


Cyrano

Tutorial

CYRANO TUTORIAL

Form 704-961010 — October, 1996 

OPTO 22

43044 Business Park Drive, Temecula, CA 92590-3614

Phone: 800/321-OPTO (6786) or 909/695-3000

Fax: 800/832-OPTO (6786) or 909/695-2712

Internet Web site: <http://www.opto22.com>

Product Support Services:

800/TEK-OPTO (835-6786) or 909/695-3080

Fax: 909/695-3017

E-mail: support@opto22.com

Bulletin Board System (BBS): 909/695-1367

Cyrano Tutorial
Form 704-961010—October, 1996

The information in this manual has been checked carefully and is believed to be accurate; however, Opto 22 assumes no responsibility for possible inaccuracies or omissions. Specifications are subject to change without notice.

Opto 22 warrants all of its products to be free from defects in material or workmanship for 24 months from the manufacturing date code. This warranty is limited to the original cost of the unit only and does not cover installation, labor or any other contingent costs.

FactoryFloor, OptoControl, OptoServer, OptoDisplay, SNAP, and Mystic are registered trademarks of Opto 22. Cyrano, G4, Optomux and Pamux are trademarks of Opto 22.

ARCNET is a registered trademark of Datapoint Corporation.

IBM is a registered trademark and IBM PC, XT, AT, and PS/2 are trademarks of International Business Machines Corporation.

Microsoft and MS-DOS are registered trademarks and Windows is a trademark of Microsoft Corporation.

TABLE OF CONTENTS

Welcome	vii
What Is Cyrano?	vii
What Is This Tutorial?	vii
About This Manual	viii
Document Conventions	viii
About Opto 22	ix
Introduction	1-1
What Topics Will I Cover?	1-1
What Will I Do?	1-1
How Will I Do It?	1-1
Tutorial Requirements	1-1
General Control Concepts	1-5
Cyrano Concepts	1-6
General Programming Terms	1-10
What Did I Do?	1-11
Where Do I Go From Here?	1-11
Getting Started	2-1
What Topics Will I Cover?	2-1
What Will I Do?	2-1
How Will I Do It?	2-1
Starting Cyrano	2-1
Using the Environment Manager	2-2
Getting Around in Cyrano	2-4
Working with Programs	2-6
Exiting Cyrano	2-13
What Did I Do?	2-15
Where Do I Go From Here?	2-15

Configuring the Hardware	3-1
What Topics Will I Cover?	3-1
What Will I Do?	3-1
How Will I Do It?	3-1
Open the Configurator Module	3-1
Configure the Controller	3-2
Configure I/O Units	3-5
Analog I/O Units	3-7
Configure I/O Points	3-8
Digital Points	3-9
Analog Points	3-14
What Did I Do?	3-18
Where Do I Go From Here?	3-18
Creating a New Chart	4-1
What Topics Will I Cover?	4-1
What Will I Do?	4-1
How Will I Do It?	4-1
Create a New Chart	4-1
Draw the Chart Strategy	4-2
Start the Chart from the POWERUP Chart	4-12
Download and Run the Program	4-18
What Did I Do?	4-23
Where Do I Go From Here?	4-23
Editing the Chart	5-1
What Topics Will I Cover?	5-1
What Will I Do?	5-1
How Will I Do It?	5-1
Redraw the Flowchart Logic	5-1
Enter Commands in the New Blocks	5-5
Create Three New Blocks	5-17
Download and Run the Program	5-24
Turn on Auto Stepping	5-26
Observe Items in the Watch Window	5-33
What Did I Do?	5-41
Where Do I Go From Here?	5-41



<i>Creating a Second Chart</i>	6-1
What Topics Will I Cover?	6-1
What Will I Do?	6-1
How Will I Do It?	6-1
Create a New Chart	6-1
Draw the Flowchart Logic	6-2
Enter Commands in the Blocks	6-3
Start the New Chart from the POWERUP Chart	6-17
Download and Run the Program	6-20
What Did I Do?	6-20
Where Do I Go From Here?	6-20
<i>Summary</i>	7-1
General Steps in Creating a Cyrano Program	7-1
Need More Help?	7-1
Product Support	7-1
Training Programs	7-1
<i>Product Support</i>	A-1

Index

WELCOME

WHAT IS CYRANO?

The Cyrano 200 Visual Control Language (“Cyrano,” for short) is a powerful, easy-to-use program that enables you to develop control applications for Opto 22 systems right from your PC. These applications are based on simple flowcharts familiar to anyone involved in process control design. Because these flowchart concepts are fundamental, and because the terminology used to program Cyrano is plain English rather than techno-jargon, you will find Cyrano easy to learn and intuitive, whether or not you have any previous programming experience.

But don’t be fooled by Cyrano’s ease of use. Features such as multicharting, full debugging capabilities, and an extensive set of built-in advanced tools combine to make Cyrano the most powerful and versatile control design program you will ever need.

An inexpensive and readily available IBM-compatible PC workstation, equipped with color graphics and a mouse, is all that you need to run Cyrano. By making selections from Cyrano’s color graphic menus on your PC and using your mouse to draw interconnections, you can create a control chart that defines how you want your application to work. Cyrano then does the rest of the work for you by creating a computer program that runs your application on the Opto 22 Controller.

WHAT IS THIS TUTORIAL?

The Cyrano tutorial is designed with a hands-on approach to help you learn the basics of programming control applications with Cyrano. Depending on your control knowledge and general software experience, the tutorial will require a few hours to complete. If you have previous programming experience, you may be more familiar with some of the terminology used in this tutorial; in any case, most terms are completely defined and **no prior programming experience is necessary!**

The tutorial is designed for use with the Opto 22 demo box as the basic hardware system. The demo box is described in detail in Chapter 1. For those of you comfortable with screwdrivers and wire strippers, an equivalent system is easy to construct.

Because Cyrano is a versatile control language, it would be impossible to cover all of its features and all possible applications in one tutorial. Instead, this tutorial covers the basics of how to get started, introducing many general concepts along the way. Should you desire more in-depth Cyrano instruction, Opto 22 also offers training classes. For information on the availability and types of courses offered, call 800/396-OPTO (6786) and ask for Cyrano training information.

ABOUT THIS MANUAL

The *Cyrano Tutorial* is the third of three volumes in the Cyrano documentation set. This manual offers a detailed, step-by-step introduction to Cyrano application development. It also provides general information on Cyrano concepts and usage.

The other two Cyrano manuals are:

- *Cyrano User's Guide* (Opto 22 form 702) — general information on installing and using Cyrano plus a description of all tools, menus, and dialog box options
- *Cyrano Command Reference* (Opto 22 form 703) — detailed information on every Cyrano command plus important technical information on various command groups

This manual is organized as follows:

- **Chapter 1: Introduction** — tutorial requirements, control concepts, Cyrano concepts, and general programming terminology
- **Chapter 2: Getting Started** — starting Cyrano, using the Environment Manager, getting around in Cyrano, and working with programs
- **Chapter 3: Configuring the Hardware** — configuring the controller, I/O units, and I/O points
- **Chapter 4: Creating a New Chart** — creating a new chart, drawing the chart strategy, starting the chart from the POWERUP chart, and downloading and running the program
- **Chapter 5: Editing the Chart** — redrawing a flowchart, entering commands to light LEDs based on button presses, and using auto stepping and the watch window
- **Chapter 6: Creating a Second Chart** — creating a new chart, drawing new flowchart logic, entering commands to average sensor readings and display the results, and downloading and running the new program
- **Chapter 7: Summary** — brief synopsis of Cyrano programming procedures
- **Appendix A: Product Support** — how to reach Opto 22



DOCUMENT CONVENTIONS

- **Bold** typeface indicates text to be typed. Unless otherwise noted, such text may be entered in upper or lower case. (Example: “At the DOS prompt, type **cd \windows.**”)
- *Italic* typeface indicates emphasis and is used for book titles. (Example: “See the *Cyrano User’s Guide* for details.”)
- Names of menus, commands, dialog boxes, fields, and buttons are capitalized as they appear in the product. (Example: “From the File menu, select Print to bring up the PRINT TOPIC dialog box.”)
- File names appear in all capital letters. (Example: “Open the file TEST1.TXT.”)
- Key names appear in small capital letters. (Example: “Press SHIFT.”)
- Key press combinations are indicated by hyphens between two or more key names. For example, SHIFT-F1 is the result of holding down the SHIFT key, then pressing and releasing the F1 key. Similarly, CTRL-ALT-DELETE is the result of pressing and holding the CTRL and ALT keys, then pressing and releasing the DELETE key.
- “Press” (or “click”) means press and release when used in reference to a mouse button.
- Menu commands are sometimes referred to with the Menu-->Command convention. For example, “Select File-->Run” means to select the Run command from the File menu.
- Numbered lists indicate procedures to be followed sequentially. Bulleted lists (such as this one) provide general information.

ABOUT OPTO 22

Opto 22's quest to deliver total control to industrial automation customers dates back to its beginnings in 1974 with the introduction of optically-isolated solid-state relays. Today, Opto 22 is the number one provider of I/O systems, with more than 45 million points of I/O working reliably worldwide. After earning a reputation for consistent innovation and leadership in automation hardware, Opto 22 realized it was time to take a new approach to control software. In 1988, Opto 22 introduced the first flowchart-based control programming language. Opto 22 continues to deliver successively more advanced generations of hardware and software.

Opto 22's products are organized in a comprehensive 7-layer architecture that establishes the foundation for future innovation and provides strong support for today's customers. (See 7-layer architecture at right.)

Opto 22 has targeted four key markets: Enterprise, Systems, Components, and Verticals. Opto 22 provides bi-directional data exchange software for integrating manufacturing data into the Enterprise Market, software-driven control solutions for the Systems Market, hardware and drivers for the Components Market, and Vertical Market solutions for the water, energy, and semiconductor industries.

All Opto 22 products are manufactured in the U.S. at the company's headquarters in Temecula, California, and are sold through a global network of distributors, system integrators, and OEMs. Sales offices are located throughout the United States. For more information, contact Opto 22, 43044 Business Park Drive, Temecula, CA 92590-3614. Phone Opto 22 Inside Sales at 1-800-452-OPTO or Opto 22 Headquarters at 909-695-3000. Fax us at 909-695-3095.

For fast delivery of marketing and technical documentation, try our free OptoFaxBack Service at 1-800-474-6786.

You can also visit our Web site at <http://www.opto22.com>.

**SOFTWARE
CONTROLLERS
INTERFACES
BRAIN BOARDS
RACKS
I/O MODULES
SOLID-STATE RELAYS**

INTRODUCTION



WHAT TOPICS WILL I COVER?

- Tutorial requirements
- General control concepts
- Flowchart structure
- Multitasking
- POWERUP and INTERRUPT charts
- Internal/external values (IVALs/XVALs)
- General programming terms

WHAT WILL I DO?

- Learn some basic programming concepts.
- Become familiar with the Cyrano language structure.

HOW WILL I DO IT?

TUTORIAL REQUIREMENTS

The minimum hardware and software requirements for this tutorial are listed below.

Hardware Requirements

1. An IBM or compatible personal computer with the following recommended minimum configuration:
 - 80386 microprocessor-based system running at 33 MHz
 - 1 MB RAM
 - Opto 22 AC37 AT Bus Serial Communications Coprocessor or ARCNET Adapter and appropriate cables

- VGA display adapter
 - Hard disk drive
 - Mouse
2. Opto 22 Demo Box (or similarly configured system)

The demo box consists of a G4LC32SX controller with EEPROM (version 3.0 or higher), two remote digital bricks, and a remote analog brick. Each digital brick includes eight input (G4IDC5) and eight output (G4ODC5) modules. The analog brick includes five input modules (two G4AD4, two G4AD6, one G4AD13) and three analog output modules (G4DA5).

Figure 1-1 presents a diagram of the demo box. Tables 1-1, 1-2, and 1-3 list the hardware wired to each module.

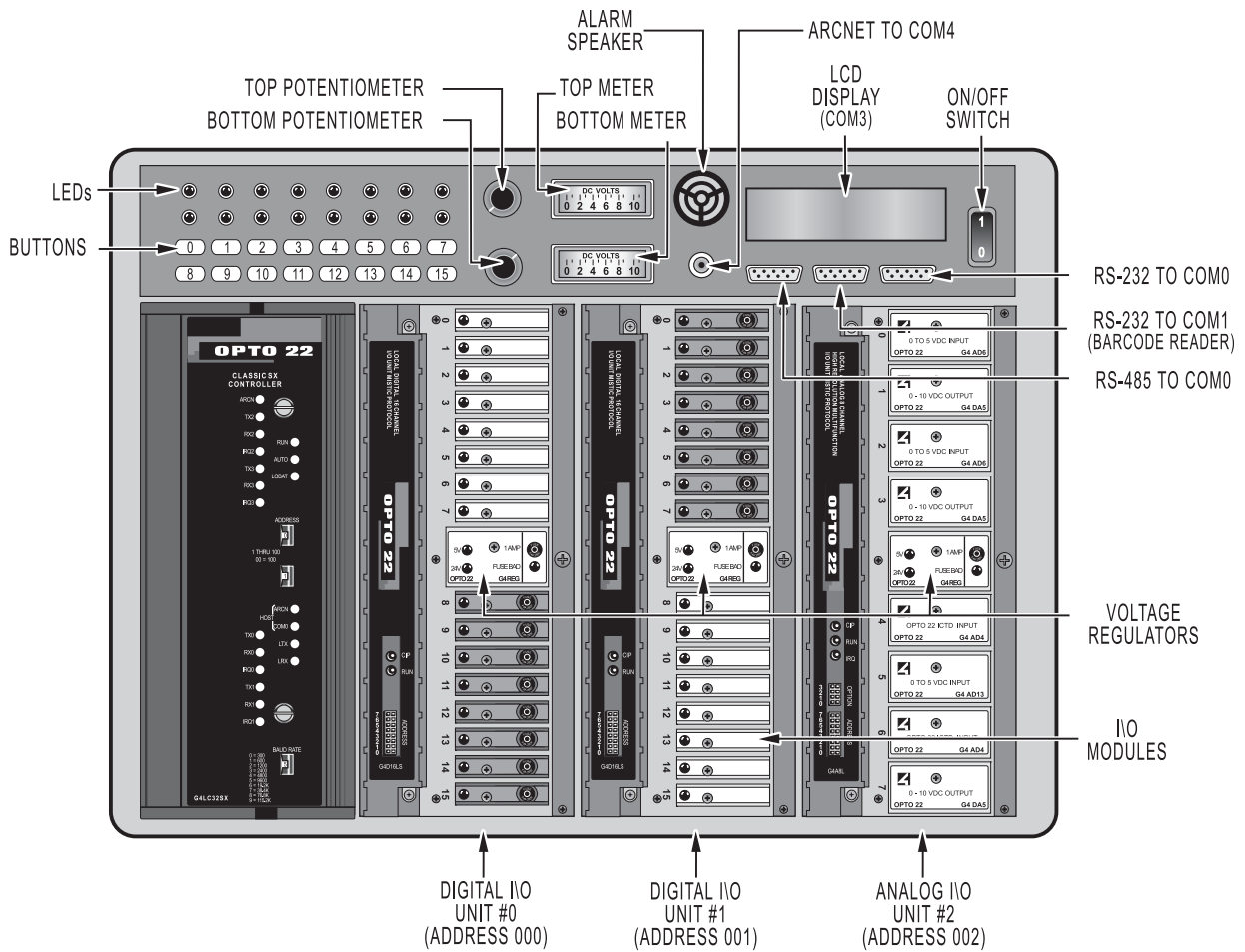


Figure 1-1: Opto 22 Demo Box

Table 1-1: Configuration of Digital Brick at Address 0

Address	Type of Module	Wired Hardware
00	Input - G4IDC5	Button 0
01	Input - G4IDC5	Button 1
02	Input - G4IDC5	Button 2
03	Input - G4IDC5	Button 3
04	Input - G4IDC5	Button 4
05	Input - G4IDC5	Button 5
06	Input - G4IDC5	Button 6
07	Input - G4IDC5	Button 7
08	Output - G4ODC5	LED 0
09	Output - G4ODC5	LED 1
10	Output - G4ODC5	LED 2
11	Output - G4ODC5	LED 3
12	Output - G4ODC5	LED 4
13	Output - G4ODC5	LED 5
14	Output - G4ODC5	LED 6
15	Output - GRODC5	LED 7

Table 1-2: Configuration of Digital Brick at Address 1

Address	Type of Module	Wired Hardware
00	Output - G4ODC5	LED 8
01	Output - G4ODC5	LED 9
02	Output - G4ODC5	LED 10
03	Output - G4ODC5	LED 11
04	Output - G4ODC5	LED 12
05	Output - G4ODC5	LED 13
06	Output - G4ODC5	LED 14
07	Output - G4ODC5	LED 15/Beeper
08	Input - G4IDC5	Button 8
09	Input - G4IDC5	Button 9
10	Input - G4IDC5	Button 10
11	Input - G4IDC5	Button 11
12	Input - G4IDC5	Button 12
13	Input - G4IDC5	Button 13
14	Input - G4IDC5	Button 14
15	Input - G4IDC5	Button 1

Table 1-3: Configuration of Analog Brick at Address 2

Address	Type of Module	Wired Hardware
00	Input - G4AD6	Top Potentiometer
01	Output - G4DA5	Top Meter
02	Input - G4AD6	Bottom Potentiometer
03	Output - G4DA5	Bottom Meter
04	Input - G4AD4	(No hardware connection)
05	Input - G4AD4	(No hardware connection)
06	Input - G4AD13	Input - PID Simulator
07	Output - G4DA5	Output - PID Simulator

Software Requirements

- Cyrano version 3.0
- DOS version 5.0 or higher
- The following suggested CONFIG.SYS and AUTOEXEC.BAT file configurations:

CONFIG.SYS

```
DOS=HIGH, UMB
DEVICE=C:\HIMEM.SYS
FILES=30
BUFFERS=10
SHELL=C:\DOS\COMMAND.COM C:\DOS\ /e:512 /p
```

AUTOEXEC.BAT

```
LOADHIGH C:\SMARTDRV.EXE
LOADHIGH C:\MOUSE.COM
PATH C:\;C:\DOS;C:\CYRANO
PROMPT $P$G
```

GENERAL CONTROL CONCEPTS

Automation

Automation is a means of adding intelligence to an industrial process. Automating a process decreases the need for active personal participation in the process. It also improves the performance, accuracy, and reliability of a process.

Programmable electronic components called “controllers” provide the intelligence required for automation. An Opto 22 Mystic controller receives a set of instructions on how to control every aspect of a process. A programmer creates an instruction set and passes it to the controller (downloads it) using a software package called the Cyrano 200 Visual Control Language (“Cyrano,” for short). Since the instruction set is stored in the controller’s electronic memory, much as a small computer would store it, the instructions can easily be modified.

Digital and Analog Inputs and Outputs

An industrial process can be composed of many different components. All of the system components communicate with the controller by way of input/output (I/O) points, also called I/O modules.

Input points are wired to hardware that brings information *into* the controller from the process. Examples of devices wired to input points are buttons, switches, and sensors. The controller takes the information from the input points, processes it using the software instruction set, and returns information pertinent to the process. This information is returned to the process by means of output points.

Output points are wired to hardware that receives information *from* the controller and use this information to control different components of the process. For example, lights, motors, and valves are all devices wired to output points.

There are two types of I/O points, digital and analog. **Digital** points have only two values, on or off (true or false). Push buttons and LEDs are examples of digital devices. **Analog** points have a range of values. Temperature and pressure are examples of analog information.

CYRANO CONCEPTS

Cyrano allows you to write control applications with little or no prior programming experience. Cyrano is a self-documenting language based on flowcharts that makes it easy to turn thoughts into action.

There are three fundamental steps to creating and running a Cyrano program:

1. *Develop an application in the Cyrano Configurator.*
2. *Download the application to a Mystic controller and debug it in the Cyrano Debugger.*
3. *Run the application stand-alone on the Mystic controller.*

Program Structure

A Cyrano program is divided into segments called flowcharts ("charts," for short). A chart has three different status levels: running, suspended, and stopped. A running chart is actively performing its assigned task. A suspended chart is temporarily paused. A stopped chart is inactive. Every chart in a Cyrano program can change the status of any other chart in the program. Every chart is independent of every other chart. Any combination of charts may be running simultaneously.

Every chart in Cyrano has a BLOCK-0, which is the location at which each chart begins executing its logic. BLOCK-0 can be renamed but never deleted.

Every Cyrano program has a POWERUP chart and an INTERRUPT chart. The POWERUP chart is the only chart started automatically at power-up or run time; hence, program execution begins in BLOCK-0 of the POWERUP chart. All other charts must be started with a START CHART command at the appropriate time in the control strategy.

The ability to run several charts, each performing a different task, is accomplished through a time-slicing technique called **multitasking** (also called multicharting). The Mystic controller contains a multitasking kernel that gives a single controller the ability to simultaneously run up to 31 charts, in addition to the HOST task, by assigning each task a 500-microsecond time slice.

The HOST task, an invisible "chart" whose purpose is to communicate to a Cyrano Debugger or to a Mystic Man-Machine Interface (MMI), is assigned the first 500-microsecond time slice. *Each chart in a running or suspended state is considered a task and is allocated a 500-microsecond time slice.* Charts that are stopped are not assigned a time slice.

When the POWERUP chart is running, it is always assigned the time slice after the HOST task. When a chart ends or its time slice expires (whichever occurs first), control passes to another chart or task in the program. Control continues passing in a round-robin manner, eventually returning to the HOST task and repeating the cycle. The list of ordered tasks is called the task queue.

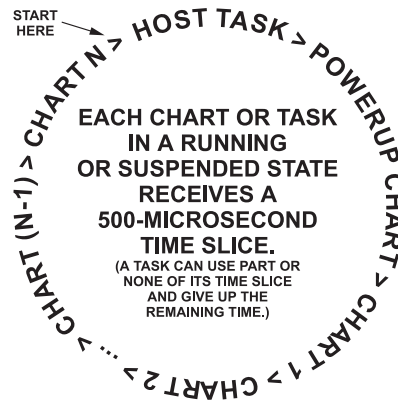


Figure 1-2: Time Slicing or Multicharting

The INTERRUPT chart is special in that it is normally in a suspended state. If an interrupt generated by an I/O unit causes the INTERRUPT chart to start running, it is always the next chart to receive a time slice. The INTERRUPT chart floats in the task queue.

Any chart can be allocated several time slices in a row if its priority is changed through the SET PRIORITY command. When 32 charts are running, each with a single time slice, each chart will be executed at least once every 16 milliseconds, sooner if fewer charts are running. With this quick control pass, all charts appear to run simultaneously. By placing time-critical tasks, such as the monitoring of an emergency stop, in separate charts, you will ensure that these processes receive attention at predictable intervals.

Program Design

When a new Cyrano program is created, there are two predetermined charts: POWERUP and INTERRUPT. Every other chart in Cyrano is added by the programmer.

Each chart should control one aspect of the application. Hence, the first step in designing a Cyrano program is to take the desired control application and divide it into related tasks. When dividing your application into tasks, keep in mind that only 32 charts (including the HOST task) can be running or suspended at any time. However, you can write up to 1,295 charts in a single program, given enough controller memory.

Tasks that need to be continuously monitored can be placed in a chart that is continuously running. Tasks that need to be performed only at given times can be stopped and started by other charts. When the application is complex and consists of many tasks, it is beneficial to have one chart that starts and stops the other charts. The POWERUP chart or another user-created chart called MAIN can be used for this purpose. The POWERUP chart should be used to initialize variables, perform setup commands, and start other charts. However, the POWERUP chart should *not* be used for major control strategy.

Chart Design

After dividing your application into tasks, it is time to start planning each chart's strategy. There are essentially two types of flowchart logic:

Flow-Through Logic

A chart with flow-through logic performs a set of specific commands and then stops. A flow-through logic chart has a beginning and an end. The end of a chart is a condition or operation block that has no exit.

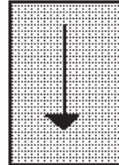


Figure 1-3: Flow-Through Logic

The POWERUP and INTERRUPT charts (as well as subroutine charts) should always have flow-through logic. Any user-created chart that does not need to run continuously should be designed with flow-through logic.

Although iteration looping can be used in flow-through logic, condition looping (such as waiting for something to occur) cannot be used. Delays are allowed in flow-through logic but should be used sparingly.

Loop Logic

A chart that needs to run continuously should be designed with loop logic. A chart with loop logic has no end. It loops through a set of operations and conditions continuously. A loop logic chart can have several paths through which the direction of the logic may flow depending on certain criteria.

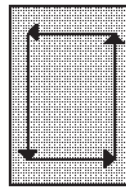


Figure 1-4: Loop Logic

There are three basic building blocks you can use to create your program logic:

Operation Blocks

Rectangles that contain instructions to be executed in a program. Several instructions or commands can be placed in one block. Operation blocks can have many entrances but only *one* exit.



Figure 1-5: Operation Block

Condition Blocks

Diamonds that contain questions that control the logical flow of a program. Condition blocks can have many entrances, but only *two* exits: True and False. More than one question may be asked within a single condition block. If *all* questions within the block evaluate True, the block will exit True. If any question in the block evaluates False, the block will exit False.



Figure 1-6: Condition Block

Connection Lines

Arrows that define the path of the program logic.



Figure 1-7: Connection Line

IVALs and XVALs

Every input and output point in Cyrano has a hardware value (XVAL) and a software value (IVAL) associated with it. The hardware value is the measurable value at the module. The software value is the value of the I/O point in the Cyrano program running on the controller.

The XVAL and IVAL share information when the point is enabled and referenced (used) in the Cyrano program. As soon as the Cyrano program encounters an I/O point that is enabled, information is copied between the hardware and software values. For output points, the software value (IVAL) is copied to the hardware value (XVAL). For input points, the hardware value (XVAL) is copied to the software value (IVAL).

GENERAL PROGRAMMING TERMS

Variables

A variable represents information. The information that a variable represents is called the value of the variable. The name of a variable remains the same during program execution. However, the value of a variable can change. There are three types of variables: floating point, integer, and string. The difference between them is the type of data they store.

Floats

A floating point value (or “float”) is a numeric value that contains a decimal point, such as 3.14159, 1.0, and 1234.2. Cyrano uses IEEE single-precision floats with rounding errors of no more than one part per million.

Integers

An integer value is a whole number with no fractional part. Examples of integer values are -1, 0, 1, 999, and -456. Cyrano uses 32-bit signed integers.

Strings

A string variable stores text and any combination of ASCII characters, including control codes and extended characters. When defining a string variable, you must specify the width of the string. The width is the maximum number of characters that the variable may hold.

A string variable can contain numeric characters. However, ASCII characters representing a number are *not* the same as the numeric value. In other words,

“3.33” does not equal 3.33
string does not equal *value*

A string of numeric characters can be converted into a number and vice versa. *A numeric value must be converted into a string of characters to be displayed on a screen.*

When converting floating point values into strings, you must describe the format, that is, how many characters and decimal places should be displayed. Floating point numbers can be formatted to display in many different ways. For example, the float variable $PI = 3.14159$ can be displayed with the following formats:

Table 1-4: Format for Floating Point Numbers

Format	String
X	3.
X.X	3.1
X.XX	3.14
X.XXX	3.141

Constants

A constant is a string or numeric value that *never* changes.

Tables

Tables in Cyrano are one-dimensional arrays. A one-dimensional array is a single variable name that represents several values. Each different value is referenced by an index number. For example, a string table named OPTO_SSR_RELAYS with a length of 99 can store 100 different relay names (note that 0 is a valid index value). Each model of relay can then be referenced by an index number. Table 1-5 shows a portion of this example string table. For example, the value at index 2 is equal to the string "240A45." Tables can store numeric values as well as strings.

Table 1-5: Example String Table OPTO_SSR_RELAYS

Index	Value
0	120D10
1	120D25
2	240A45
3	480D15
4	575D25

WHAT DID I DO?

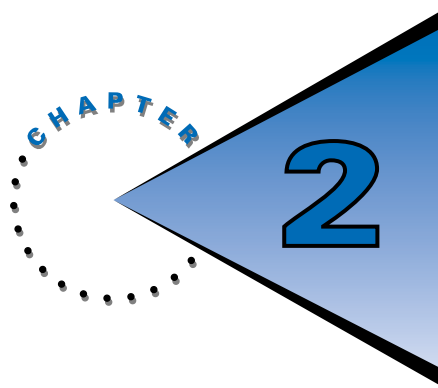
In this chapter you reviewed the tutorial requirements, became familiar with general control concepts, and got an introductory look at several Cyrano concepts, including multitasking, IVALs/XVALs, program execution, and the program and chart design. You also did a quick review of general programming terminology.

WHERE DO I GO FROM HERE?

Now that you have some familiarity with the control process and introductory Cyrano concepts, you are ready to launch Cyrano and create a new program. You will become familiar with using the Cyrano Environment Manager, saving Cyrano programs, and backing up your work.

Proceed to Chapter 2.

GETTING STARTED



WHAT TOPICS WILL I COVER?

- Starting Cyrano
- Using the Environment Manager
- Getting around in Cyrano
- Working with programs (creating, saving, and making backups)
- Exiting Cyrano

WHAT WILL I DO?

- Create a new program called TUTOR in your Cyrano directory.
- Save the new program.
- Make a backup copy of the TUTOR program in a separate directory.
- Create a second backup of the program called TUTOR2 on a floppy disk drive.
- Exit the Configurator and the Environment Manager.

HOW WILL I DO IT?

STARTING CYRANO

To run Cyrano from DOS, change directory to your Cyrano directory (C:\CYRANO, by default), then type **cyrano** at the prompt. For example:

```
C:\>CD CYRANO
```

```
c:\cyrano>cyrano
```

To launch Cyrano from Windows, double-click the left mouse button over the Cyrano icon in the Opto 22 program group:

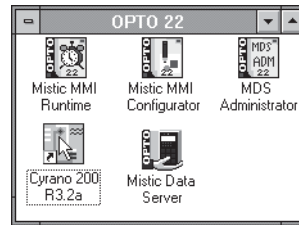


Figure 2-1: Cyrano Program Icon

USING THE ENVIRONMENT MANAGER

The Cyrano Environment Manager is the first screen displayed when Cyrano is launched. The Environment Manager consists of a menu bar and the Cyrano logo. The menu bar includes three menu choices: Cyrano, Tools, and System. Each menu includes several options.

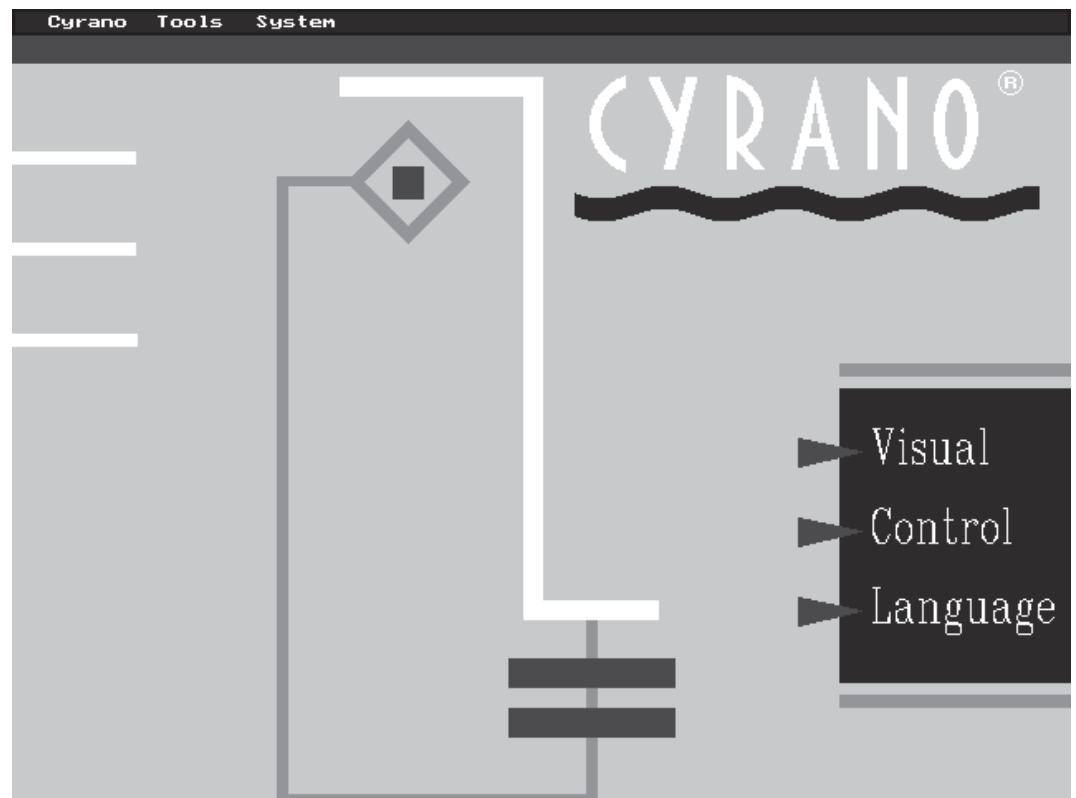


Figure 2-2: Cyrano Environment Manager

Cyrano Menu

The following options are available from the Cyrano menu.

Configurator

Launches the Configurator, the module in which Cyrano programs are written and edited.

Debugger

Launches the Debugger, the module that compiles programs and downloads them to the Mystic processor. From this module you can examine and modify I/O points and variables. You can also monitor the step-by-step execution of your Cyrano programs using the SINGLE STEP and AUTO STEP tools.

OnLine

Launches the OnLine module, which allows you to make changes to a completed Cyrano program after it is running on the processor without stopping the process. This module enables you to make changes to applications that cannot be interrupted for minor modifications or enhancements. The OnLine module contains a subset of the commands and options available in the Configurator module. Many modifications can be made to an active Cyrano program using the OnLine module; however, new variables cannot be defined and new I/O points cannot be configured from this module. If the process you are designing cannot be interrupted after it has been installed, it may be a good idea to configure a few unused variables and I/O points for future use.

Subroutine

Launches the Subroutine module, a subset of the Configurator module in which subroutines can be written. Subroutines are customized procedures that can be used by multiple Cyrano programs or multiple charts within a single Cyrano program. Subroutines are created to perform tasks that may be executed several times. Once created, subroutines are used like regular commands in Cyrano.

Version

Displays the version of Cyrano you are using.

Quit

Exits the Environment Manager.

Tools Menu

The Tools menu lists some of the many tools and utilities available with Cyrano.

Init

Initializes variables and table values in a Cyrano program using a MISTIC.INC file. The MISTIC.INC file contains initialization values for variables. Init is useful for initializing tables, such as the string table shown in Table 1-4.

Backup/Restore

Creates a full backup of a Cyrano program, including all associated files, such as the MISTIC.INC and subroutine files. This utility also restores a program from the backup.

ChartDXF

Converts Cyrano chart files into .DXF format, allowing them to be plotted on CAD-supported devices.

Bill of Material

Generates a file listing all hardware referenced in a Cyrano program.

System Menu

Only one option is available under this menu.

Enable NULOGIC

Toggles access to the Cyrano motion commands on and off. If your application does not require motion control, this option can be left disabled.

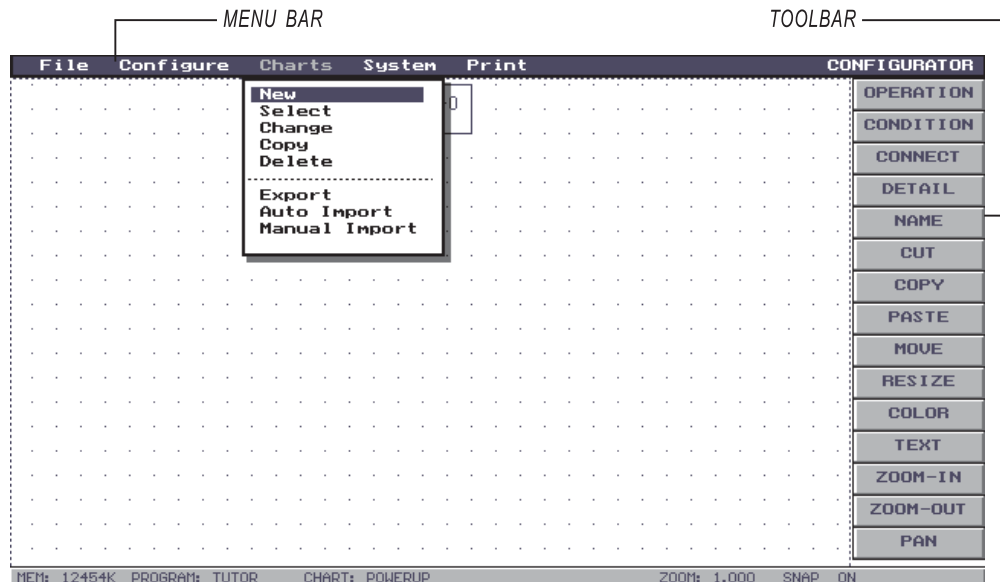
GETTING AROUND IN CYRANO

Figure 2-3: Menu Bar and Toolbar

Menu Bar

Items on the menu bar in Cyrano can be accessed using the keyboard, mouse, or any combination of mouse and keyboard. The menu bar toggles on and off from the keyboard when you press the ESC key. To select a menu on the menu bar using the mouse, position the cursor on the item and press the left mouse button. Click the right mouse button to release the menu bar.

Once a menu is selected (as evidenced by a highlighted item), you can use the four arrow keys or mouse movements to maneuver through the available commands. Press ENTER or click the left mouse button to execute a highlighted command.

In summary, the left mouse button functions like the ENTER key and the right mouse button performs like the ESC key:



LEFT CLICK = ENTER

RIGHT CLICK = ESC

Figure 2-4: Mouse and Keyboard Equivalents

Toolbar

The side toolbars in the Configurator, OnLine, and Subroutine modules contain tools to create a program or subroutine. The toolbar in the Debugger module contains tools to debug and monitor a program.

Select tools from the side toolbar by using the left mouse button. *When a tool is highlighted, you cannot use another tool or make choices from the menu bar.* To release the selected tool, press the right mouse button or the ESC key.

Dialog Boxes

Dialog boxes can be closed using the ESC key or right mouse button. Dialog boxes that contain an ACCEPT button **must** be closed using ACCEPT or the new information entered in that dialog box will be disregarded.

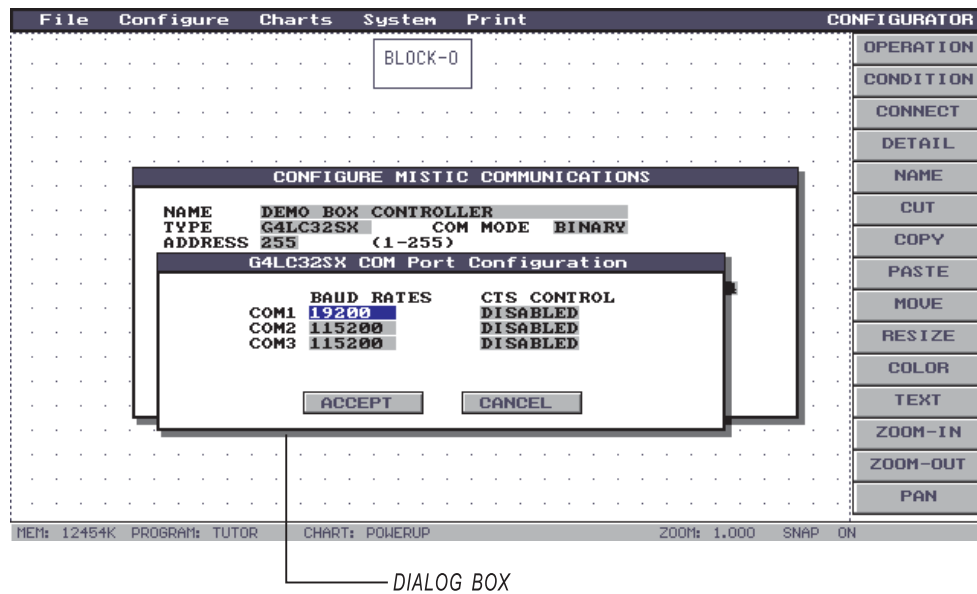


Figure 2-5: Dialog Boxes

Information entered in dialog box fields should be followed by a press of the left mouse button or the ENTER key to accept the entered information. Note that a selection in one field may restrict access to other fields. If the information for a field is predetermined or unnecessary, the field will be inaccessible.

In general, you can use the up and down arrow keys to move from field to field in a dialog box. An exception to this involves empty fields. To return to a previous field from an empty field, press the ESC key once.

Caution: If you have made changes to a field, do not move out of the field by using arrow keys. Every change or entry must be followed by an ENTER keystroke or a left mouse button click.

WORKING WITH PROGRAMS

We will now proceed step by step through the process of launching the Configurator, opening and saving a new program, and making a backup copy of the program.

Launch the Configurator module

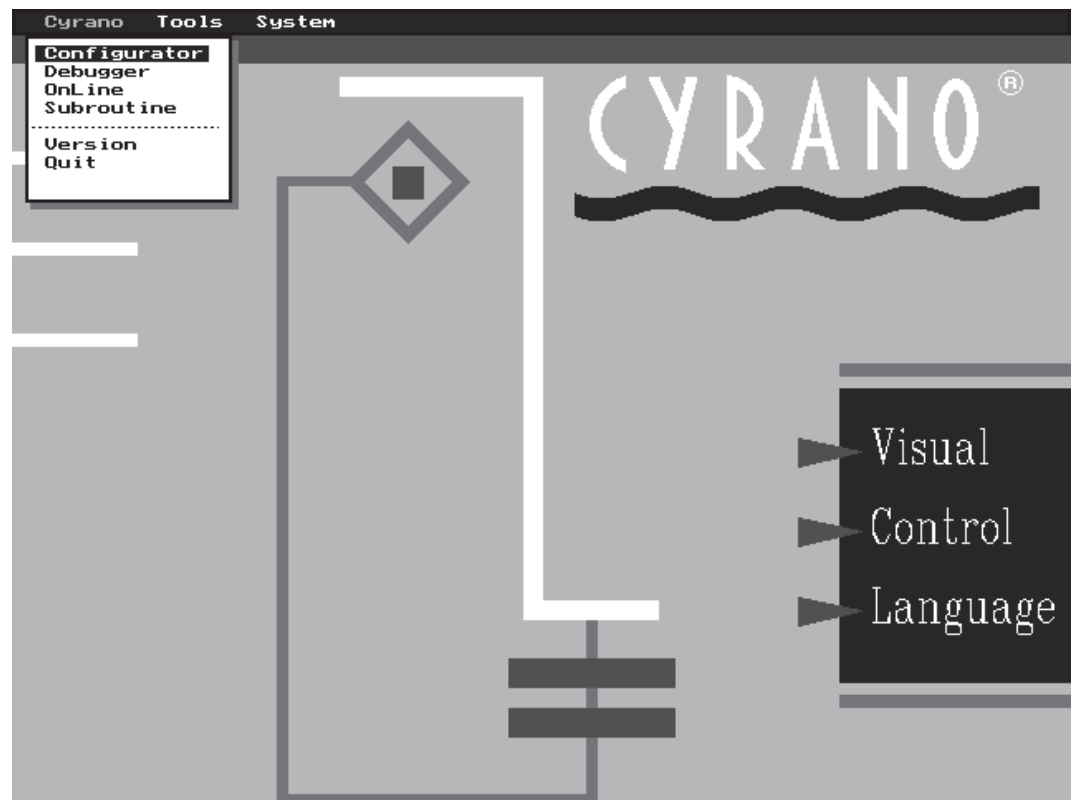


Figure 2-6: Opening the Configurator Module

All programs are written in the Configurator module. To launch this module, select Configurator from the Cyrano menu in the Environment Manager. The Configurator automatically loads the last Cyrano program that was edited on your computer. If the last edited program could not be located, or if you have just installed a new version of Cyrano, a default program called NONAME will load.

Open a new program

To load a different program or to create a new program, select Load from the Configurator's File menu. (Note that in Figure 2-7 below, the grid is enabled. To turn the grid on or off, select Grid from the System menu.)

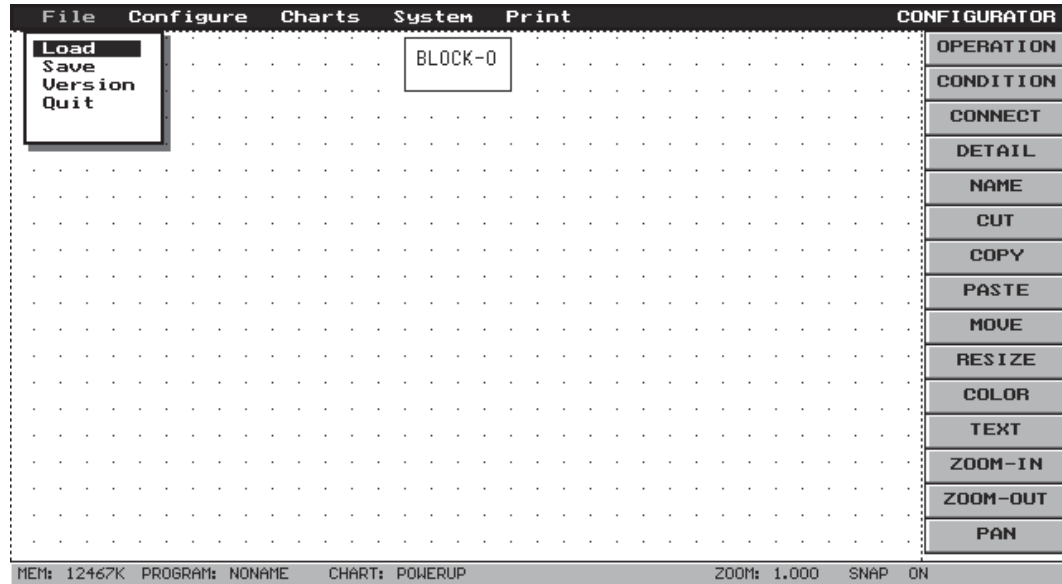


Figure 2-7: Creating a New Program

If Cyrano has loaded the default program called NONAME, or if you have made changes to the currently loaded program, a dialog box will prompt you to save your program. If you have **not** been editing a program you wish to save, select NO by pressing the right arrow key and then pressing ENTER.



Figure 2-8: Canceling a Save

The LOAD FILE dialog box will now appear. The name and path of the currently loaded Cyrano program are displayed in the FILE NAME and PATH boxes. Use the down arrow key to highlight the FILE NAME option and press ENTER. The cursor now advances to the FILE NAME field. Type the name of the new program. As an example, the name TUTOR has been entered as the new program name in Figure 2-9.

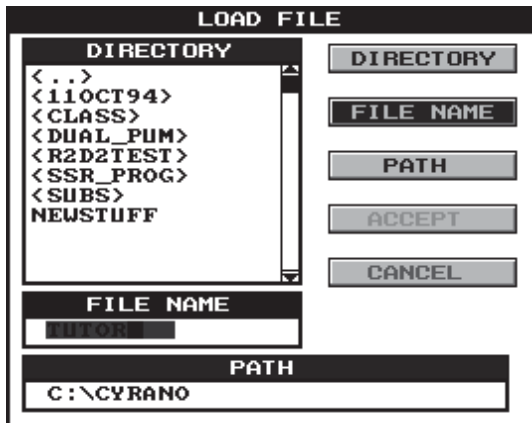


Figure 2-9: Entering a File (Program) Name

After typing the program name, press ENTER and ACCEPT will become highlighted. Verify that the program name and file path are correct, then press ENTER while ACCEPT is highlighted to load this program. If you select CANCEL or press the ESC key, the LOAD FILE dialog box will close *without* loading a new program into the Configurator.

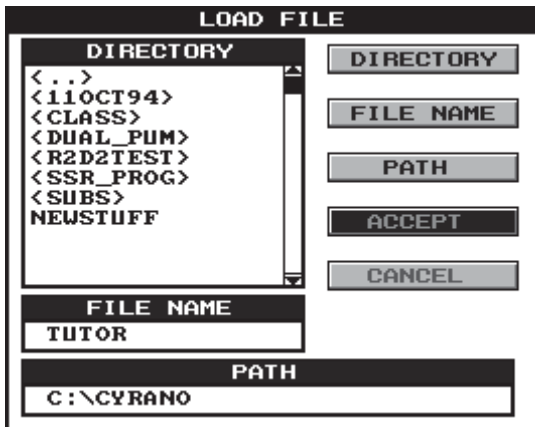


Figure 2-10: Accepting a File to Be Loaded

The DIRECTORY option can be used to select the path and/or file name of a previously written program. The path can also be typed directly by selecting the PATH option and pressing ENTER.

Save the program

To save your program, choose Save from the File menu.

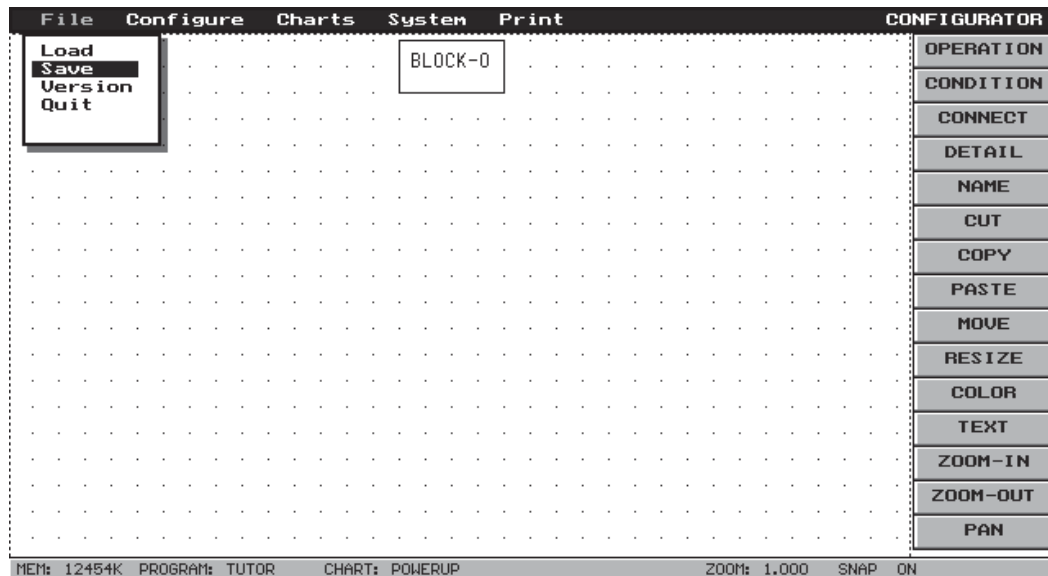


Figure 2-11: Saving a Program

The file name and directory path of the currently loaded program will appear in the SAVE FILE dialog box.

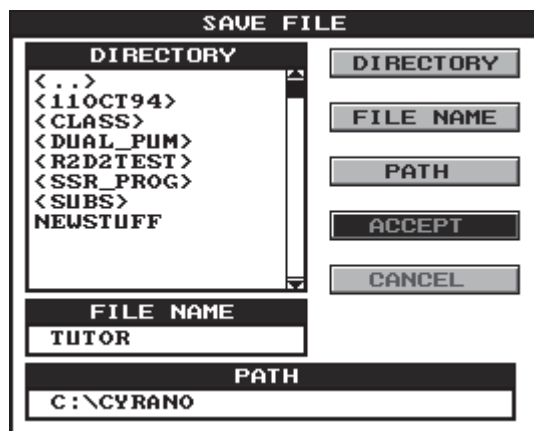


Figure 2-12: Accepting the Save

Select ACCEPT to save your program with the file name and path displayed in the appropriate fields. It is good programming practice to save your program every 15–30 minutes depending on how much editing you are doing.

Make a backup of your program

It is also good practice to create backup copies of your Cyrano program. Frequent backups of your Cyrano program should be made while you are in the development stages. It is even more important to keep several copies of your completed Cyrano project. Uploading of Cyrano programs from the Mystic controller is not currently supported.

Since modifications *cannot* be made to a Cyrano program without the original source code, accurate record keeping is a necessity. Several copies of a program should be kept in different locations. Three copies will usually suffice: one on a hard drive, one on a floppy, and one stored off-site.

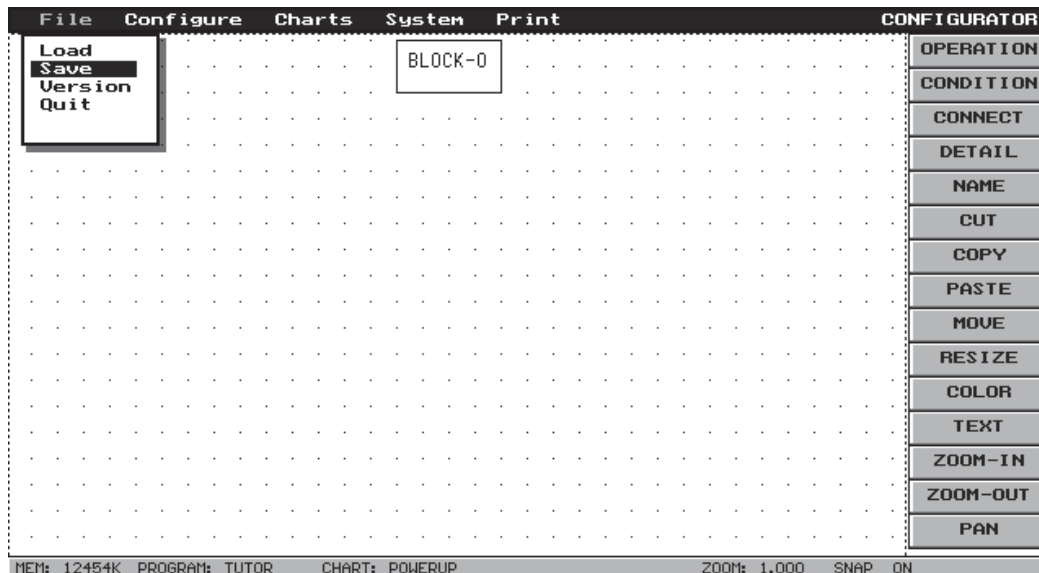


Figure 2-13: Saving a Program

Open the SAVE FILE dialog box by selecting Save from the File menu.

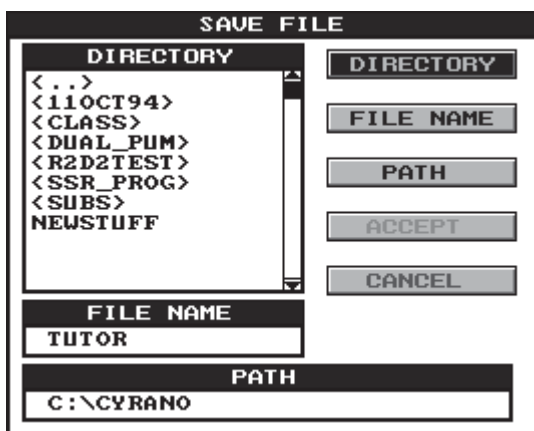


Figure 2-14: Changing Directories

Make a backup of your program by changing the name or directory where you are going to save your program. The DIRECTORY option allows you to change paths. Highlight DIRECTORY and press ENTER to advance your cursor to the DIRECTORY list.

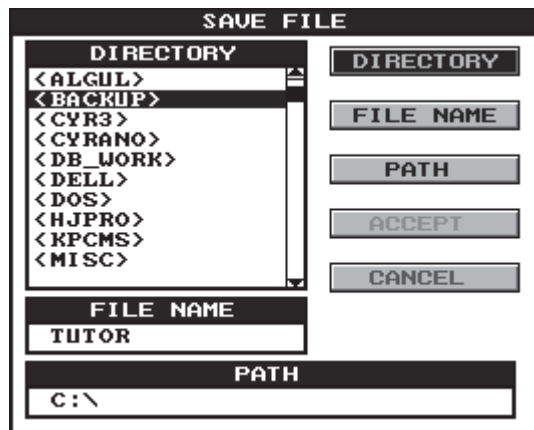


Figure 2-15: Selecting a Directory

Scroll through the DIRECTORY list by using the up or down arrow keys. To change to a subdirectory, highlight the directory (for example, BACKUP in Figure 2-15) and press ENTER. You can move to a higher directory by selecting <..> from the list (see Figure 2-16). Once you have chosen a new directory, press ESC to remove your cursor from the DIRECTORY list.

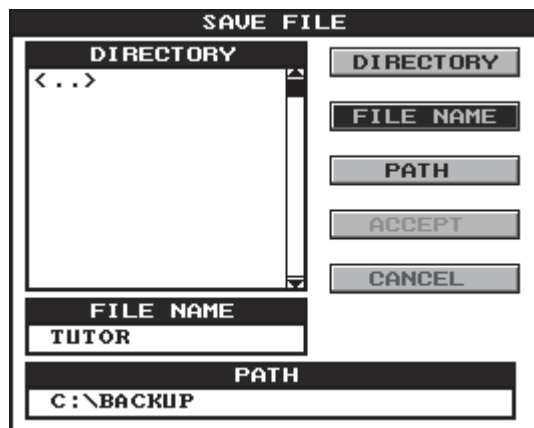


Figure 2-16: Changing a File Name

You may wish to change the file name to distinguish it from the current working program. To do so, select the FILE NAME option and press ENTER. Type a new name (such as TUTOR2) in the FILE NAME field.

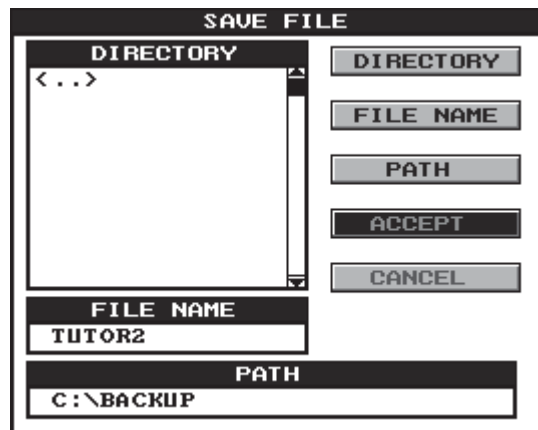


Figure 2-17: Accepting the Save

Once you have chosen the desired path and file name for the backup, highlight ACCEPT and press ENTER.

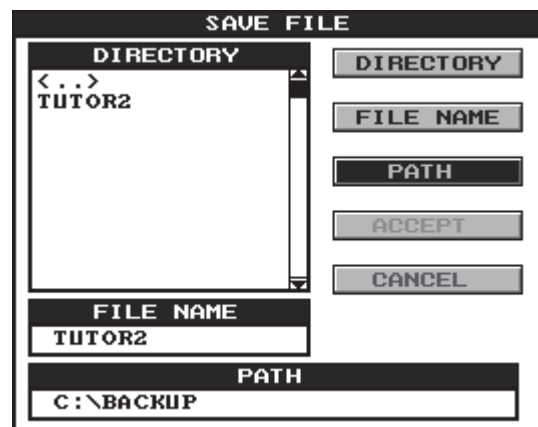


Figure 2-18: Changing the Path

A backup can also be made to a disk drive by changing the path. Insert a formatted disk into the drive. Select the PATH option and press ENTER. Type in the appropriate path to the disk drive (for example, B:\) and press ENTER. After the path has been changed, highlight ACCEPT and press ENTER. See Figure 2-19.

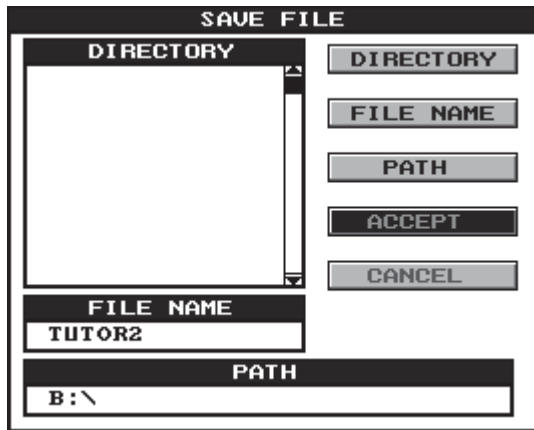


Figure 2-19: Accepting the Save

Program backups can also be made by using the Backup/Restore utility. This utility makes a backup of the Cyrano program and all related files, such as the MISTIC.INC and subroutine files.

EXITING CYRANO

To exit the Configurator, select Quit from the File menu.

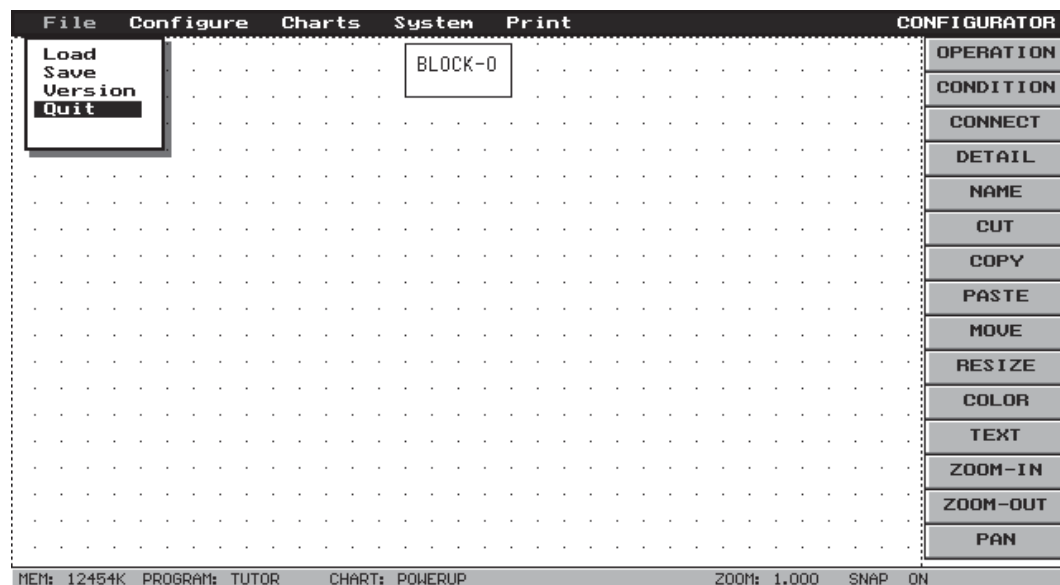


Figure 2-20: Exiting the Configurator

A dialog box (Figure 2-21) will appear to ask you if you are sure you wish to exit. To return to the Configurator, use the right arrow to highlight NO and press ENTER. To exit the Configurator, press ENTER while YES is highlighted.

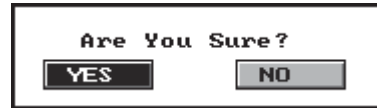


Figure 2-21: Confirming the Exit

When you exit the Configurator you will return to the Environment Manager. You will then be able to access many of the other tools, utilities, and modules in Cyrano.

To exit Cyrano altogether, select Quit from the Environment Manager's Cyrano menu.

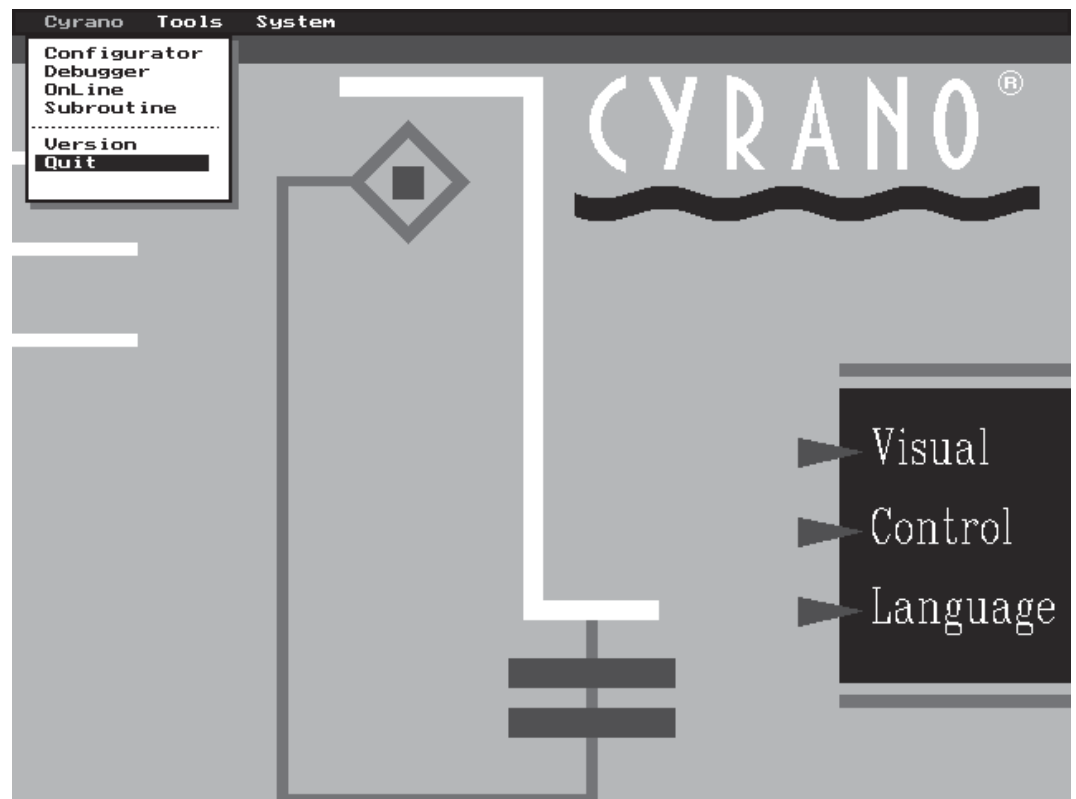


Figure 2-22: Exiting the Environment Manager

WHAT DID I DO?

In this chapter you got your first taste of hands-on work with Cyrano programs. After starting Cyrano and launching the Configurator from the Environment Manager, you created a new program, saved it, and saved backup copies as well. Along the way you became familiar with how to get around in Cyrano. Finally, you exited the Configurator and quit out of Cyrano.

WHERE DO I GO FROM HERE?

You are now ready to use the Cyrano Configurator to configure the hardware in your demo box. In so doing you will be configuring the Mystic controller as well as a variety of input/output (I/O) units and points.

Proceed to Chapter 3.

CONFIGURING THE HARDWARE



WHAT TOPICS WILL I COVER?

- Configuring a Mystic controller
- Configuring digital and analog I/O units (bricks)
- Configuring digital points with various features
- Configuring analog points using engineering units

WHAT WILL I DO?

- Configure the demo box hardware to be used in the Cyrano program. This includes the Mystic controller, two digital I/O units, an analog I/O unit, and several digital and analog I/O points.

HOW WILL I DO IT?

When writing a Cyrano program, the first step is to identify the type of hardware and communication parameters you will be using. This process is called configuration.

OPEN THE CONFIGURATOR MODULE

If you have exited the Configurator module, reopen it (see Chapter 2 for details). Verify that the program name listed in the status bar at the bottom of the window is TUTOR. If a different program name appears, select Load from the File menu and locate the TUTOR program. Note that if you created a backup program before exiting the Configurator, Cyrano will automatically load the backup, since it is the last program edited in the Configurator.

CONFIGURE THE CONTROLLER

From the Configurator's Configure menu, select Mystic.

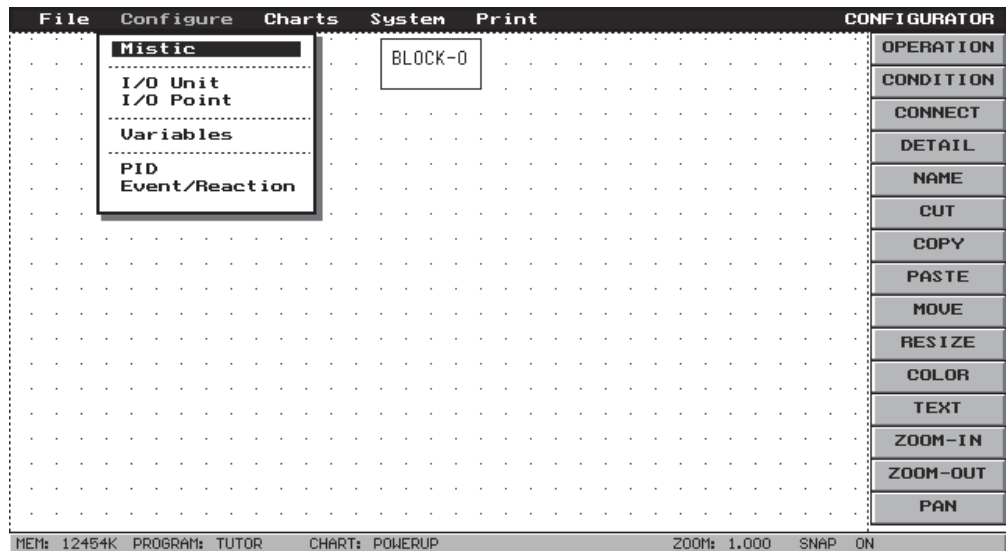


Figure 3-1: Configuring the Controller

The CONFIGURE MISTIC COMMUNICATIONS dialog box will open. In this dialog box, you will enter the information defining the controller where the downloaded Cyrano program will run.

In the NAME field, enter a descriptive name for the controller. For our example, use DEMO BOX CONTROLLER. Press ENTER and the next field, TYPE, will be highlighted.

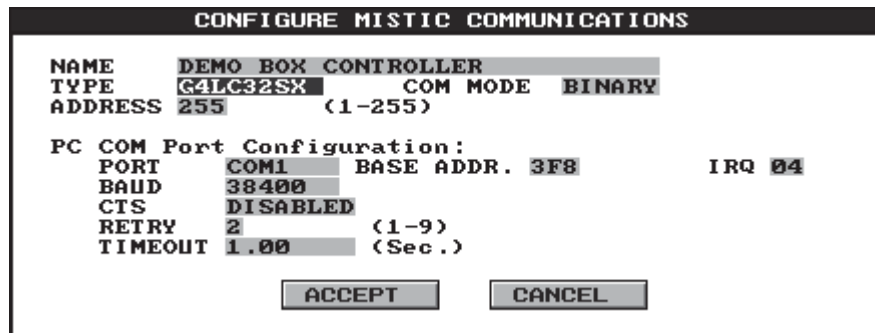


Figure 3-2: Selecting the Controller Type

In the TYPE field, use the left or right arrow key (or move the mouse) to view the types of controllers. Select G4LC32SX as the controller type and press ENTER.



A G4LC32SX COM Port Configuration dialog box will then appear. You can now enter communication information for three of the four COM ports of the G4LC32SX Mystic controller. The fourth COM port is set by a thumb wheel on the G4LC32SX controller.

	BAUD RATES	CTS CONTROL
COM1	19200	DISABLED
COM2	115200	DISABLED
COM3	38400	DISABLED

ACCEPT CANCEL

Figure 3-3: Configuring the COM Ports

Use the left/right arrow keys (or move the mouse) to view the available baud rates. Select the appropriate baud rate for the device that will connect to each COM port. For our example, configure COM1 to attach to a barcode reader, which has a baud rate of 19200, and press ENTER. COM2 is used for communication with the I/O bricks in the demo box. Configure the baud rate on COM2 to 115200 and press ENTER. COM3 is wired to the display panel of the demo box that communicates with a baud rate of 38400. Select this baud rate for COM3 and press ENTER.

Disable CTS control for all three ports. CTS control is used for handshaking with RS-232 communications.

After configuring the third COM port, press ENTER and the ACCEPT button will become highlighted. With ACCEPT highlighted, press ENTER (or the left mouse button) to save your communication settings. Note that if you press ENTER while CANCEL is highlighted (or if you press the ESC key), you will close the dialog box **without** making changes to the communication settings.

CONFIGURE MISTIC COMMUNICATIONS

NAME DEMO BOX CONTROLLER
 TYPE G4LC32SX COM MODE BINARY
 ADDRESS 001 (1-200)

PC COM Port Configuration:
 PORT ARCNET BASE ADDR. 260 RAM ADDR. D4000
 BAUD 38400
 CTS DISABLED
 RETRY 2 (1-9)
 TIMEOUT 1.00 (Sec.)

ACCEPT CANCEL

Figure 3-4: Configuring the Communications Port

After you accept the information, the G4LC32SX COM Port Configuration dialog box will close. The COM MODE field in the CONFIGURE MISTIC COMMUNICATIONS dialog box will then be highlighted. Choose the COM mode appropriate to your communication. There are two choices, BINARY and ASCII. Binary mode is a special 11-bit communication protocol that is **not** supported by most standard PC-type modems. If you are communicating via a modem, you should use the ASCII protocol. Otherwise, choose the binary mode for faster communication. The field has no effect when using ARCNET for host PC communication.

Configure the G4LC32SX controller at address 001 and verify that the address is set to 001 on the front of the controller in the demo box. For purposes of this tutorial, ARCNET is used to communicate with the PC at base address 260 and RAM address D4000. Fill in the appropriate fields to reflect these communication parameters.

Note: The base address and RAM address of your ARCNET board may differ.

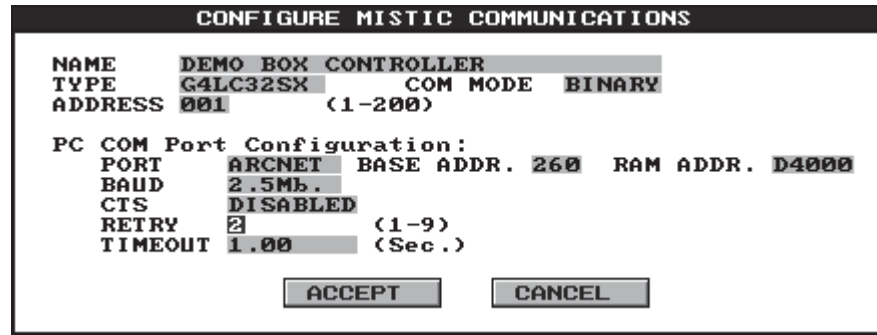


Figure 3-5: Entering the Communications Retry Number

After you have selected ARCNET and set the address for the ARCNET board, the baud rate will automatically be set to 2.5Mb and CTS (clear to send) will be disabled. (CTS enables or disables handshaking with RS-232 communication.) Notice that you **cannot** access the baud rate and CTS fields, as the cursor will skip to the RETRY field. Some field selections in Cyrano restrict access to other fields. This occurs if the information in those fields is not required or if it is predetermined by previous parameter settings. In this case, ARCNET communicates at 2.5 Mb and does not need CTS enabled, so this information is preset once ARCNET has been selected as the port.

Leave the number of retries set to the default value of 2. This number specifies how many additional times Cyrano will attempt to communicate with the control processor before it generates a timeout error.

The timeout value indicates how long Cyrano will wait for a response from the processor before it will try to communicate again. In the TIMEOUT field, enter 1.0 for a timeout of one second.

Close the CONFIGURE MISTIC COMMUNICATIONS dialog box by selecting ACCEPT.



CONFIGURE I/O UNITS

Digital I/O Units

Each I/O unit (brick) must be configured in the Cyrano program. There are two methods for configuring I/O units: all-at-once and on-the-fly. All-at-once configuration defines the units before they are actually used in the Cyrano program. The on-the-fly method configures units as they are referenced in the program.

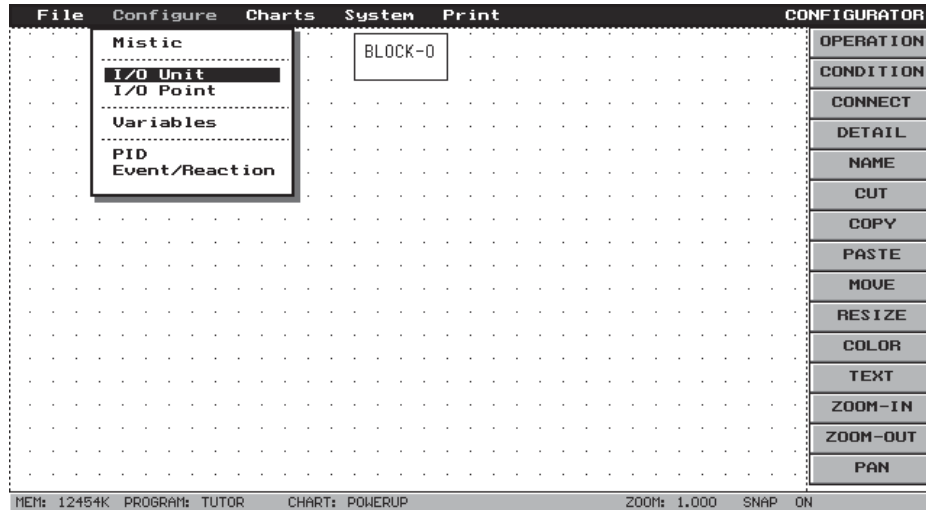


Figure 3-6: Configuring an I/O Unit

Select I/O Unit from the Configure menu. Since no I/O units have been defined, a dialog box displays the message "No I/O Unit(s) Defined." Press any key to close this message dialog box.



Figure 3-7: No I/O Unit(s) Defined Message Box

From the CURRENT I/O UNITS dialog box, highlight the ADD button and press ENTER. The DELETE button is used to remove configured I/O units and the CHANGE button is used to modify the configuration of a previously configured unit.

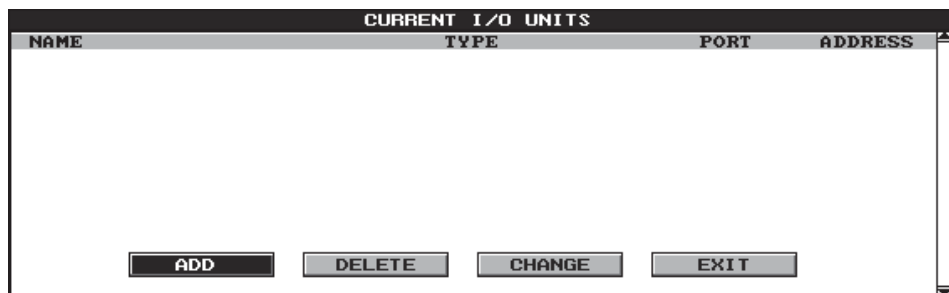


Figure 3-8: Adding an I/O Unit

The ADD I/O UNIT dialog box prompts you for configuration information regarding the brick you wish to add. Type LEFT_DIGITAL_BRICK as the name of the I/O unit. (If you type LEFT DIGITAL BRICK, Cyrano will automatically replace the spaces with underscores when you press ENTER.) Names are limited to 32 characters. Use as many characters as necessary when naming items in Cyrano. Descriptive names make debugging and program maintenance easier; however, they also require more memory and thus can slow communication to HOST port devices.

The screenshot shows a dialog box titled "ADD I/O UNIT" with the following configuration:

NAME	LEFT_DIGITAL_BRICK
TYPE	DIGITAL MF
PORT	REMOTE 2
ADDRESS	000 (0-255)
WATCHDOG	DISABLED
SECURITY	0
ENABLE	YES

At the bottom of the dialog box are two buttons: "ACCEPT" and "CANCEL".

Figure 3-9: Configuring a Digital I/O Unit

The left digital brick in the demo box is a digital MF (multifunction) brick located at address 000. Check the jumpers at the bottom of the I/O units in the demo box to verify that the address of this brick is 000. (Address 000 has no jumpers.)

All three I/O units in the demo box are connected to the controller via an RS-485 serial port (using twisted-pair wiring) with a baud rate of 115K. Bricks connected via twisted-pair wiring are attached to remote ports, where REMOTE 0 corresponds to COM0, REMOTE 1 corresponds to COM1, REMOTE 2 corresponds to COM2, and REMOTE 3 corresponds to COM3. Recall that the controller's COM2 port was configured with a baud rate of 115K for communication with the I/O units. Use the left or right arrow keys to select REMOTE 2 as the communication port.

WATCHDOG is an option that sets the I/O points on a brick to a given state (defined with the I/O point) if there is a communication line failure between the brick and the processor. For this tutorial, leave the watchdog disabled.

Security levels can be set from 0 to 3, where 0 is the lowest level of security (*no* password required). Security levels 1 to 3 require a password to make changes. A level 3 password is required to add or change passwords. Since there is no security risk for our tutorial, select the lowest level of security, 0.



The ENABLE field allows you to enable or disable communication to an I/O unit. Selecting NO disables communication to the brick and all its modules. You may want to disable an I/O unit to simulate an input or output value in the DEBUGGER. For our example, leave the I/O unit enabled by selecting the default value of YES.

After you complete the information in the ADD I/O UNIT dialog box, highlight ACCEPT and press ENTER.

```

ADD I/O UNIT
NAME      CENTER_DIGITAL_BRICK
TYPE      DIGITAL MF
PORT      REMOTE 2
ADDRESS   001      (0-255)
WATCHDOG  DISABLED
SECURITY  0
ENABLE    YES

ACCEPT    CANCEL
  
```

Figure 3-10: Accepting a Digital I/O Unit Configuration

Configure the second digital I/O unit with the name CENTER_DIGITAL_BRICK using the same configuration selections as those for the first digital I/O unit. When choosing the address for this unit, note that the address defaults to the lowest *available* address, 001 (recall that LEFT_DIGITAL_BRICK was configured at address 000). You may change the address by typing in another value. Check the jumpers on the center I/O unit in the demo box to verify that the brick is located at address 001.

ANALOG I/O UNITS

```

ADD I/O UNIT
NAME      ANALOG_BRICK
TYPE      ANALOG MF      SINGLE POINT
PORT      REMOTE 2
ADDRESS   002      (0-255)
WATCHDOG  DISABLED
SECURITY  0
ENABLE    YES
TEMP CNU  Degrees F

ACCEPT    CANCEL
  
```

Figure 3-11: Configuring an Analog I/O Unit

The third I/O unit, located on the right side of the demo box, is a multifunction analog brick. After selecting ANALOG MF in the TYPE field, you will be prompted for a secondary type. The various HRD options available as a secondary type are used to configure high-resolution, high-density analog units. The brick in the demo box is a single-point analog unit. This brick communicates via REMOTE 2 and resides at address 002. The TEMP CNV field allows you to choose temperature units of Celsius or Fahrenheit.

Complete the information as illustrated in Figure 3-11 and close the dialog box by selecting ACCEPT.

After you configure the three bricks, they will be listed in the CURRENT I/O UNITS dialog box. Confirm that all three bricks are listed, then close this dialog box by pressing the ESC key or EXIT at the bottom of the dialog box.

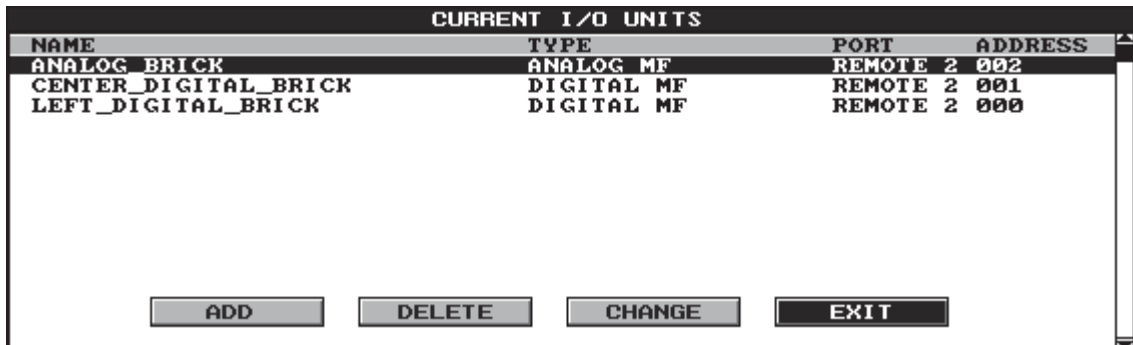


Figure 3-12: Current I/O Units Dialog Box Showing Three Configured Bricks

CONFIGURE I/O POINTS

Every I/O point used in a Cyrano program must be configured. I/O points can be configured through all-at-once or on-the-fly configuration methods. In this exercise, we will configure most of the I/O points we will need all at once. Some I/O points will not be configured initially so that we can configure them later using the on-the-fly method.

Select I/O Point from the Configure menu, as shown in Figure 3-13. The SELECT I/O UNIT dialog box will open, listing the three bricks previously configured, as shown in Figure 3-14.

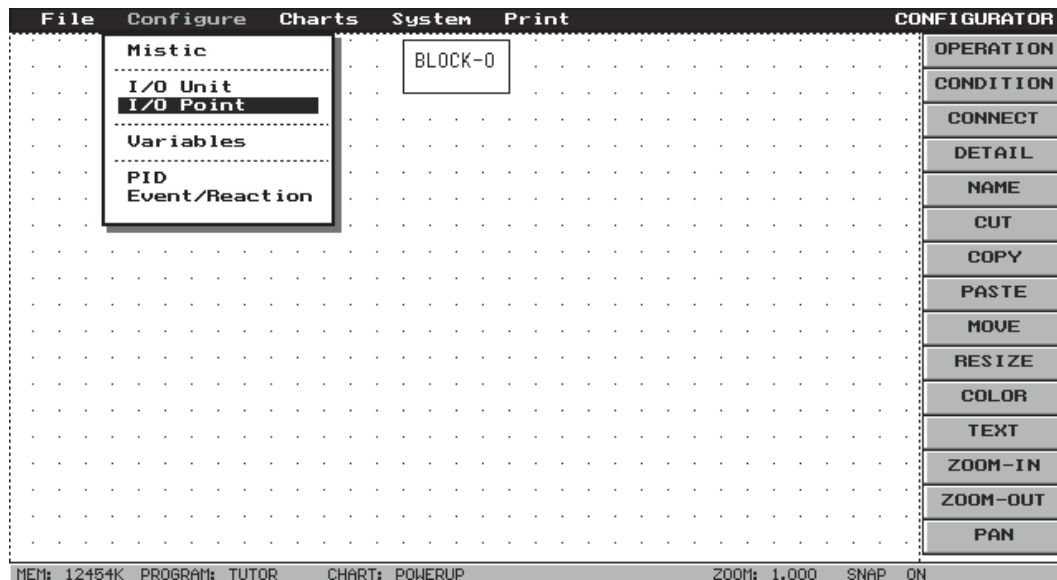


Figure 3-13: Configuring an I/O Point



DIGITAL POINTS

SELECT I/O UNIT			
NAME	TYPE	PORT	ADDRESS
ANALOG_BRICK	ANALOG MF	REMOTE 2	002
CENTER_DIGITAL_BRICK	DIGITAL MF	REMOTE 2	001
LEFT_DIGITAL_BRICK	DIGITAL MF	REMOTE 2	000

Figure 3-14: Selecting an I/O Unit

From the SELECT I/O UNIT dialog box, select LEFT_DIGITAL_BRICK. The CONFIGURE DIGITAL I/O POINT dialog box will then open, displaying the 16 digital channels located on that brick. Highlight the module located at channel 00 and press ENTER while ADD is highlighted.

Other buttons in this dialog box include DELETE, used to remove configured I/O points, and CHANGE, used to change the configuration of a previously configured point. The DUP command will be used later in this tutorial to copy the configuration of one point to another. MOVE can be used to move a configured I/O point to another channel on the same brick or to a channel on another brick.

CONFIGURE DIGITAL I/O POINT				
UNIT	LEFT_DIGITAL_BRICK	ADDRESS	000	
CH#	NAME	TYPE	FEATURES	USED
00	UNUSED			
01	UNUSED			
02	UNUSED			
03	UNUSED			
04	UNUSED			
05	UNUSED			
06	UNUSED			
07	UNUSED			
08	UNUSED			
09	UNUSED			
10	UNUSED			
11	UNUSED			
12	UNUSED			
13	UNUSED			
14	UNUSED			
15	UNUSED			

Figure 3-15: Configuring a Digital I/O Point

The module located at channel 00 of the left digital brick is a G4IDC5 wired to BUTTON_0 on the demo box. Name this I/O point BUTTON_0 and configure it as a digital input point by selecting the type DIN.

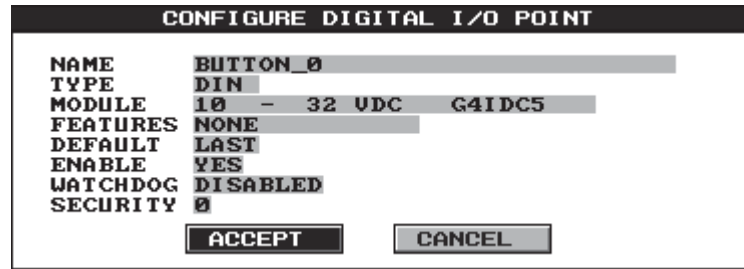


Figure 3-16: Configuring a Digital Input Point

Use the left or right arrow key to find the correct module type, then press ENTER repeatedly to accept the default values for the remaining fields. FEATURES allow you to utilize the flexibility of the I/O modules by configuring them as latches, counters, etc. DEFAULT defines the value the point has when the Cyrano program is first executed. This value can be set to ON, OFF, or LAST.

The remaining fields — ENABLE, WATCHDOG, and SECURITY — at the point level are very similar to the brick options. Refer to pages 3-6 and 3-7 for details.

After accepting this configuration, you will be returned to the CONFIGURE DIGITAL I/O POINT dialog box.

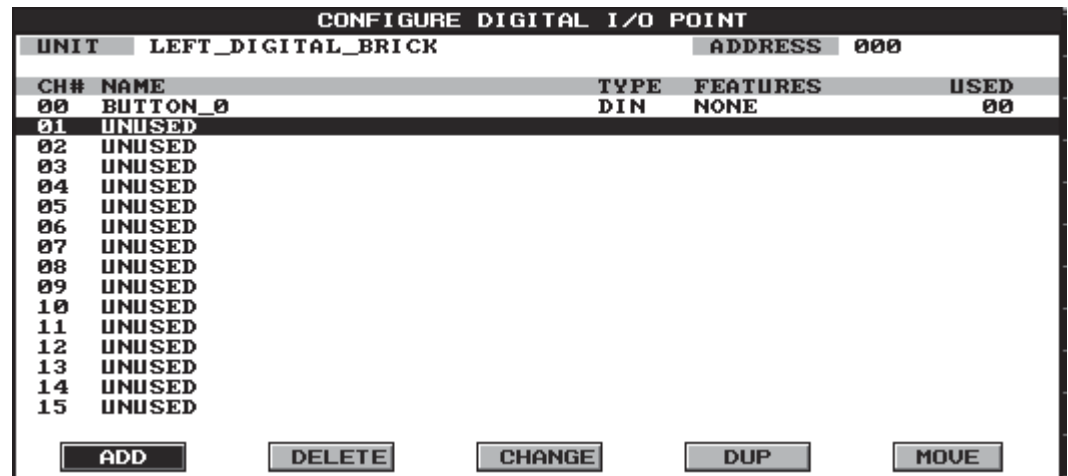


Figure 3-17: Adding an I/O Point



Highlight channel 01 and select ADD to configure BUTTON_1. Configure this module as an ON-latch by selecting ON LATCH in the FEATURES field. A latch is like a switch. As soon as the button is pushed, the latch is triggered. Pushing the button again has no effect until the latch is cleared (or the switch is reset).

CONFIGURE DIGITAL I/O POINT	
NAME	BUTTON_1
TYPE	DIN
MODULE	10 - 32 UDC G4IDC5
FEATURES	ON LATCH
DEFAULT	LAST
ENABLE	YES
WATCHDOG	DISABLED
SECURITY	0
<input type="button" value="ACCEPT"/> <input type="button" value="CANCEL"/>	

Figure 3-18: Configuring a Digital Input Point

After accepting the configuration of BUTTON_1, you will return to the main CONFIGURE DIGITAL I/O POINT dialog box, as shown in Figure 3-19.

CONFIGURE DIGITAL I/O POINT				
UNIT	LEFT_DIGITAL_BRICK		ADDRESS 000	
CH#	NAME	TYPE	FEATURES	USED
00	BUTTON_0	DIN	NONE	00
01	BUTTON_1	DIN	ON LATCH	00
02	UNUSED			
03	UNUSED			
04	UNUSED			
05	UNUSED			
06	UNUSED			
07	UNUSED			
08	UNUSED			
09	UNUSED			
10	UNUSED			
11	UNUSED			
12	UNUSED			
13	UNUSED			
14	UNUSED			
15	UNUSED			
<input type="button" value="ADD"/> <input type="button" value="DELETE"/> <input type="button" value="CHANGE"/> <input type="button" value="DUP"/> <input type="button" value="MOVE"/>				

Figure 3-19: Copying an I/O Point

Once again highlight channel 01, but this time use the right arrow key to select the DUP button. Press ENTER and a second highlight bar will appear in the dialog box.

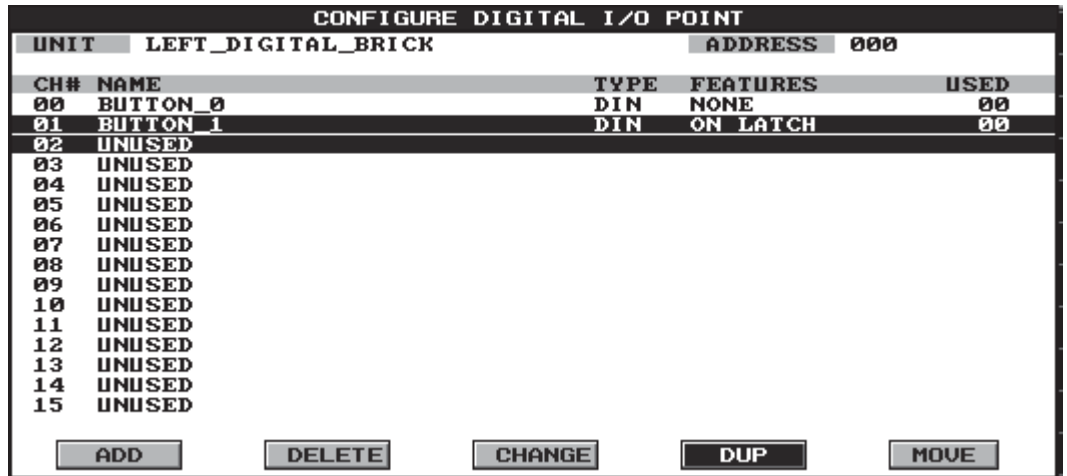


Figure 3-20: Pasting an I/O Point

Use the up/down arrow keys to move the second highlighted row through the list of unconfigured channels. Highlight channel 02 and press ENTER. The dialog box in Figure 3-21 will appear.

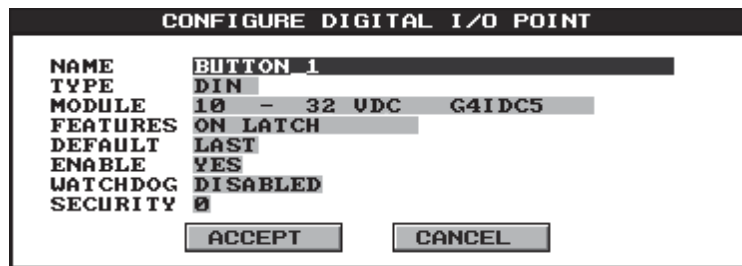


Figure 3-21: Editing the Copied I/O Point

Notice that Figure 3-21 shows configuration information for channel 02 that is identical to that for channel 01 (see Figure 3-16). Press the END key while you are in the NAME field and the number 1 will be highlighted. Type 2 over the number 1 and press ENTER several times until ACCEPT is highlighted. Press ENTER while ACCEPT is highlighted to accept the configuration information in Figure 3-22.

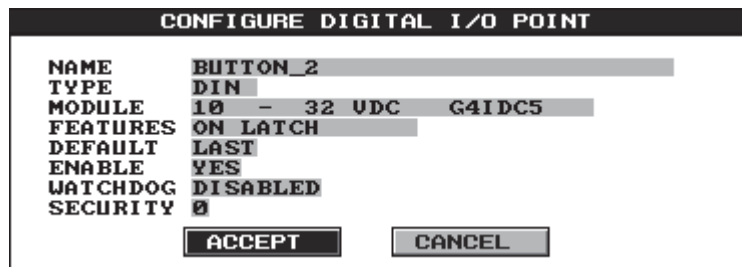


Figure 3-22: Accepting an Analog I/O Unit Configuration



Using the DUP command, you have copied the configuration of one I/O point to another. Three digital input points will now be listed in the CONFIGURE DIGITAL I/O POINT dialog box, as shown in Figure 3-23.

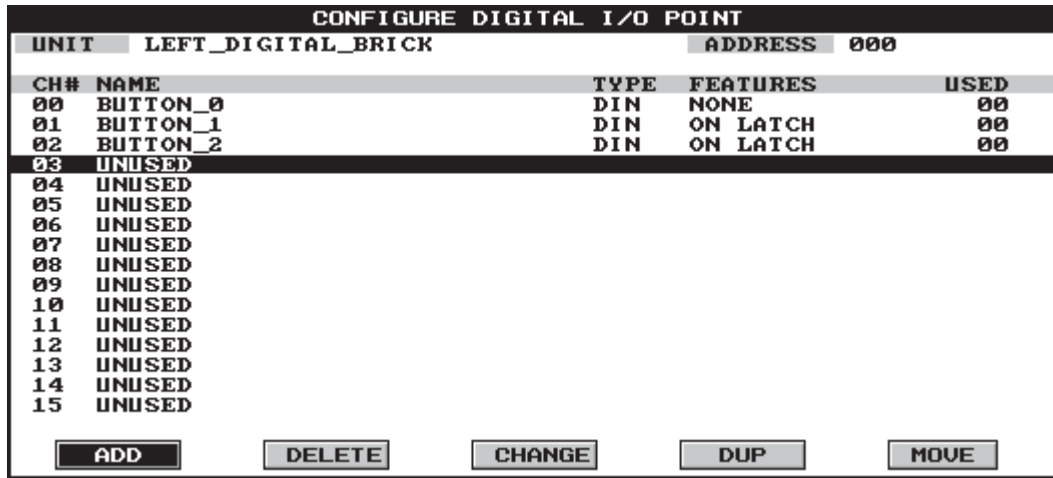


Figure 3-23: Adding an I/O Point

The digital modules at channels 08 through 15 are output modules controlling LEDs 0 through 7 on the front panel of the demo box. Using the ADD and DUP commands, configure two digital output points at channels 08 and 09. Name these points LED_0 and LED_1. Configure these points as digital output points by selecting DOUT as the TYPE and selecting G40DC5 in the MODULE field. Choose the default values for the remaining fields. See Figure 3-24.

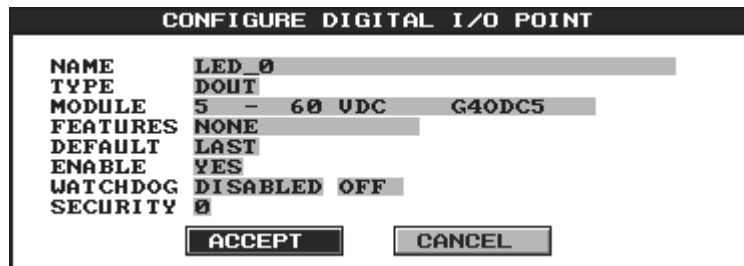


Figure 3-24: Configuring a Digital Output Point

After configuring the digital output points, a total of three digital input and two digital output points should be listed in the CONFIGURE DIGITAL I/O POINT dialog box for LEFT_DIGITAL_BRICK, as shown in Figure 3-25. After confirming this list, press ESC to close this dialog box.

CONFIGURE DIGITAL I/O POINT				
UNIT	LEFT_DIGITAL_BRICK		ADDRESS	000
CH#	NAME	TYPE	FEATURES	USED
00	BUTTON_0	DIN	NONE	00
01	BUTTON_1	DIN	ON LATCH	00
02	BUTTON_2	DIN	ON LATCH	00
03	UNUSED			
04	UNUSED			
05	UNUSED			
06	UNUSED			
07	UNUSED			
08	LED_0	DOUT	NONE	00
09	LED_1	DOUT	NONE	00
10	UNUSED			
11	UNUSED			
12	UNUSED			
13	UNUSED			
14	UNUSED			
15	UNUSED			

ADD DELETE CHANGE DUP MOVE

Figure 3-25: Viewing the Configured Points

ANALOG POINTS

The SELECT I/O UNIT dialog box should be open, as shown in Figure 3-26. If you closed this dialog box, open it again by selecting I/O Point from the Configurator's Configure menu.

SELECT I/O UNIT			
NAME	TYPE	PORT	ADDRESS
ANALOG_BRICK	ANALOG MF	REMOTE 2	002
CENTER_DIGITAL_BRICK	DIGITAL MF	REMOTE 2	001
LEFT_DIGITAL_BRICK	DIGITAL MF	REMOTE 2	000

SELECT EXIT

Figure 3-26: Selecting an I/O Unit



We will now configure several points on the analog brick. Highlight the analog brick and click SELECT to open the CONFIGURE ANALOG I/O POINT dialog box, which is similar to the digital configuration dialog box. Highlight the first point at channel 00 and select ADD.

CH#	NAME	TYPE	UNITS	USED
00	UNUSED			
01	UNUSED			
02	UNUSED			
03	UNUSED			
04	UNUSED			
05	UNUSED			
06	UNUSED			
07	UNUSED			
08	UNUSED			
09	UNUSED			
10	UNUSED			
11	UNUSED			
12	UNUSED			
13	UNUSED			
14	UNUSED			
15	UNUSED			

Figure 3-27: Adding an Analog I/O Point

A second configuration dialog box will open to allow you to enter configuration details. Name this analog point TEMPERATURE_SENSOR_#1.

NAME	TEMPERATURE_SENSOR_#1
TYPE	AIN
MODULE	ANALOG INPUT
UNITS	DEGREES F
LO SCALE	-50.0000
HI SCALE	150.0000
DEFAULT	NO
ENABLE	YES
WATCHDOG	DISABLED
SECURITY	0

Figure 3-28: Configuring an Analog Input Point

This module is a G4AD6 input module wired to the top potentiometer in the demo box. Select AIN (analog input) as the TYPE.

Note: Scaling Analog Ranges in Engineering Units

1. Select ANALOG INPUT in the module field.
2. Type in the engineering units in the Units field.
3. Set the LO SCALE.
4. Set the HI SCALE

The range for this module is 0–5 V. However, we are going to use this potentiometer to simulate a temperature sensor. Scale the range of values between -50°F and 150°F. It is easy to scale values in engineering units in Cyrano, as indicated on the following page.

Configure the I/O point as illustrated in Figure 3-28. The 0–5 V reading is remapped to a temperature range of -50°F to 150°F as shown in Table 3-1:

Table 3-1: Remapping Voltage to Temperature

Voltage Reading	Corresponding Temperature
0	-50°F
1	-10°F
2	30°F
3	70°F
4	110°F
5	150°F

Scaling to engineering units eliminates cumbersome conversions in your program. All values are scaled in units appropriate to their function, for example, GPM, percent, degrees F, grams, or psi.

Select DUP to add a second analog point to this brick at channel 02. Configure this point the same as the point at channel 00 and name it TEMPERATURE_SENSOR_#2.

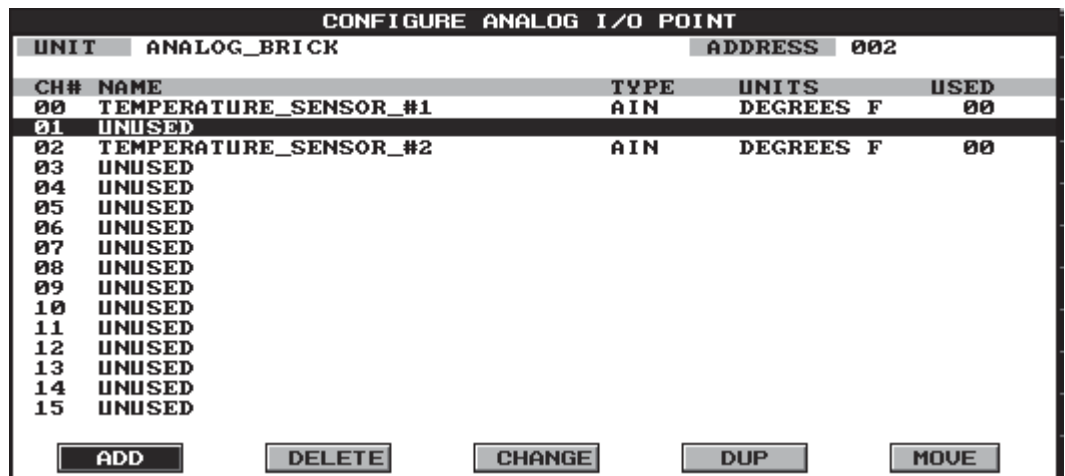


Figure 3-29: Adding an I/O Point



Add a third point to the analog brick at channel 01. This module is a G4DA5 output module for the top meter. Name this point TOP_METER. Since we will use this meter to display the value of the top pot (the simulated temperature sensor), set the range the same as the top pot and accept the configuration.

```

CONFIGURE ANALOG I/O POINT
NAME      TOP_METER
TYPE      AOUT
MODULE    ANALOG OUTPUT
UNITS     DEGREES F
LO SCALE  -50.0000
HI SCALE  150.0000
DEFAULT   NO  0.0000
ENABLE    YES
WATCHDOG  DISABLED 0.0000
SECURITY  0
ACCEPT
CANCEL
  
```

Figure 3-30: Configuring an Analog Output Point

Select DUP to configure the module at channel 03 the same as channel 01. Since this module is wired to the bottom meter on the demo box, name this point BOTTOM_METER.

```

CONFIGURE ANALOG I/O POINT
NAME      BOTTOM_METER
TYPE      AOUT
MODULE    ANALOG OUTPUT
UNITS     DEGREES F
LO SCALE  -50.0000
HI SCALE  150.0000
DEFAULT   NO  0.0000
ENABLE    YES
WATCHDOG  DISABLED 0.0000
SECURITY  0
ACCEPT
CANCEL
  
```

Figure 3-31: Configuring an Analog Output Point

The first four points on the analog brick should now be configured. Verify the configuration in the CONFIGURE ANALOG I/O POINT dialog box, as shown in Figure 3-32.

CONFIGURE ANALOG I/O POINT				
UNIT	ANALOG_BRICK	ADDRESS 002		
CH#	NAME	TYPE	UNITS	USED
00	TEMPERATURE_SENSOR_#1	AIN	DEGREES F	00
01	TOP_METER	AOUT	DEGREES F	00
02	TEMPERATURE_SENSOR_#2	AIN	DEGREES F	00
03	BOTTOM_METER	AOUT	DEGREES F	00
04	UNUSED			
05	UNUSED			
06	UNUSED			
07	UNUSED			
08	UNUSED			
09	UNUSED			
10	UNUSED			
11	UNUSED			
12	UNUSED			
13	UNUSED			
14	UNUSED			
15	UNUSED			

ADD DELETE CHANGE DUP MOVE

Figure 3-32: Viewing the Configured Analog I/O Points

WHAT DID I DO?

In this chapter you configured all the hardware used in this tutorial. You first configured the Mystic controller by selecting the controller type and configuring the communication ports. Next you configured the three I/O units in the demo box: the left digital brick, the center digital brick, and the right analog multifunction brick. Finally, you configured most of the digital and analog input and output points you will need by using the all-at-once configuration method. Later you will configure additional I/O points using the on-the-fly method.

WHERE DO I GO FROM HERE?

With your hardware configured, you are ready to create your first chart. This simple chart will respond to presses of buttons on your demo box by lighting corresponding LEDs. As you create the chart, you will become familiar with drawing and connecting command blocks, entering commands, creating variables, starting charts from the POWERUP chart, and downloading programs.

Proceed to Chapter 4.

CREATING A NEW CHART



WHAT TOPICS WILL I COVER?

- Creating a new chart
- Using drawing tools
- Entering commands
- Defining variables
- Starting a chart from the POWERUP chart
- Downloading and running the program

WHAT WILL I DO?

- Create a chart called BUTTON_CONTROL.
- Draw a flowchart that includes an operation block named LED_0_CONTROL.
- Enter a command in this operation block to copy the state of the digital input point BUTTON_0 to the digital output point LED_0. This will cause button 0 to control the state of LED_0. Hence, when button 0 is pressed (on), LED_0 will be lit (on). When BUTTON_0 is off, LED_0 will be off.
- Add a command to the POWERUP chart to run the new BUTTON_CONTROL chart.
- Download the program using the Debugger module and run it.

HOW WILL I DO IT?

CREATE A NEW CHART

If you have exited the Configurator module, reopen it . Verify that the program name listed in the status bar at the bottom of the window is TUTOR.

To create a new chart in your program, select New from the Charts menu.

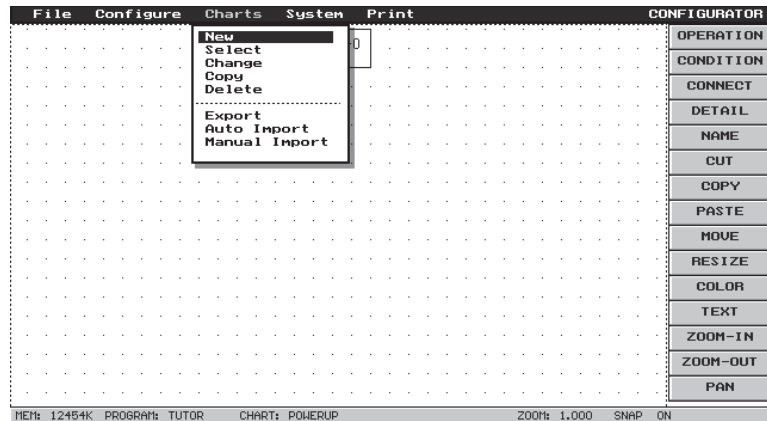


Figure 4-1: Creating a New Chart

The Enter New Chart Name dialog box will appear to prompt you for the name of the new chart. Type in the name `BUTTON_CONTROL`. Notice that if you type `BUTTON CONTROL`, Cyrano will automatically fill in the space with an underscore. Cyrano always replaces spaces with underscores in the names of objects and variables.



Figure 4-2: Naming a New Chart

DRAW THE CHART STRATEGY

Select the `OPERATION` tool from the top of the toolbar. To select the tool, point to it with the mouse cursor and press the left mouse button.

A rectangular object will appear in the drawing window. Place the operation block below BLOCK-0 in the drawing window by moving the mouse and pressing the left mouse button when the block is in the desired location. The block can be resized while it is being drawn by pressing and holding the left mouse button as you move the mouse. Release the left mouse button when the block is the desired size. (A block can also be resized after it is drawn by using the RESIZE tool.)

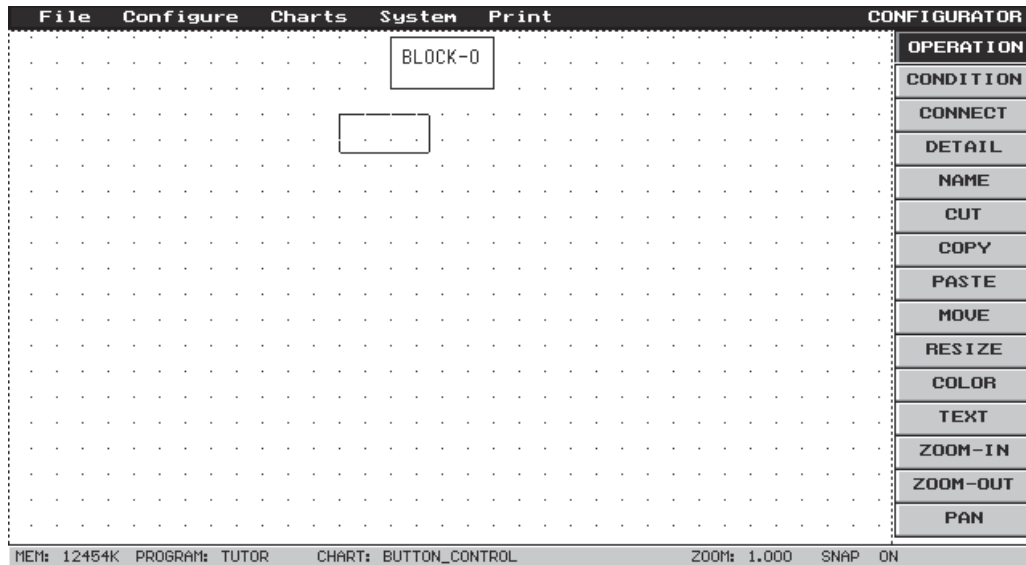


Figure 4-3: Drawing an Operation Block

Release the OPERATION tool by pressing the right mouse button or the ESC key. Note that in Cyrano, you cannot select a new tool until you have released the previous tool.

You have created and placed the BLOCK-1 operation block. When blocks are created in Cyrano, they are given default names of BLOCK-#, where # increments with each new block that you create.

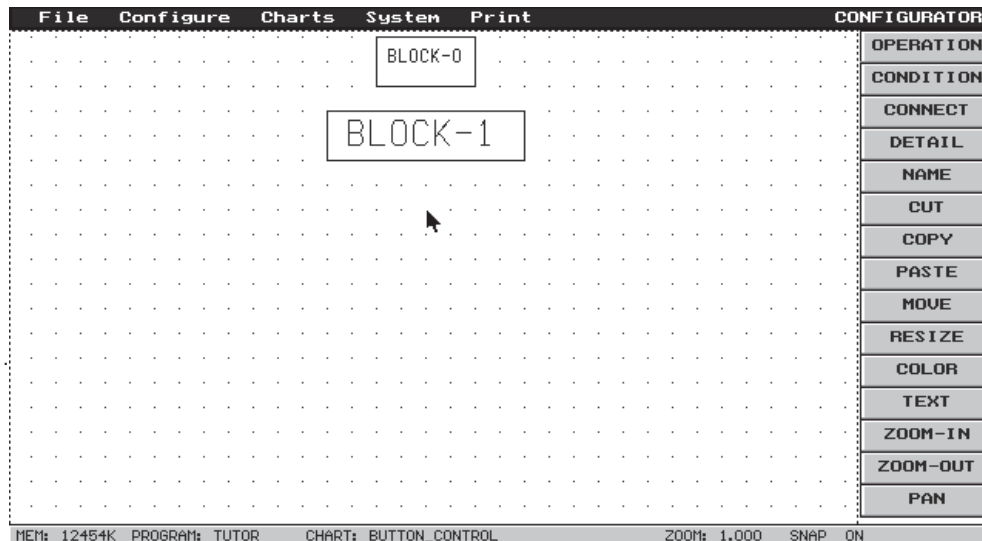


Figure 4-4: Viewing the Completed Operation Block

Draw a connection line from BLOCK-0 to BLOCK-1 as follows. Select the CONNECT tool from the toolbar and move the mouse cursor over the center of BLOCK-0. This will cause the block to start blinking. Click the left mouse button after the block starts blinking, then move the mouse cursor to the center of BLOCK-1 and click the left mouse button again. This will connect BLOCK-0 and BLOCK-1.

Draw a second connection line that exits BLOCK-1 and re-enters the same block. To bend a connection line at a right angle, press the left mouse button where you want the line to bend. By connecting BLOCK-1 to itself, you will cause the commands in BLOCK-1 to be executed repeatedly.

Release the CONNECT tool by pressing the right mouse button.

Keep in mind that when drawing connection lines, you should select blocks in the order of the logic flow.

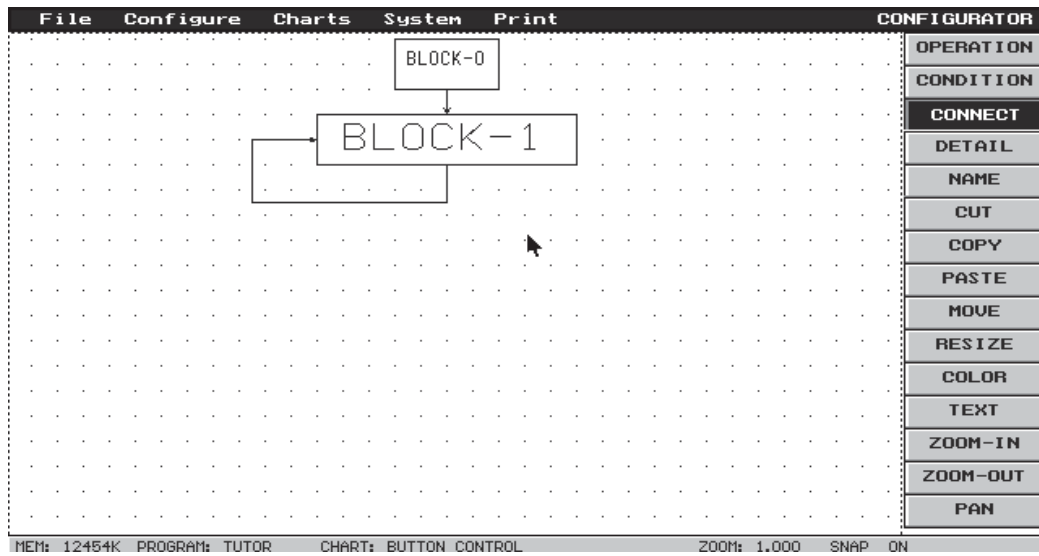


Figure 4-5: Drawing Connection Lines

You can now use the NAME tool to rename BLOCK-1. Select the NAME tool, move the mouse cursor over BLOCK-1, and click the left mouse button.

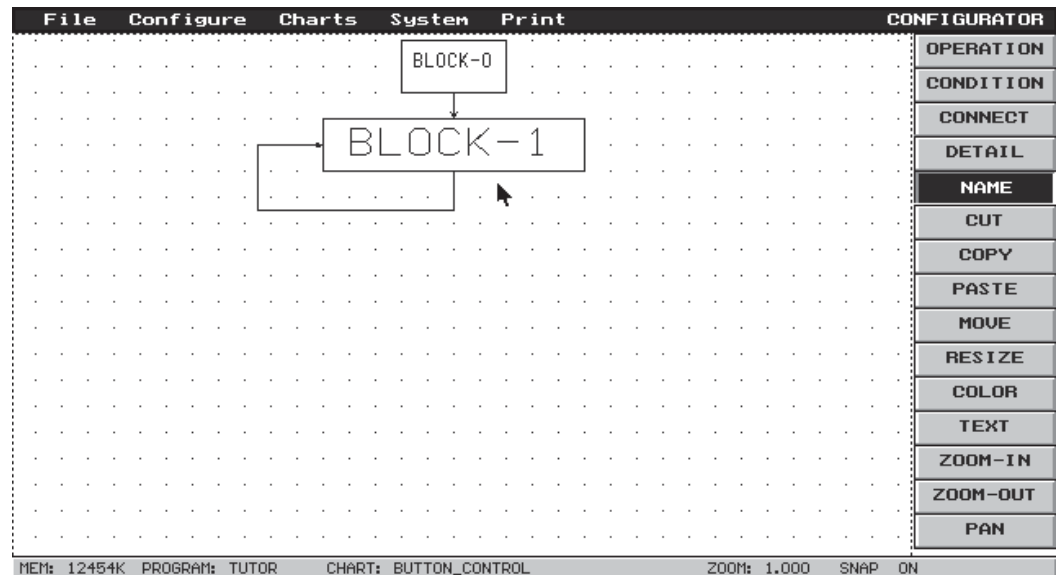


Figure 4-6: Naming an Operation Block

This will cause the Enter New Block Name dialog box to appear. Type the name LED_0_CONTROL right over the original name (BLOCK-1). Press ENTER twice to accept the new name.

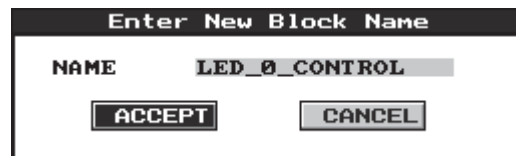


Figure 4-7: Entering a Block Name

Note that block names must be unique; however, since block names in Cyrano are case-sensitive, two

names with the same text but different case usage may coexist. Thus, for example, one block can be named DELAY and a second block in the same chart can be named Delay.

Now open the detail window for the LED_0_CONTROL block by pointing to the block and double-clicking the left mouse button. Notice that the DETAIL tool is highlighted. You can also open a detail window by selecting the DETAIL tool and clicking the left mouse button over the desired block.



Figure 4-8: Adding a New Command

The command buttons at the bottom of the detail window allow you to add, change, cut, copy, or paste Cyrano commands. Highlight ADD and press ENTER. This will open the ITEM EDITOR dialog box.

You will now enter a command in the ITEM EDITOR dialog box to copy the state of the input point BUTTON_0 to the output point LED_0. If BUTTON_0 is on, LED_0 will be on. If BUTTON_0 is off, LED_0 will be off.

A very powerful and useful command in Cyrano called MOVE allows you to **copy** the value of inputs, outputs, or numeric variables to other outputs or variables. MOVE automatically converts the value being copied into the right format. Thus, input values can be copied to outputs, inputs can be copied to floating point variables, floating point values can be copied to integer variables, etc.

There are several ways to enter commands in the ITEM EDITOR dialog box:

1. *Type the complete command.*

After you become familiar with Cyrano commands, you will generally know the complete command you wish to enter. The command can be typed directly into the OPERATION field. Note that if you type an unrecognized command, the GROUP SELECTIONS dialog box will appear. (See method 3 on the following page.)

Example: Type *MOVE* and press *ENTER*.

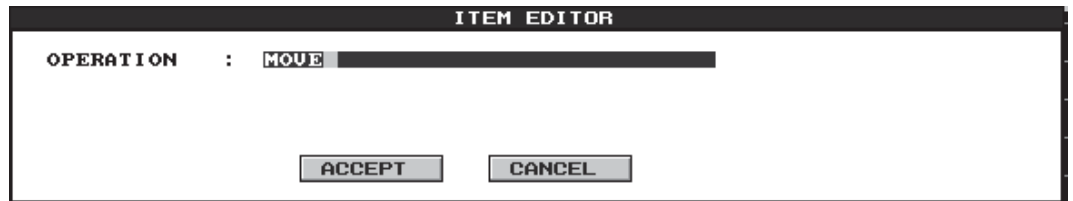


Figure 4-9: Entering a Full Operation Name

2. *Type the first few letters of the command.*

If you know the first few letters of a command, you can type those letters to get a list of all the commands beginning with those letters.

Example: Type *MO*, press *ENTER*.



Figure 4-10: Entering a Partial Operation Name

An OPERATIONS dialog box will open listing all the commands that begin with MO. Use the up or down arrow keys to highlight the desired command, then press ENTER.

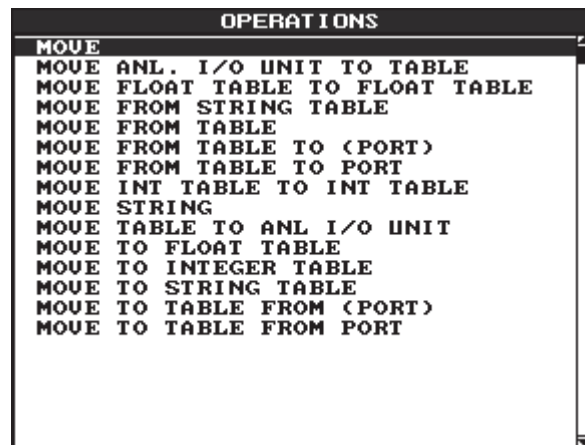


Figure 4-11: Selecting an Operation

MOVE will now appear in the ITEM EDITOR dialog box, as shown in Figure 4-9.

3. Press ENTER without typing any characters.

If you have no idea what command name to type, press ENTER before typing any characters.

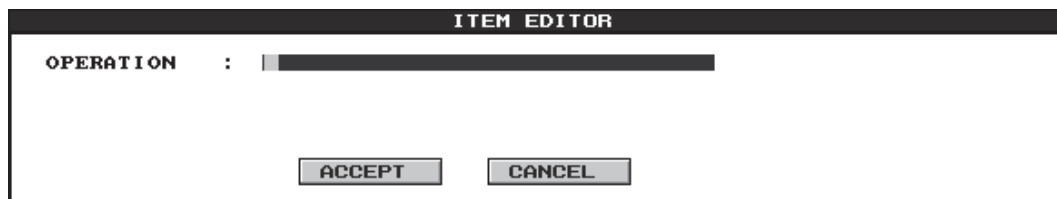


Figure 4-12: Accessing a List of Operation Names

This will open the GROUP SELECTIONS dialog box, which lists all Cyrano command groups. Use the up or down arrow keys to highlight the command group most closely related to the command you wish to use, then press ENTER.

Example: Choose the command group GENERAL PURPOSE.

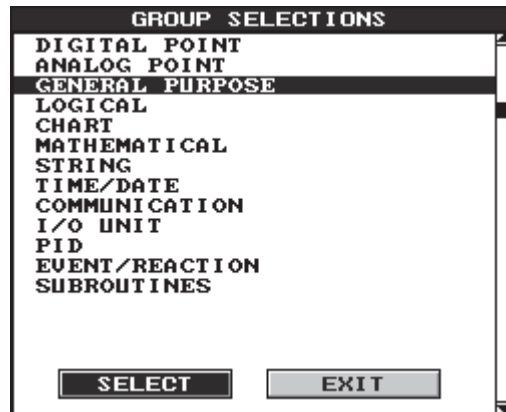


Figure 4-13: Selecting a Group

A list of all Cyrano commands in the selected group will now appear. Use the up or down arrow keys to highlight the MOVE command, then press ENTER.

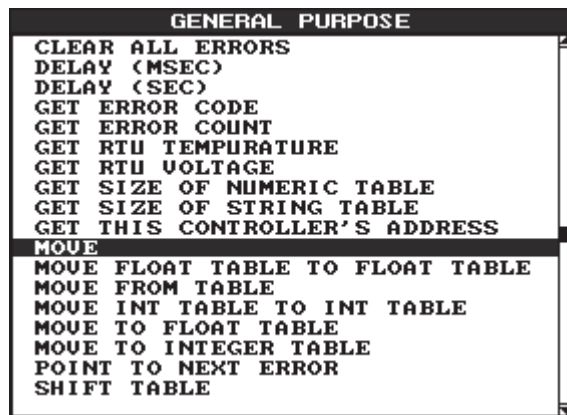


Figure 4-14: Selecting an Operation

MOVE will now appear in the ITEM EDITOR dialog box, as shown in Figure 4-9.

Once you have selected MOVE as the operation in the ITEM EDITOR dialog box, a new field will appear next to the label From, prompting you for the type of item being copied. Use the left or right arrow keys to browse through the list of valid types.

Since a button is a digital input point, select DIGITAL IN from the list and press ENTER.

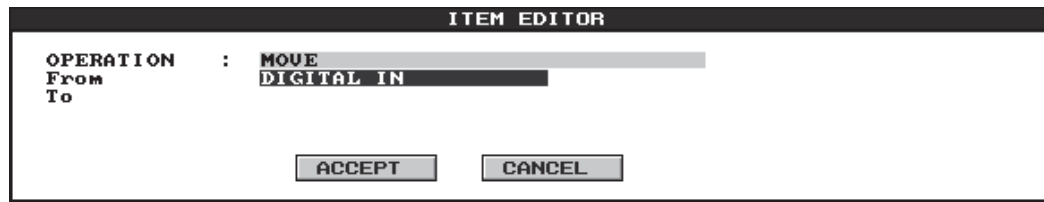


Figure 4-15: Selecting an I/O (Variable) Type

A new field will appear to the right of the first field, prompting you for the name of the item being copied. As with the operation name, you can enter the name of the variable in three ways:

1. Type the complete name in the field.
2. Type the first few letters and choose from a list.
3. Press `ENTER` before typing any characters and choose from the list of configured variables of the type of you have chosen (in this case, digital inputs). This is often the quickest and easiest method.

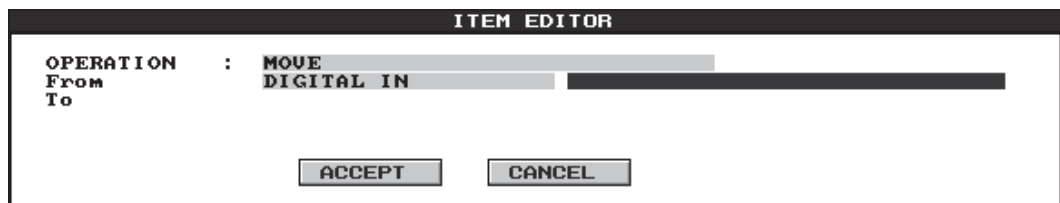


Figure 4-16: Entering an I/O (Variable) Name

You will now use the third method to bring up a list of configured digital input points.

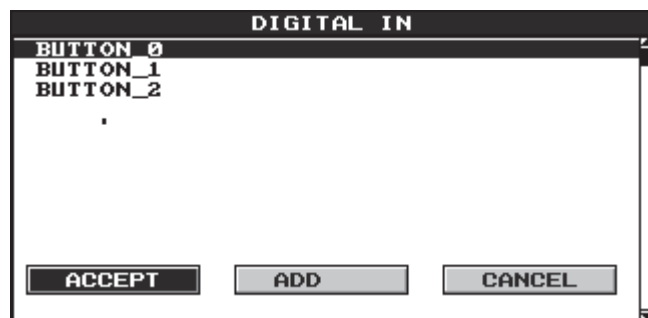


Figure 4-17: Selecting a Digital Input Point

Highlight `BUTTON_0` in the list and press `ENTER` while `ACCEPT` is highlighted. `BUTTON_0` will now appear alongside `DIGITAL IN` in the `ITEM EDITOR` dialog box.



Figure 4-18: Selecting an I/O (Variable) Type

A new field will now appear next to the label To, prompting you for the type of variable to copy the original item to. For our example, the digital input BUTTON_0 will be copied to a digital output point. Select DIGITAL OUT as the type (this type appears automatically by default) and press ENTER.

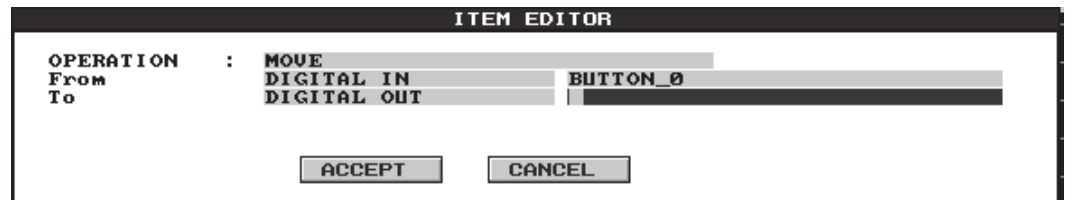


Figure 4-19: Entering an I/O (Variable) Name

Once again a new field will appear, prompting you for the type of item being copied to. You can use the same three methods previously described to enter the variable name. For our example, press ENTER without typing any characters and select LED_0 as the variable name from the list of configured digital output points.

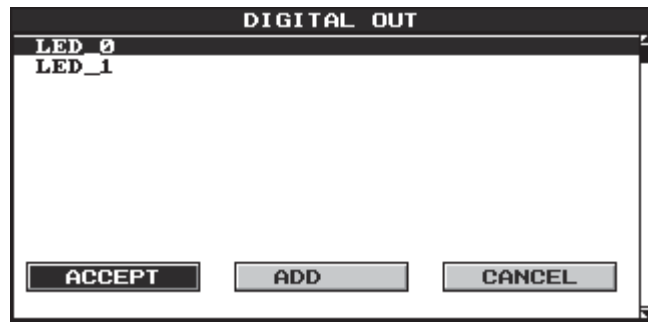


Figure 4-20: Selecting a Digital Output Point

Once a type and name have been filled in for both the *From* and *To* parameters, ACCEPT will be highlighted. Press ENTER to accept the new command and close the ITEM EDITOR dialog box.



Figure 4-21: Accepting a Completed Command

The new command will now appear in the LED_0_CONTROL detail window. Press ESC or the right mouse button once to close this window and again to release the DETAIL tool.

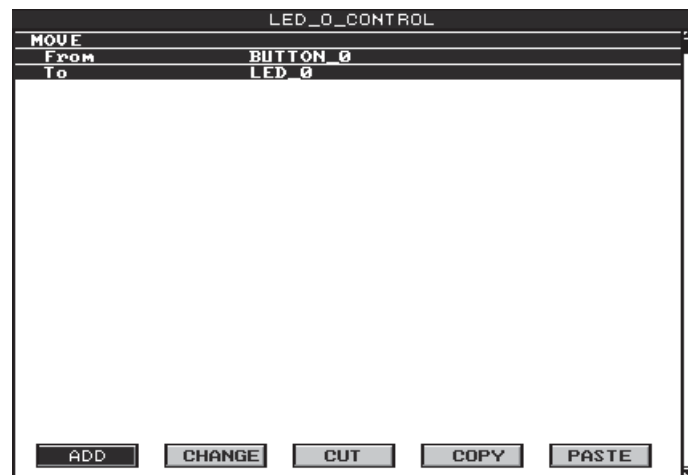


Figure 4-22: Viewing an Added Command

START THE CHART FROM THE POWERUP CHART

Every Cyrano program has a POWERUP chart. The POWERUP chart receives the first time slice after the HOST task and begins running without being started by a command. Any user-created charts that need to begin running when a program is executed should be started from the POWERUP chart.

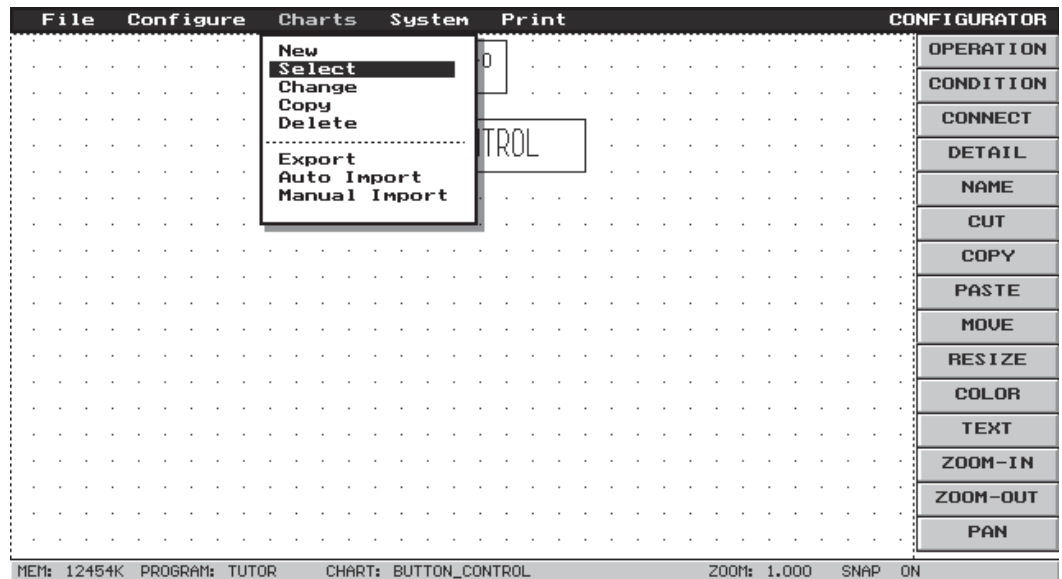


Figure 4-23: Changing Chart Views

You will now enter the POWERUP chart. Choose Select from the Charts menu to bring up the SELECT CHART dialog box, which will display all charts in your program. Select POWERUP from the list and press ENTER or the left mouse button.

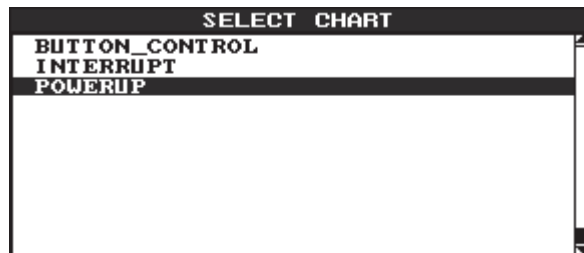


Figure 4-24: Selecting a Chart

This will open the POWERUP chart. Select the OPERATION tool from the toolbar and draw a new operation block below BLOCK-0.

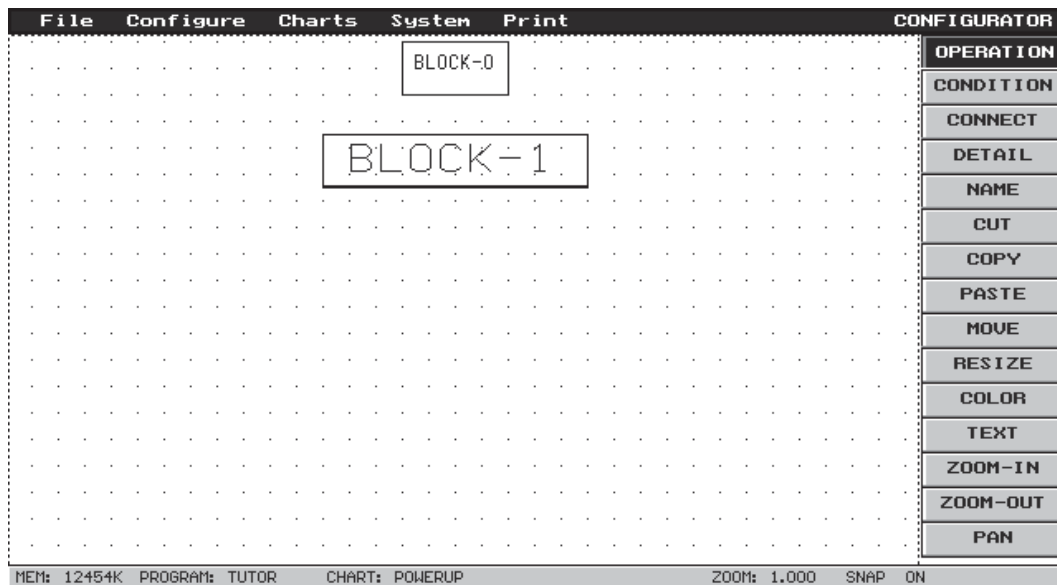


Figure 4-25: Drawing an Operation Block

Use the NAME tool on the toolbar to name the new block START_CHARTS.

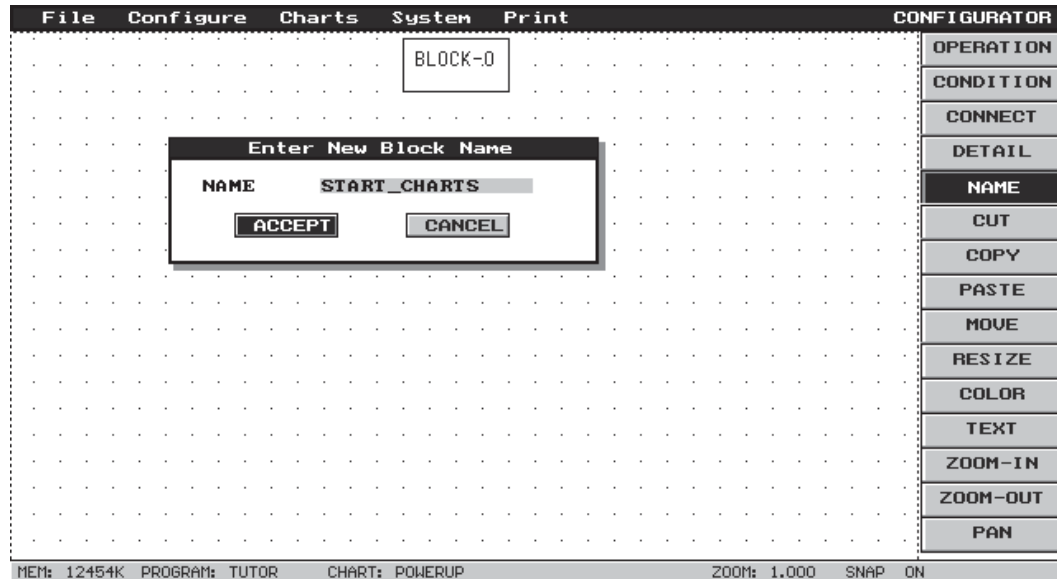


Figure 4-26: Naming an Operation Block

Use the CONNECT tool on the toolbar to connect BLOCK-0 to the START_CHARTS operation block.

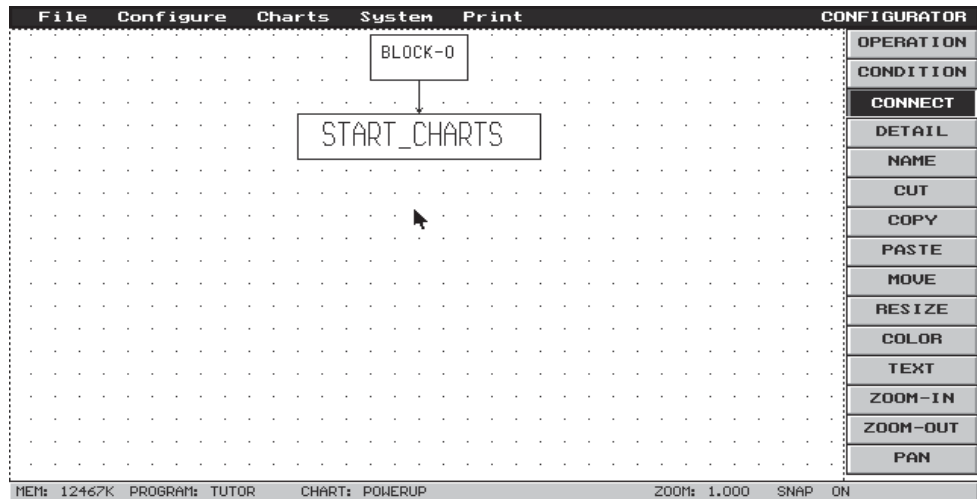


Figure 4-27: Drawing Connection Lines

Open the detail window for the START_CHARTS block by double clicking the left mouse button over the block. Highlight ADD at the bottom of the detail window and press ENTER. This will open the ITEM EDITOR dialog box. Press ENTER at the OPERATION field to bring up the GROUP SELECTIONS dialog box.

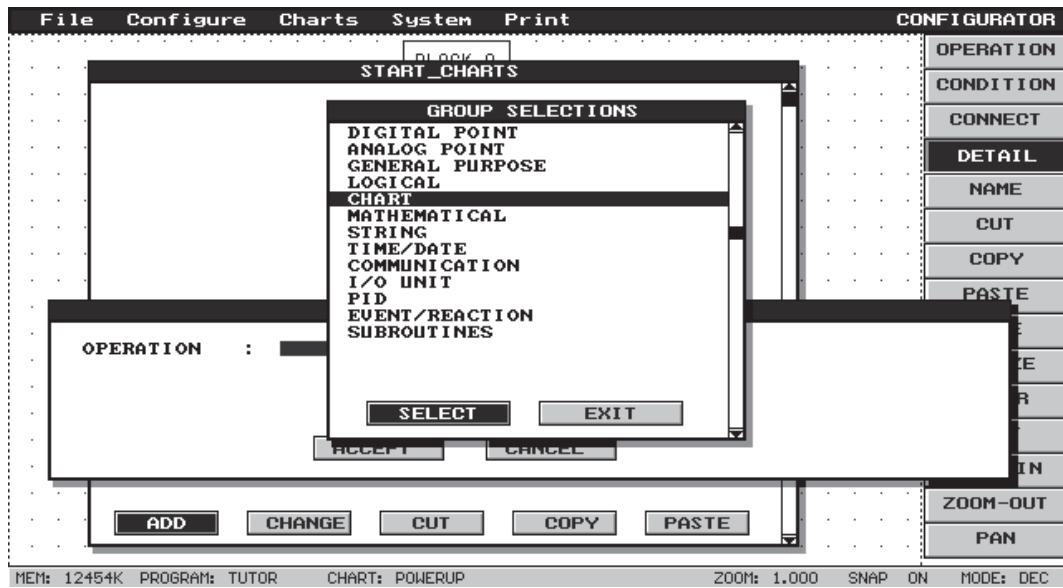


Figure 4-28: Selecting a Group

Use the up or down arrow keys to highlight CHART, then press ENTER to view all commands in the Chart command group.

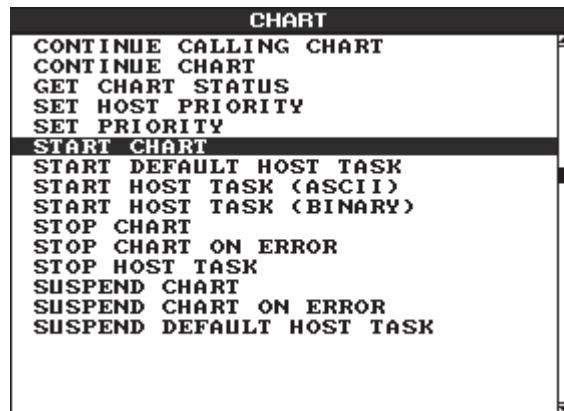


Figure 4-29: Selecting an Operation

Select the START_CHART command from the CHART dialog box and press ENTER. This will return you to the ITEM EDITOR dialog box. Press ENTER once to advance to the chart name field, then press

ENTER again without typing any characters to bring up a list of available charts. Select `BUTTON_CONTROL` and press ENTER.

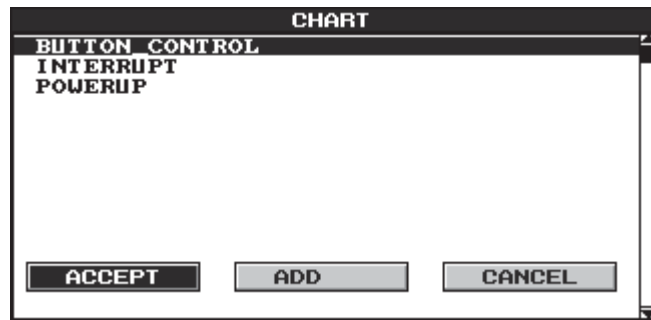


Figure 4-30: Selecting a Chart

When a `START CHART` command is executed, Cyrano returns a status of True (-1) if the requested chart starts successfully. If the chart was already running or is unable to start due to the 32-task queue's being full, Cyrano returns a False (0) status. You will thus need to create a numeric variable, using the on-the-fly method, to store the status returned when the program attempts to start the `BUTTON_CONTROL` chart.

Use the left or right arrow keys to select the variable type as `VARIABLE INTEGER`, then press ENTER. When selecting variable types for numeric variables in Cyrano, it is good programming practice to match the type of data to be stored in the variable. For example, although integers can be stored in floating point variables, Cyrano must first convert the value to a float before it can be stored. Unnecessary conversions can slow the execution of your program.

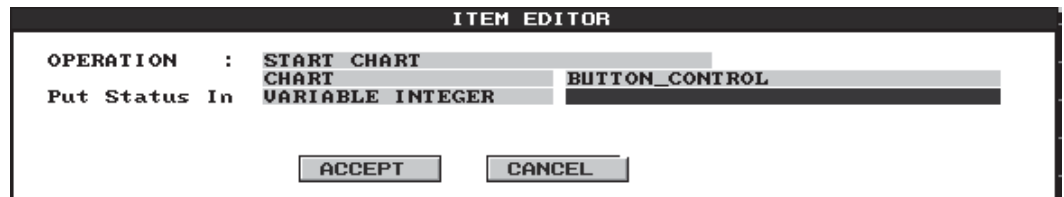


Figure 4-31: Selecting a Variable Type

You will now need to enter a name for the status variable in the field to the right of `VARIABLE INTEGER`. Press ENTER without typing any characters in this field. Since no variables of this type have been defined in this program, a dialog box will ask you if you want to define a new variable. Select YES.



Figure 4-32: Confirming a Variable Definition

The `ADD NUMERIC VARIABLE` dialog box will appear. In the `NAME` field, type `BUTTON_CONTROL_CHART_STATUS` as the name of the variable to be added. The `TYPE` field will default to `INTEGER`, since we already requested a `VARIABLE INTEGER` in the `ITEM EDITOR` dialog box. Initialize the variable to 0 by selecting the default values of YES and 0 in the `INIT.` and `INIT VAL`

fields. Also leave the SECURITY entry at its default value of 0. Press ENTER when ACCEPT is highlighted to accept the new variable.

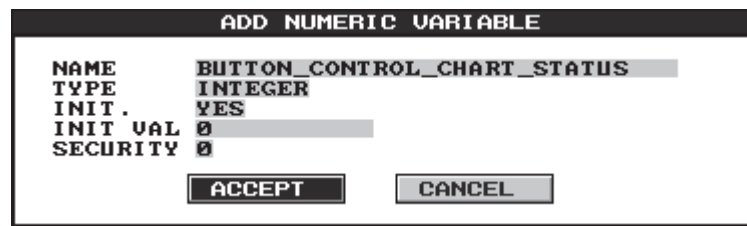


Figure 4-33: Adding a Numeric Variable

This will return you to the ITEM EDITOR dialog box, where all information will now be complete. Press ENTER when ACCEPT is highlighted to accept the new command. The completed command can be viewed in the detail window for START_CHARTS.

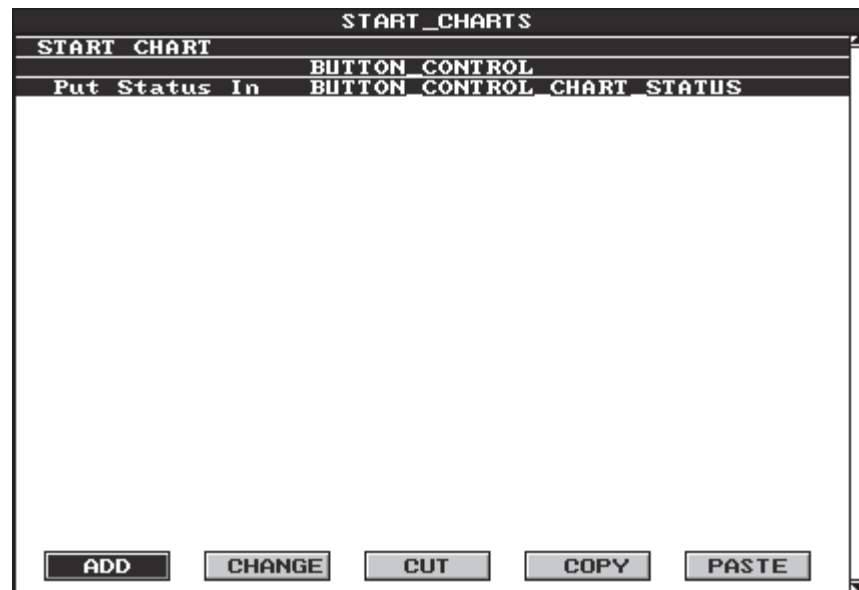


Figure 4-34: Viewing an Added Command

Press ESC or the right mouse button once to close this window and again to release the DETAIL tool.

DOWNLOAD AND RUN THE PROGRAM

You are now ready to download the program to the control processor and enter the Debugger module. Select Debugger from the Configurator's System menu.

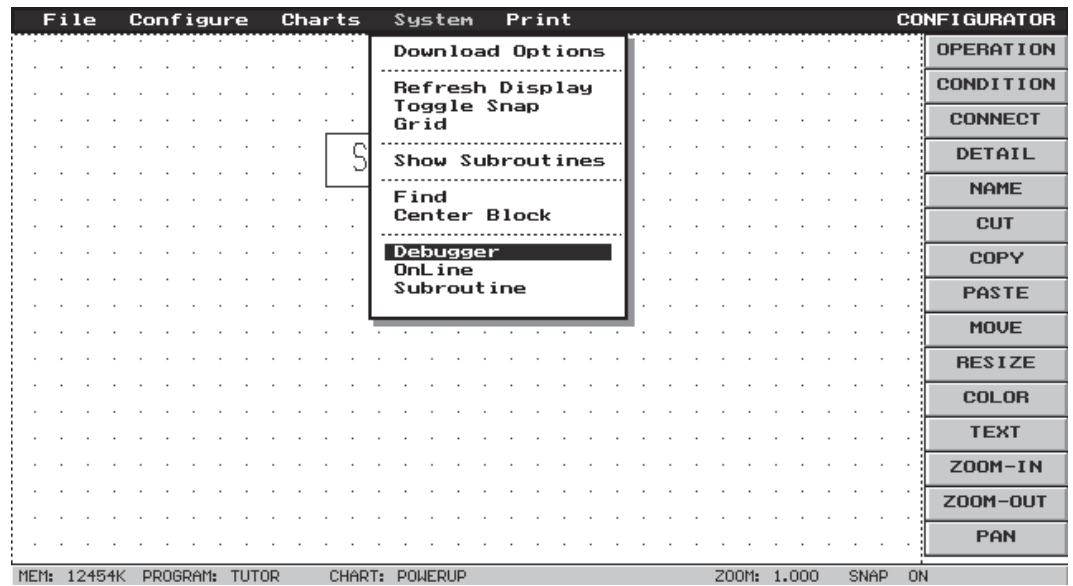


Figure 4-35: Opening the Debugger from the Configurator

A prompt will appear to ask you if the program should be saved. Highlight YES and press ENTER to save the program before it is downloaded to the processor. The program will be saved with the current path and program name.



Figure 4-36: Confirming a Save

If power to the controller has been cycled, a POWERUP message will appear, indicating that the controller has been reset. Press any key.



Figure 4-37: Closing Message Dialog Box

A DOWNLOAD WARNING dialog box will appear next, displaying information about the program you are attempting to download and the program currently running on the controller. The programs are date- and time-stamped to distinguish different versions of the same program. You can continue the download by selecting YES. Choosing YES erases the program currently running on the controller and replaces it with your program. To cancel the download request, select NO. For our example, choose YES.



Figure 4-38: Confirming the Download

Several dialog boxes will display information regarding the download process. Each dialog box will close automatically when the process is complete. The last dialog box in the series will display the amount of memory available in the controller processor after your program has been loaded.



Figure 4-39: Closing Memory Information Dialog Box

Press ENTER to close this dialog box and the Debugger module will open, displaying the last chart viewed in the Configurator.

Notice that the word DEBUGGER appears at the far right of the menu bar above the toolbar. Also, the menu bar and toolbar include different commands and tools in this module than in the Configurator.

The status of the program and the name of the currently displayed chart appear in the lower left corner of the drawing window. Notice that the program and chart are currently stopped.

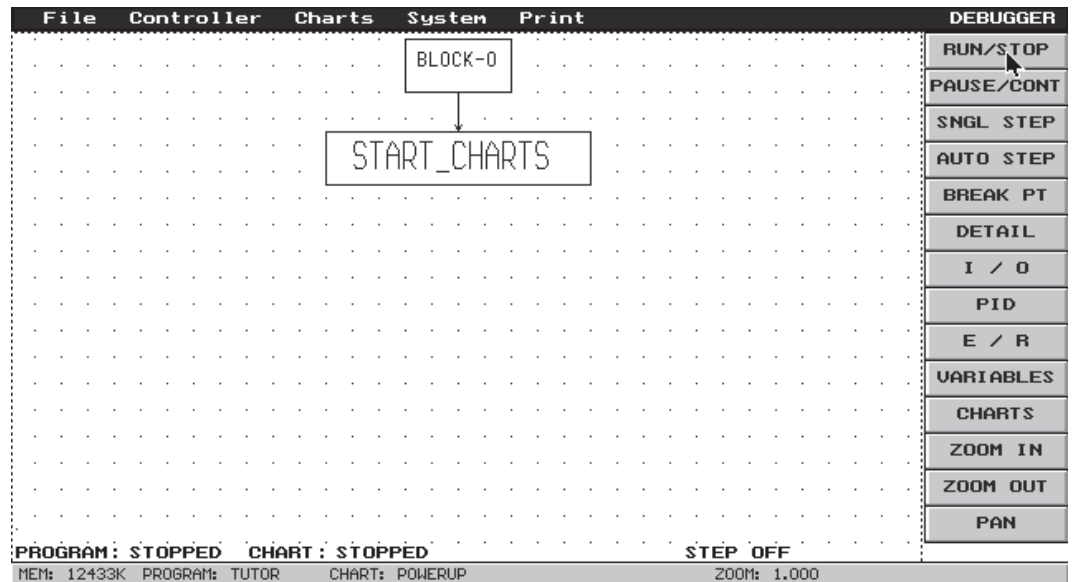


Figure 4-40: Starting a Program

Start the program by selecting the RUN/STOP tool on the toolbar. This tool works as a toggle to run a stopped program or stop a running program.

Notice that even though the program began running, the status of the POWERUP chart is still stopped. This is because when the program started, the POWERUP chart executed the commands in the START_CHARTS block. Since there is no exit for this block, the chart stopped once the commands were executed.

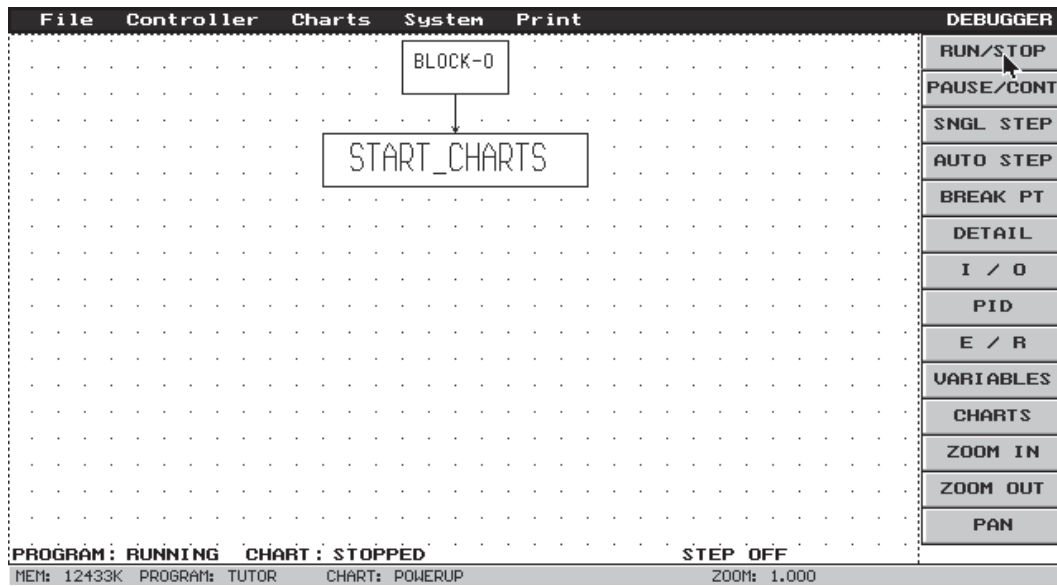


Figure 4-41: Viewing the POWERUP Chart Status

View the BUTTON_CONTROL chart by choosing Select from the Charts menu.

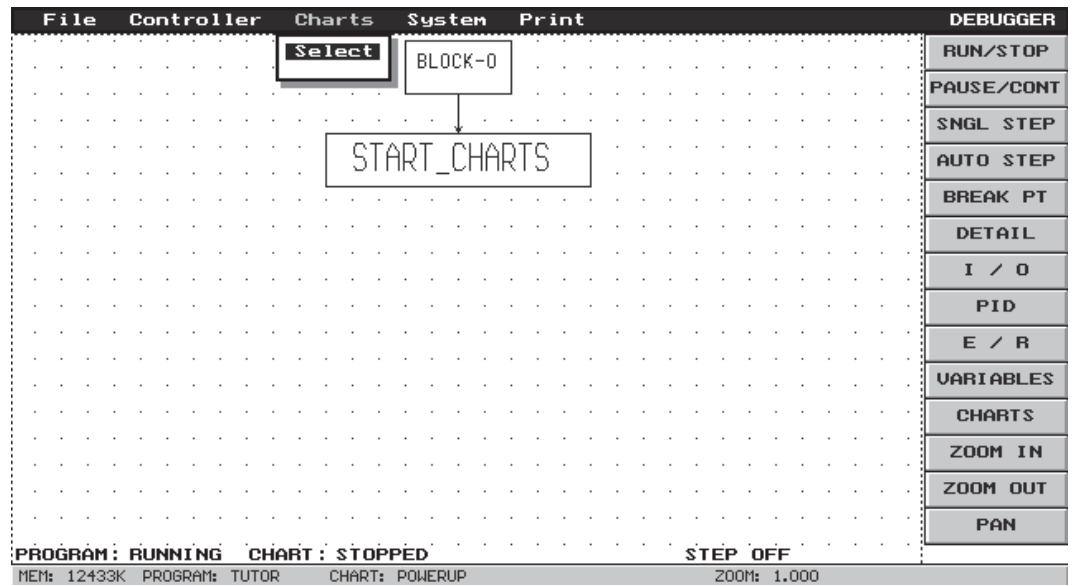


Figure 4-42: Changing Chart Views

Select `BUTTON_CONTROL` from the list in the `SELECT CHART` dialog box.

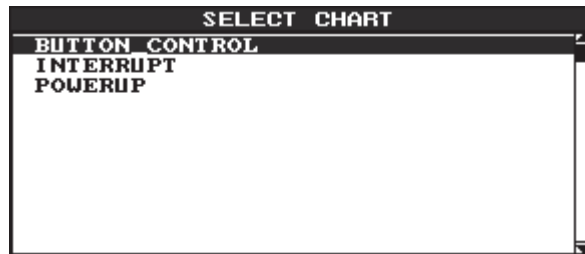


Figure 4-43: Selecting a Chart

Notice that the `BUTTON_CONTROL` chart is running. Check the operation of this chart by pressing the 0 button on the demo box. Notice that the first LED light on the box turns on when the button is pressed and turns off when the button is released.

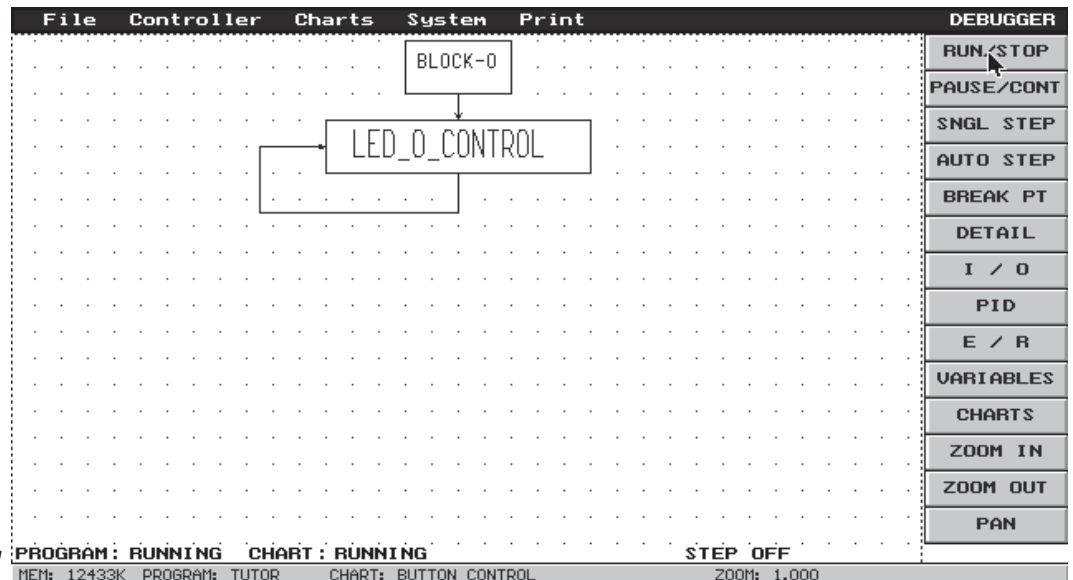


Figure 4-44: Viewing the `BUTTON_CONTROL` Chart Status

EDITING THE CHART

CHAPTER

5



WHAT TOPICS WILL I COVER?

- Using condition blocks
- Using the CUT, COPY and PASTE tools
- Modifying commands
- Using on-latches
- Configuring I/O points on the fly
- Using auto stepping
- Using the watch window

WHAT WILL I DO?

- Modify the BUTTON_CONTROL chart structure by adding one new condition block and two new operation blocks, redrawing connection lines as needed.
- Add commands to turn on LED_1 and turn off LED_2 when the latch at BUTTON_1 is set.
- Use the on-the-fly method to configure the point LED_2.
- Duplicate the newly added blocks and modify them to turn off LED_1 and turn on LED_2 when the latch at BUTTON_2 is set.
- Download the modified program to the Debugger.
- Use auto stepping and the watch window to examine the program as it is running.

HOW WILL I DO IT?

REDRAW THE FLOWCHART LOGIC

If you have exited the Configurator module, reopen it. Verify that the program name listed in the status bar at the bottom of the window is TUTOR.

We will now erase the connection line exiting from the LED_0_CONTROL block. Select the CUT tool from the toolbar and point to the exit line. Once the connection line begins flashing, press the left mouse button.

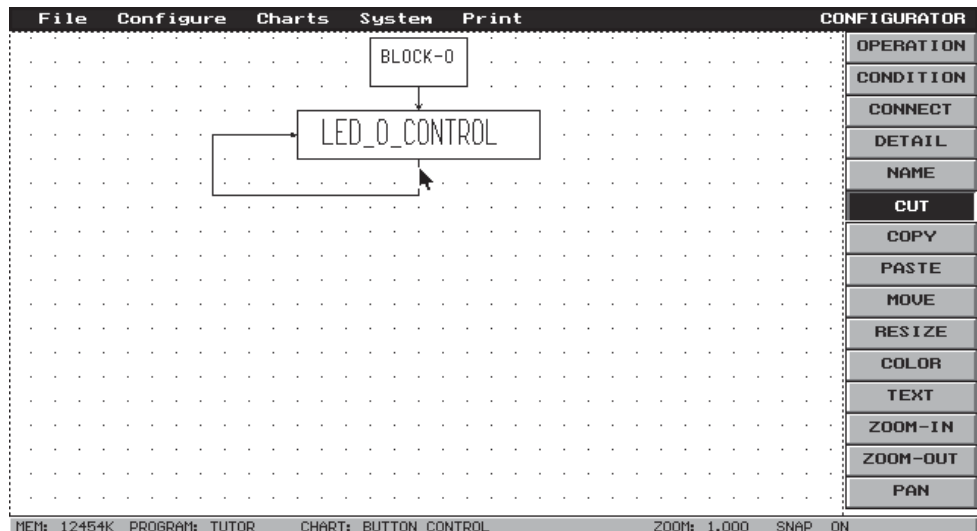


Figure 5-1: Using the CUT Tool

A dialog box will appear to ask you to confirm this action. While YES is highlighted, press ENTER. This will erase the selected connection line.



Figure 5-2: Confirming a Cut

We will now draw a CONDITION block below the LED_0_CONTROL operation block. Select the CONDITION tool on the toolbar. A diamond object will appear in the drawing window. Place the condition block below the LED_0_CONTROL block in the drawing window by moving the mouse and pressing the left mouse button when the block is in the desired location. The block can be resized while it is being drawn by pressing and holding the left mouse button as you move the mouse. Release the left mouse button when the block is the desired size. (A block can also be resized after it is drawn by using the RESIZE tool.)

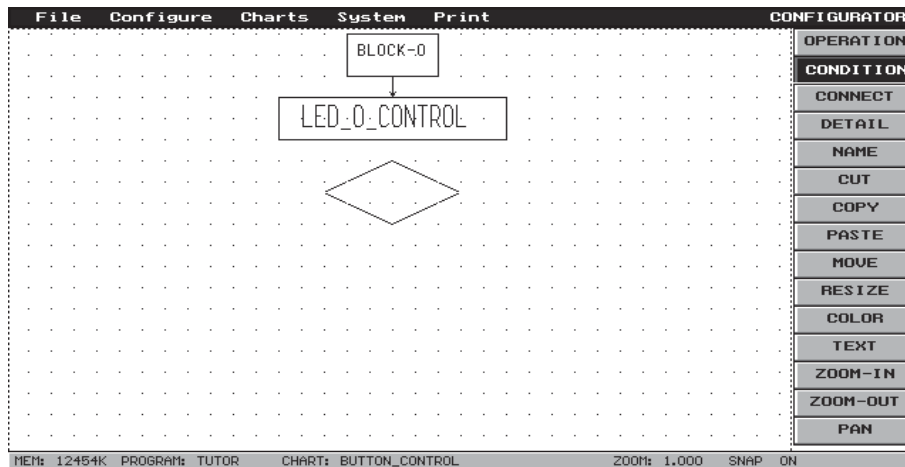


Figure 5-3: Drawing a Condition Block

Release the CONDITION tool by pressing ESC or the right mouse button.

Now select the OPERATION tool. Place two operation blocks side by side to the right of the condition block. Remember, operation blocks can be resized as they are drawn. Once they are placed, they can be resized using the RESIZE tool.

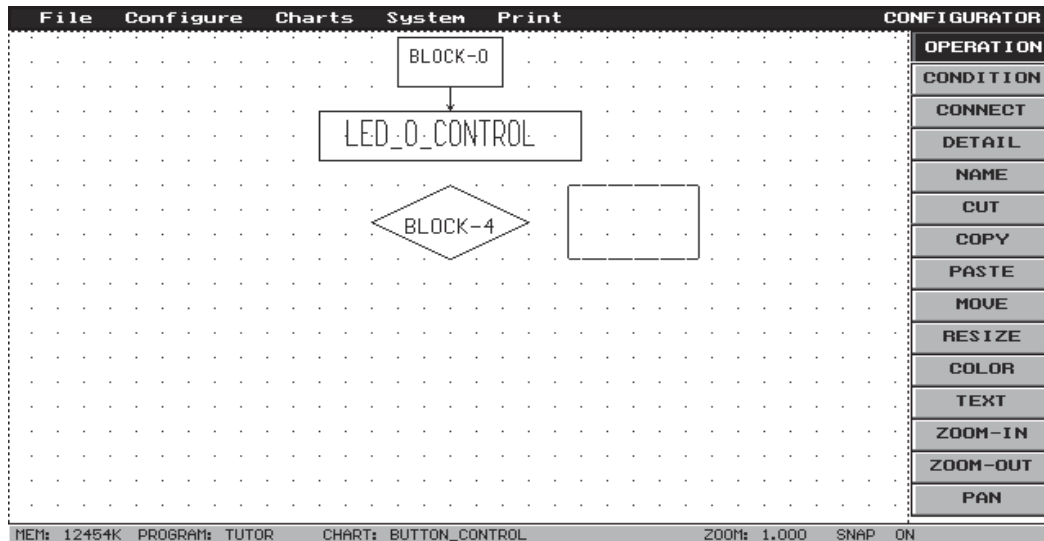


Figure 5-4: Drawing an Operation Block

Use the CONNECT tool to draw appropriate connection lines between the blocks. Notice that when you draw the first exit from a condition block, a SELECT CONNECTION dialog box will appear to ask you whether to draw a True or False exit (the True route will be taken if every condition in the block evaluates True, the False route will be taken otherwise). No prompt will appear when you draw the second exit, since at that point only one exit type can be drawn.

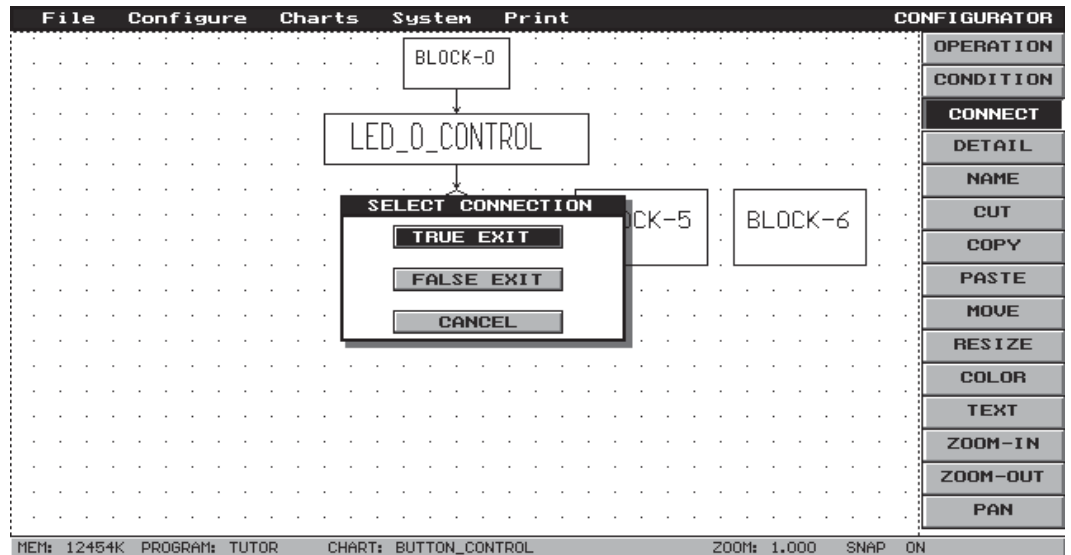


Figure 5-5: Selecting a Connection Exit

Note that if you attempt to draw too many exits from a block, you will be unable to select the block with the CONNECT tool.

Now use the NAME tool to name the new blocks. Block names should describe the function of the commands the blocks contain. It is good practice to include in each block only as many commands as can be completely described by the block name. This practice promotes the self-documenting nature of Cyrano and makes program debugging and maintenance easier and quicker.

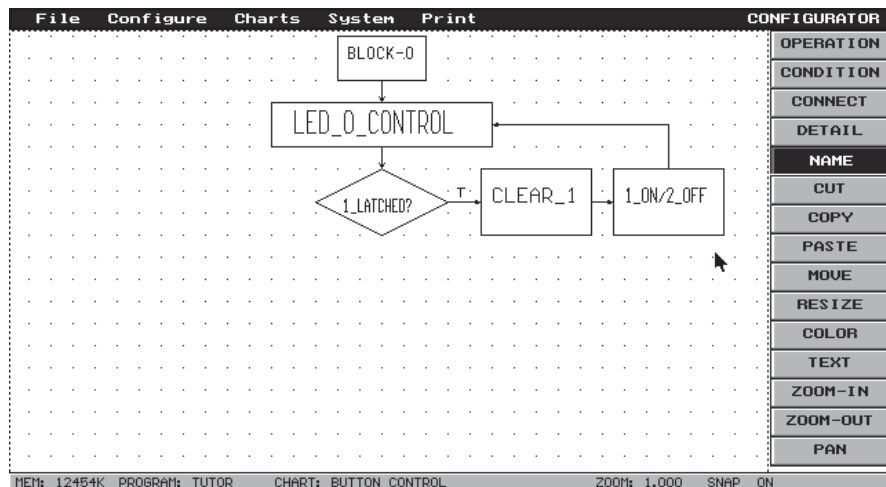


Figure 5-6: Naming an Operation Block

ENTER COMMANDS IN THE NEW BLOCKS

We will set up the condition block to check whether the latch at BUTTON_1 has been set. If the condition is True, the exit from this block enters an operation block that clears the latch. It is always good programming practice to clear a latch immediately after it has been triggered. If other operations are performed before the latch is cleared, the latch may be turned on again without triggering a second latching event.

After clearing the latch, we will include commands to turn on LED_1 and turn off LED_2 in the next operation block.

Move the mouse cursor over the condition block and double-click the left mouse button to open the detail window.

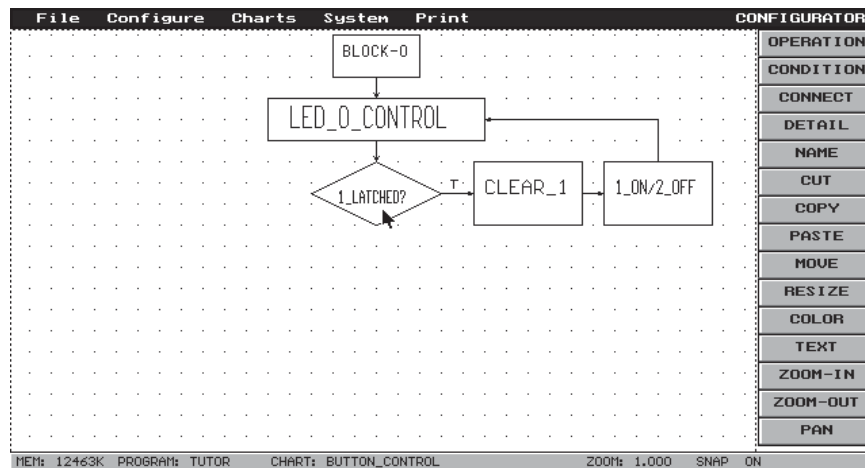


Figure 5-7: Opening a Detail Window

From the detail window for the block 1_LATCHED?, select ADD.

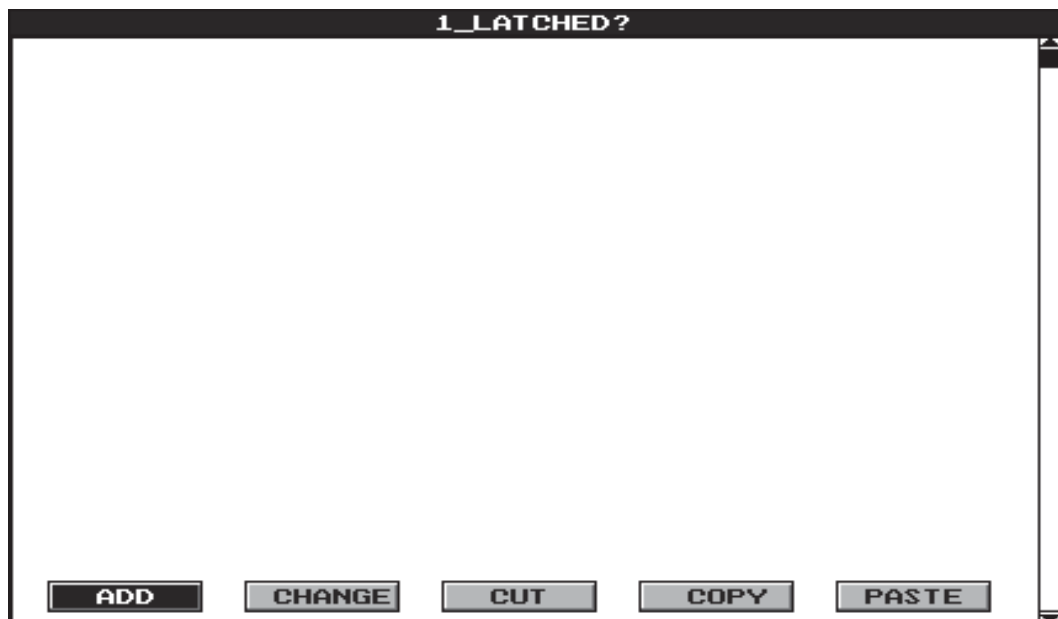


Figure 5-8: Adding a New Command

The ITEM EDITOR dialog box will open.



Figure 5-9: Entering a Condition

Press ENTER from the CONDITION field to open the GROUP SELECTIONS dialog box.

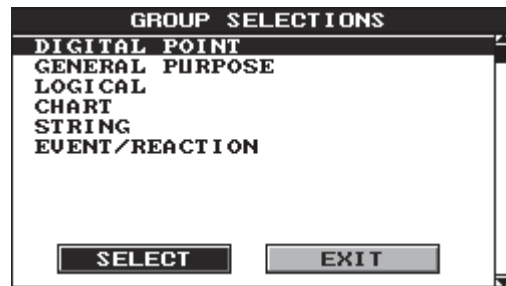


Figure 5-10: Selecting a Group

Select the DIGITAL POINT group from the list and a list of digital point conditions will appear.

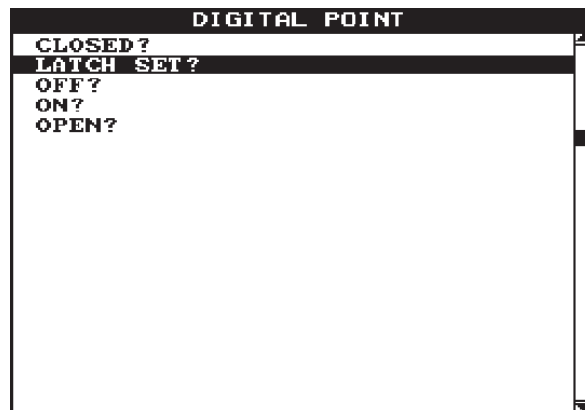


Figure 5-11: Selecting a Condition

Select the condition LATCH SET? from the list of commands. This condition returns True if the specified latch is set, False otherwise.



Figure 5-12: Selecting an I/O (Variable) Type

In the ITEM EDITOR dialog box, select the variable type as ON LATCH. Recall that the digital point BUTTON_1 was configured as an on-latch.



Figure 5-13: Entering an I/O (Variable) Name

In the field to the right of the variable type, you will need to supply the name of the latch. Press ENTER without typing any characters and a dialog box will display a list of all configured ON-latches.



Figure 5-14: Selecting an ON-Latch

Select BUTTON_1 from the list and press ENTER while ACCEPT is highlighted to open the ITEM EDITOR dialog box for BUTTON_1.



Figure 5-15: Accepting a Completed Command

When the information in the ITEM EDITOR dialog box is complete, ACCEPT will become highlighted. Press ENTER while ACCEPT is highlighted and the new command will appear in the detail window for 1_LATCHED?

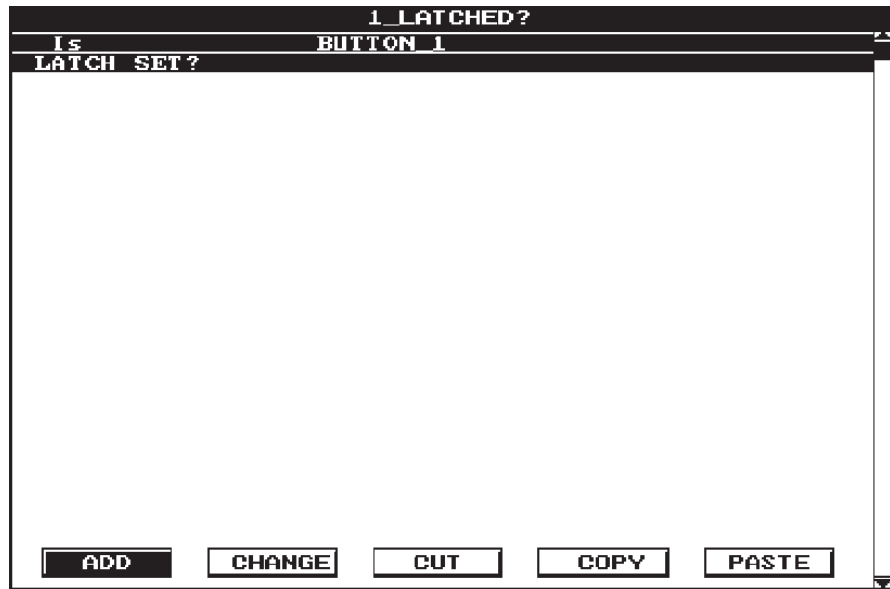


Figure 5-16: Viewing the Detail Window

Close the detail window by pressing ESC or the right mouse button. The DETAIL tool will still be highlighted. Using the mouse, point to the CLEAR_1 operation block and press the left mouse button to open the detail window for this block.

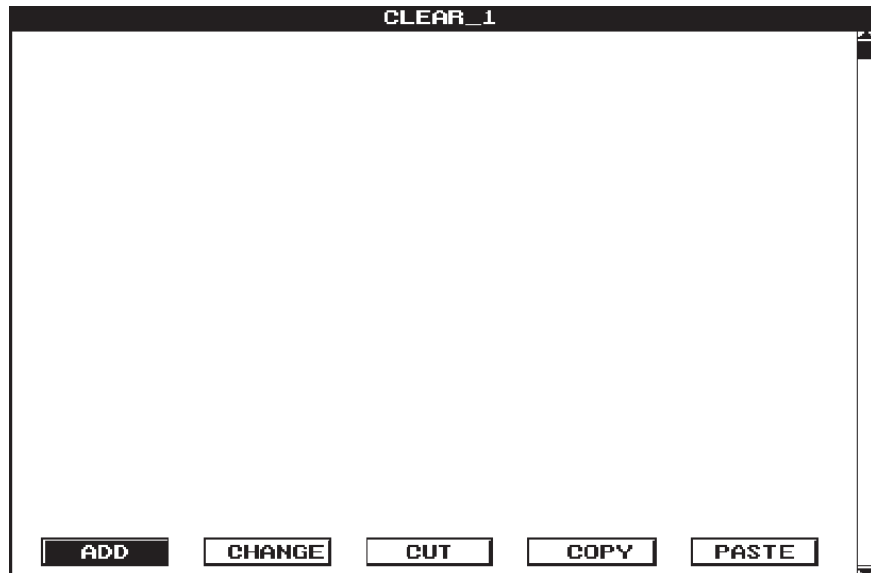


Figure 5-17: Adding a New Command

Press ADD to open the ITEM EDITOR dialog box.



Figure 5-18: Entering an Operation

Press ENTER to open the GROUP SELECTIONS dialog box.

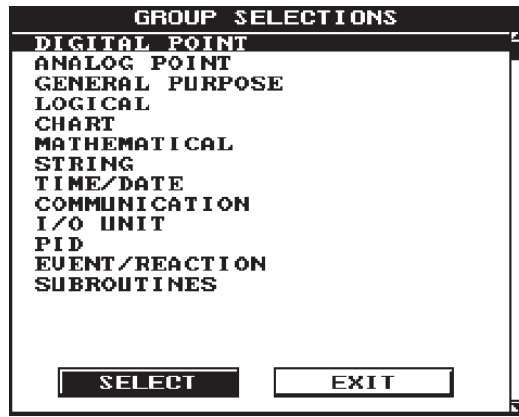


Figure 5-19: Selecting a Group

Select the DIGITAL POINT group and a list of digital point operations will appear.

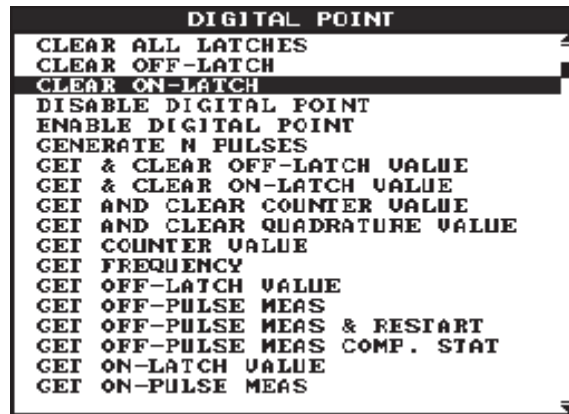


Figure 5-20: Selecting an Operation

Select CLEAR ON-LATCH from the DIGITAL POINT dialog box to enter this command in the ITEM EDITOR dialog box.



Figure 5-21: Accepting a Completed Command

Complete the information in the ITEM EDITOR dialog box as illustrated in Figure 5-21 and press ENTER when ACCEPT becomes highlighted.

The new command will appear in the detail window for the CLEAR_1 operation block.

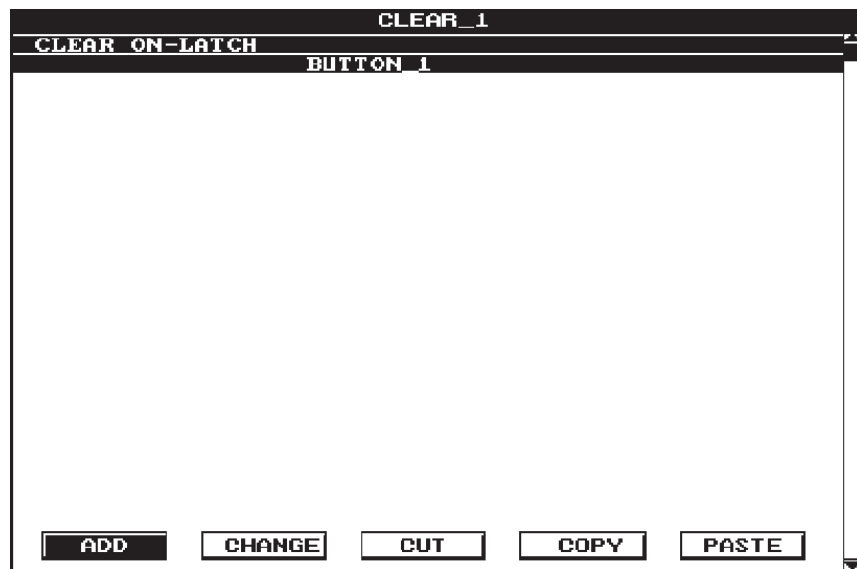


Figure 5-22: Viewing the Detail Window

Press ESC or the right mouse button to close the detail window. The DETAIL tool will still be highlighted. Using the mouse, point to the 1_ON/2_OFF operation block and press the left mouse button to open the detail window for this block.

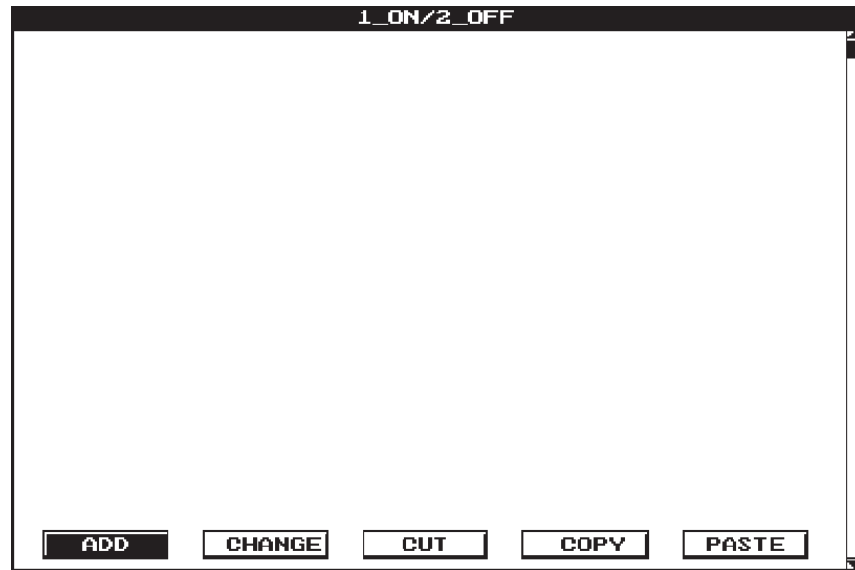


Figure 5-23: Adding a Command

Press ADD to open the ITEM EDITOR dialog box.



Figure 5-24: Entering an Operation

Press ENTER to open the GROUP SELECTIONS dialog box.

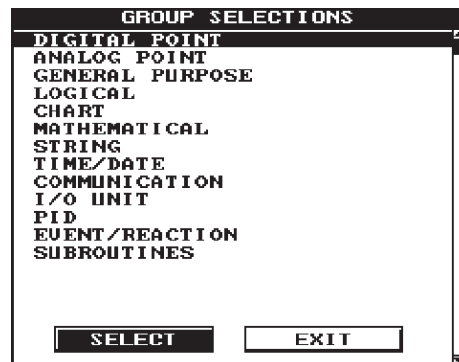


Figure 5-25: Selecting a Group

Select the DIGITAL POINT group.

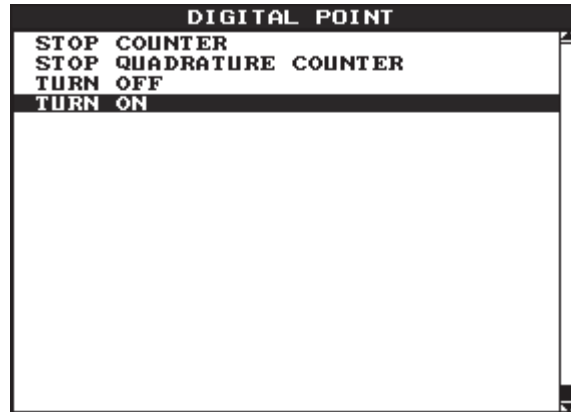


Figure 5-26: Selecting an Operation

Select TURN ON from the list to enter this command in the ITEM EDITOR dialog box.

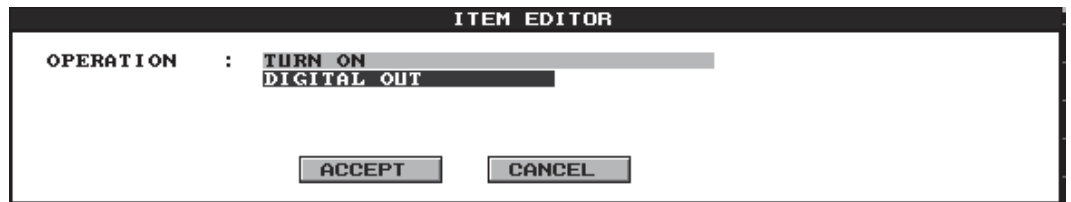


Figure 5-27: Selecting an I/O (Variable) Type

Use the left or right arrow keys to scroll through the list of I/O variable types. Since an LED is a digital output point, select DIGITAL OUT.

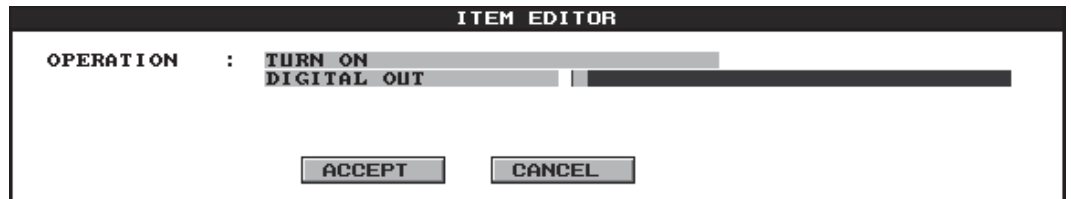


Figure 5-28: Entering an I/O (Variable) Name

In the field to the right of the variable type, you will need to supply the name of the digital output. Press ENTER and a dialog box will display a list of all configured digital output points.

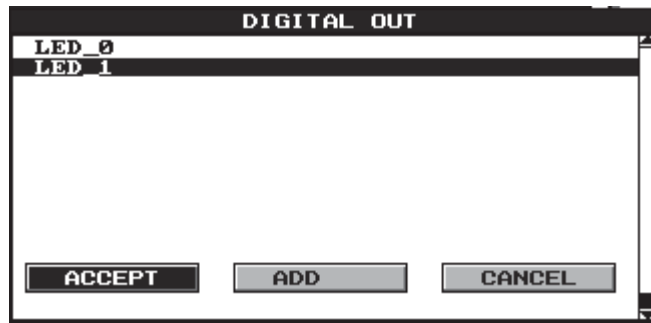


Figure 5-29: Selecting a Digital Point

Select LED_1 from the list. In the ITEM EDITOR dialog box, press ENTER while ACCEPT is highlighted to close the dialog box.

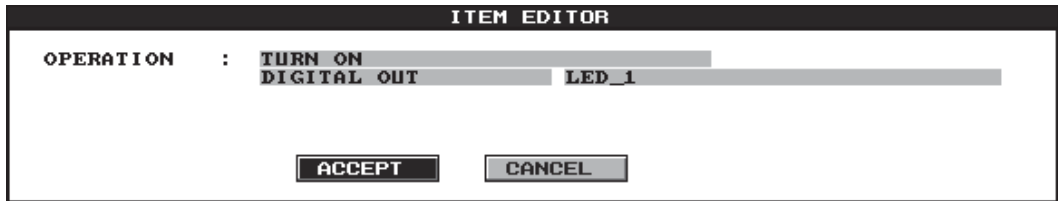


Figure 5-30: Accepting a Completed Command

The new command will appear in the detail window for the block 1_ON/2_OFF.

Now press ENTER while ADD is highlighted to add a second command to this block.

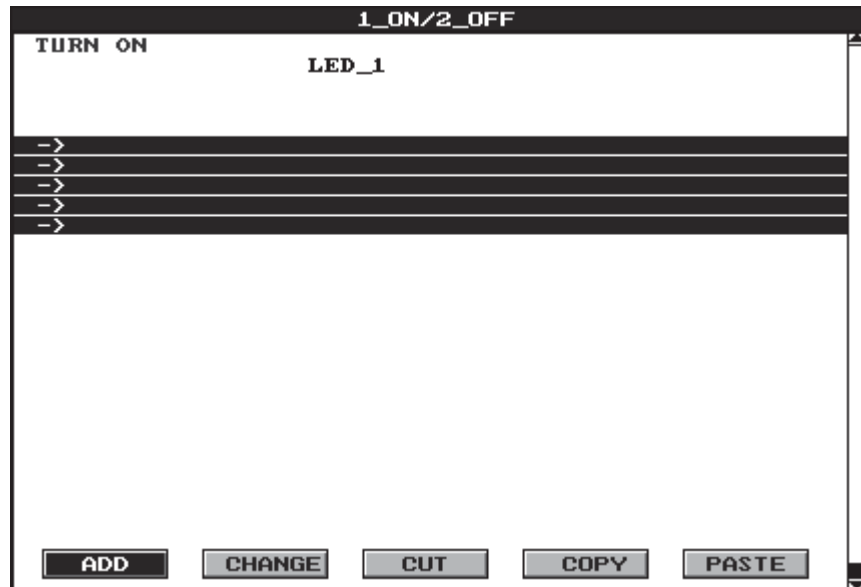


Figure 5-31: Adding a Second Command

Notice that five highlighted rows appear in the window, representing the command to be added. Using the up and down arrow keys, position these highlighted rows where the new command should be. In this case we want the new command to be executed after the TURN ON command, so the original position of the highlighted row is correct.

With the command rows correctly positioned, press ENTER to open the ITEM EDITOR dialog box.

We will now add a command to turn off the digital output point LED_2. The command TURN OFF can be found in the Digital Point group (or, as always, the command name can be typed directly into the OPERATION field).



Figure 5-32: Selecting an I/O (Variable) Name

In the field to the right of the variable type, you will need to supply the name of the digital output. Press ENTER and a dialog box will display a list of all configured digital output points.

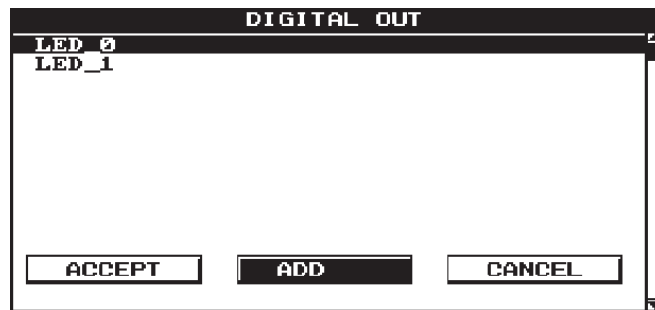


Figure 5-33: Adding an I/O Point On the Fly

Since the point LED_2 has not been configured, select ADD in the DIGITAL OUT dialog box to add this point using the on-the-fly method.

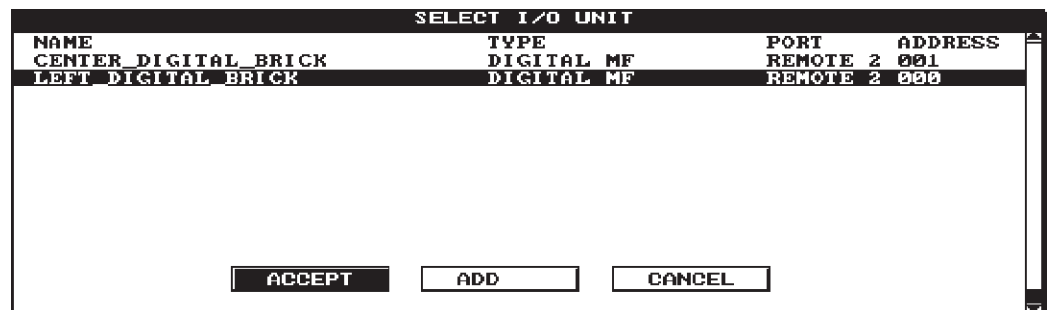


Figure 5-34: Selecting an I/O Unit

Select the LEFT_DIGITAL_BRICK from the SELECT I/O UNIT dialog box.

CONFIGURE DIGITAL I/O POINT				
UNIT	LEFT_DIGITAL_BRICK	ADDRESS 000		
CH#	NAME	TYPE	FEATURES	USED
00	BUTTON_0	DIN	NONE	01
01	BUTTON_1	DIN	ON LATCH	02
02	BUTTON_2	DIN	ON LATCH	00
03	UNUSED			
04	UNUSED			
05	UNUSED			
06	UNUSED			
07	UNUSED			
08	LED_0	DOUT	NONE	01
09	LED_1	DOUT	NONE	01
10	UNUSED			
11	UNUSED			
12	UNUSED			
13	UNUSED			
14	UNUSED			
15	UNUSED			

ADD DELETE CHANGE DUP MOVE

Figure 5-35: Adding a New I/O Point

Use the up or down arrow keys to highlight channel 10 of this brick, then select ADD to configure the point LED_2 at this channel.

CONFIGURE DIGITAL I/O POINT	
NAME	LED_2
TYPE	DOUT
MODULE	5 - 60 UDC G40DC5
FEATURES	NONE
DEFAULT	LAST
ENABLE	YES
WATCHDOG	DISABLED OFF
SECURITY	0

ACCEPT CANCEL

Figure 5-36: Configuring a Digital I/O Point

Configure the new point as shown above, then press ENTER while ACCEPT is highlighted to add the new command to the ITEM EDITOR dialog box.

ITEM EDITOR	
OPERATION	: TURN OFF DIGITAL OUT LED_2

ACCEPT CANCEL

Figure 5-37: Accepting a Completed Command

Press ENTER while ACCEPT is highlighted in the ITEM EDITOR dialog box and the new TURN OFF command will appear below the TURN ON command in the 1_ON/2_OFF detail window.

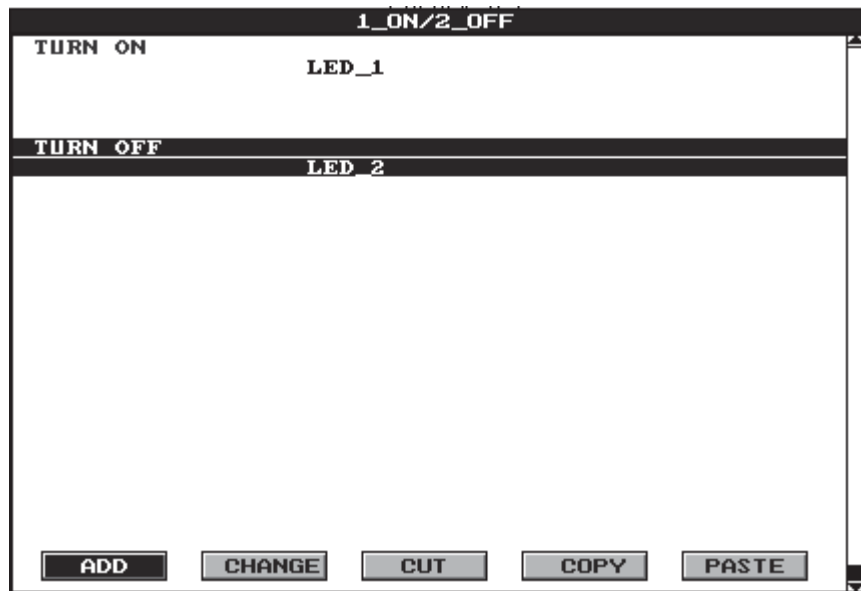


Figure 5-38: Viewing the Detail Window

Press ESC or the right mouse button twice, once to close the detail window and again to release the DETAIL tool.

CREATE THREE NEW BLOCKS

We will now copy and paste the 1_LATCHED?, CLEAR_1, and 1_ON/2_OFF blocks.

Select the COPY tool. Place the mouse cursor just above and to the left of the condition block 1_LATCHED? and press and hold the left mouse button. While holding the mouse button, drag the mouse so that the box in the drawing window surrounds the condition block and the operation blocks CLEAR_1 and 1_ON/2_OFF. After surrounding the three blocks, release the mouse button. This will copy the three blocks to a temporary buffer.

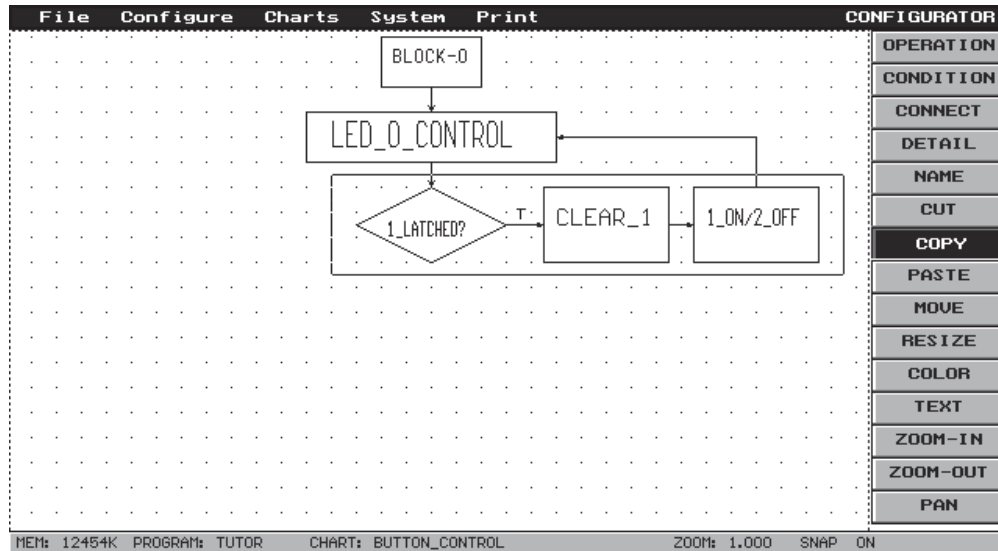


Figure 5-39: Using the COPY Tool

Select the PASTE tool and a rectangle will appear representing the blocks in the buffer. Position this rectangle just below the three original blocks and press the left mouse button.

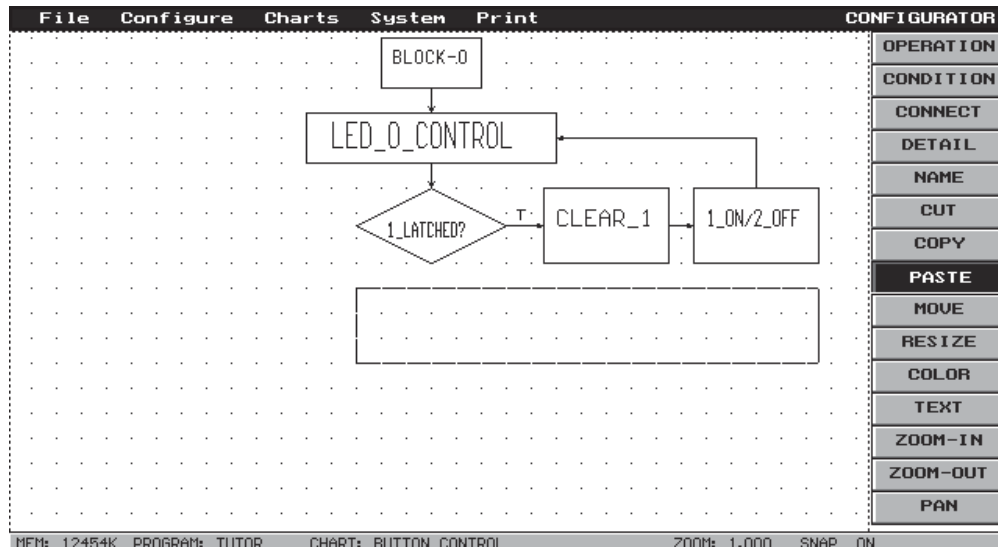


Figure 5-40: Using the PASTE Tool

This will paste copies of the three original blocks into the chart at the position you have indicated. These three new blocks will contain the same commands as the original blocks.

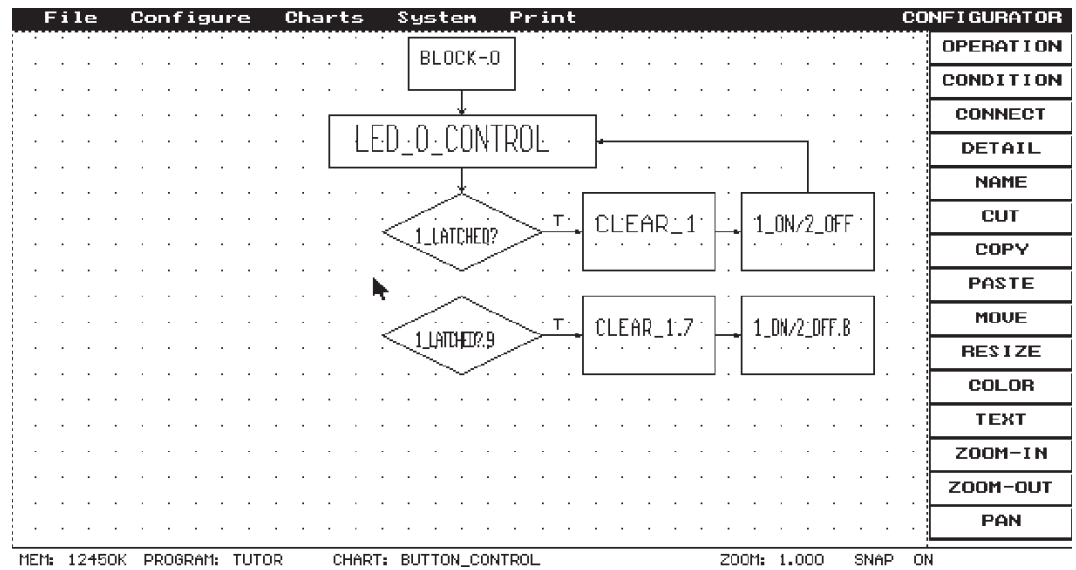


Figure 5-41: Viewing the Pasted Blocks

Use the NAME tool to rename the new blocks. The new condition block will monitor the latch at BUTTON_2. The first new operation block will clear the latch at BUTTON_2. The second operation block will turn on LED_2 and turn off LED_1. Name the blocks accordingly.

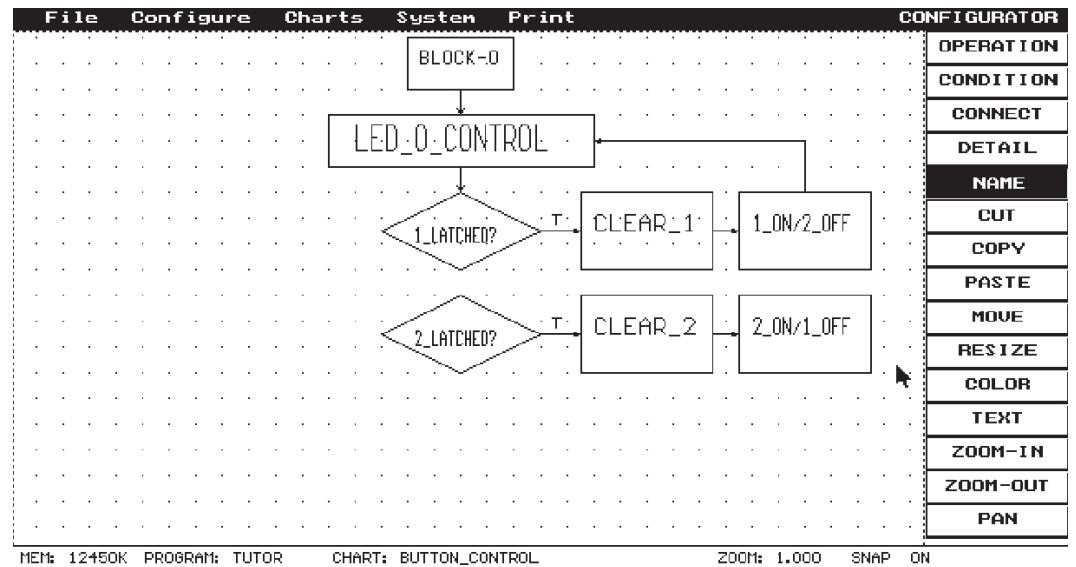


Figure 5-42: Naming the Pasted Blocks

Use the CONNECT tool to complete the logic flow in the chart, as shown below. Release the CONNECT tool when finished by pressing the ESC key.

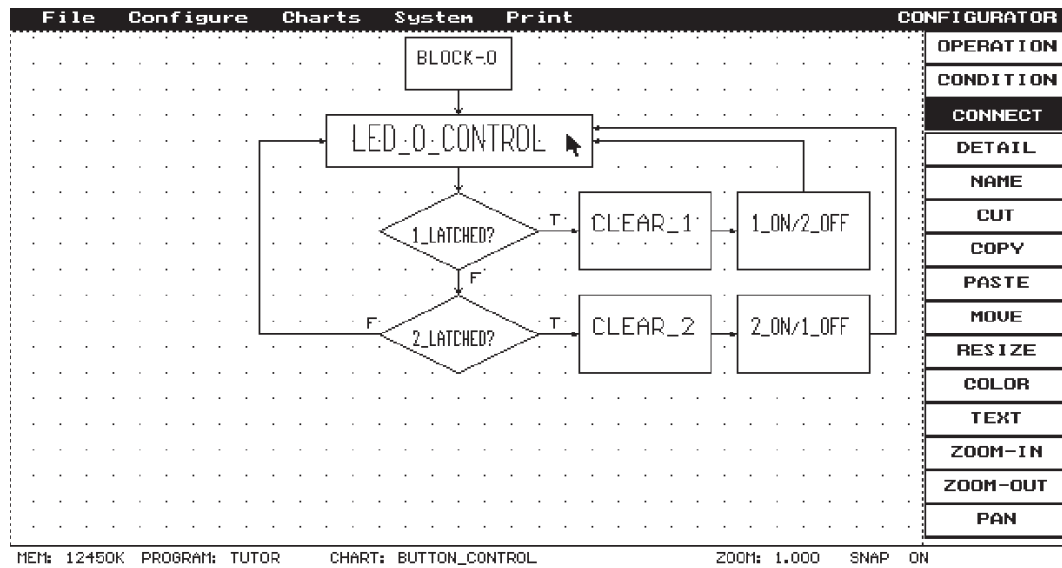


Figure 5-43: Drawing New Connection Lines

Double-click the left mouse button over the 2_LATCHED? condition block to open the detail window.

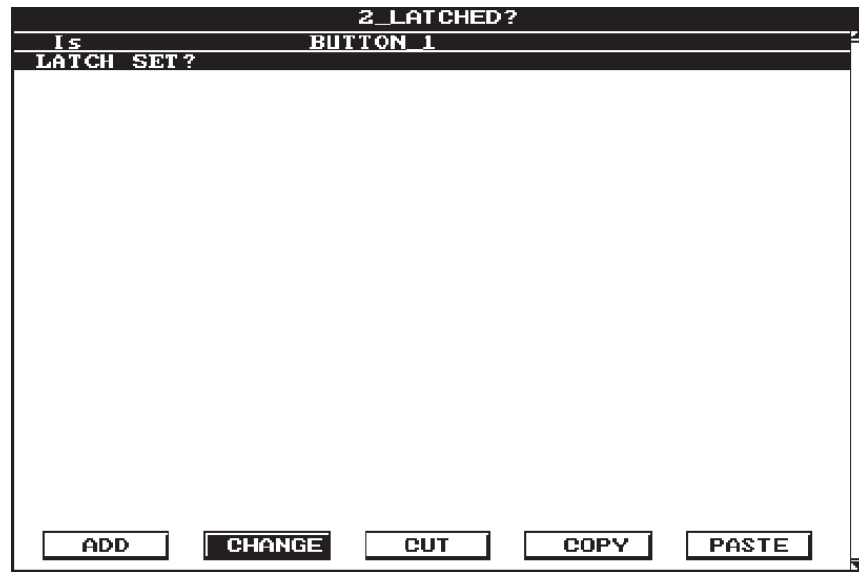


Figure 5-44: Changing a Command

Select CHANGE to modify the command in this block. Press ENTER while CHANGE is highlighted to open the ITEM EDITOR dialog box.

Change the name of the on-latch from BUTTON_1 to BUTTON_2. Press ACCEPT to complete the change.



Figure 5-45: Accepting the Edited Command

Verify that the new LATCH SET? command checks the status of the BUTTON_2 latch. Close the detail window for the 2_LATCHED? condition block by pressing ESC.

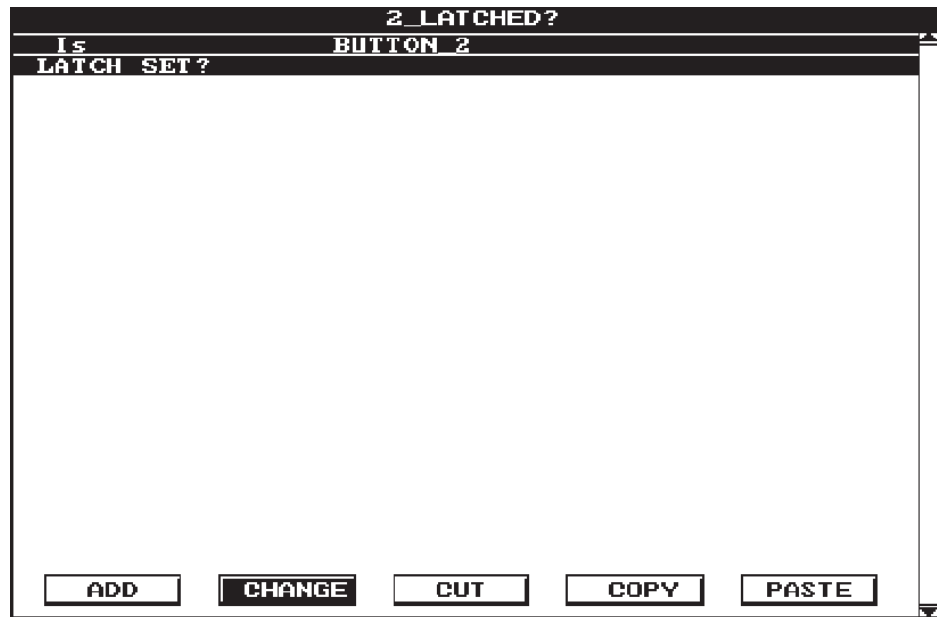


Figure 5-46: Viewing the Changed Command

With the DETAIL tool still selected, click the left mouse button over the CLEAR_2 operation block to open its detail window.

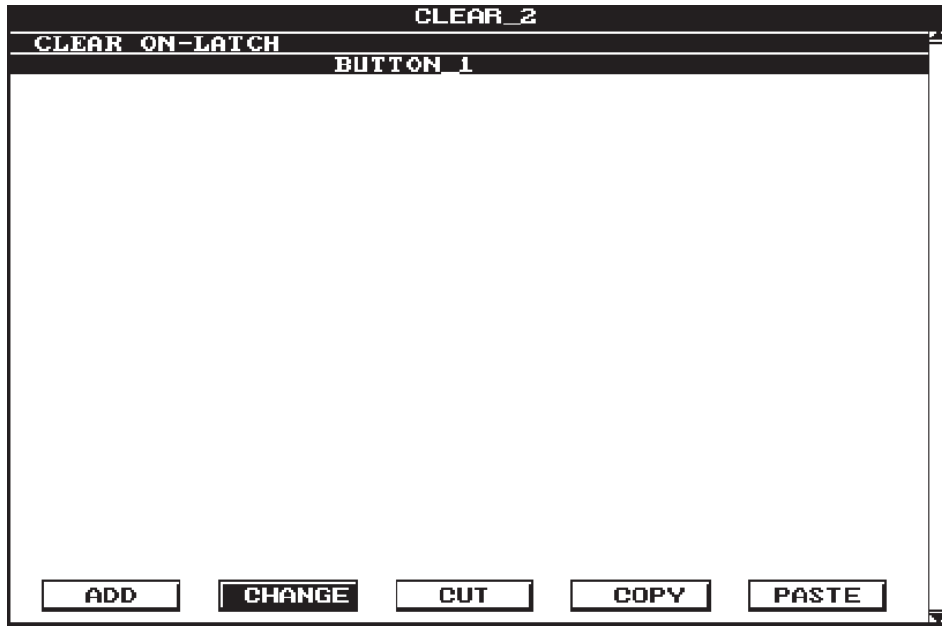


Figure 5-47: Changing a Command

Select CHANGE button to change the command in this block so that it clears the latch for BUTTON_2.



Figure 5-48: Editing a Command

After changing the latch name from BUTTON_1 to BUTTON_2, press ACCEPT in the ITEM EDITOR dialog box to complete the change.

The changed command will be displayed in the detail window for this block. Close the detail window by pressing ESC.

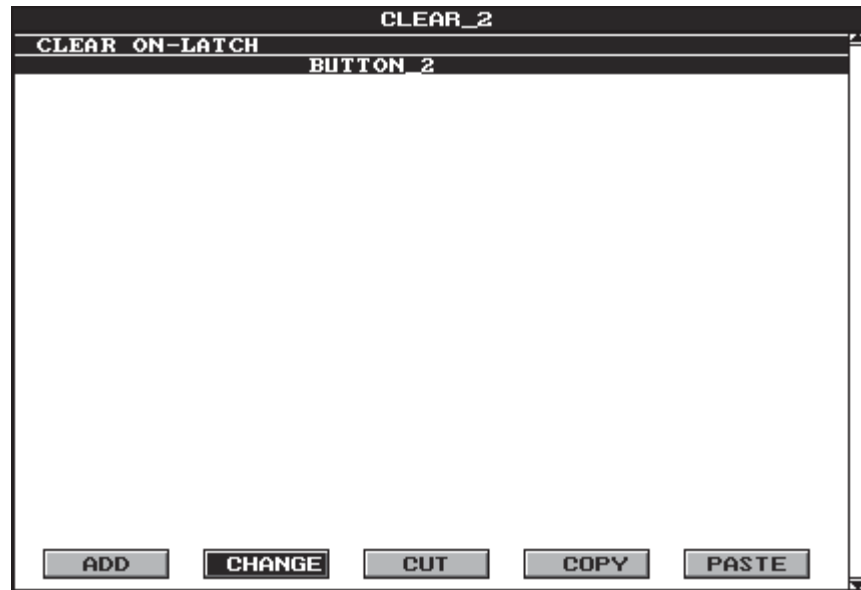


Figure 5-49: Viewing the Changed Command

Open the detail window for operation block 2_ON/1_OFF. Notice that you can use the up or down arrows to highlight either command. Highlight the TURN ON command and press ENTER while CHANGE is highlighted to modify this command.

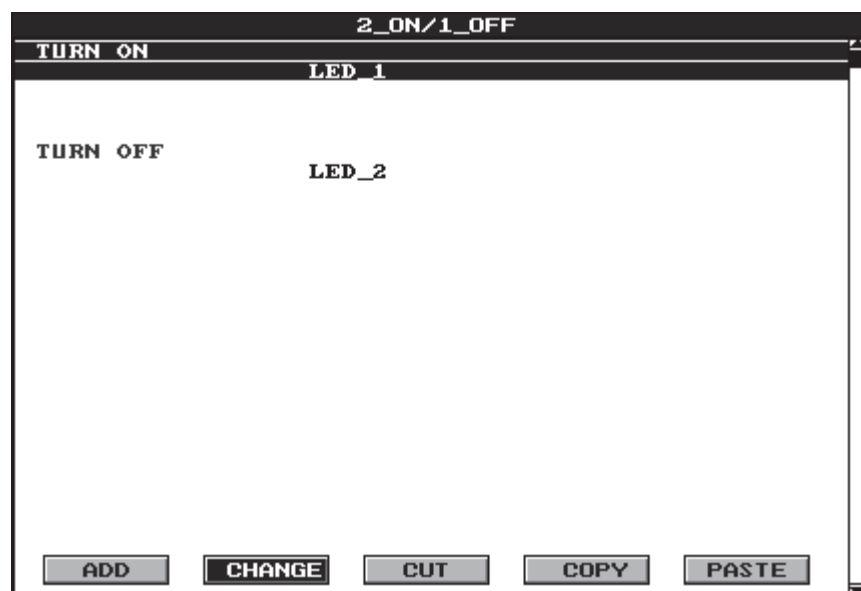


Figure 5-50: Changing the First Command

In the ITEM EDITOR dialog box, change the name of the digital point to be turned on from LED_1 to LED_2. Press ACCEPT to close the dialog box.

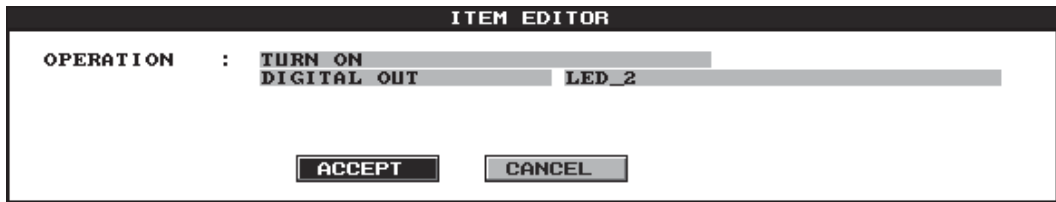


Figure 5-51: Accepting a Changed Command

Now highlight the TURN OFF command in the detail window and press ENTER while CHANGE is highlighted.

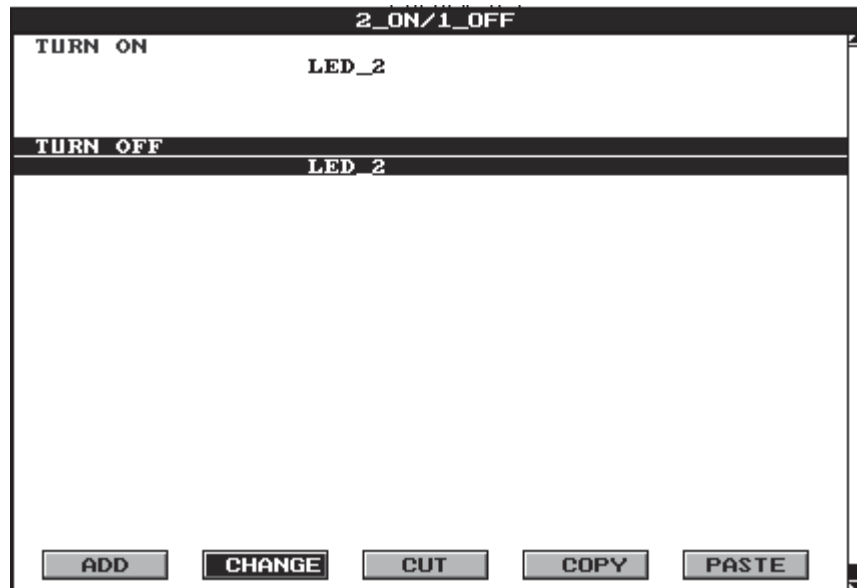


Figure 5-52: Changing the Second Command

In the ITEM EDITOR dialog box, change the point to be turned off from LED_2 to LED_1. Press ACCEPT to close the dialog box.

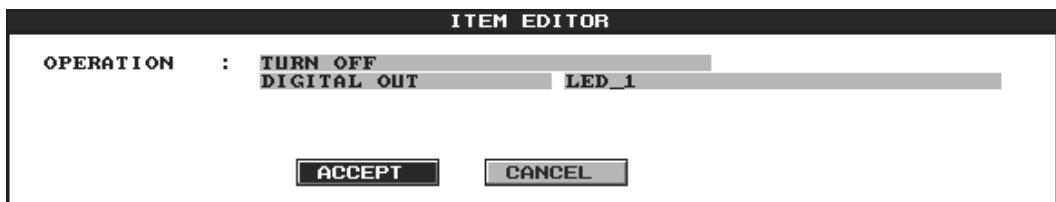


Figure 5-53: Accepting an Edited Command

Verify that LED_2 is turned on and LED_1 is turned off by the commands in the detail window. Close the window by pressing ESC.

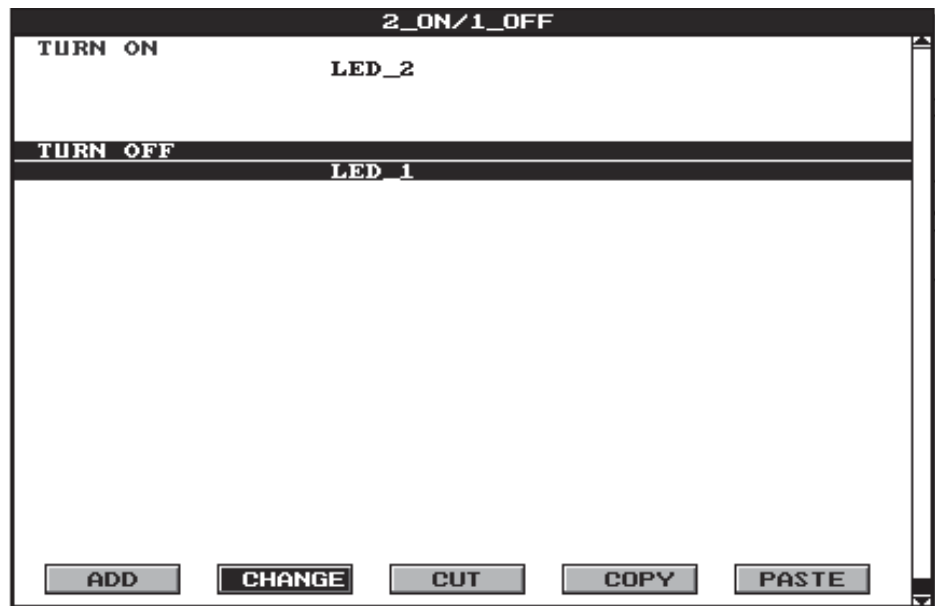


Figure 5-54: Viewing the Changed Commands

DOWNLOAD AND RUN THE PROGRAM

You are now ready to download the program to the control processor and enter the Debugger module. Select Debugger from the Configurator's System menu.

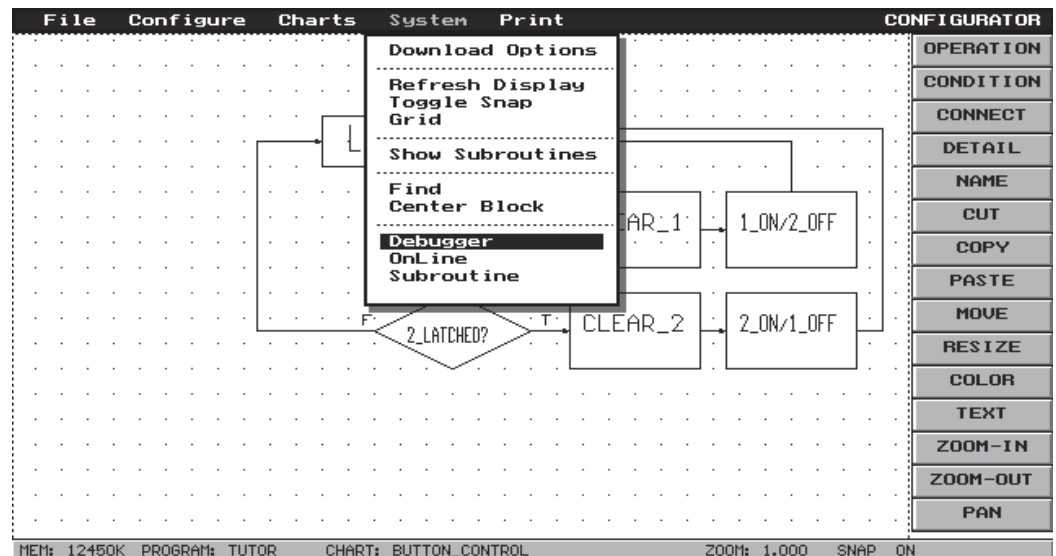


Figure 5-55: Opening the Debugger from the Configurator

A dialog box will appear to ask you if the program should be saved. Select YES to save the program before it is downloaded to the processor. The program will be saved with the current path and program name.



Figure 5-56: Confirming a Save

A DOWNLOAD WARNING dialog box will appear next, displaying information about the program you are attempting to download and the program currently running on the controller. Select YES to continue the download.

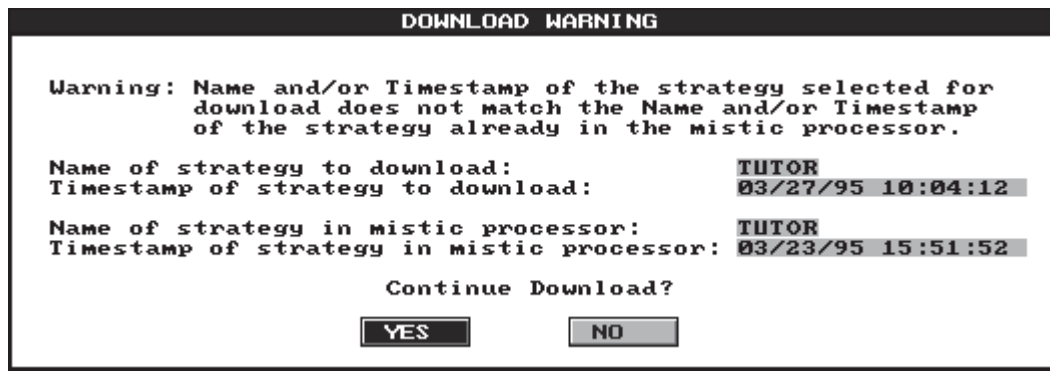


Figure 5-57: Confirming the Download

Another dialog box will indicate the amount of memory left in the controller after the program has been downloaded.



Figure 5-58: Closing Memory Information Dialog Box

Press ENTER to close this dialog box and the Debugger module will open, displaying the last chart viewed in the Configurator.

Start the program by selecting the RUN/STOP tool on the toolbar.

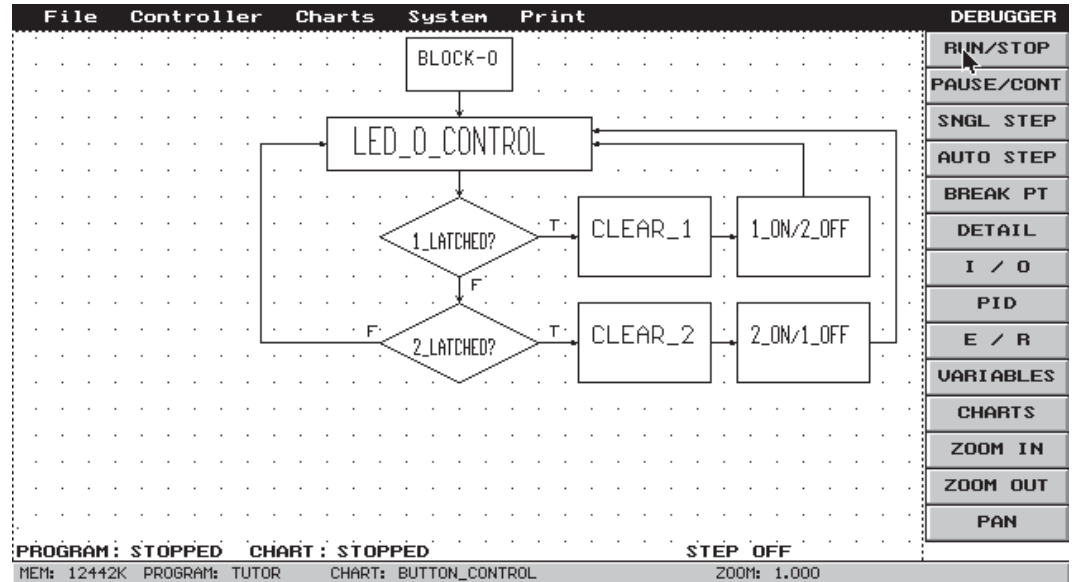


Figure 5-59: Starting a Program

TURN ON AUTO STEPPING

Turn on auto stepping by selecting the AUTO STEP tool on the toolbar. AUTO STEP is a toggle that turns auto stepping on when it is off and off when it is on. Notice that the message at the bottom of the drawing window changes from STEP OFF to STEP AUTO when auto stepping is turned on.

Watch the flow of logic through the main loop, which consists of the LED_0_CONTROL operation block and two condition blocks. See Figures 5-60 through 5-63.

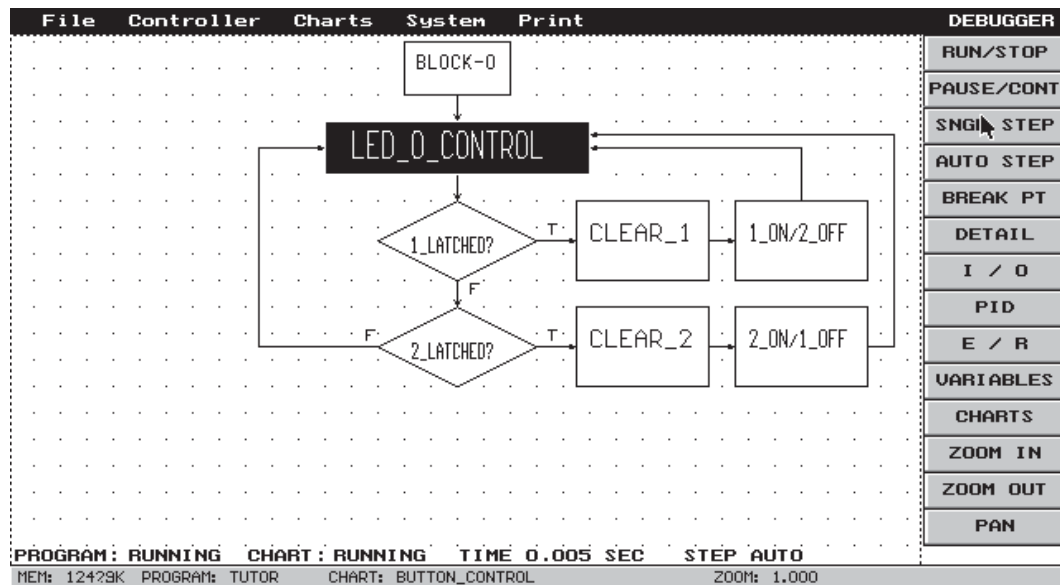


Figure 5-60: Auto Stepping Through Main Loop (1 of 4)

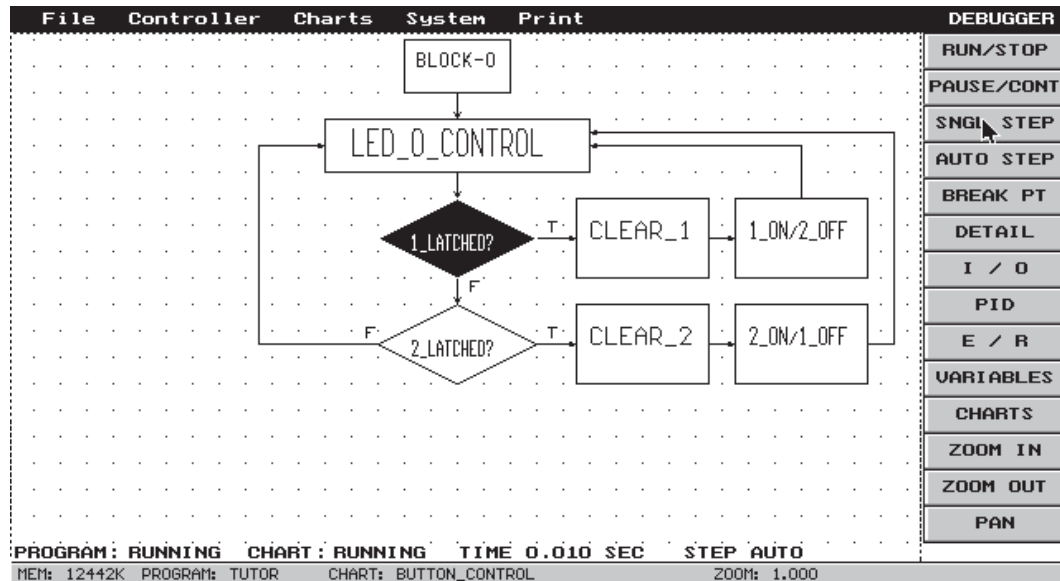


Figure 5-61: Auto Stepping Through Main Loop (2 of 4)

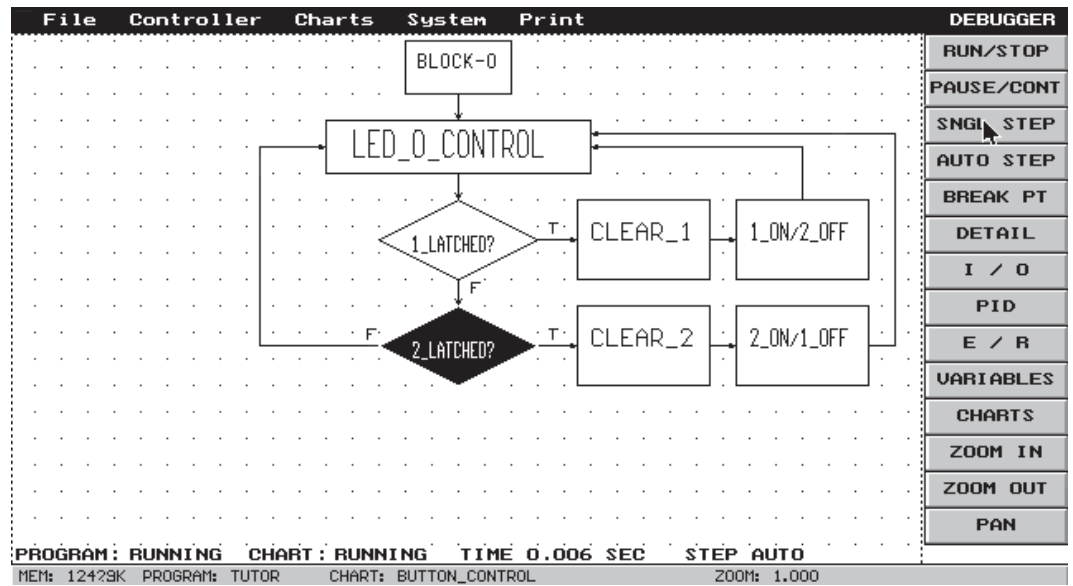


Figure 5-62: Auto Stepping Through Main Loop (3 of 4)

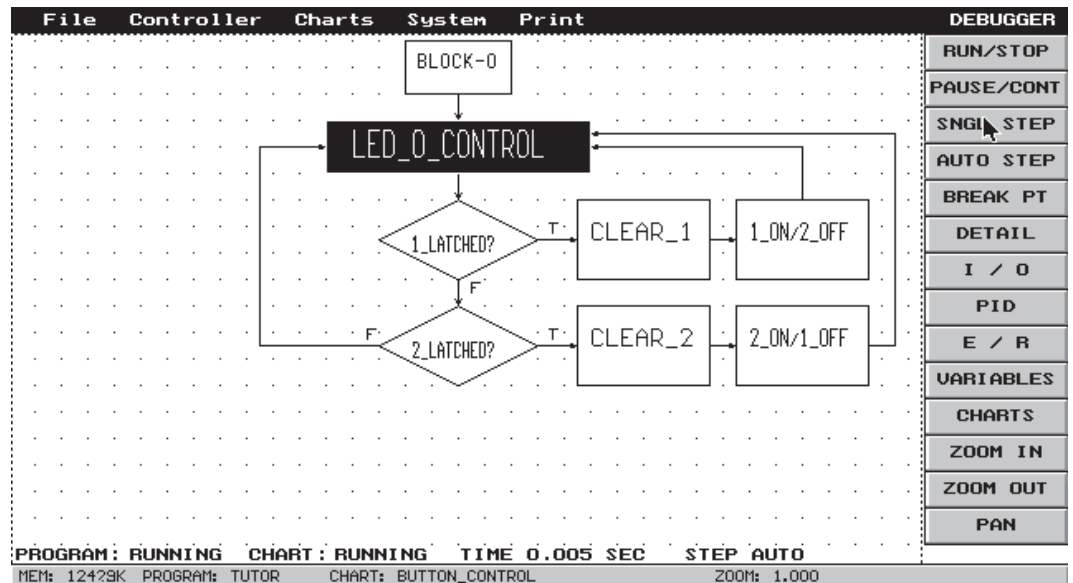


Figure 5-63: Auto Stepping Through Main Loop (4 of 4)

Push Button 1 on the demo box and observe the path of the logic change. This will set latch 1, so control will flow through the True exit of the first condition block. The logic path will take this loop once every time Button 1 is pressed. Remember that the latch is cleared in the first operation block of this loop. Notice that LED 1 will turn on and LED 2 will turn off.

See Figures 5-64 through 5-67.

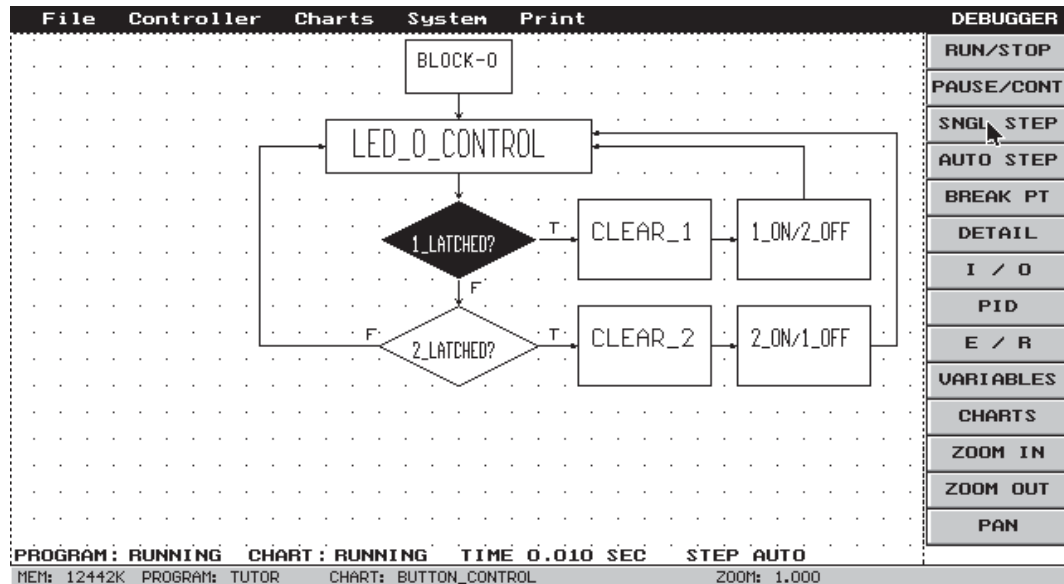


Figure 5-64 Auto Stepping Through BUTTON_1 Loop (1 of 4)

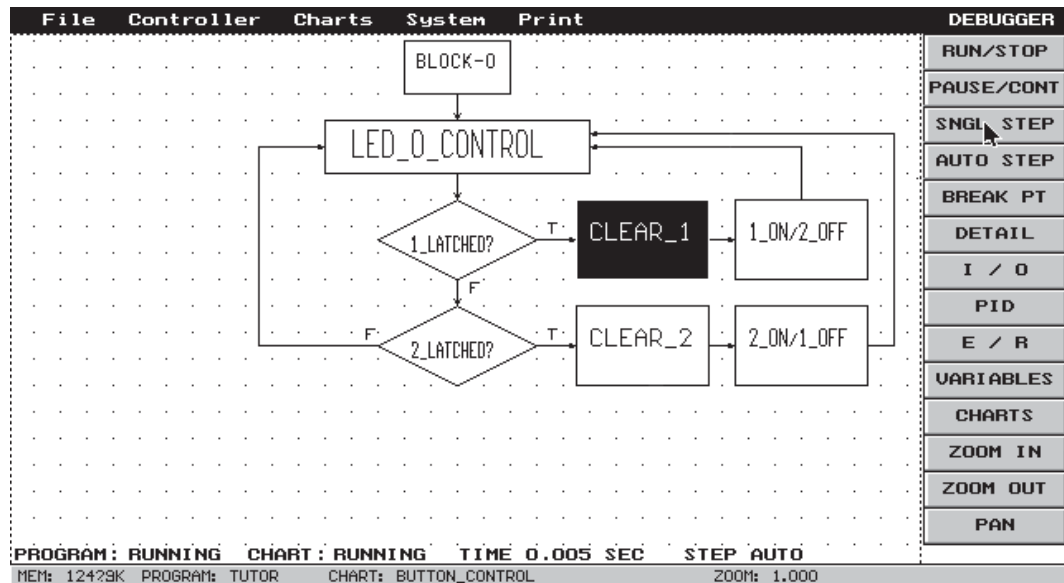


Figure 5-65: Auto Stepping Through BUTTON_1 Loop (2 of 4)

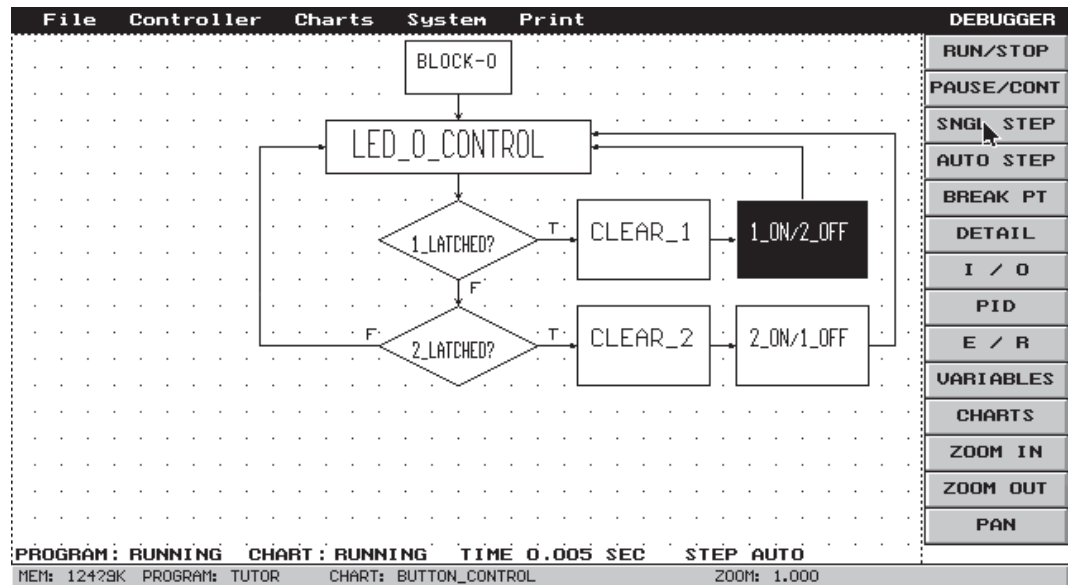


Figure 5-66: Auto Stepping Through BUTTON_1 Loop (3 of 4)

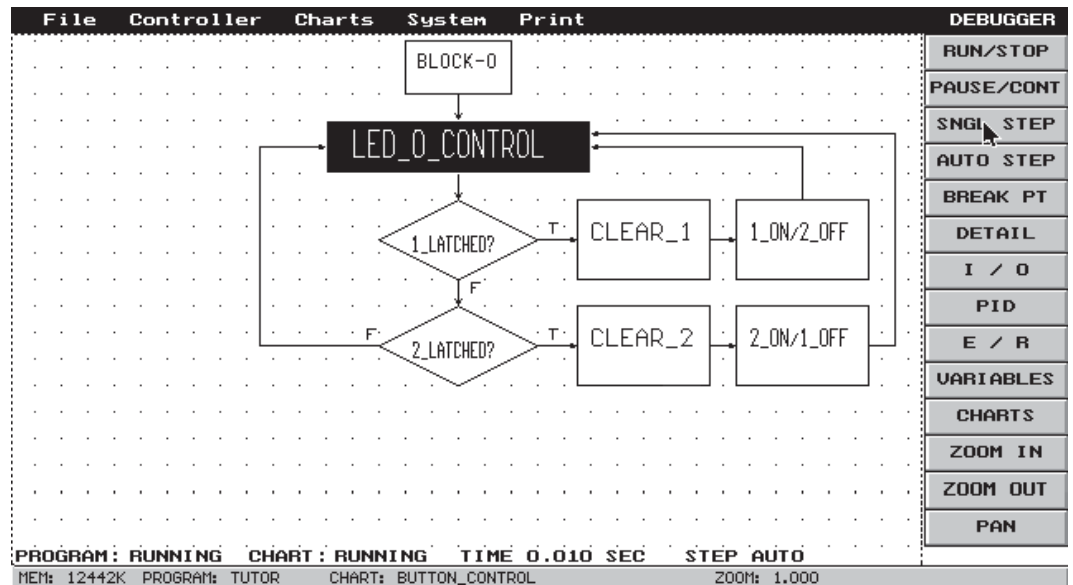


Figure 5-67: Auto Stepping Through BUTTON_1 Loop (4 of 4)

Push Button 2 on the demo box. This will set latch 2, so control will flow through the True exit of the second condition block. The logic path will take this loop once every time Button 2 is pressed. Notice that LED 2 will turn on and LED 1 will turn off.

See Figures 5-68 through 5-72.

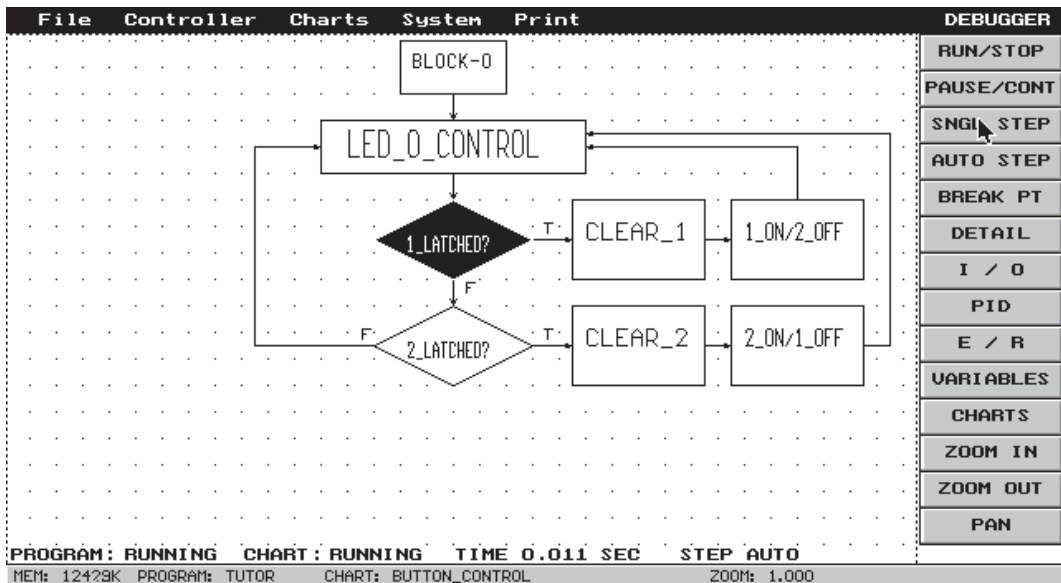


Figure 5-68: Auto Stepping Through BUTTON_2 Loop (1 of 5)

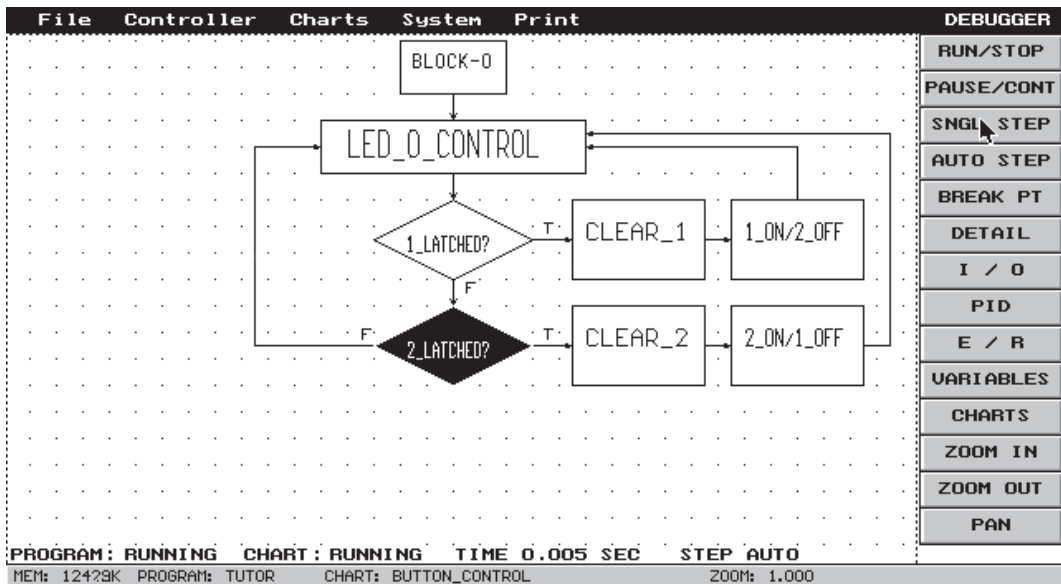


Figure 5-69: Auto Stepping Through BUTTON_2 Loop (2 of 5)

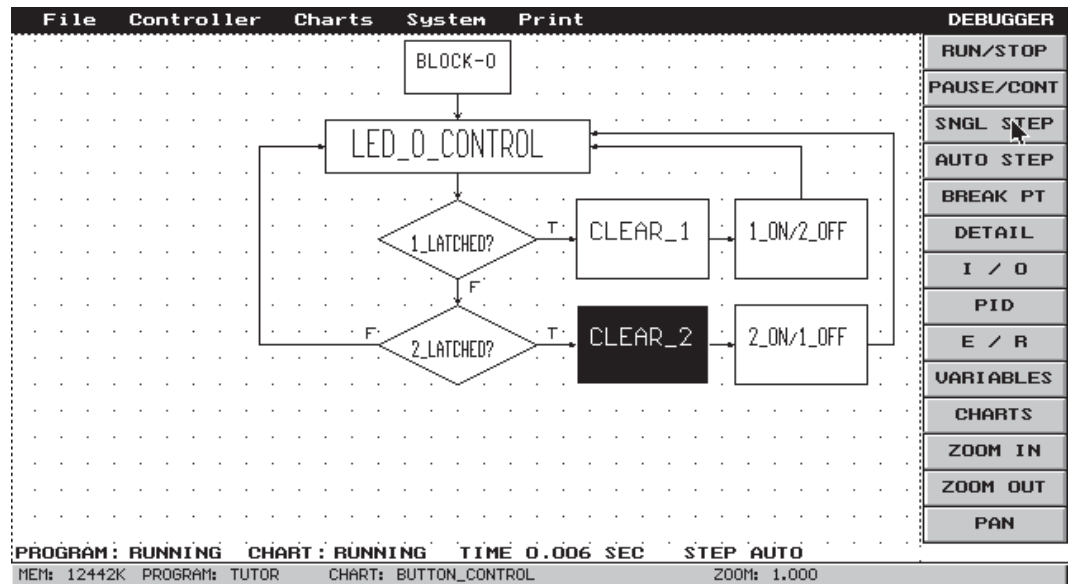


Figure 5-70: Auto Stepping Through BUTTON_1 Loop (3 of 5)

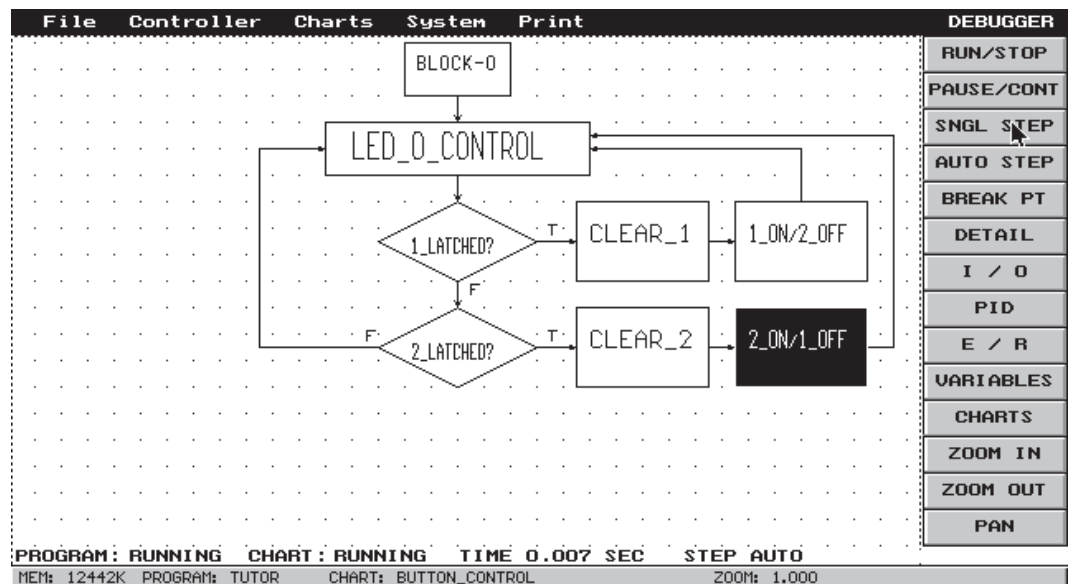


Figure 5-71: Auto Stepping Through BUTTON_1 Loop (4 of 5)

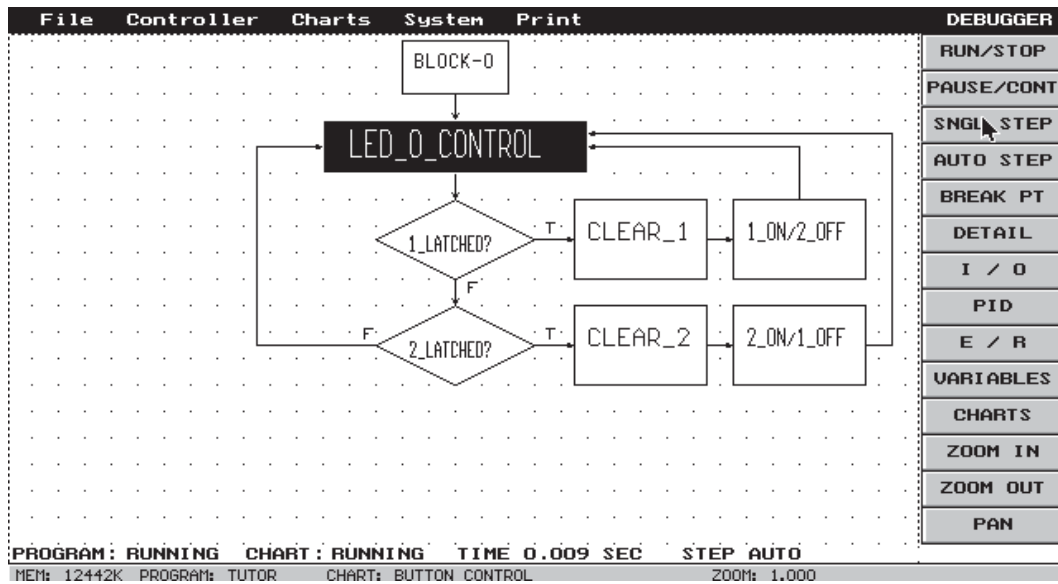


Figure 5-72: Auto Stepping Through BUTTON_1 Loop (5 of 5)

OBSERVE ITEMS IN THE WATCH WINDOW

The watch window allows you to monitor the values of variables or I/O points in your program. Open the watch window by selecting Open Watch Window from the Controller menu.

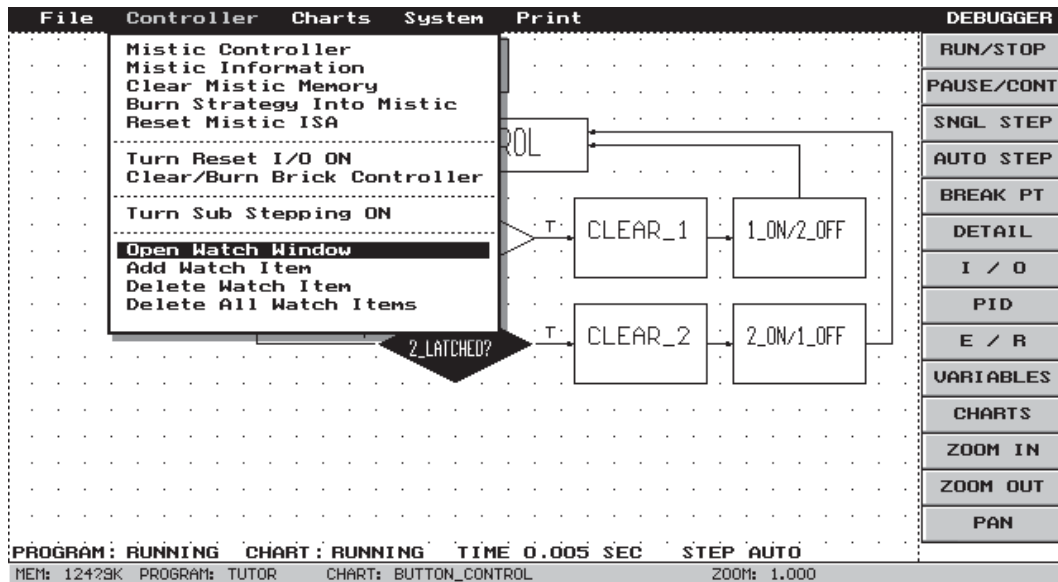


Figure 5-73: Opening the Watch Window

A dialog box will notify you that no watch window items have been defined. Press any key to continue.



Figure 5-74: Closing the No Watch Items Defined Information Dialog Box

Add an item to the watch window by selecting Add Watch Item from the Controller menu.

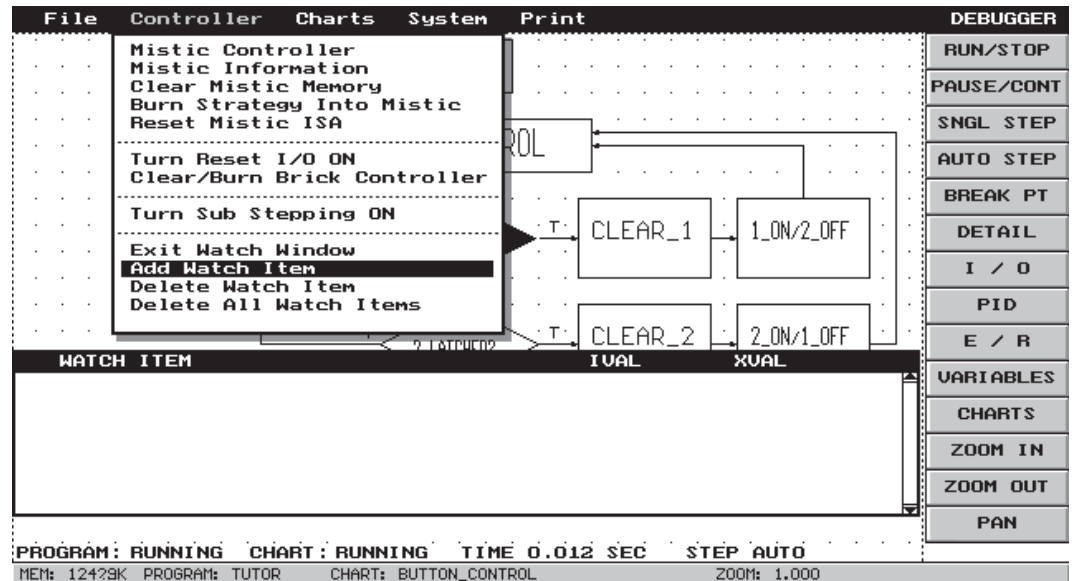


Figure 5-75: Adding an Item to the Watch Window

A dialog box will appear, listing the types of items that can be monitored in the watch window. Select I/O from the list.



Figure 5-76: Selecting a Watch Window Item Type

Use the up or down arrow keys to select the brick where the I/O point is configured. In this case, highlight LEFT_DIGITAL_BRICK and press ENTER.

SELECT I/O UNIT			
NAME	TYPE	PORT	ADDRESS
ANALOG_BRICK	ANALOG MF	REMOTE 2	002
CENTER_DIGITAL_BRICK	DIGITAL MF	REMOTE 2	001
LEFT_DIGITAL_BRICK	DIGITAL MF	REMOTE 2	000

Figure 5-77: Selecting an I/O Unit

The SELECT I/O POINT dialog box will appear. Use the up or down arrow keys to select the point BUTTON_1 from the list of configured I/O points.

SELECT I/O POINT		
UNIT	LEFT_DIGITAL_BRICK	
ADDRESS	000	REMOTE 2
CH#	NAME	TYPE
00	BUTTON_0	DIN
01	BUTTON_1	DIN
02	BUTTON_2	DIN
03	UNUSED	
04	UNUSED	
05	UNUSED	
06	UNUSED	
07	UNUSED	
08	LED_0	DOUT
09	LED_1	DOUT
10	LED_2	DOUT
11	UNUSED	
12	UNUSED	
13	UNUSED	
14	UNUSED	
15	UNUSED	

Figure 5-78: Selecting an I/O Point for the Watch Window

BUTTON_1 is configured as a latch. The state of either the button (On or Off) or the latch (True or False) can be monitored. To monitor the button state, highlight STATE in the SELECT WATCH ITEM dialog box and press ENTER.

SELECT WATCH ITEM	
STATE	
ON LATCH	
<input type="button" value="SELECT"/>	<input type="button" value="EXIT"/>

Figure 5-79: Selecting the Attribute to Be Watched

Select Add Watch Item from the Controller menu. Following the same procedure used to add the state of BUTTON_1 to the watch window, add the state of the BUTTON_1 latch and the state of the digital output point LED_1 to the watch window.

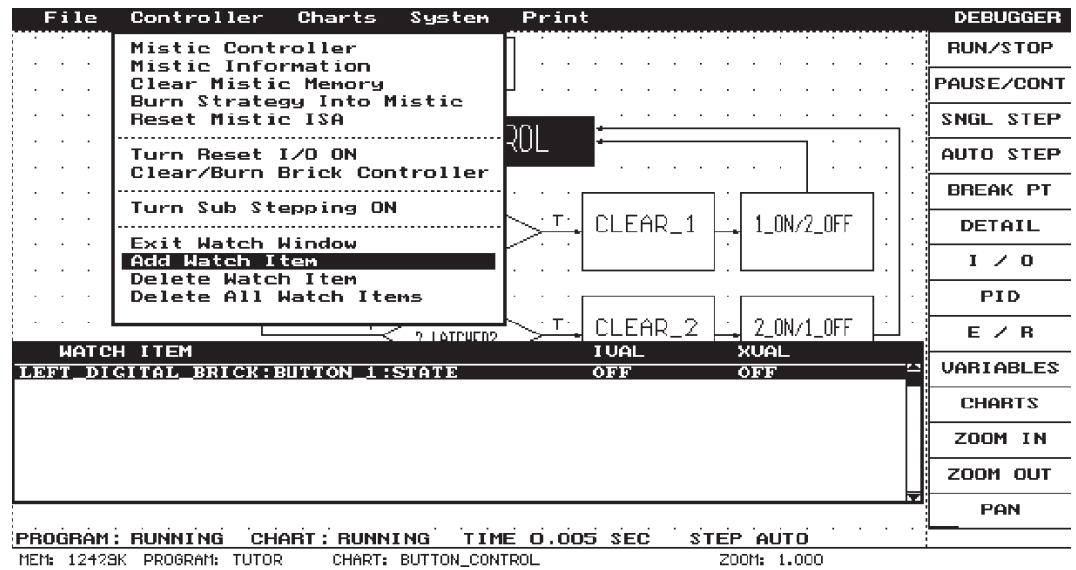


Figure 5-80 Adding a New Watch Window Item

With auto stepping on, press the PAUSE/CONT tool on the toolbar. PAUSE/CONT is a toggle that pauses or continues a program. After the program is paused, push the SNGL STEP tool on the toolbar. Notice that the program advances one block every time SNGL STEP is pressed. The program pauses just *before* executing the commands in the highlighted block.

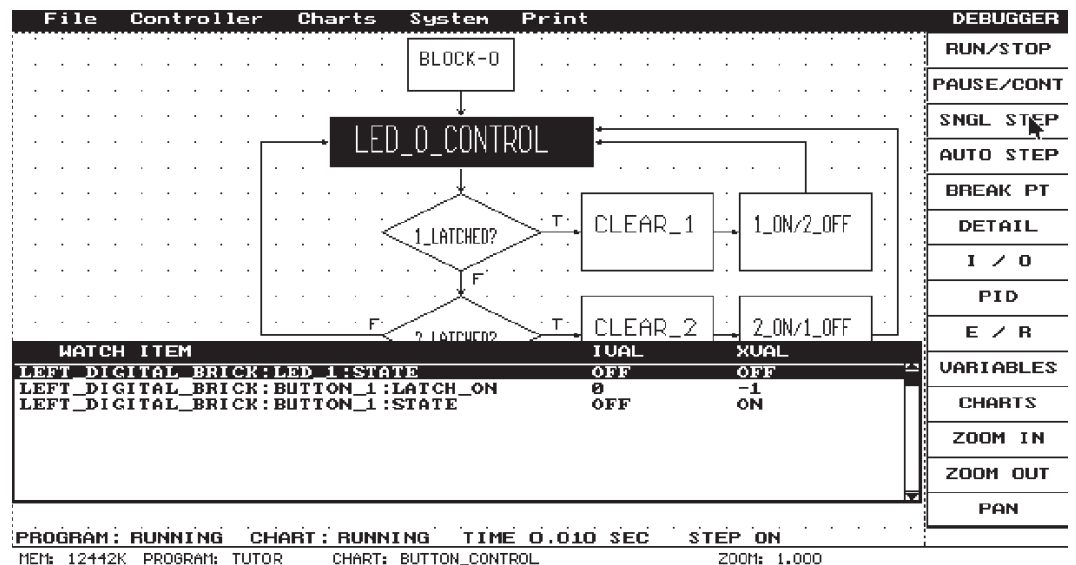


Figure 5-81: Viewing Items in the Watch Window

Pause the program at the LED_0_CONTROL block, then press and hold Button 1 on the demo box. Notice that the hardware value (XVAL) of the button state changes to On and the latch is set to True (-1). Remember, when the input point is referenced in the Cyrano program, the XVAL (hardware value) is copied to the IVAL (software value).

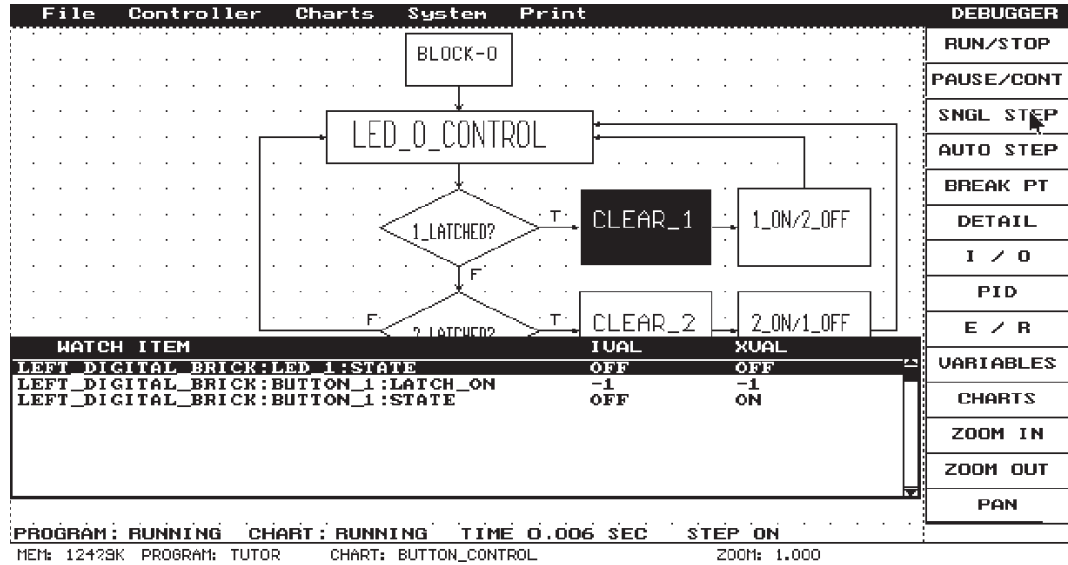


Figure 5-82: Viewing Items in the Watch Window

Press SNGL STEP twice to advance to the CLEAR_1 operation block. The commands in the 1_LATCHED? condition block have been evaluated. Since the latch has been referenced, the True (-1) XVAL is copied to the IVAL. Release Button 1 on the demo box and notice that the state of BUTTON_1 returns to Off.

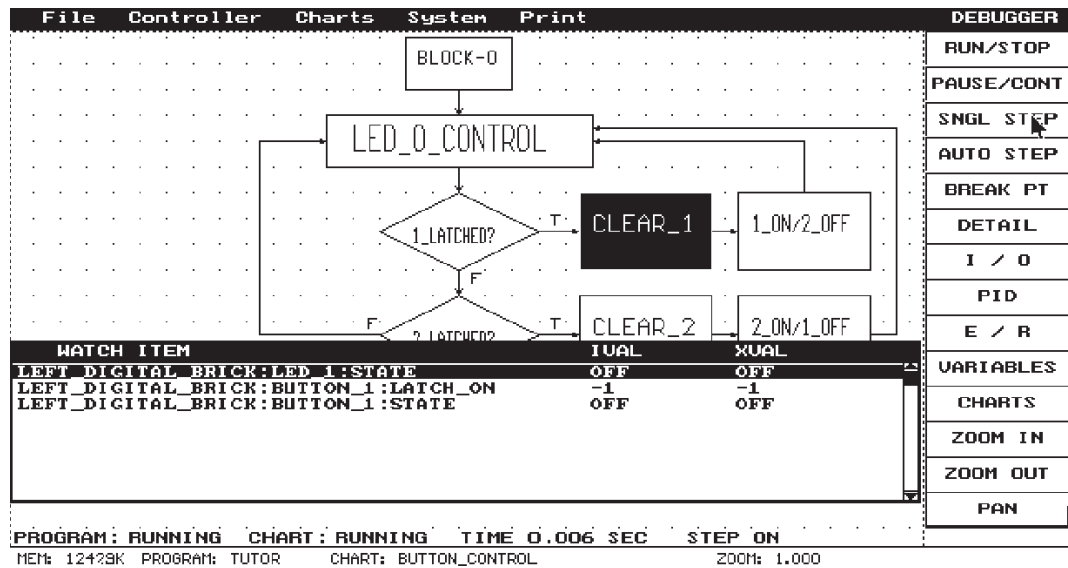


Figure 5-83: Viewing Items in the Watch Window

Press SNGL STEP to advance to the next operation block. The commands in the CLEAR_1 operation block have now been executed. Notice that the hardware value of the latch has been reset to False (0).

Figure 5-84: Viewing Items in the Watch Window

Press SNGL STEP again to advance to the LED_O_CONTROL block. The commands in the 1_ON/2_OFF block have now been executed. Notice that LED 1 on the demo box has been turned On and that the state of LED_1 is displayed as ON in the watch window. The IVAL and XVAL are equal because the point LED_1 was referenced in the Cyrano program.

Figure 5-85: Viewing Items in the Watch Window

Advance the program to the first condition block. Notice that the software (IVAL) value of the latch is still True (-1).

The screenshot shows a debugger interface with a ladder logic chart and a watch window. The chart starts with a 'BLOCK-0' box, followed by an 'LED_O_CONTROL' block. Below this is a diamond-shaped condition '1_LATCHED?'. From the 'T' (True) side of the diamond, the flow goes to a 'CLEAR_1' block, then to a '1_ON/2_OFF' block, and loops back to the 'LED_O_CONTROL' block. From the 'F' (False) side of the diamond, the flow goes to a '2_LATCHED?' condition. From the 'T' side of '2_LATCHED?', it goes to a 'CLEAR_2' block, then to a '2_ON/1_OFF' block, and loops back to the 'LED_O_CONTROL' block. The watch window at the bottom shows the following data:

WATCH ITEM	IVAL	XVAL
LEFT_DIGITAL_BRICK:LED_1:STATE	ON	ON
LEFT_DIGITAL_BRICK:BUTTON_1:LATCH_ON	-1	0
LEFT_DIGITAL_BRICK:BUTTON_1:STATE	OFF	OFF

At the bottom of the debugger, it says 'PROGRAM: RUNNING CHART: RUNNING TIME 0.010 SEC STEP ON'. The status bar shows 'MEM: 12442K PROGRAM: TUTOR CHART: BUTTON_CONTROL ZOOM: 1.000'.

Figure 5-86: Viewing Items in the Watch Window

Advance the program one block so that the point BUTTON_1 is referenced. Notice that the software (IVAL) value of the latch is set equal to the hardware (XVAL) value, since the point was referenced in the previous 1_LATCHED? condition block.

This screenshot is similar to Figure 5-86, but the program has advanced. The watch window now shows the following data:

WATCH ITEM	IVAL	XVAL
LEFT_DIGITAL_BRICK:LED_1:STATE	ON	ON
LEFT_DIGITAL_BRICK:BUTTON_1:LATCH_ON	0	0
LEFT_DIGITAL_BRICK:BUTTON_1:STATE	OFF	OFF

The debugger status at the bottom now shows 'PROGRAM: RUNNING CHART: RUNNING TIME 0.005 SEC STEP ON'. The status bar remains 'MEM: 12442K PROGRAM: TUTOR CHART: BUTTON_CONTROL ZOOM: 1.000'.

Figure 5-87: Viewing Items in the Watch Window

Close the watch window by selecting Exit Watch Window from the Controller menu.

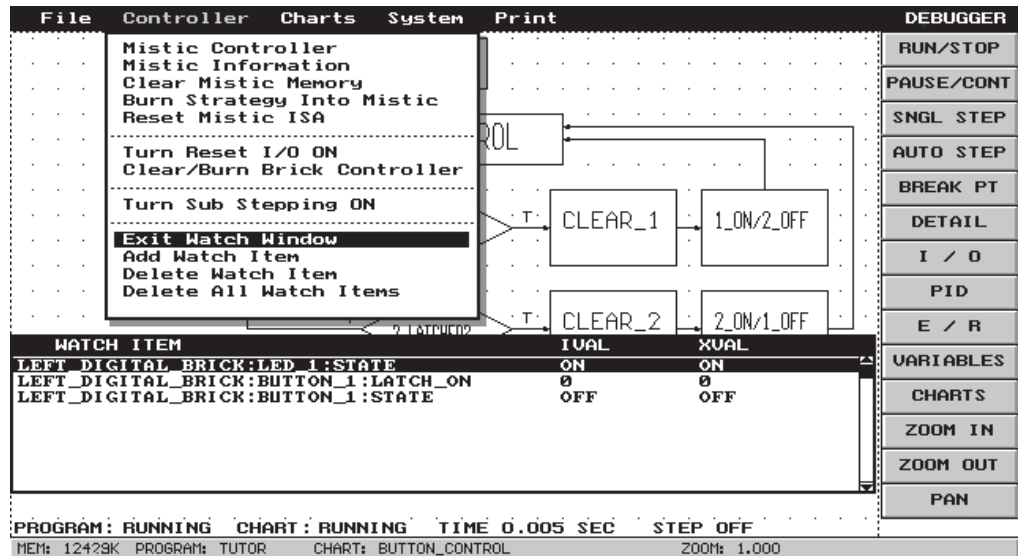


Figure 5-88 Exiting the Watch Window

Return to the Configurator module by selecting Configurator from the System menu.

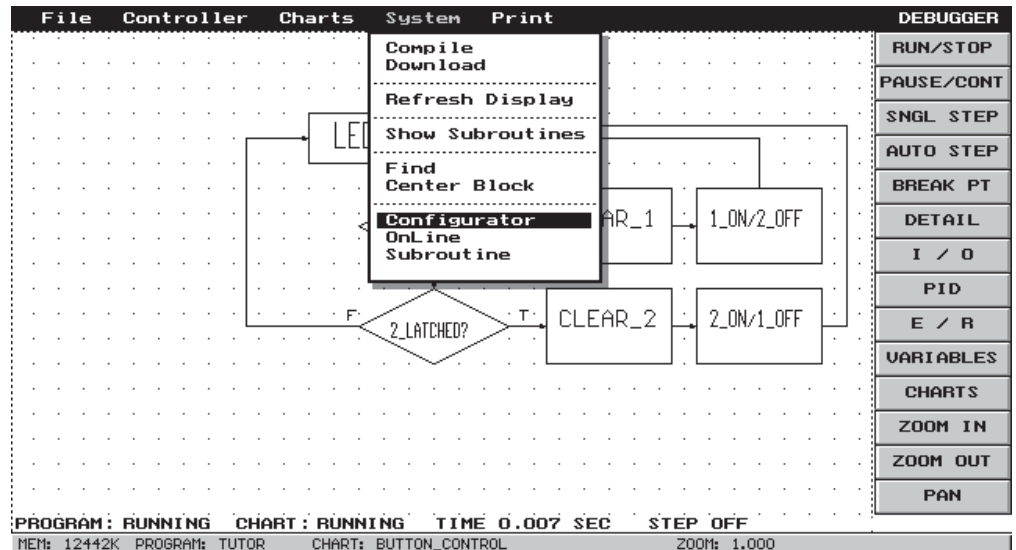


Figure 5-89: Returning to the Configurator from the Debugger

A dialog box will appear to ask you to confirm your exit from the Debugger. Select YES.

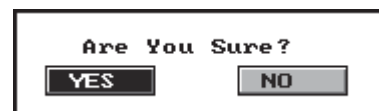


Figure 5-90: Confirming Debugger Exit

WHAT DID I DO?

In this chapter you added complexity to a simple chart by adding two new condition blocks and four new operation blocks. You learned how to redraw a chart strategy, add multiple commands to a block, configure an I/O point on the fly, copy and paste command blocks, and modify commands. After downloading the program, you used the auto step and watch window features to see what actions were taken and how variables changed as the program ran.

WHERE DO I GO FROM HERE?

You are ready to take advantage of Cyrano's multitasking capability by creating a second chart. This chart will monitor analog values at predetermined time intervals, use mathematical functions to manipulate the values, convert the results into a string, and print the string to the demo box display panel. All of this will occur while the chart created in this chapter runs concurrently.

Proceed to Chapter 6.

CREATING A SECOND CHART

CHAPTER

6

WHAT TOPICS WILL I COVER?

- Displaying analog values on a meter
- Using a timer
- Copying a command
- Using mathematical functions
- Converting numbers to strings
- Printing to a display panel

WHAT WILL I DO?

- Create a new chart called ANALOG to display the values of the top and bottom potentiometers on the top and bottom meters.
- Simulate temperature sensors using the top and bottom potentiometers.
- Add commands to calculate the average value for the sensors every second and display this value on the first line of the demo box display panel.
- Add a command to the POWERUP chart to run this new chart when you start the program.
- Download and run the new program.

HOW WILL I DO IT?

CREATE A NEW CHART

Return to the Configurator module. Verify that the program name listed in the status bar at the bottom of the window is TUTOR.

Select New from the Charts menu, type in the chart name ANALOG, and press ENTER. This will create a new chart called ANALOG.

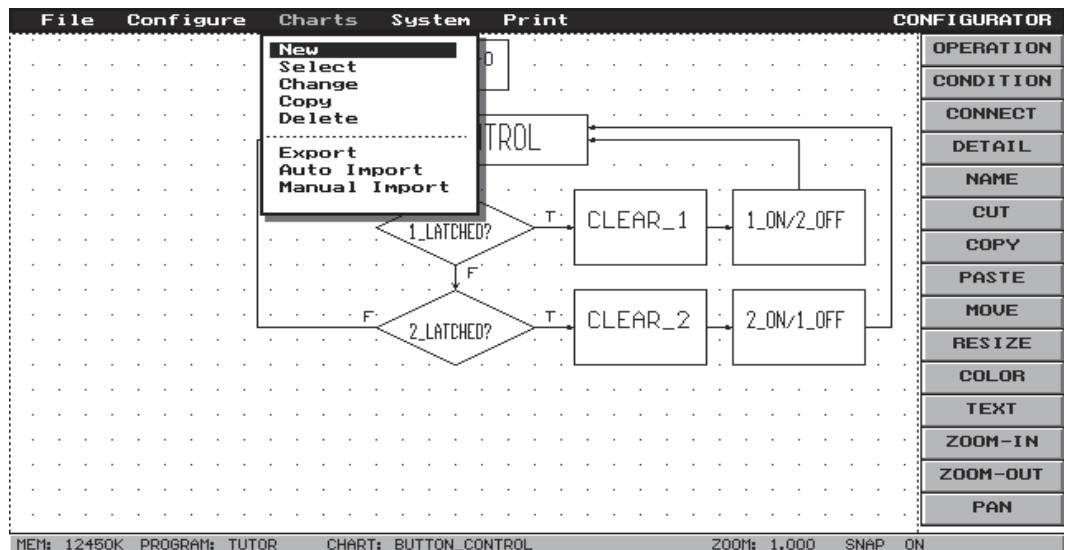


Figure 6-1: Creating a New Chart

DRAW THE FLOWCHART LOGIC

Use the OPERATION and CONDITION tools from the toolbar to create the flowchart structure shown in Figure 6-2. Connect the blocks using the CONNECT tool and name the blocks using the NAME tool.

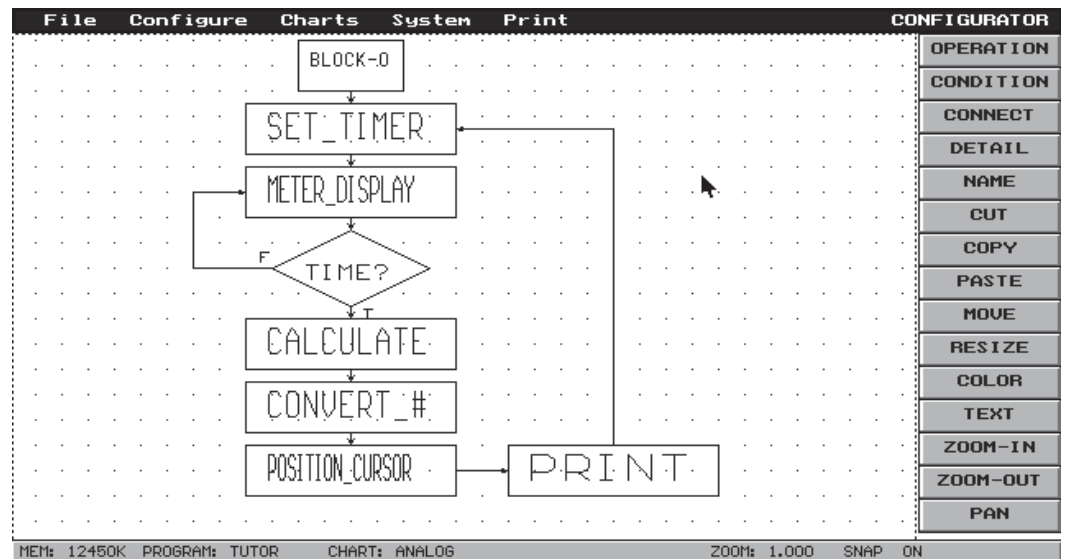


Figure 6-2: Drawing the Flowchart

The chart's first operation block will start a timer with a value of one second. The next operation block will display the values of the two sensors on the two meters. The condition block will check to see if the timer has expired. Until the one-second timer value has elapsed, the program will continually loop on the block

that displays the values of the sensors. Once the timer has expired, the program will calculate the average value of the two sensors and convert the average value from a number to a string. The last two operation blocks in the loop will clear the first line of the display and print the string representing the average value of the sensors.

ENTER COMMANDS IN THE BLOCKS

SET_TIMER

Open the detail window for the operation block SET_TIMER. Select ADD to create a command that starts a timer with a value of one second. To start a timer, simply move a numeric value into a variable defined as a Timer type, as shown in Figure 6-3.



Figure 6-3: Accepting a Completed Command

Use the MOVE command to start a timer named TIMER with a value of one second. If the interval of the timer is always going to be one second, use a constant float to set this value. To allow the timer to be set to different intervals based on other conditions in the program, a variable float (or variable integer) would have to be used. Since timers store values as floats, it is best to use a floating point variable rather than an integer variable to start a timer. When speed is important, this practice eliminates needless data conversions (from integer to float) and cuts program execution time.

No timers have been defined in the Cyrano program. After typing TIMER as the name of the variable timer, a dialog box will appear to ask you if you would like to define this variable. Select YES.



Figure 6-4: Confirming New Variable Creation

Add a timer variable named TIMER that is initialized to a value of 0. Notice that the default values of the fields are all correct.

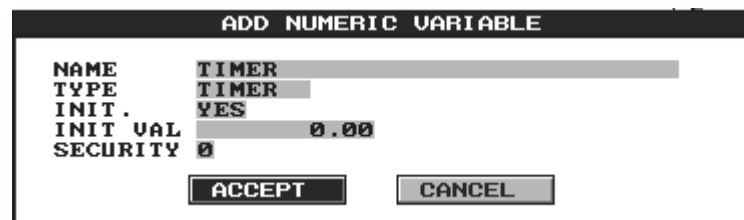


Figure 6-5: Defining a New Variable

Press enter once while ACCEPT is highlighted in the ADD NUMERIC VARIABLE dialog box to accept the variable and again in the ITEM EDITOR dialog box to accept the command. The new command will appear in the SET_TIMER detail window. Press ESC or the right mouse button to close this window.

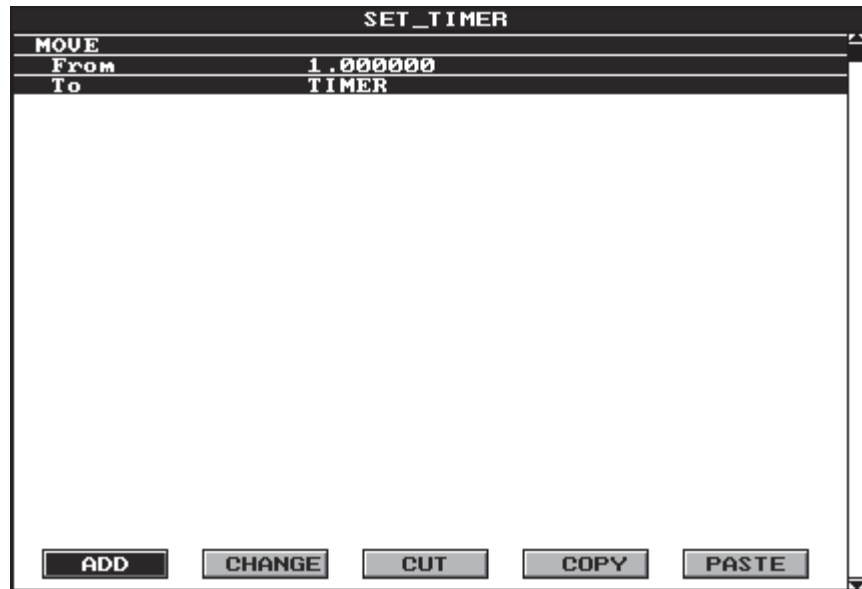


Figure 6-6: Viewing the Detail Window

METER_DISPLAY

Open the detail window for the METER_DISPLAY operation block. Add two MOVE commands to copy the value of the input points configured as temperature sensors to the output points wired to the meters. Display the value of TEMPERATURE_SENSOR_#1 on the TOP_METER and the value of TEMPERATURE_SENSOR_#2 on the BOTTOM_METER. See Figures 6-7 and 6-8 for command details.

Since the four I/O points have already been configured in Chapter 3, the names of these points can be selected from lists by pressing ENTER without typing any characters in the field to the right of each variable type.

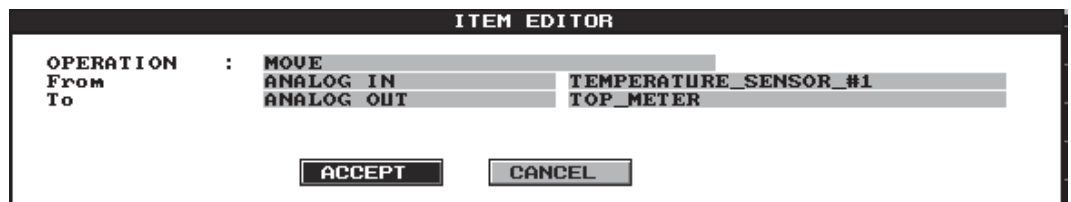


Figure 6-7: Accepting a Completed Command



Figure 6-8: Accepting a Completed Command

After the two commands have been created, they will appear in the METER_DISPLAY detail window. Close this window by pressing ESC or the right mouse button.

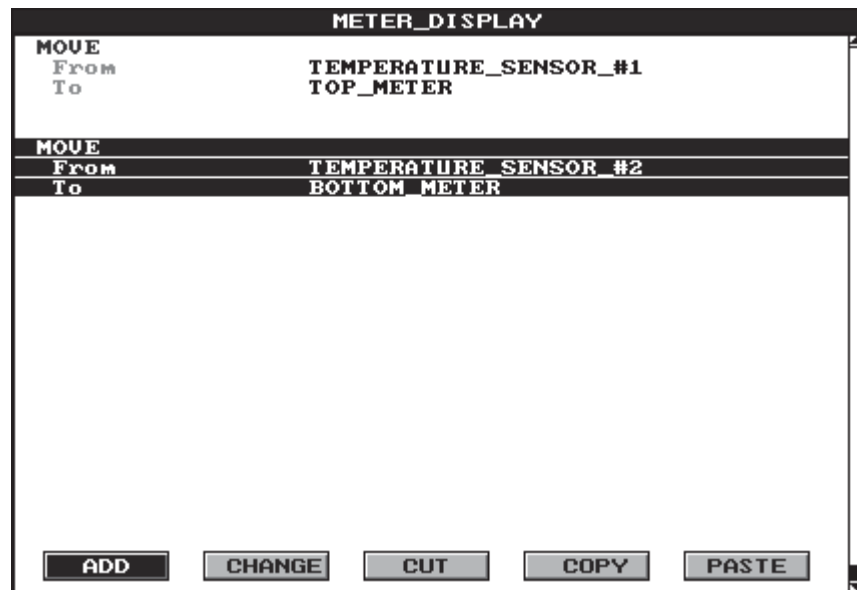


Figure 6-9: Viewing the Detail Window

TIME?

In the TIME? condition block, enter a command that checks the value of the TIMER. The command TIMER EXPIRED can be entered by typing the first few letters, such as TIM, and pressing ENTER. A list of commands matching the sequence you have typed will then appear. Select the command from this list. In this case, only one condition begins with TIM.

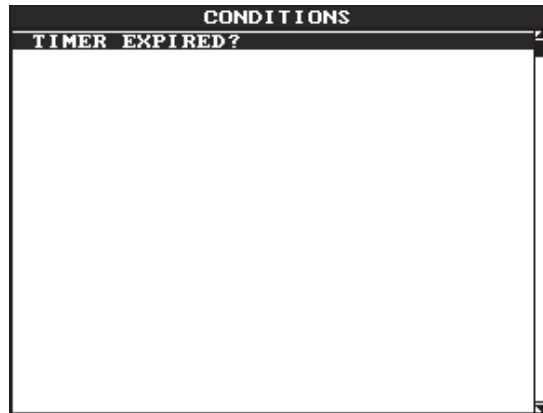


Figure 6-10: Selecting a Condition

The variable TIMER has already been defined and can be chosen from a list by pressing ENTER without typing any characters in the name field (to the right of the variable type field).

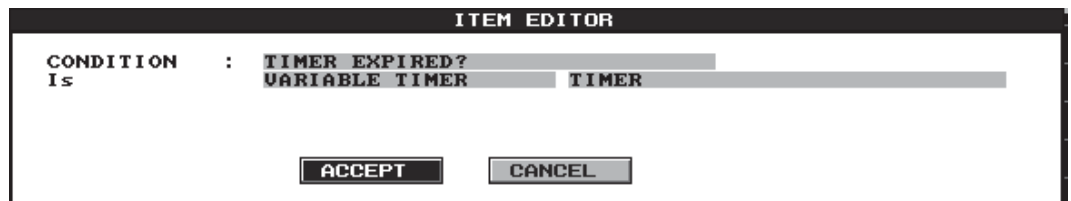


Figure 6-11: Accepting a Completed Command

Accept the new command, verify that the command is correct in the detail window, then close this window.



Figure 6-12: Viewing the Detail Window

CALCULATE

Enter a command in the CALCULATE operation block to add the two sensor values together and store them in a variable float called TOTAL.

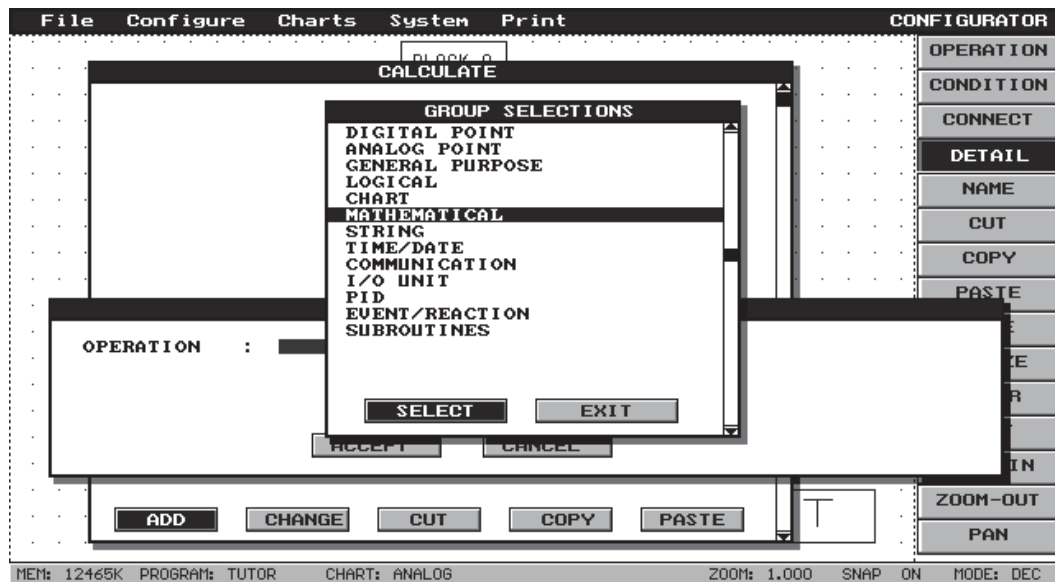


Figure 6-13: Selecting a Group

The command, DO ADDITION, can be found in the Mathematical group.

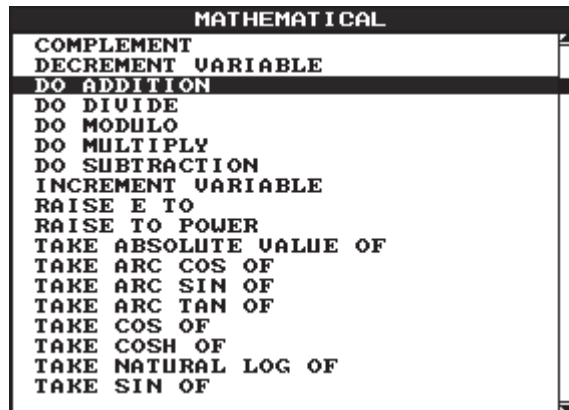


Figure 6-14: Selecting an Operation

Complete the information in the ITEM EDITOR dialog box, as shown below.

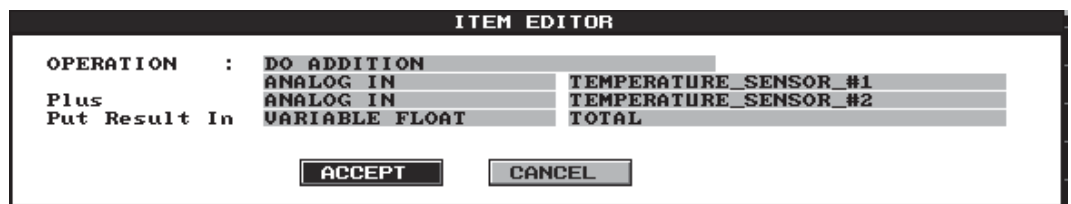


Figure 6-15: Accepting a Completed Command

The variable TOTAL has not been defined in the program. After you have typed TOTAL in the name field for the *Put Result In* parameter, a dialog box will appear to allow you to define this variable.



Figure 6-16: Confirming New Variable Creation

Select YES to open the ADD NUMERIC VARIABLE dialog box. Accept the defaults as shown in Figure 6-17.

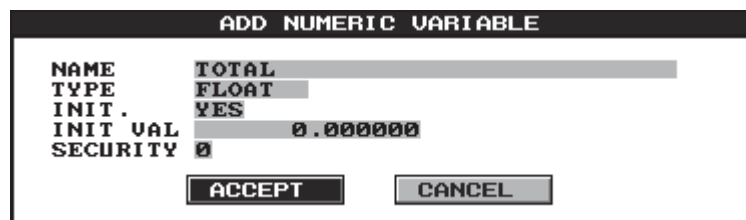


Figure 6-17: Defining a New Variable

After accepting the new variable and the new command, verify that the command is correct in the detail window. Then select ADD from the CALCULATE dialog box to create a second command.

This command, DO DIVIDE, will divide TOTAL by a constant float equal to 2.0000, storing the result in a variable float called AVERAGE. Notice that all the variables are of the same type (float). It is good programming practice when doing mathematical operations to keep all relevant variables the same type. While mixing of types is allowed, unnecessary data conversions waste processor time. Also, mixed-type calculations can produce unexpected results.

Create the new command as shown in Figure 6-18.

The screenshot shows a dialog box titled "ITEM EDITOR". It contains the following text:

```

OPERATION      : DO DIVIDE
By             : VARIABLE FLOAT      TOTAL
Put Result In : CONSTANT FLOAT     2.000000
               : VARIABLE FLOAT      AVERAGE
  
```

At the bottom of the dialog box are two buttons: "ACCEPT" and "CANCEL".

Figure 6-18: Accepting a Completed Command

Note that the variable AVERAGE must be added to the program. Type the name AVERAGE in the name field for the *Put Result In* parameter and press ENTER. You will again be asked if you want to define a new variable. Select YES.

The screenshot shows a dialog box with the text "Do you want to define one?" and two buttons: "YES" and "NO".

Figure 6-19: Confirming New Variable Creation

Enter the required information in the ADD NUMERIC VARIABLE dialog box, as shown below. Complete the variable definition by selecting ACCEPT.

The screenshot shows a dialog box titled "ADD NUMERIC VARIABLE". It contains the following text:

```

NAME          : AVERAGE
TYPE          : FLOAT
INIT.         : YES
INIT VAL     : 0.000000
SECURITY     : 0
  
```

At the bottom of the dialog box are two buttons: "ACCEPT" and "CANCEL".

Figure 6-20: Defining a New Variable

Complete the new command by selecting ACCEPT in the ITEM EDITOR dialog box.

The two new commands will appear in the CALCULATE detail window. Verify these commands before closing the window.

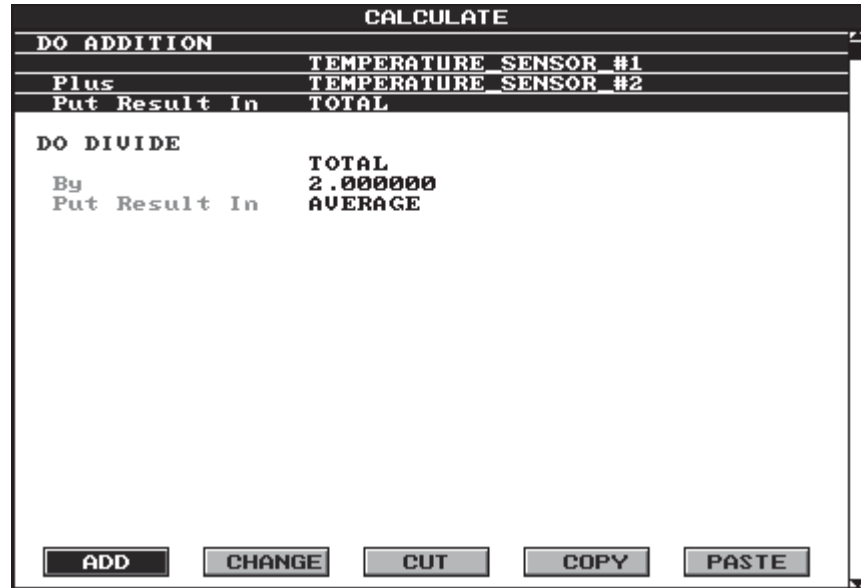


Figure 6-21: Viewing the Detail Window

CONVERT_#

Open the detail window for the CONVERT_# operation block. We will add a command to convert the numeric variable AVERAGE into a string variable so that it can be displayed. This is necessary because *strings are the only variable type that can be sent to a display screen*. You cannot directly print any number before converting it to a string.

The CONV. FLOATING POINT # TO STR. command converts numbers to strings and allows you to format the result. This formatting includes specifying the total number of characters and the number of decimal places.

From the detail window, select ADD. In the ITEM EDITOR dialog box's OPERATION field, type CONV and press ENTER to view a list of Cyrano's conversion operations.

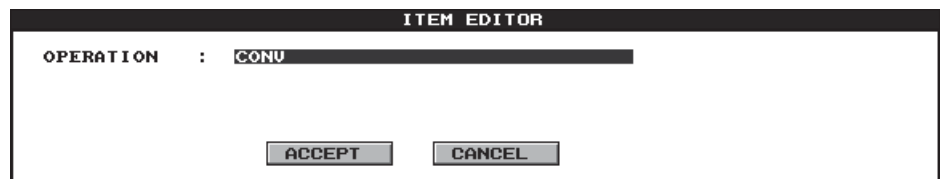


Figure 6-22: Entering an Operation

Select CONV. FLOATING POINT # TO STR. from the list.

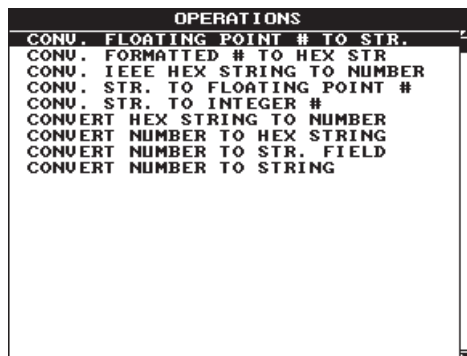


Figure 6-22: Selecting an Operation

Complete the information in the ITEM EDITOR dialog box to convert the floating point variable into a string displaying a total of six characters with two decimal places. The length of the string should always be defined with enough characters to hold the largest possible value, with the decimal point counting as one character. In this case, the largest number this variable will hold is 150.00, which contains six characters. Numbers are truncated to the number of decimal places specified and are right justified.

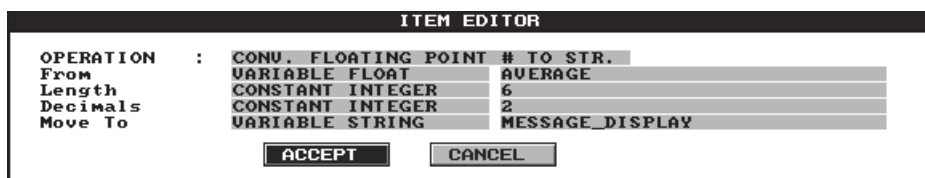


Figure 6-24: Defining a Command

The string variable MESSAGE_DISPLAY must be added to hold the converted number. As noted previously, the width of the string must be large enough to hold all the characters being displayed (six, in this case).

Type MESSAGE_DISPLAY as indicated in Figure 6-24 and you will be asked if you want to define a new variable. Select YES.



Figure 6-25: Confirming New Variable Creation

Define the string variable as indicated in Figure 6-26. Select ACCEPT from this dialog box to complete the variable definition, then select ACCEPT from the ITEM EDITOR dialog box to complete the new command.

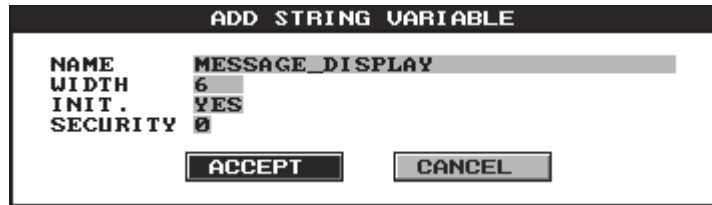


Figure 6-26: Defining a New Variable

The new command will appear in the CONVERT_# detail window.

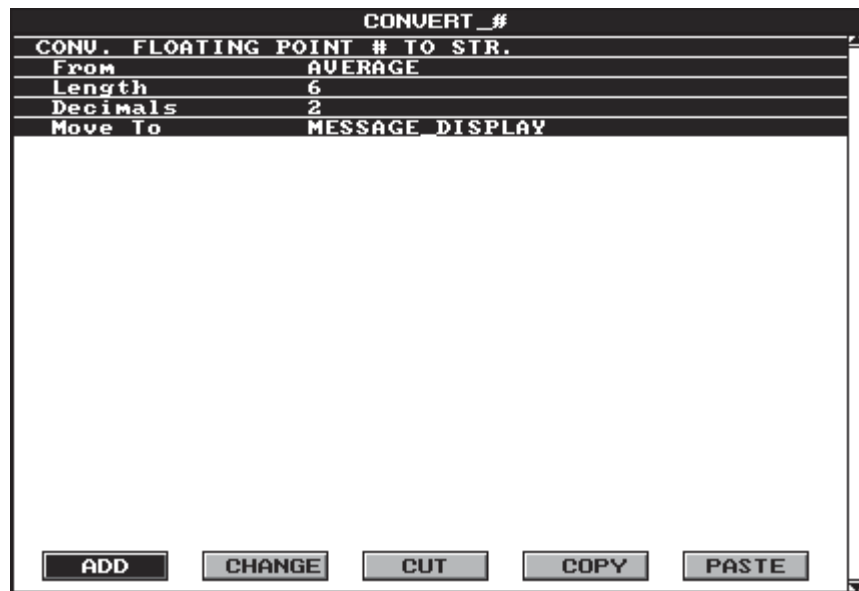


Figure 6-27: Viewing the Detail Window

POSITION_CURSOR

We will now add commands to the POSITION_CURSOR block to send two characters to port 3, which is wired to the demo box display panel.

ASCII character 5 erases all characters from the cursor position to the end of the current line of the display. ASCII character 6 causes the cursor to return to the home position on the display. Before printing text to the display, we will use two PRINT CHR TO PORT commands to send these values to port 3.

Open the detail window for the POSITION_CURSOR block and select ADD. Enter information for the PRINT CHR TO PORT command as shown below.

The ITEM EDITOR dialog box contains the following configuration:

OPERATION	:	PRINT CHR TO PORT	
From		CONSTANT INTEGER	6
To Port		CONSTANT INTEGER	3
Put Status In		VARIABLE INTEGER	

Buttons: ACCEPT, CANCEL

Figure 6-28: Defining a Command

The PRINT CHR TO PORT command will return a status of True (-1) or False (0). Store this status in a variable integer.

To supply the variable integer name, press ENTER without typing any characters and a dialog box will appear, listing the one variable integer previously defined.

The VARIABLE INTEGER dialog box shows a list with one entry:

BUTTON_CONTROL_CHART_STATUS

Buttons: ACCEPT, ADD, CANCEL

Figure 6-29: Adding a Variable "On the Fly"

Add a new variable integer by selecting ADD. The ADD NUMERIC VARIABLE dialog box will open. Complete the required information as shown below and select ACCEPT to define a new variable called PORT_STATUS.

The ADD NUMERIC VARIABLE dialog box contains the following configuration:

NAME	PORT_STATUS
TYPE	INTEGER
INIT.	YES
INIT VAL	0
SECURITY	0

Buttons: ACCEPT, CANCEL

Figure 6-30: Defining a New Variable

Close the ITEM EDITOR dialog box by selecting ACCEPT.

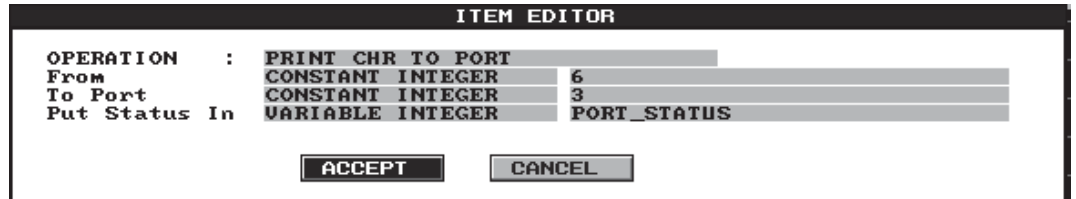


Figure 6-31: Accepting a Completed Command

The new command will appear in the detail window.

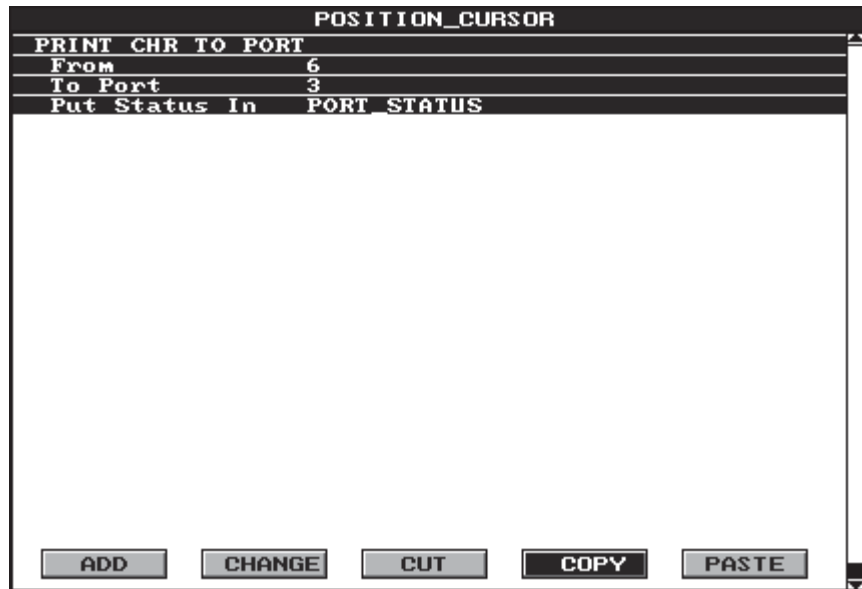


Figure 6-32: Viewing the Detail Window

Because the second command to be added is nearly identical to the first, we will copy the first command, paste it, and modify it as needed.

From the detail window, highlight the PRINT CHR TO PORT command and use the right arrow key to highlight COPY. Press ENTER to copy the command to the buffer. Now highlight PASTE and press ENTER.

Two identical commands will be displayed in the detail window. Highlight the second command and select CHANGE.

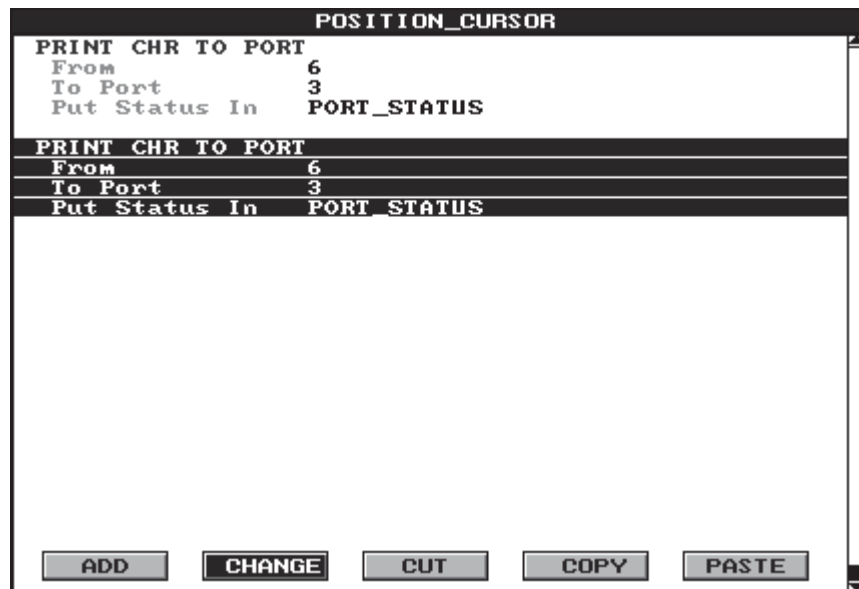


Figure 6-33: Changing a Command

From the ITEM EDITOR dialog box, change the value of the constant integer in the *From* field from 6 to 5.

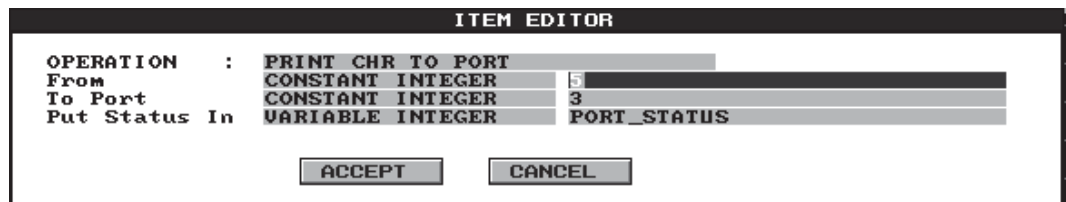


Figure 6-34: Editing a Command

Select ACCEPT and the modified command will appear below the original command in the POSITION_CURSOR detail window.

The effect of these two commands will be to print ASCII character 6 followed by ASCII character 5 to the demo box display panel. These actions will return the cursor to the home position and clear the first line of the display.

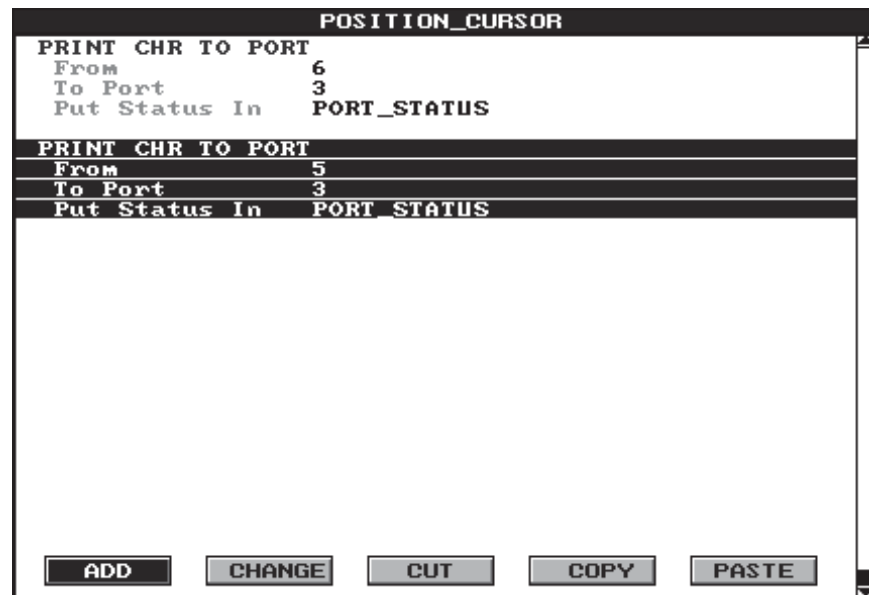


Figure 6-35: Viewing the Detail Window

PRINT

The final operation block into which commands must be added is PRINT. Open the detail window for this operation block and select ADD.

To print a string at a specified port, we must use the PRINT TO PORT command.

Create a PRINT TO PORT command to send the string MESSAGE_DISPLAY, created by the convert command, to port 3, which is wired to the demo box display panel. Enter the information as shown below and select ACCEPT.

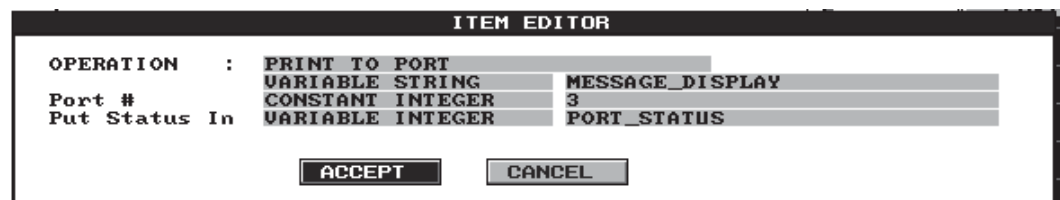


Figure 6-36: Accepting a Completed Command

Verify the command in the PRINT detail window, then close the window.

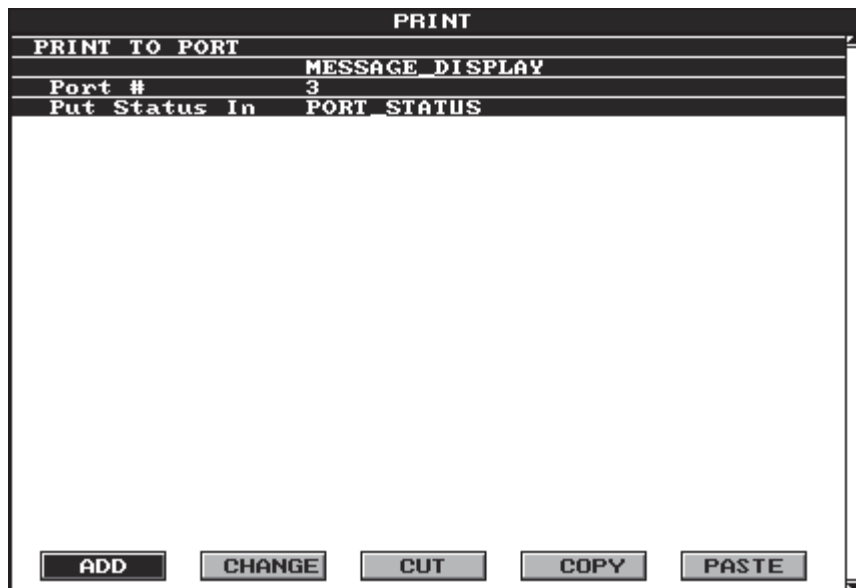


Figure 6-37: Viewing the Detail Window

START THE NEW CHART FROM THE POWERUP CHART

The newly created ANALOG chart must be started from the POWERUP chart if it is to start when the program is run. Select the POWERUP chart by choosing Select from the Charts menu.

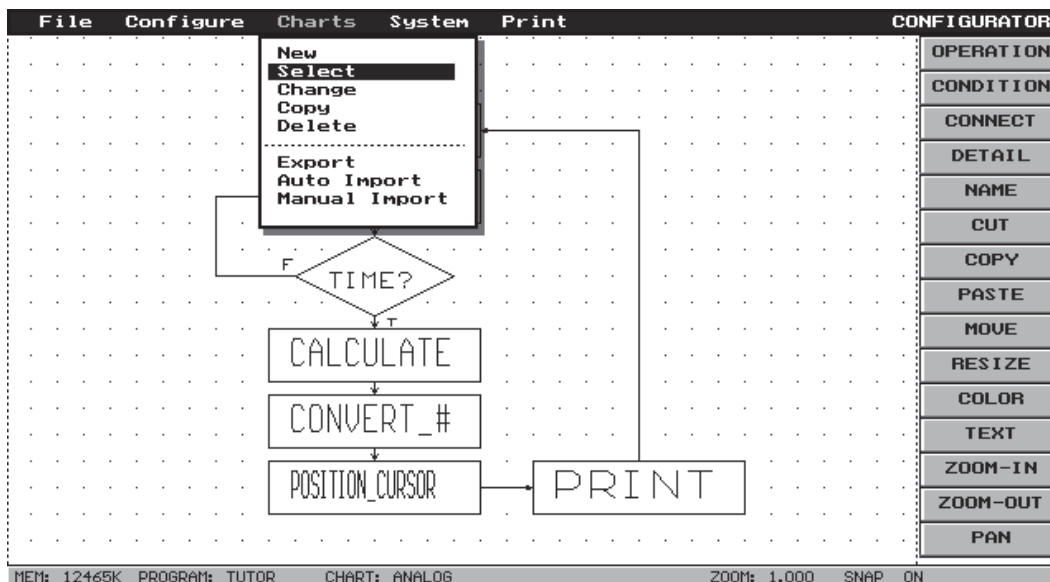


Figure 6-38: Changing Chart Views

Select POWERUP from the list of charts.

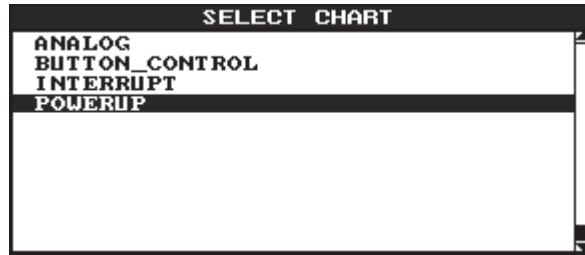


Figure 6-39: Selecting a Chart

This will load the POWERUP chart into the Configurator. Double-click the left mouse button over the START_CHARTS operation block to open the detail window for this block.

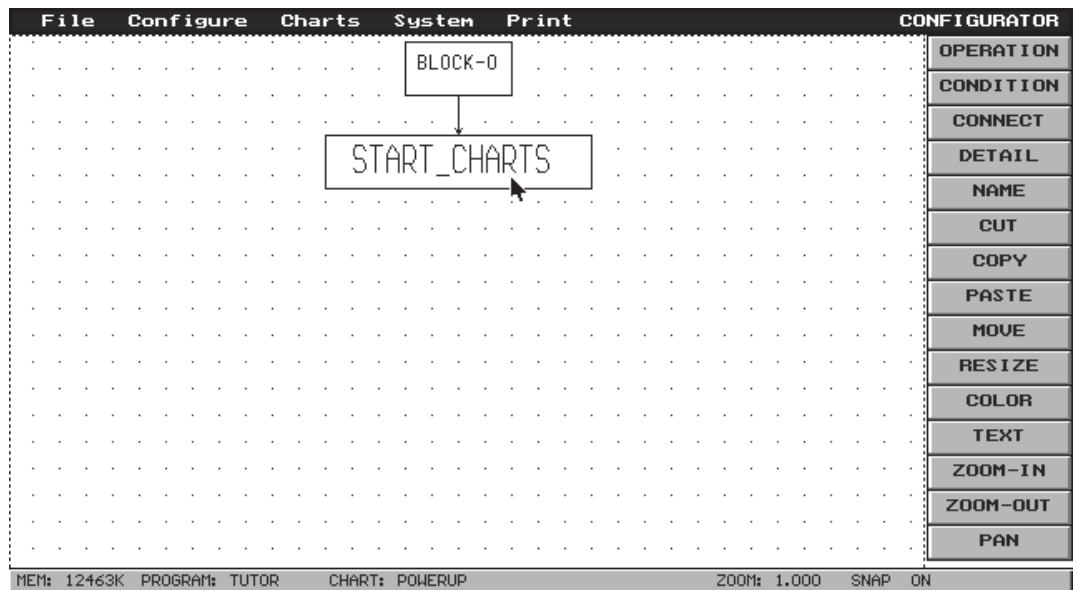


Figure 6-40: Opening a Detail Window

The detail window will contain one START CHART command. Select ADD to create a second command to start the ANALOG chart.

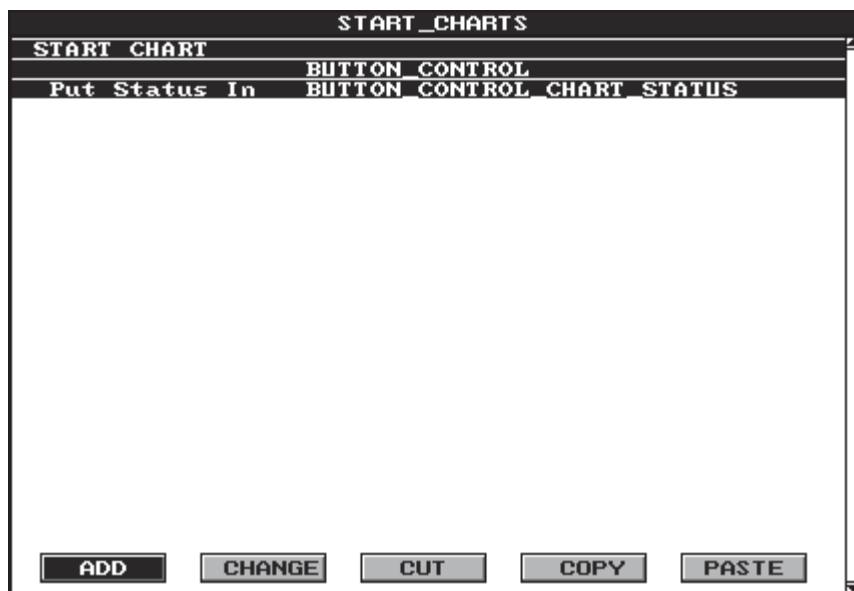


Figure 6-41: Adding a Second Command

Complete the information in the ITEM EDITOR dialog box as shown below.

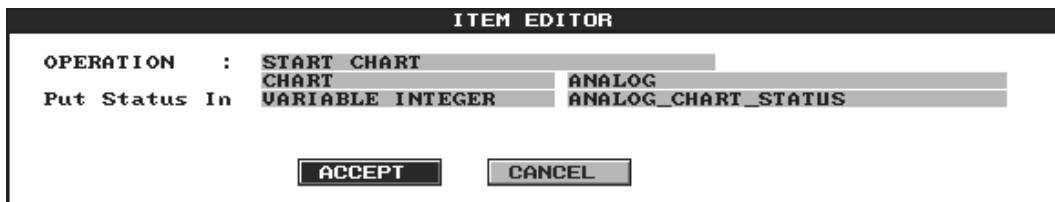


Figure 6-42: Defining a Command

Add a new variable called ANALOG_CHART_STATUS to hold the returned status when the ANALOG chart is started.

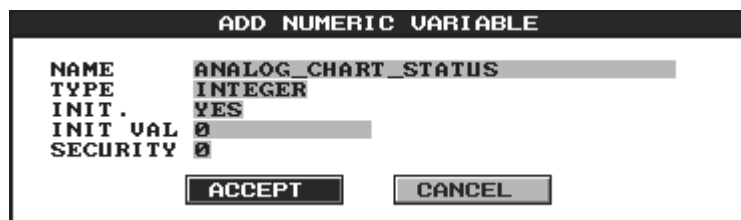


Figure 6-43: Defining a New Variable

Select ACCEPT from the ADD NUMERIC VARIABLE and ITEM EDITOR dialog boxes, then verify the new command in the START_CHARTS detail window before closing the window.

DOWNLOAD AND RUN THE PROGRAM

Select Debugger from the System menu.

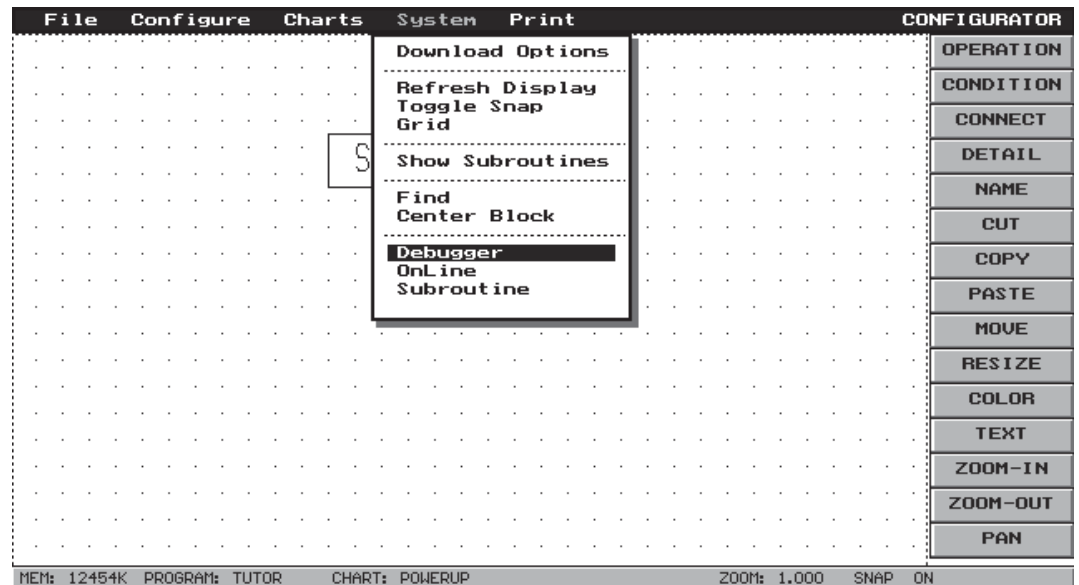


Figure 6-44: Opening the Debugger from the Configurator

After the program has been downloaded to the Mystic controller, start the modified program running.

Verify that the values of the simulated temperature sensors are displayed on the meters. Also verify that the average of the two sensors is calculated and displayed on the demo box display panel once every second.

Notice Cyrano's multitasking capability. The BUTTON_CONTROL chart is controlling the operation of the push buttons, while the ANALOG chart is controlling the temperature sensors display at the same time.

WHAT DID I DO?

In this chapter you created a new chart called ANALOG. After drawing the flowchart structure, you added commands to start a timer, display the values of two sensors on two meters, check if the timer has expired, calculate the average of two sensors, convert the result into a string, clear and reposition the cursor on a display panel, and print the string to the display. You then downloaded and ran the program to verify that both the new chart and the previously created BUTTON_CONTROL chart were running.

WHERE DO I GO FROM HERE?

You are now ready to create your own Cyrano programs! See Chapter 7 for a brief review of the overall Cyrano programming process.

SUMMARY



GENERAL STEPS IN CREATING A CYRANO PROGRAM

1. Divide the application into separate tasks
2. Plan the control strategy
3. Configure the hardware
4. Draw the flowchart structure
5. Enter commands into the condition and operation blocks
6. Download and run the program to test its operation
7. Modify and enhance the program
8. Repeat steps 6 and 7 until the application is complete

NEED MORE HELP?

PRODUCT SUPPORT

Opto 22 products are easy to use. However, should you have a problem, contact Opto 22 Product Support and we will be more than happy to help you. A good rule of thumb is, **Never spend more than an hour on a single problem** before calling Opto 22. Your success is our success.

See Appendix A for complete information on contacting Product Support.

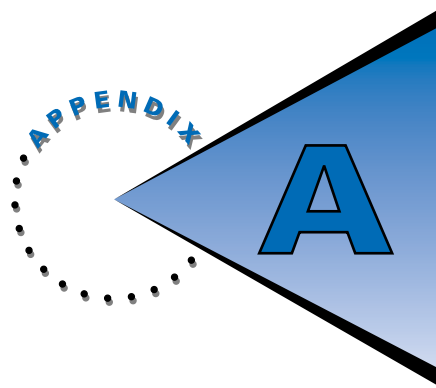
TRAINING PROGRAMS

Opto 22 offers Cyrano training classes throughout the year at our facility in Temecula, California, located approximately one hour north of San Diego.

“Introduction to Cyrano” is a four-day, hands-on, learn-by-example course designed to enhance your Cyrano skills. Every attendee receives a training manual with step-by-step details documenting each classroom exercise.

For further details or to enroll, call 800/396-OPTO and ask for Cyrano training information.

PRODUCT SUPPORT



If you have any questions about this product, contact Opto 22 Product Support Monday through Friday, 8 a.m. to 5 p.m. Pacific Time.

Phone: 800/TEK-OPTO (835-6786)
909/695-3080

Fax: 909/695-3017

E-mail: support@opto22.com

Bulletin Board System (BBS): 909/695-1367
(24 hours a day, 7 days a week)

When accessing the BBS, use the following modem settings:

- No parity, 8 data bits, 1 stop bit
- Baud rates up to 28,800
- Z-modem protocol for uploads and downloads (optional but recommended)

When calling for technical support, be prepared to provide the following information about your system to the Product Support engineer:

- Software and version being used
- Controller firmware version
- PC configuration
- A complete description of your hardware and operating systems, including:
 - jumper configuration
 - accessories installed (such as expansion daughter cards)
 - type of power supply
 - types of I/O units installed
 - third-party devices installed (e.g., barcode readers)
- Specific error messages seen

