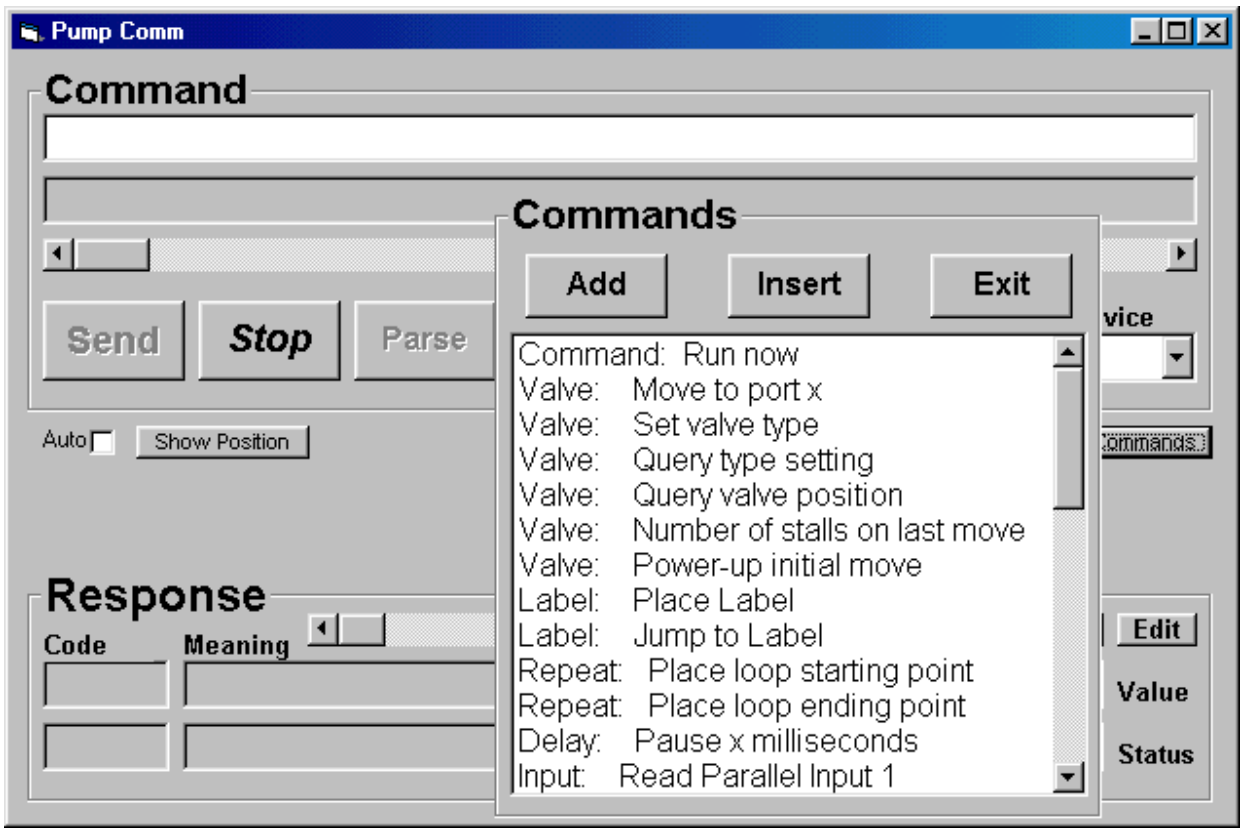


Figure 3 Commands List Box

To select a command, left click on the desired command in the list. The **Add** button appends the command to the end of the command string. The **Insert** button places the command into the string at the location following the cursor position in the text box. The command selection process may be terminated at any time by left clicking on **Exit**.

When a command is selected, it is automatically placed into the command string if it requires no values for completion. Many commands require a value, called an *argument*, for completion. For example, the command "Valve: Move to port x" requires the argument "x" to be entered to identify the port number. When an argument is required, a prompt box will appear to tell what is required and the range of acceptable argument values. Commands which require two arguments will display a second prompt box after the first argument is entered. When the command is completed, it is placed into the command string. If an error is made in entering an argument, the prompt box will clear itself until a correct entry is made.

2.3 Reading the Device Responses

Each time a command string is sent to a device, the device checks for proper command structure (*syntax*) and correct values (*arguments*). If an error is found, an error code is returned as the status. If the command string is querying something in the device, a reply containing the queried value will be added to the status. The *status* and *value* (if any) are then returned to the **Response** section.

The *status* is displayed in the horizontal **Status** row and the *value* is displayed in the horizontal **Value** row. Each bar consists of two vertical columns: the **Code** column and the **Meaning** column, as shown in Figure 1.

The device returns the *status* and *value* in a coded format. The actual codes returned for status and value are displayed in the **Code** column of the **Status** and **Value** rows, respectively. The meanings of these codes are displayed in the **Meaning** column for the **Status** and **Value** rows. A typical response is shown in Figure 4.

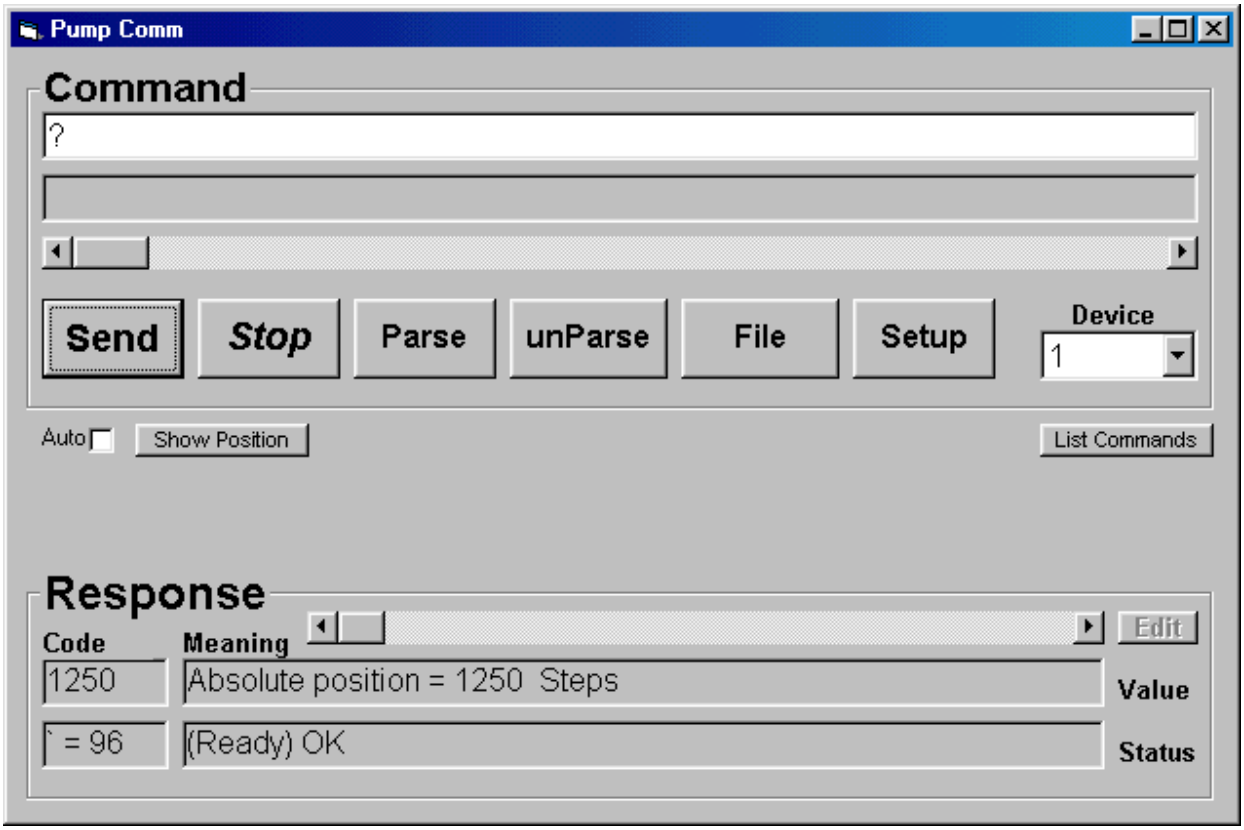


Figure 4 Typical Device Response

In Figure 4, the “?” command was sent and the response codes were “`” (= 96 as a decimal value for the ASCII code) for status and “1250" for value. The status code means “The device is ready and OK”. The value means “The syringe absolute position is 1250 steps from the zero position.” If there had been an error to report, the code for that error would be displayed in place of the “`”.

If the addressing mode is *multiple-device*, responses are withheld until an individual device is addressed. The individually-addressed device will then send its response. This prevents the communications bus conflicts which would occur if multiple devices attempted to reply at the same time.

3.0 SAVING AND RECALLING COMMAND STRINGS

The command string in the **Command** text box can be saved into a program file and later recalled from the file back into the **Command** text box. Each command string saved is given a name to identify it. All saved command strings, also called *programs*, are saved as records into the same file, ***programs.txt***, located in the same directory as the ***Kcomm.exe*** program. Saving and recalling command strings is done through the **Program** screen, shown in Figure 5.

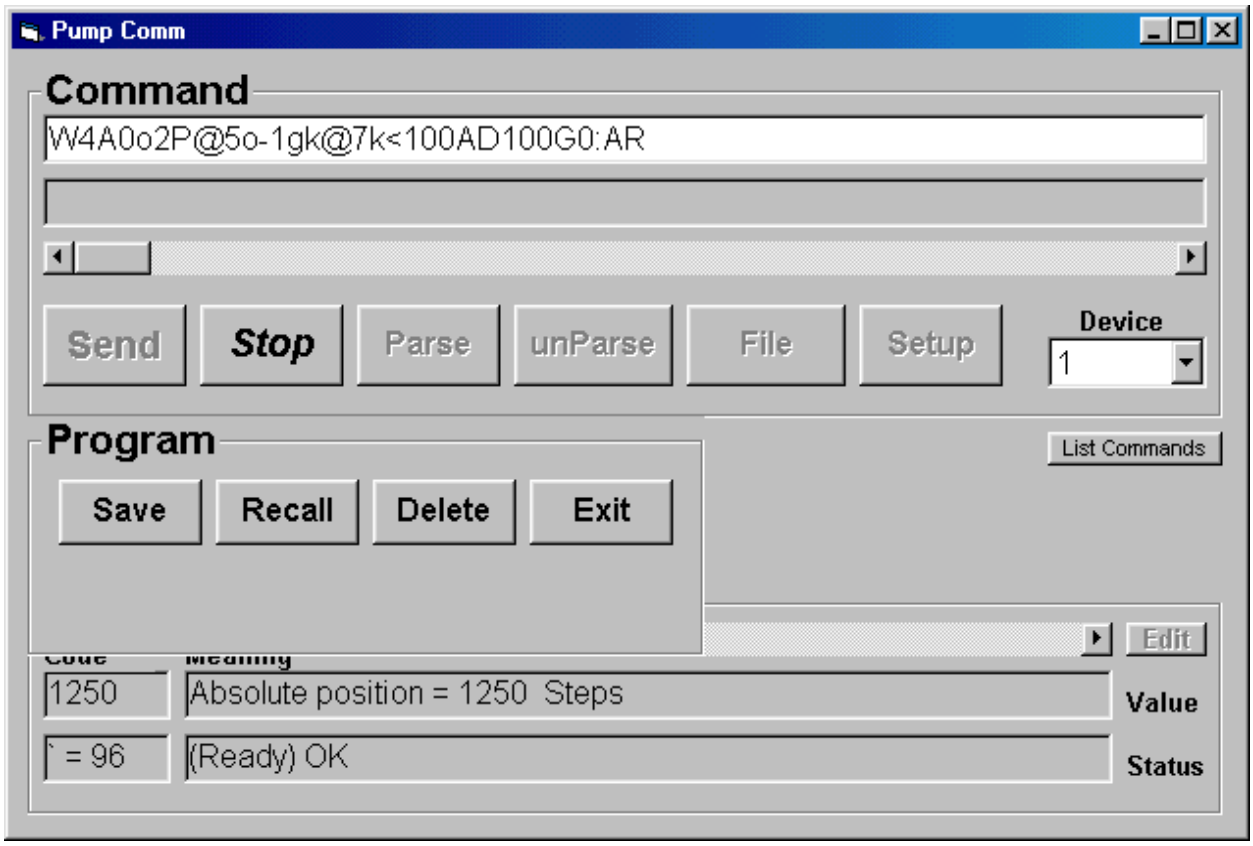


Figure 5 The Program Screen

The **Program** screen is accessed by clicking on the **File** button.

3.1 Saving Command Strings

Command strings (programs) are saved by clicking on the **Save** button, choosing a name for the string to be saved, and then clicking on **Save** button again. A name can be selected from the **Save** drop list, or a new name can be entered into the

Save program list text box.

When the **Save** button is clicked, a program list box appears. Only the text entry box is initially shown. Clicking on the text box produces a list of previously saved programs.

The traditional naming restrictions of Microsoft Windows© do not apply to creating names for your programs. You may type any string of characters you wish, including spaces, punctuation, special keyboard characters, and numbers. The name should contain not more than 36 characters if it is to be entirely visible in the program listing. Longer names can be accommodated by scrolling horizontally with the cursor and arrow keys.

Command strings are stored as records into a single file, ***program.txt***, located in the same directory as the ***KComm.exe*** program. Each record in the file is an ASCII text string terminated by a carriage return/linefeed character sequence. The syntax of a record is

{command string name};{command string}

This simple format permits editing by any text editor or word processor, in addition to using the KComm program. A text editor can also be used to insert and rearrange the command string records, and to copy strings from the file to the Windows clipboard for use in other Windows programs.

3.2 Recalling Command Strings

Command strings which have been given names and saved can be recalled into the **Command** text box by clicking **File –> Recall** and then clicking on a record name. When **Recall** is clicked, only the top text box appears with no name in it. The list of saved names is accessed by clicking in the text box. When a name is clicked, the command string corresponding to that name is placed into the **Command** text box, ready to send or edit. The new command string replaces whatever was previously in the **Command** text box.

3.3 Deleting Saved Command Strings

Command strings saved in the command string file can be deleted by left clicking on the **Delete** button and then clicking on the program name to be deleted.

4.0 PARSING THE COMMAND STRING

The term *parse* means to search through a text or to examine a text for the organization of its components parts, to analyze its grammar. A parsing feature is included in ***KComm*** to support reading and error checking a command string in the **Command** text box. The two buttons which support this are **Parse** and **unParse**.

4.1 Parse

The **Parse** button causes the command string in the **Command** text box to be broken into the individual commands in it, and to display the separated commands in the display bar just below the **Command** text box. While parsing the command string, the commands are checked for errors in syntax (structure) and arguments (values). If an error is encountered, the parsing routine stops and displays the word "Error" at the point following the error. A message box appears telling the nature of the error. By correcting errors in the command string as they are found, a correct command string will eventually result. Errors can be corrected in the **Command** text box string, not the parsed string display.

4.2 unParse

The **unParse** button reverses the parsing process performed by the **Parse** button. The effect is to remove all the command separations and restore the string in the parsing bar back to its unparsed form. This is useful for locating an error in the **Command** text box, since the unparsed string characters line up one-for-one with the characters in the **Command** text box.

4.3 Parsing Example

The concept of parsing, as it is implemented here, can be illustrated by the following example. Note how the parsed string is easier to read.

Unparsed: W4:AAOo2P24000o3P10000M5000o4A0R

Parsed: W4 :A A0 o2 P24000 o3 P10000 M5000 o4 A0 R

5.0 OTHER FEATURES

KComm offers some other convenient features. These are position tracking and the ability to transfer a program string recalled from the device's memory directly into the **Command** text box for editing and reuse.

5.1 The Edit Button

In the **Response** section, at the upper, right end, is a button labeled **Edit**. The **Edit** button allows a stored program to be recalled from a device's memory and transferred to the **Command** text box where it can be edited and saved or sent back to the device for execution or to replace the recalled version.

5.2 Syringe and Valve Position Tracking

KComm provides a mechanism for tracking the position of the syringe and the valve while a move is in progress. There are two screen features associated with this function: **Auto** and **Show Position**.

5.2.1 Show Position

The **Show Position** button is located just under the Command section on the left side. Each click of the button updates a small text display which appears to the right of the button. This text shows the current positions of the syringe and the valve on a pump. The syringe position from zero is shown first, followed by the valve port.

This feature operates by sending queries of the current syringe and valve positions. The status response is not evaluated and the **Response** portion of the program is not affected. There is a small time delay in the command processing, so the displayed positions lag slightly behind the actual hardware positions.

5.2.2 Auto

When you wish the position display to be continuously updated, you could keep clicking the **Show Position** button, but this would quickly become tiresome! The **Auto** check box takes care of this for you. Remember that the display lags behind the actual positions. This feature can be used only after at least one command has been sent.

5.3 *Stop*

The **Stop** button sends a “Terminate” command each time it is clicked. This button stops any running command or program in the addressed device and is always active regardless of what other screen functions are activated.

5.4 Faded Button Legends

When some buttons are clicked and activated, the labels on others may become faded, or “grayed-out”. This indicates the faded buttons are not active and will not respond to clicking. When the active function terminates, the faded buttons are restored. This prevents inappropriate combinations of feature selections.