Training Manual for Cscape and NX / LX

17Feb 2007 MAN0800-05

PREFACE

This manual provides introductory level training for Cscape Software users using NX / LX.

For user manual updates, contact Technical Support:

North America:

(317) 916-4274

www.heapg.com

email: techsppt@heapg.com

Europe:

(+) 353-21-4321-266

www.horner-apg.com

email: techsupport@hornerirl.ie

Horner APG,LLC. ("HE-APG") warrants to the original purchaser that the Cscape Software manufactured by HE-APG is free from defects in material and workmanship under normal use and service. The obligation of HE-APG under this warranty shall be limited to the repair or exchange of any part or parts which may prove defective under normal use and service within two (2) years from the date of manufacture or eighteen (18) months from the date of installation by the original purchaser whichever occurs first, such defect to be disclosed to the satisfaction of HE-APG after examination by HE-APG of the allegedly defective part or parts. THIS WARRANTY IS EXPRESSLY IN LIEU OF ALL OTHER WARRANTIES EXPRESSED OR IMPLIED INCLUDING THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR USE AND OF ALL OTHER OBLIGATIONS OR LIABILITIES AND HE-APG NEITHER ASSUMES, NOR AUTHORIZES ANY OTHER PERSON TO ASSUME FOR HE-APG, ANY OTHER LIABILITY IN CONNECTION WITH THE SALE OF THIS Cscape Software. THIS WARRANTY SHALL NOT APPLY TO THIS Cscape Software OR ANY PART THEREOF WHICH HAS BEEN SUBJECT TO ACCIDENT, NEGLIGENCE, ALTERATION, ABUSE, OR MISUSE. HE-APG MAKES NO WARRANTY WHATSOEVER IN RESPECT TO ACCESSORIES OR PARTS NOT SUPPLIED BY HE-APG. THE TERM "ORIGINAL PURCHASER", AS USED IN THIS WARRANTY, SHALL BE DEEMED TO MEAN THAT PERSON FOR WHOM THE Cscape Software IS ORIGINALLY INSTALLED. THIS WARRANTY SHALL APPLY ONLY WITHIN THE BOUNDARIES OF THE CONTINENTAL UNITED STATES.

In no event, whether as a result of breach of contract, warranty, tort (including negligence) or otherwise, shall HE-APG or its suppliers be liable of any special, consequential, incidental or penal damages including, but not limited to, loss of profit or revenues, loss of use of the products or any associated equipment, damage to associated equipment, cost of capital, cost of substitute products, facilities, services or replacement power, down time costs, or claims of original purchaser's customers for such damages.

To obtain warranty service, return the product to your distributor with a description of the problem, proof of purchase, post paid, insured and in a suitable package.

ABOUT PROGRAMMING EXAMPLES

Any example programs and program segments in this manual or provided on accompanying diskettes are included solely for illustrative purposes. Due to the many variables and requirements associated with any particular installation, Horner APG cannot assume responsibility or liability for actual use based on the examples and diagrams. It is the sole responsibility of the system designer utilizing Cscape Software to appropriately design the end system, to appropriately integrate the Cscape and to make safety provisions for the end equipment as is usual and customary in industrial applications as defined in any codes or standards which apply.

Note: The programming examples shown in this manual are for illustrative purposes only. Proper machine operation is the sole responsibility of the system integrator.

Revisions to this Manual

This version (MAN0800-04) of the ***Training Manual for Cscape and NX / LX*** contains the following revisions, additions, and deletions:

1. **Added a new Lab 3 and re-numbered the subsequent labs accordingly.**

2. **Made minor corrections throughout the document as needed.**

TABLE OF CONTENTS

Introduction to Cscape

Quick Start Guide

Intro to Cscape: Quick Start Guide

Objective:

The objective of this Quick-Start Guide is to familiarize yourself with some of the features and functionality of the Cscape programming software.

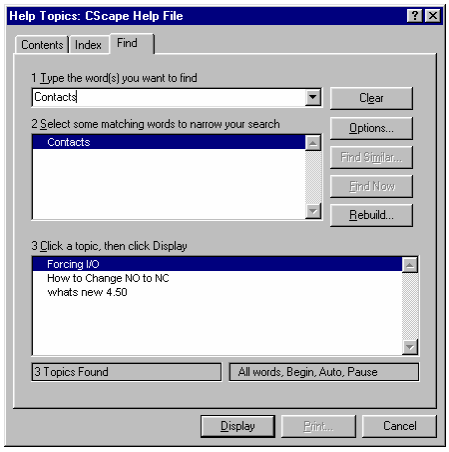
Equipment Needed:

A PC with Cscape loaded.

1.0 Help File
- 1.1 Open the help file. The help file is located under **Help** from the main tool bar.
- 1.2 Select **Contents** to access the help file.
- 1.3 The first screen has a lot of useful information that is listed below.
  - 1.3.1 What's New in Version X.XX – This section will include all of the additions that were added to that particular release of Cscape.
  - 1.3.2 Cscape Reference Manual – This section allows the user to navigate to all of the information in the help file.
  - 1.3.3 The User Interface – This describes some of the user features of Cscape and how to navigate through the software.
  - 1.3.4 Creating and Editing Ladder Programs – This section does a multitude of things from the different ladder elements to clearing out an old program.
  - 1.3.5 Creating and Editing Text Screens – This discusses how to create and manipulate the HMI portion of an OCS program.
  - 1.3.6 Networking and Communication – This section discusses the different aspects of the CsCan network and serial communications.
  - 1.3.7 I/O and CPU Configuration – This section covers how to configure a controller and a quick reference to a few of the I/O cards like the High Speed Counter, Stepper Module, and more.
  - 1.3.8 Debugging – This section covers the aspects of running the debug option in the software.
  - 1.3.9 Inside the Controller – This section covers the system resources of the controller, updating the firmware, cabling, and other features
  - 1.3.10 Project Management – This covers how to build a CsCan project for more than 1 node system.
  - 1.3.11 How Do I? – This is a quick start guide on how to get started on certain task.
  - 1.3.12 Additional Technical Support – This covers information on how to contact Horner APG.

1.4 Searches can be done through selecting **Find** from the top of the screen.

1.4.1 Upon opening the Find portion of the help file, type in “Contacts” and the following will be shown on the screen.



1.5 The programmer also has the ability to open the help file by pressing F1 on the keyboard of the PC.

2.0 Getting Started

2.1 There are 2 ways to create a new program. A new program will have a name of “Untitled” until the program is saved as its file name.

2.1.1 Create a program under the **File** selection on the main menu

2.1.2 Create a new program by pressing the New File from the Tool Bar at the top of the screen.

2.2 There are 2 ways to save a program. All programs will be saved as the “filename”.csp

2.2.1 Save a program under the **File** selection on the main menu

2.2.2 Save a program from the shortcut on the Tool Bar at the top of the screen.

2.3 There are 3 ways to open a program.

2.3.1 Open the program under the **File** selection on the main menu.

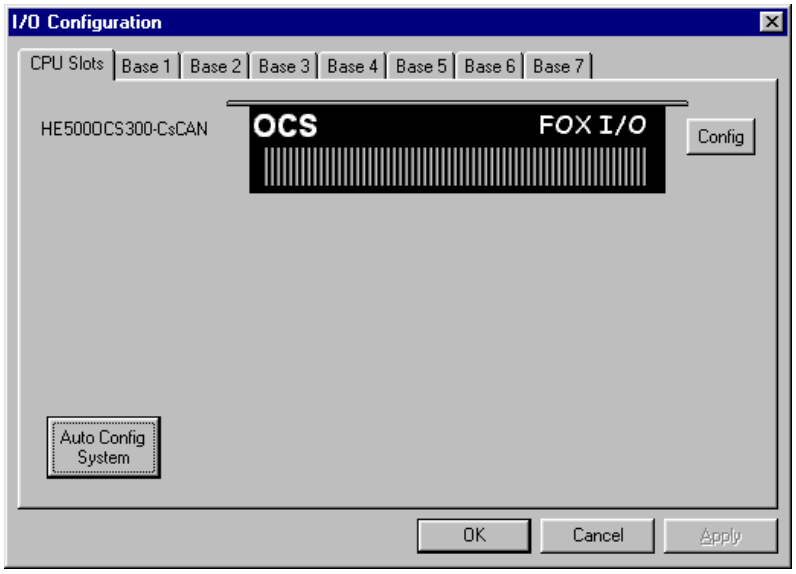
2.3.2 Open a program from the shortcut on the Tool Bar at the top of the screen.

2.3.3 The program will automatically open if the program is double clicked on in the location where it is stored on your PC.

2.4 Configuring a controller is done be clicking the **Controller** menu and selecting **I/O Configuration**. This will bring up the screen below. If no controller is attached to the PC, the controller will default to the OCS300. If there is a controller attached to the PC and the target ID matches the local ID; the controller will match what the PC is attached to. There are 2 ways to configure the controller.

2.4.1 Manually configure the controller by pressing the Config button next to the controller and then select the controller from the pull down list.

2.4.2 Configure the controller from the Auto-Configure option. Keep in mind on existing programs that Auto-Configure will erase I/O configurations that deviate from the default parameters. An example of this would be in an application with a High Speed



Counter that uses an option other than option 1 or any analog modules that have the ability to change the input or the output type.

2.5 Configuring the I/O is done from the same place as configuring a controller. I/O is never automatically configured without the user telling it to happen, unlike the controller that will automatically configure if the PC is connected to it when Cscape is opened.

2.5.1 If the Auto-Configuration option is used, the I/O will be recognized when you Auto-Config. On OCS units utilizing the FOX I/O system, the I/O will appear on the base where the I/O is connected. On the OCS250 and below, the I/O will appear on the stack with the controller. The one exception is for Ethernet cards, which will always be connected directly to the controller, regardless of controller type.

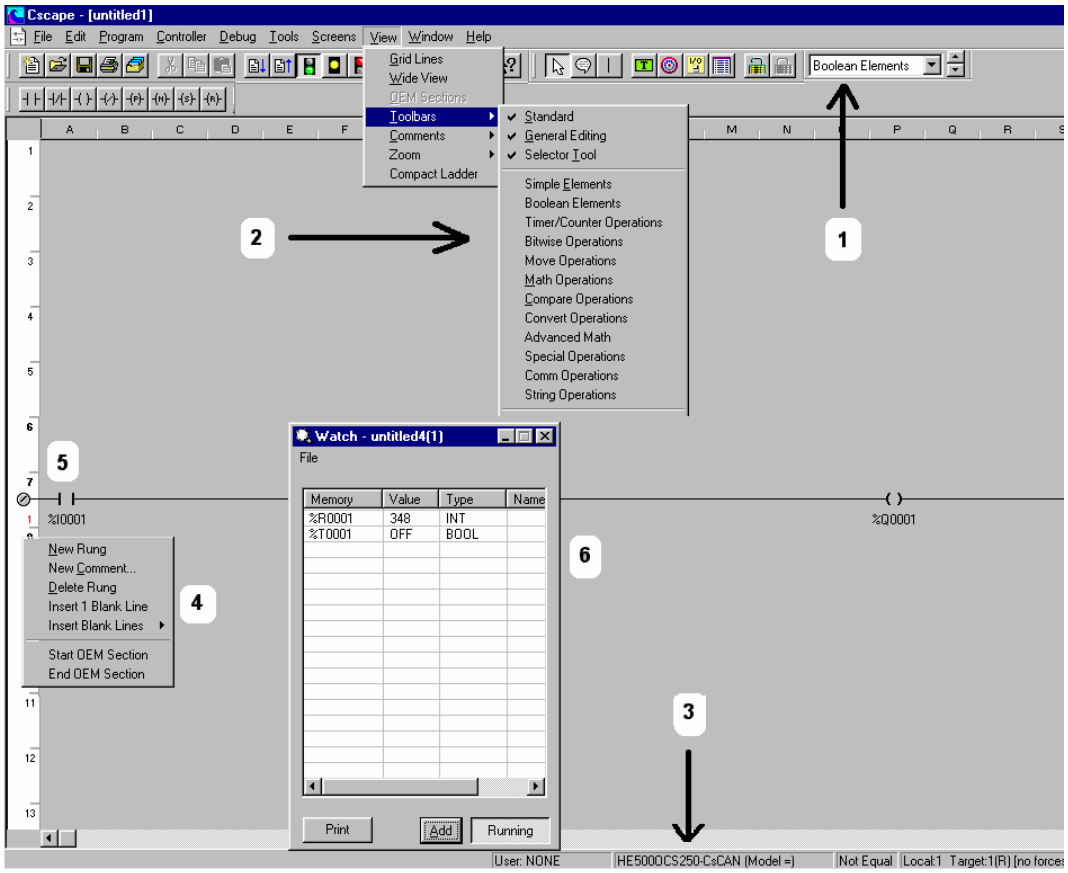
2.5.2 If the I/O is manually configured, go to the position that the module is to be configured and click on the Config button or double click on the position. The screen shown below will appear. Select the appropriate module for the slot. For FOX I/O systems, select the tab corresponding to the FOX base address.

2.6 Toolbars are used to place Ladder elements and functions.

2.6.1 Selector Tool – This allows the programmer to select between the different tool bars with 1 shown on the screen at a time. This is achieved through the pull down menu at the top of the screen. **#1** in the picture below illustrates the location of the pull down selection.

2.6.2 Menu Toolbar Selection – The user can setup Cscape to display multiple Toolbars at a time. This is done through selecting multiple Toolbars under **View** and **Toolbars**. **#2** in the picture below illustrates this. The toolbars can be left floating over the main

Cscape program or can be dragged and "docked" to the top or left side of the screen.



2.7 The status bar has many useful features. **#3** points to the status bar.

2.7.1 **User** – The User field indicates which user is currently logged into the program via use of the Security features. If security is not configured or if no one is currently logged in, this will indicate NONE as it does in the illustration.

2.7.2 **Model** – This will let the programmer know which unit the program is configured for and whether the configured model is equal to the model that the PC is connected to.

2.7.3 **Program Equality** – This is the box to the right of the Model box. This will let the user know if the program in the unit and the program in Cscape are equal. If the status indicates Unknown, the user might need to perform a verify between the controller and the software.

2.7.4 **Local and Target** – The Local ID indicates the node ID of the controller that the PC is directly connected to while the Target ID indicates the node ID of the controller that Cscape is trying to talk to. The Target ID does not need to match the local id. If programming is to be performed across the CsCAN Bus, then the Target will be the node that will receive the download. The (R) indicates that the controller is in RUN mode, (I) indicates that the controller is in STOP or IDLE mode, and (D) indicates that the controller is in DO/IO state. If a (B) is shown, it means the

controller is Busy because another computer is trying to talk to it at that moment.

2.8 Starting a New Rung of logic can be done in either of two different ways.

2.8.1 **Placing a contact** – A new rung can be started by dropping a contact on to the screen. The user needs to drop the contact in A column for this to occur. To verify that a new rung has been started, look at the left margin. If there is a screw head in the margin, a new rung has been started. See **#5** in the picture on the previous page. Another thing to consider when programming a parallel contact is that placing the parallel contact in the A column will start a new rung. To get around this, place the branches first.

2.8.2 **Right clicking in the margin** – right clicking in the left-hand margin and selecting **New Rung** can also create a new rung. See **#4** in the picture on the previous page.

2.9 Data Watch enables the user to monitor and/or change values in a table. **#6** is what Data Watch looks like. Data Watch is selected from the magnifying glass on the Toolbar or through selecting it from the **Controller** menu. New fields are added to Data Watch by clicking Add and then keying in the register and the type. Ranges of addresses can be added at one time by using the notation ‘r15-25’, which will add 11 registers from %R15 through %R25.

Notes:

LAB 1

Basic OCS Programming and Configuration

Lab 1: Basic OCS Programming and Configuration

Lab 1: Basic OCS Programming and Configuration

Objective:

The objective of this lab is to give you the knowledge to use Cscape to create a program including hardware configuration, logic design, and screen development.

This foundation will then be used to help you expand your skills in the use of Cscape and the OCS.

Procedure:

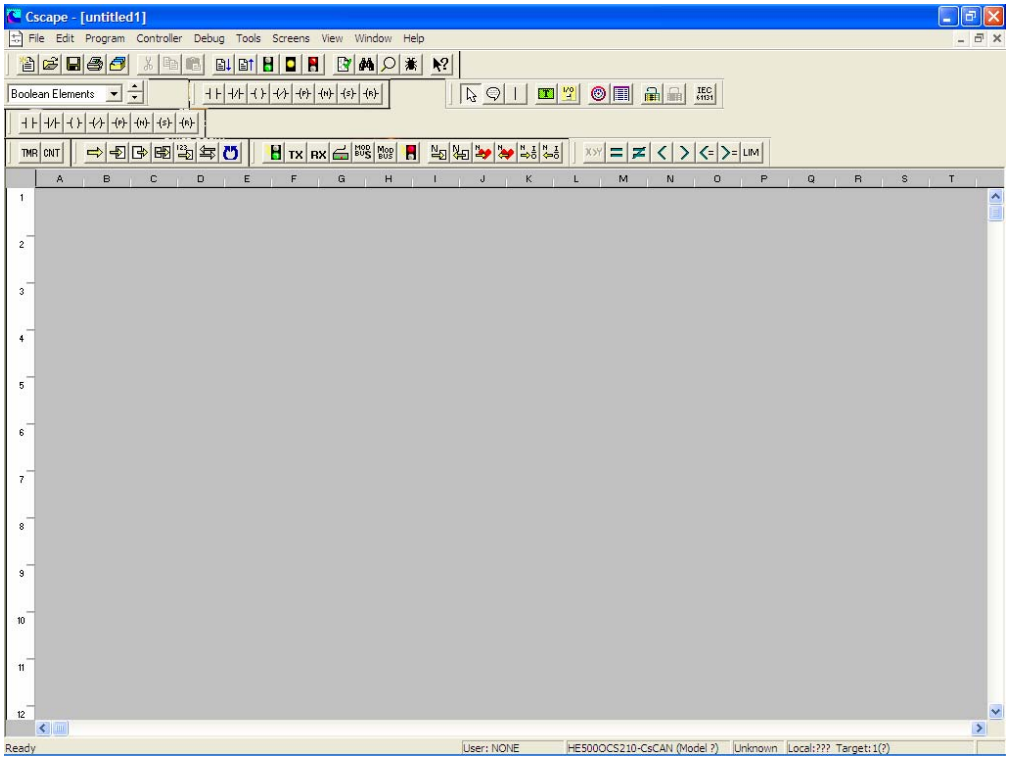
Step 1

- **Connect the Demo Case to your PC.** Connect the serial cable provided to the NX 9 pin programming port and the 9 pin serial port on your PC.

Step 2

- **Power up the OCS and start Cscape on your PC.** Connect the power supply to the NX. Open the Cscape program on your PC. A new, blank program called "untitled1" is automatically opened and should be automatically configured for your NX if the serial cable is properly connected.

  NOTE: Only the controller is automatically configured as described above. Any I/O will still have to be configured as described later in this lab.

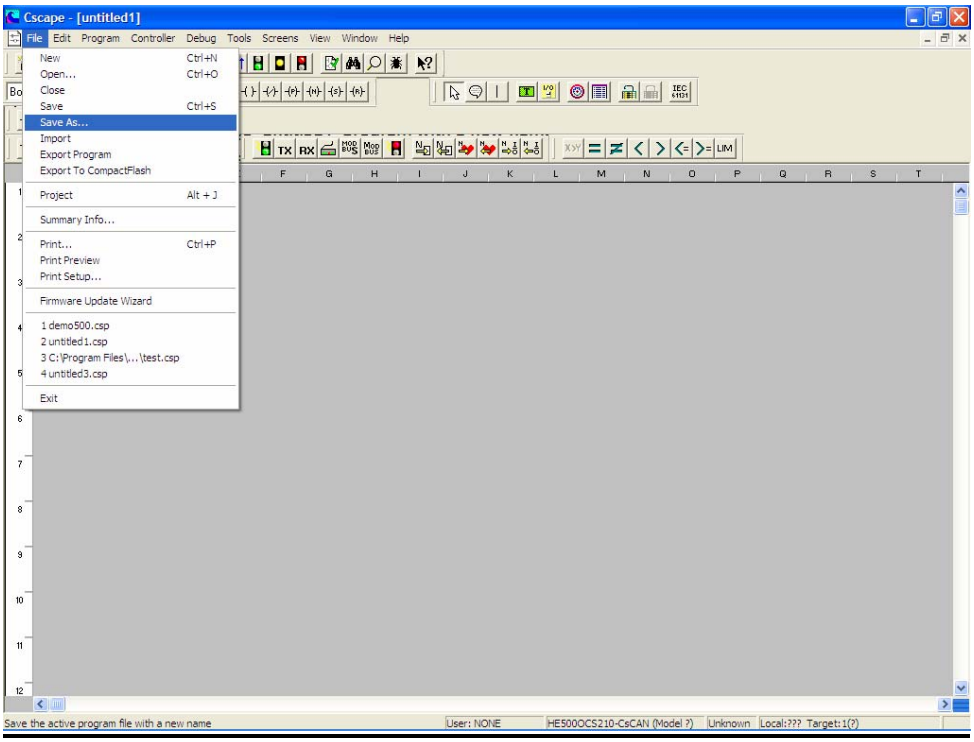


Lab 1: Basic OCS Programming and Configuration

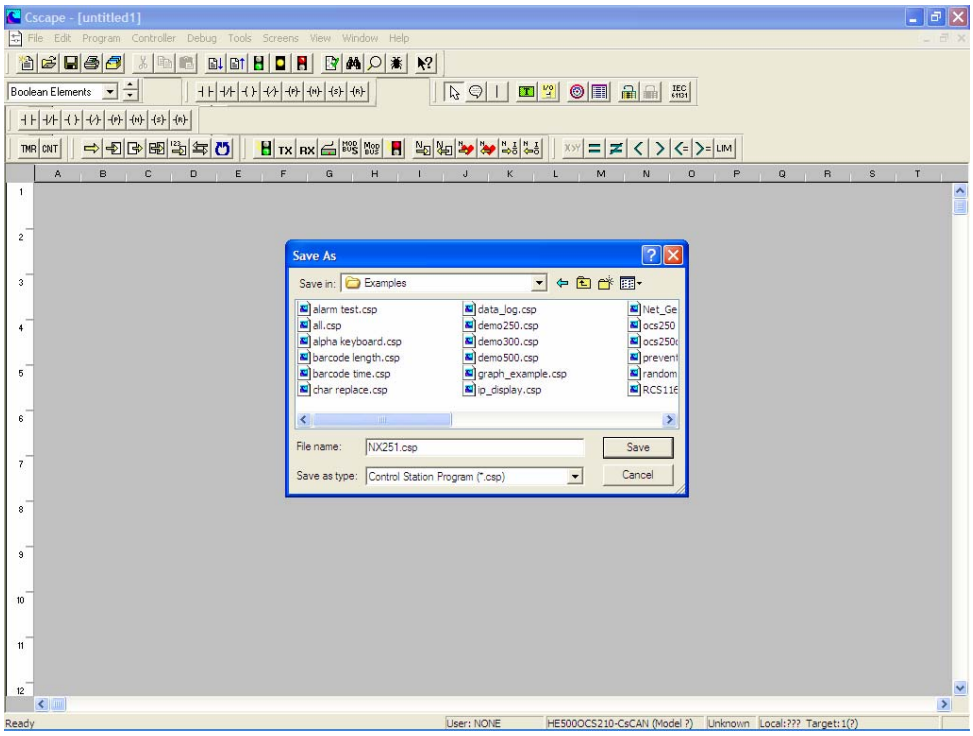
Step 3

➢ **Save the 'untitled1' program with a new name.**

Click on the **File** menu and select **Save As…**



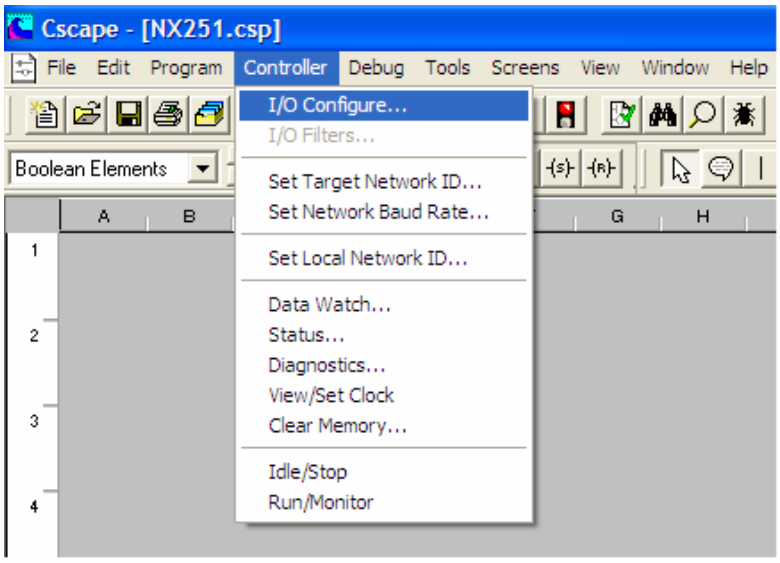
Type your program name, such as NX251.csp, in the File Name dialog box and click the Save button.



Step 4

- **Configure the OCS Controller**

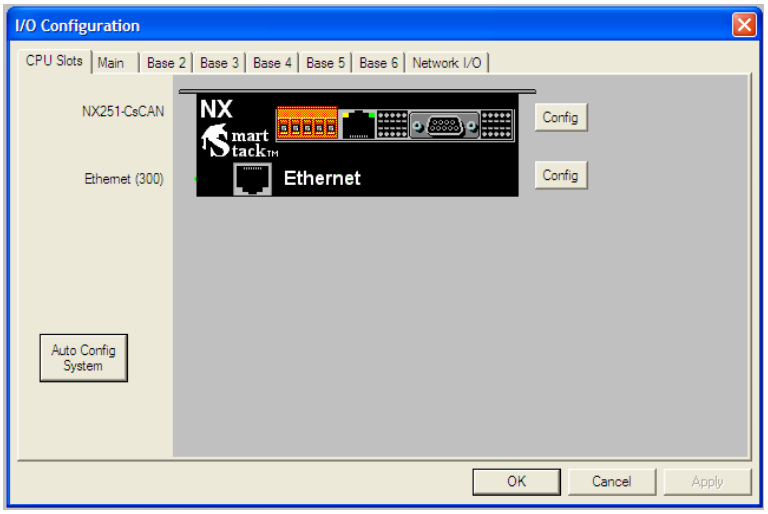
Click on the **Controller** menu and select **I/O Configure**.



If you are online with the OCS, use the Auto Config System button. Clicking it will automatically configure the controller and any attached I/O if you are connected to the OCS properly.

Otherwise, to do it manually:

1. Double click on the controller picture or click the 'Config' button next to it.
2. Select NX251 from the list and click OK
3. Select the Main tab at the top of the I/O configuration screen



4. **Double click on the first I/O module slot or click the 'Config' button next to it.**
5. **Select the Mixed Digital tab.**
6. Select DIQ611 from the list.
7. Click OK
   (To delete or replace a module, right click on the module.)
8. Double click on the second I/O module slot
9. Select the Analog In tab
10. Select the THM100.
11. Click OK.
12. Double click on the third I/O module slot.
13. Select the Analog Out tab
14. Select the DAC101
15. Click OK
16. Click OK again to exit the I/O configuration.

Step 5

- **Save the program.**

Click on the **File** menu and select **Save**.

Step 6

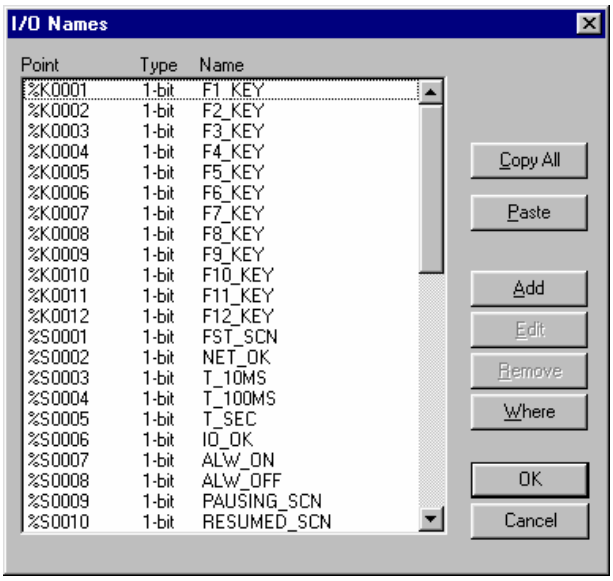
- **Name some I/O points.**

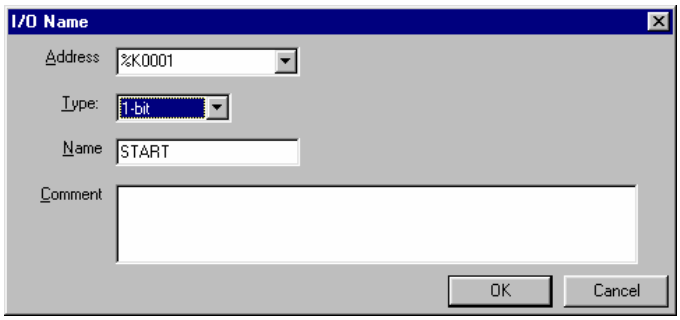
Click on the **Program** menu and select **I/O Names**.

- **Add** I/O points by clicking the 'Add' button and filling in the information.
- **Edit** an existing I/O point by finding it in the list and double-clicking it.

Add or edit the following I/O points:

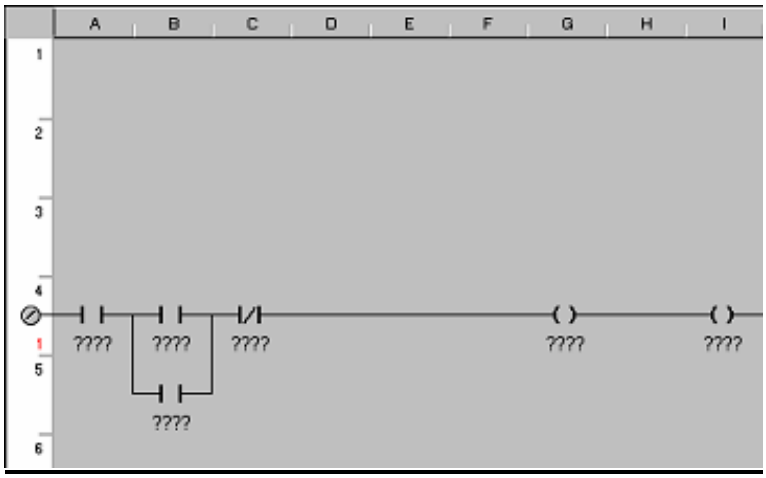
%I01	E_STOP – Configure for 1 bit
%K1	START – %K1 is named 'F1_KEY' by default so it will need to be edited instead of added. Configure for 1 bit.
%K2	STOP - %K2 is named 'F2_KEY' by default so it will need to be edited instead of added. Configure for 1 bit.
%Q1	RUN – Configure for 1 bit
%D1	Stopped_Screen – Configure for 1 bit
%D2	Running_Screen – Configure for 1 bit


I/O Names
Point Type Name
%K0001 1-bit F1_KEY
%K0002 1-bit F2_KEY
%K0003 1-bit F3_KEY
%K0004 1-bit F4_KEY
%K0005 1-bit F5_KEY
%K0006 1-bit F6_KEY
%K0007 1-bit F7_KEY
%K0008 1-bit F8_KEY
%K0009 1-bit F9_KEY
%K0010 1-bit F10_KEY
%K0011 1-bit F11_KEY
%K0012 1-bit F12_KEY
%S0001 1-bit FST_SCN
%S0002 1-bit NET_OK
%S0003 1-bit T_10MS
%S0004 1-bit T_100MS
%S0005 1-bit T_SEC
%S0006 1-bit IO_OK
%S0007 1-bit ALW_ON
%S0008 1-bit ALW_OFF
%S0009 1-bit PAUSING_SCN
%S0010 1-bit RESUMED_SCN
Copy All
Paste
Add
Edit
Remove
Where
OK
Cancel


I/O Name
Address
%K0001
Type:
1-bit
Name
START
Comment
OK
Cancel

Step 7

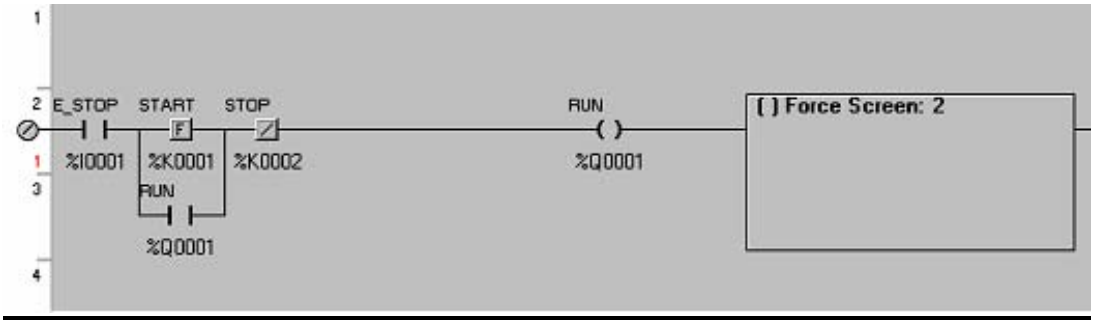
- **Program the following rung:**



1. Select and drop the three normally open contacts.
2. Select and drop the normally closed contact.
3. Add the vertical connecting lines.
4. Select and drop two normally open coils.

Step 8

- **Add the element names.**



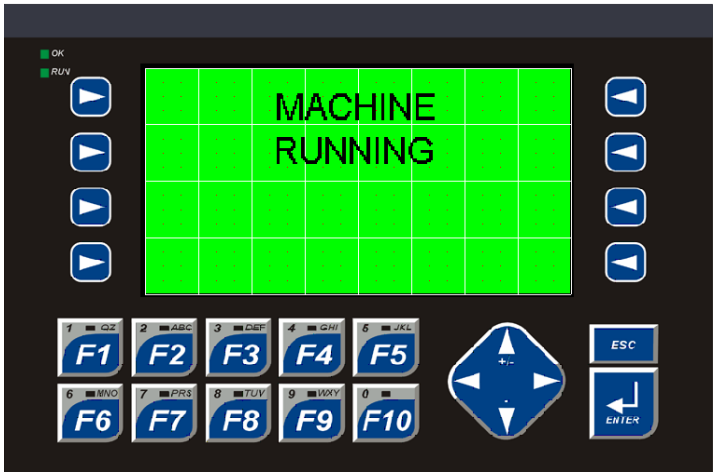
1. Double click on each element in the rung.
2. Select the name or address from the drop down list. Name the last coil %D2 and specify it as a Force Screen.
3. Click OK

Step 9

- **Add words to screen 2.**

1. Double click the screen in the ladder logic.
2. Click Edit Screen.
3. Insert Static Text at the top center of the screen. Edit the text to display MACHINE.

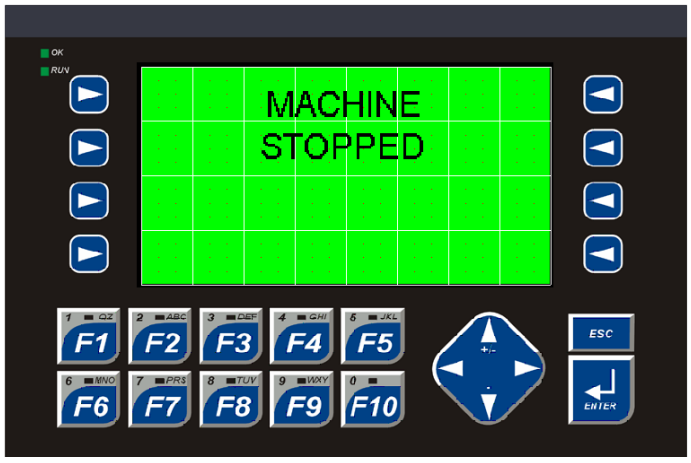
4. Insert Static Text in the center of the screen just below the text insert above. This Static Text should indicate RUNNING.
5. Note that the size of the box will need to be stretched and the font sized should be increased from the default.
6. Close the graphics editor.
7. Click OK



Step 10

- **Add Screen 1**

1. Click on the **Screens** menu and select **View / Edit Screens…**
2. Repeat steps 3 – 6 from above.



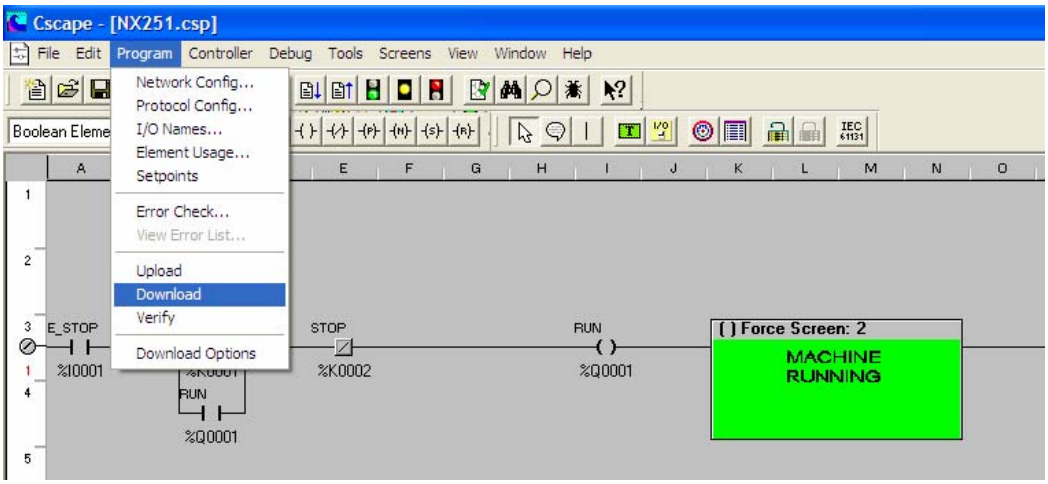
Step 11

- **Save the program.**

Step 12

- **Download the program to the NX251.**

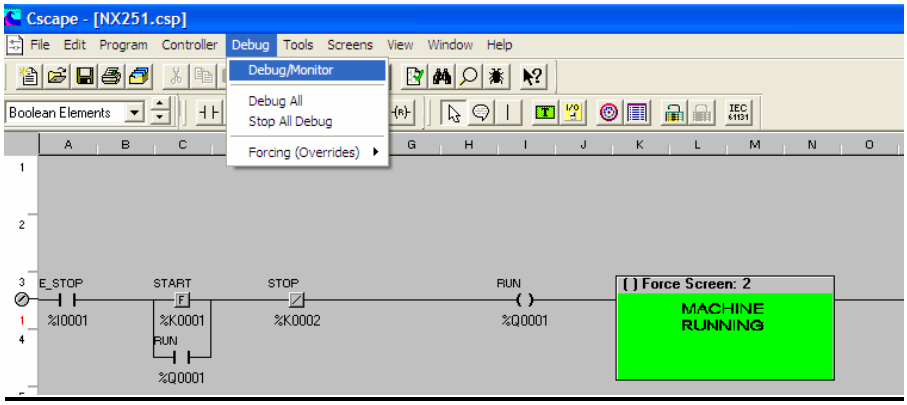
1. Select the **Program** menu and click **Download**.
2. Use the SmartLoad function when the Download dialog box appears.
3. Click OK.



Once downloaded, make sure the NX is in Run mode (the green traffic light on the toolbar).

Step 13

- **Debug your program**



Click on the **Debug** and select **Debug/Monitor**

Close switch 1 on the Demo Panel.

Switch 1 is connected to the first digital input on the DIQ611 I/O card, which is addressed to %I01.

E_STOP & STOP should be red.

The screen should show MACHINE STOPPED.

Push the F1 key.

START should turn red until you release the F1 key.

The RUN coil and contact should both turn red.

The screen should change to MACHINE RUNNING.

Output 1 should turn ON

Push F2 or open switch 1.

The output should turn OFF

The screen should show MACHINE STOPPED.

CONGRATULATIONS! You have finished your first OCS program. Now move on to LAB 2 and learn additional skills.

Lab 1: Basic OCS Programming and Configuration

Notes:

LAB 2

Text Tables

Lab 2: Text Tables

Lab 2: Text Tables

Objective:

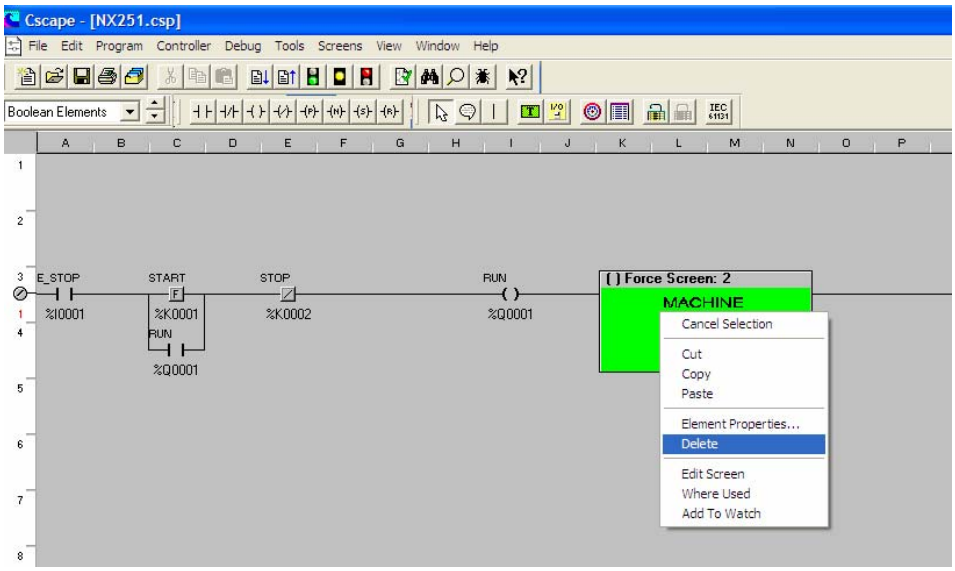
Add a variable field to screen 1 containing a Text Table and eliminate screen 2.

This lab is a continuation of LAB 1.

Procedure:

Step 1

- **Delete screen 2 from the program.**



Right click on screen 2 in the rung. Then click Delete.

Step 2

- **Edit screen 1.**

Click on **Screens**, select **View/Edit Screens**.

Step 3

- **Delete the Static Text.**

1. Delete both Static Text Fields on screen 1.

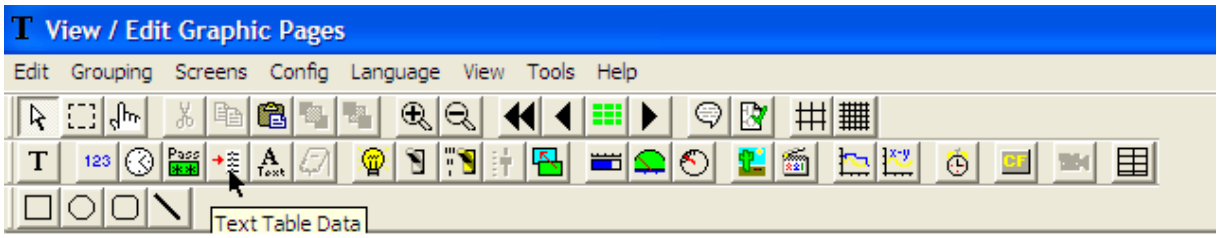
2. Delete both Static Text Fields on screen 2. Hint: CTRL A (Select All) then delete the elements.

Lab 2: Text Tables

Step 4

- **Add a field.**

1. Place the Text Table Object on screen 1. Remember that the field will need to be stretch to size and the parameters will need to be configured.



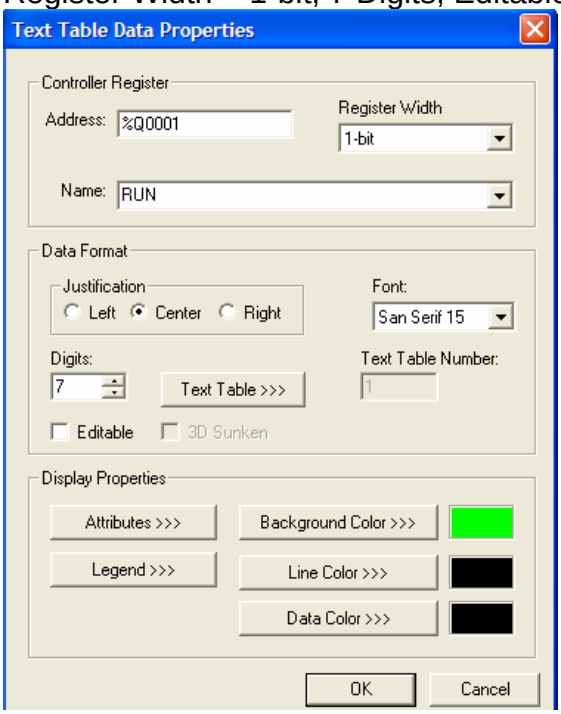
Step 5

- **Edit the field.**

1. Double Click on the field.
2. Change Address to %Q1.

In Data Format verify:

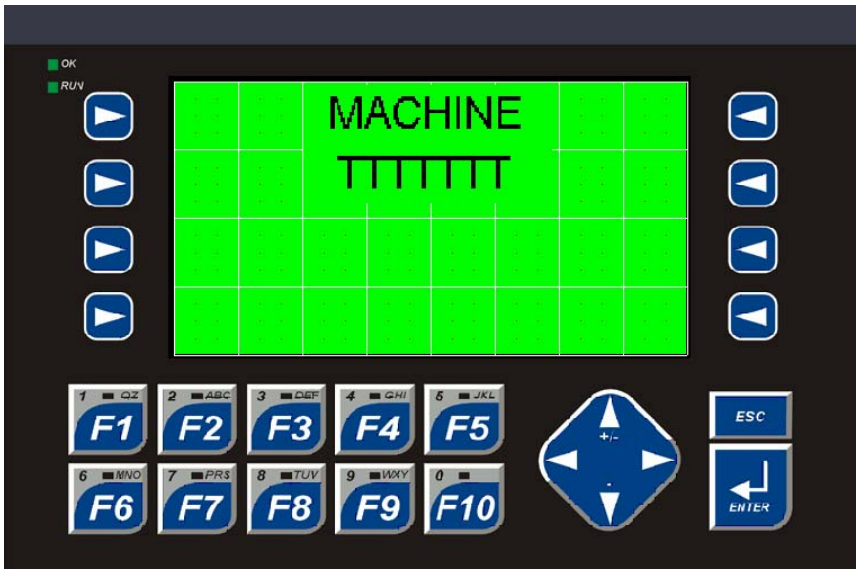
Register Width = 1-bit, 7 Digits, Editable is not checked, and Font = San Serif 15.



3. Click on Text Table.
4. Click on Add.
5. For Value = 0, add String = STOPPED.
6. Click OK.
7. Click on Add.

8. For Value = 1, add String = RUNNING.

9. Click OK.
10. Click Legend
11. Change the Legend to “MACHINE”
12. Change the Legend Font to San Serif 15
13. Click OK.
14. Click Attributes
15. Uncheck Border
16. Click OK.
17. Click OK.



18. Close the Graphics editor.

Step 6

- **Save the program.**

Click on **File** and select **Save**

Step 7

- **Download to the NX.**

Click on **Program** and select **Download**

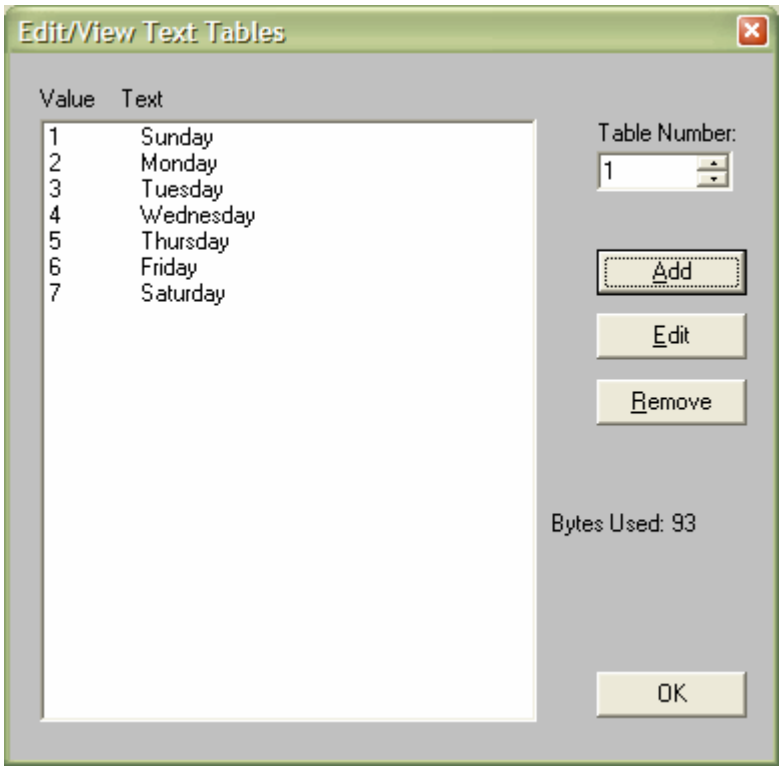
Step 8

- **Check the program operation.**

The program should operate exactly the same as it did before. The screen should look and act the same as before.

This shows some of the flexibility of the Horner controllers and how easy it is to configure some of the screen features. Text tables can often be used in place of an entirely new screen or simply to associate text with a number to make it easier to decipher that number.

Take, for example, the system register %SR50. This registers represents the Day of the Week for the controller's internal Real Time Clock. It contains a value of 1-7 depending on what day it is. Linking a text table directly to %SR50 and configuring the text table as shown would display the days of the week instead of just a number:



Lab 2: Text Tables

Notes:

Lab 2: Text Tables

Notes:

LAB 3

Screen Manipulation

Lab 3: Screen Manipulation

Lab 3: Screen Manipulation

Objective:

The objective of this lab is to demonstrate several different methods to manipulate screens through ladder logic and through the "Screen Jump" object in the graphics editor.

Screen Overview

When using Horner APG graphics-based controllers, you have 1023 screens to use in your program. There is not a built-in way of scrolling through these screens in graphics-based controllers, so screen manipulation must be done either through ladder logic or through the "Screen Jump" object in the graphics editor… or through a combination of both.

When writing a program, planning is needed to determine what screens need to be seen and when they need to be seen. Many times, a dedicated alarm screen is used so that, when the alarm occurs, the alarm screen can be forced on. Or perhaps there is a main menu screen that has links to configuration or data monitoring screens.

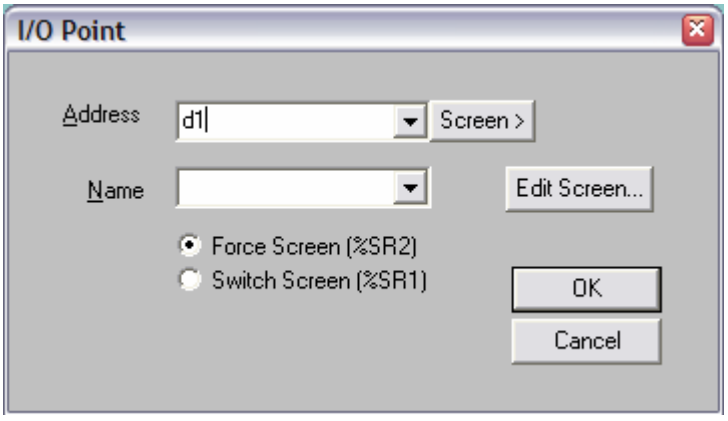
There are dedicated %D bit-length registers that are numbered the same as the screen they represent. %D1 is for screen 1, %D312 is for screen 312, and so on.

There are also three system registers that reflect exactly what the display is currently doing. These registers are the %SR1 User Screen register, the %SR2 Alarm Screen register, and the %SR3 System Screen register. The numbers in these registers reflect the number of the screen currently displaying. %SR2 takes priority over %SR1 and %SR3 takes priority over %SR2.

Part 1 – Switching and Forcing

%D registers can be used as a contact that indicates a screen being on or off. They can also be used as coils to control screens. When using them as coils, simply place a normally open coil in your ladder logic and then configure it for a %D register.

Once a coil is configured for a %D register, the configuration box changes to account for some additional options:



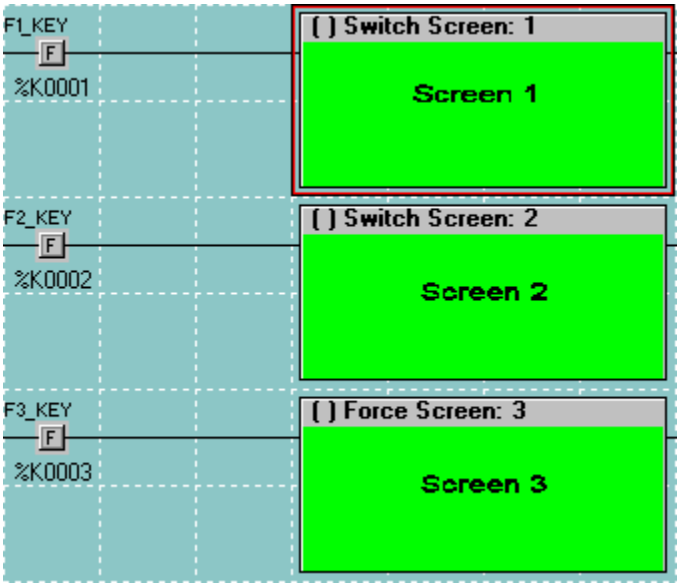
From this configuration box, there is the added ability to click the 'Edit Screen…' button and go directly to the screen editor for the screen specified. There is also an option of using this coil to Force the screen or Switch the screen.

Lab 3: Screen Manipulation

When forcing a screen, the screen will be forced to display for as long as the coil has power. The screen number of the screen being forced is also reflected in %SR2. If a screen is forced, the value in %SR1 is not affected and stays the same as it was. When the screen is no longer forced, the controller will return to the screen reflected in %SR1. For example, if screen 51 is being displayed and an alarm occurs that forces screen 20, %SR1 will have a value of 51 and %SR2 will have a value of 20. When screen 20 is released from its force, %SR1 will still have a value of 51 and %SR2 will have a value of 0.

When switching a screen, the screen specified by the %D register will be switched to and will stay there even after power to the coil has been lost. This change is reflected in %SR1. For example, if screen 51 is being displayed and the screen is switched to screen 30, %SR1 will change from 51 to 30.

1. Create a new program for the NX. Configure the I/O as shown in Lab 1.
2. Using the screen editor (Screens menu, View/Edit Screens…, or click  on the toolbar), put a Static Text label on screen 1 that says “Screen 1”. Do the same for Screen 2 and Screen 3 with Static Text labels that say “Screen 2” and “Screen 3”.
3. Exit the screen editor and save the file.
4. Add ladder logic so that the F1 key will SWITCH to Screen 1 and the F2 key will SWITCH to Screen 2. Note that when you are configuring the coil, you can click the ‘Screen>’ button and choose the screen to associate the coil with from the thumbnails shown. The %D address will automatically fill in this way.
5. Add ladder logic so that the F3 key will FORCE screen 3.



6. Save and download the program to the controller.

Screen 1 should display after the program is downloaded. Press the F2 key and note how the switch screen works. Press F1 to switch back to screen 1. Press F3 and watch how screen 3 will be on only for as long as you hold down the button. When you

let go, the screen you were previously viewing will come back up. Try pressing F3 from both screen 1 and screen 2 to see this. Open a datawatch window in Cscape and add %SR1 and %SR2 as INT values to watch the system registers and what they do when the buttons are pressed.

Part 2 – Changing the System Registers

Another way of displaying a screen is to directly move a value into one of the screen system registers. Using a simple Move function (Move functions are gone over in detail in an upcoming lab), a value representing the screen number can be moved to %SR1 to switch the screen. Cscape programming does not allow the user to write a value to %SR2 to change the screen. To turn on Alarm Screens (%SR2), either the %D coil for that screen will need to be specified as an Alarm Screen and forced on or the logic Alarm handler can be used.

Part 3 – Screen Jumps

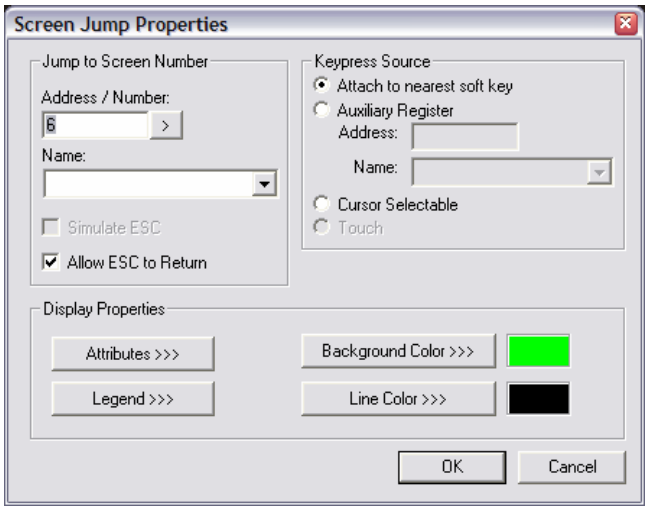
A way of letting a user change screens from the screen itself without involving any ladder logic is to use Screen Jumps. This is an object that is placed on the screen just like any other object or data field on the screen. A screen number is specified as the screen to jump to.

One advantage to using Screen Jumps, in addition to not having to program ladder logic to do it, is that a menu structure can be simulated. When configuring a Jump Screen, there is an option to “Allow ESC to Return”. With this option checked, using the jump keeps the last page on a stack so that it can be recalled. Pressing the ESC key on the NX will recall the page from which the current page was jumped to. On the LX and other touch-screen controllers that don’t have an ESC key built in, a screen jump is configured and the option to “Simulate ESC” is checked. Up to 16 “layers” can be recalled in order to back up through a menu system.

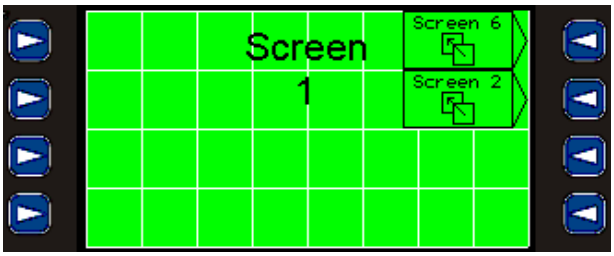
Adding to the program from Parts 1 and 2:

1. Add screen 6 with a Static Text label as done with the other screens.
2. Go to screen 1 and add a Screen Jump  . On NX controllers, this will be linked to the nearest softkey. The softkeys are the buttons on the side of the screen with arrows on them.  They can be linked to on-screen objects. On LX controllers, the Screen Jump will be a pushbutton on the touch-screen.
3. Double-click the Screen Jump and configure it to jump to screen 6. Check the “Allow ESC to Return” box. Change the Legend to something meaningful.

Lab 3: Screen Manipulation



4. Add another Screen Jump to go to screen 2. Do NOT check the "Allow ESC to Return" option. Change the Legend. Your screen might look something like this:



5. On screen 2, configure the same Screen Jump to screen 6 (copy and paste it from screen 1 if desired). Make sure “Allow ESC to Return” is checked.
6. Add another Screen Jump to go to screen 4. Again, do NOT check “Allow ESC to Return” for this jump.
7. Repeat step 5 and 6 for screen 4. Make the second Screen Jump go to screen 1.
8. If using the NX controller, skip to step 10.
9. For the LX controller, go to screen 6 and put in another Screen Jump. Double-click on it to configure it and check the “Simulate ESC” box. This is all that is needed for this jump.
10. Exit the screen editor and save the program.
11. Download the program to the controller.

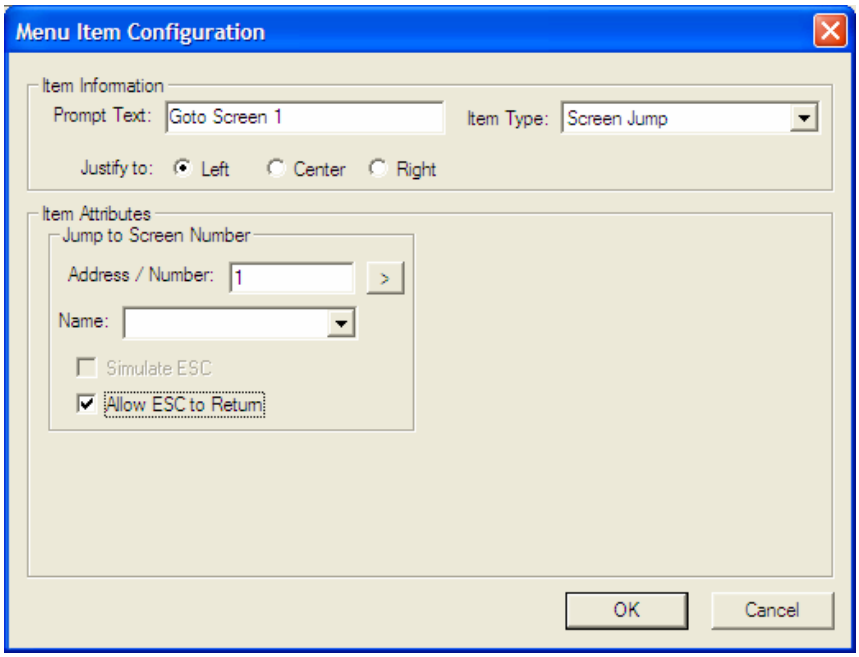
Screen 1 should display after the program is downloaded. Press the Screen Jump to go to screen 2, then to screen 4 and back to screen 1. On any of those screens, press the Screen Jump to go to screen 6. When screen 6 is displayed, pressing the ESC key on the keypad (or the pushbutton on the touch screen controllers) will return to the screen that screen 6 was called from.

Part 4 – Menu Object

On XLE controllers, the display is limited to 2 soft keys on each side of the display thus limiting the size and the number of items placed on the screen. To overcome this obstacle, the Menu Object can be used. The Menu Object has many features but for our lab, we will be using it for screen manipulation.

1. Connect to the XLE using the DB9 to RJ45 cable. The cable will be connected to MJ1 port located on the right side of the XLE.
2. Open Cscape and start a new program.
3. Change the Target ID to 4 in Cscape. This done by pressing the Bulls Eye at the top of the tool bar or by selecting Controller from the menu and then selecting Set Network Target ID.
4. Configure the Controller and the I/O like in the previous labs.
5. Open the Graphics Editor and configure screens 1 – 6 to indicate ‘SCREEN #’ using the Static Text like before. # will indicate the number of the screen that is being configured so replace # with a 1 on screen 1, 2 on screen 2, and so on.
6. Now configure screen 7 for a menu object. The Menu Object is the eight object form the left in the second row of the tool bar.
7. Stretch this to cover the entire screen.
8. Open the Menu Object and click on Configure Menu Pages.
9. Press Add.

10. In the Prompt Text, type ‘ Goto Screen 1’, select Screen Jump in the Item Type, under the Item Attributes put a 1 in the Address/Number, check the Allow ESC to Return, and press OK. The picture below is what the configuration should look like.



11. Go back to the Main Menu of the Menu object and repeat the steps for screens 2-6.
12. Configure screen 7 as the initial screen by clicking on the Screens Menu of the graphics editor and selecting Set Initial Screen and putting a 7 into the First Screen to display box.
13. Change the Legend of the Menu Object to reflect ‘ Screen Manipulation’.
14. Once done, close the graphics editor and download the program to the XLE.
15. Once done downloading the program, test the program by using the up and down arrow keys on the XLE to change the highlighted selection and pressing the Enter Key. The ESC key will allow you to navigate back to the Menu Screen.

Extra Credit #1

On NX or XLE controllers, pressing the up and down arrows simultaneously gets into the system menu. LX controllers have a System key on the keypad.

Add ladder logic to either your NX program or your XLE program to lock out the System key on the controller. %SR3 contains a number reflecting the system screen currently displayed. If none is shown, %SR3 contains a 0.

You can monitor %SR3 for a non-zero value and, if it is non-zero, move a zero back into it. Use a compare function to compare the value to zero. Compare functions will pass power to the rest of the rung if they are true.

This feature can be used to limit access to the parameters of the System Menu like placing the controller in Stop, Changing the ID, Changing the Network Baud Rate, etc…

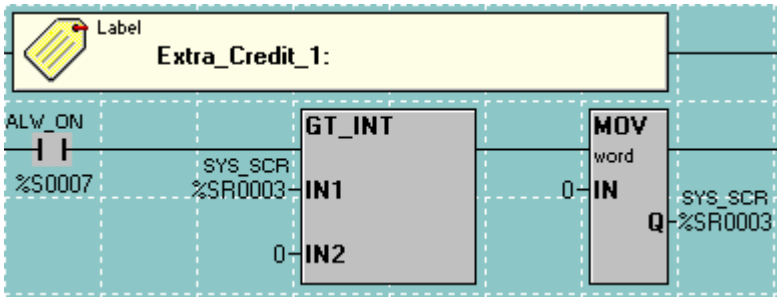
Extra Credit #2

Create ladder logic to scroll through the screens on the NX controller using the up and down arrow keys on the keypad.

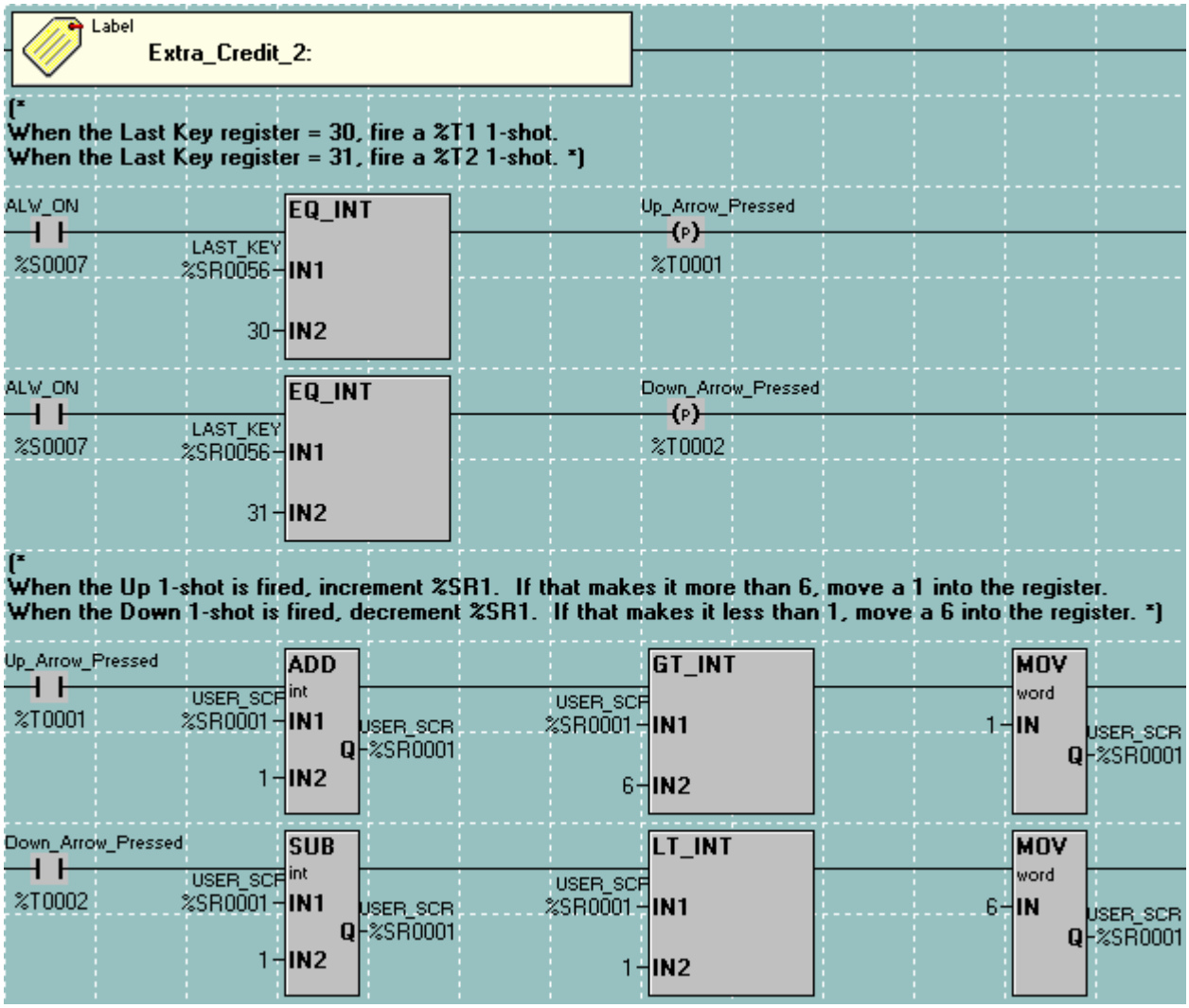
%SR56 is the 'Last Key' register and reflects a value unique to the button being pressed. The Up key is a value of 30 and the down key is a value of 31.

If %SR56 is equal to 30, increment %SR1 by 1 using an ADD Math Operation. If it is 31, decrement it by 1 using a SUB Math Operation. Be careful when pressing the Down key from screen 1… you'll have to put a value of 6 in %SR1 to "wrap around". Be careful when pressing the Up key from screen 6… you'll have to put a value of 1 in %SR1 to "wrap around". If you want to skip any screens in between 1 and 6, you'll have to program that in, too. And one last warning… Math Operations will take place on every scan if they are powered. Positive Transition (1-shot) coils will have to be thrown into the mix.

Solutions



Lab 3: Screen Manipulation



Lab 3: Screen Manipulation

Notes:

Lab 3: Screen Manipulation

Notes:

LAB 4

Timers and Counters

Lab 4: Timers and Counters

Lab 4: Timers and Counters

Objective:

Review and understand Timers and Counters.

Timers Overview:

The purpose of the Timers portion of this lab is to show how each type of Timer operates and what the difference is between them. Also, using built-in status bits in the Timer registers can be useful in many cases instead of using additional coils in the ladder logic.

Note: You will almost ALWAYS use %R registers for Timers and Counters. Also, Timers and Counters always use 2 consecutive word-length registers!

REMEMBER! If a timer is addressed to %R1, then %R2.15 will indicate whether the timer is receiving power (for Counters and TON Timers only). %R2.16 will indicate whether the timer is passing power to the rest of the rung. In the same way, if addressed to %R846, then %R847.15 and %R847.16 are those status bits.

Part 1 – TON Timers:

1. Create a new NX program.
2. Title the program “Timers.csp”.
3. Set the target ID to match the NX you are going to program.
4. Configure the NX. (Reference Lab 1 for correct procedure)
5. Configure a timer at %R1 that will pass power to a coil, %M1, when the F1 key is pressed and held for 3 seconds or more. Configure the timer for 100ms resolution.

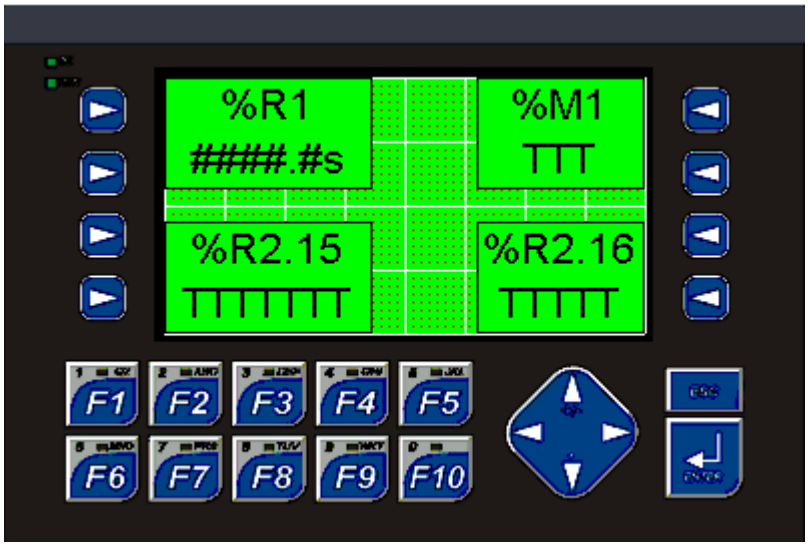
HINT: Since the timer is set for 100ms resolution, 3 seconds is equal to a “Pt” of 30. 30 100ms pulses equals 3 seconds.

6. Configure a text table on the screen to show ‘Off’ or ‘On’ depending on the state of %M1.
7. Configure a second text table to show ‘Off’ or ‘Enabled‘ depending on the state of %R2.15. %R2.15 will reflect whether or not the Timer is currently enabled.

HINT: There are up to 200 text tables to use. By default, every new text table field you make references text table 1. You will have to make new text tables and point the new text table to the appropriate table number.

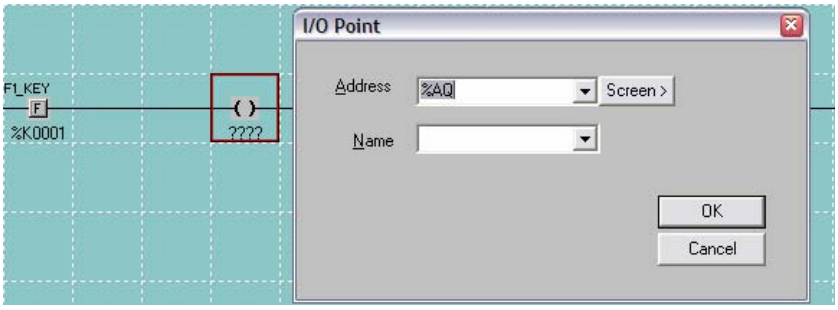
8. Configure a third text table to show ‘Off’ or ‘Power’ depending on the state of %R2.16. %R2.16 will reflect whether or not the Timer is currently passing power to the rest of the ladder rung. It will pass power when the timer is done timing.
9. Configure a Numeric data field that displays how much time has elapsed in the timer. This will be the accumulated value of the timer, %R1. Configure the data field to be un-editable and displaying a length of 5 with 1 decimal place.

10. Label each field on the screen so you can tell them apart. This can be done by modifying the legend for each of the objects placed on the screen. Your screen might look something like this in the graphics editor:

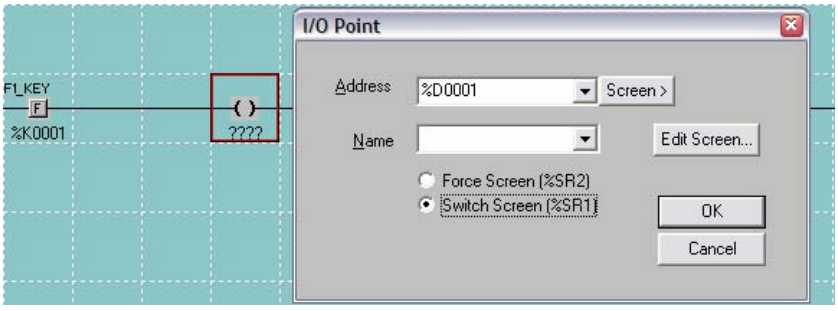


11. If this data is on a screen other than screen 1, you will need to add ladder logic or a Screen Jump to switch to this screen or back to it once you have switched away from it.

- Add a normally open contact addressed to the F1 key (%K1).
- Use it to fire a Switch Screen addressed to the screen above. Place a normally open coil and double-click on it to configure it. Click the 'Screen>' button and choose this screen from the thumbnails.



- Once the screen is chosen, make sure to specify the 'Switch Screen' option and click OK.



12. Download the program to your controller and make sure it is in RUN mode.
13. Compare the operation of %R2.15 (“Off” or “Enabled”) to the operation of the F1 key. They should be the same.
14. Compare the operation of %R2.16 (“Off” or “Power”) to %M1. They should be the same. You can use %R2.16 in place of %M1 in programming.
15. Watch the value in %R1 to see the accumulated time when F1 is pressed. Letting go of F1 before the 3 seconds is up will cause the timer to automatically reset to 0.

Part 2 – Retentive TON Timers

1. To the above program, add a timer that times and keeps track of how long the F2 key has been pressed. After the total has reached 5 seconds, the timer should pass power… unless the F10 key is pressed to reset the accumulated time. Configure the Timer for 10ms resolution.

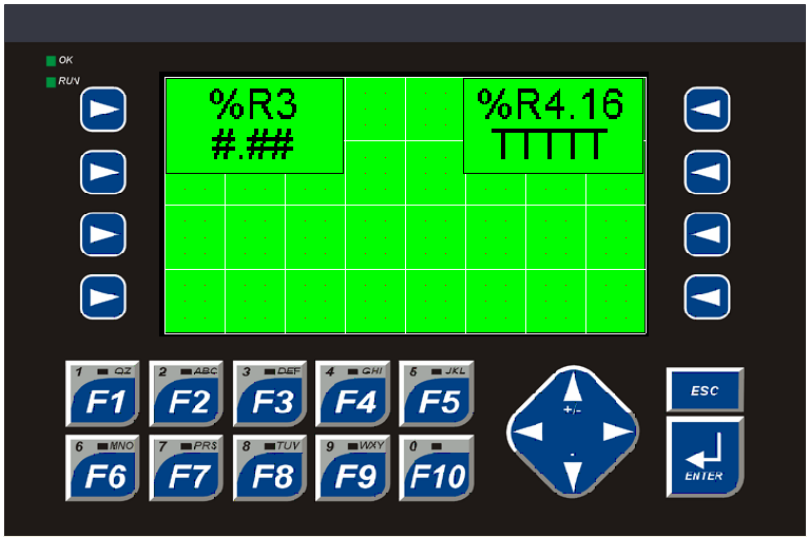
HINT: Since this timer is set for 10ms resolution, 5 seconds is equal to a ‘Pt’ of 500. 500 10ms pulses equals 5 seconds.

HINT: Remember; each Timer or Counter takes 2 word-length (%R) registers. The timer from Part 1 takes up %R1 and %R2. Don’t overlap this timer with that one!

2. Configure a text table on the screen to show ‘Off’ or ‘On’ depending on the state of the Timer “Passing Power” status bit. (You may have to start a new screen.)

HINT: Since you have already created a text table for %R2.16 with ‘Off’ and ‘Power’, you can link this timer’s status bit to the same text table. Two different registers can use the same text table.

3. Configure a numeric data field that displays how much time has accumulated in the timer. Configure the numeric field for un-editable, 3 digits, and 2 decimal places.
4. Label each field using the legend so that you can tell them apart. Your screen might look something like this:



5. If the screen information is on a screen other than Screen 1, add more logic or a Screen Jump to switch to the appropriate screen.
6. Download the program to your controller and make sure it is in RUN mode.
7. Press the F2 key and watch the time increment. Let go before the 5 seconds is up and the time should stay where it is. Pressing F2 again will resume where it left off. You will have to press F10 to get the timer to restart at 0 again.

Part 3 – TOF Timers

1. To the above program, add a timer that will immediately pass power when the F3 key is pressed and will keep passing power for 5 seconds after the F3 key is released. Configure this timer for 100ms resolution.

HINT: Remember not to overlap the timers! Use the Timer Status bit to determine when the timer is passing power.

2. Configure a text table on the screen to show 'Off' or 'Power' depending on the state of the timer's status bits to show when the timer is passing power. (You may have to start a new screen and, remember, you can re-use the text table you already have for 'Off' and 'Power'.)
3. Label everything:

4. Add logic or a Screen Jump to switch to this screen if needed.
5. Download and make sure the controller is in RUN mode.
6. Notice how this timer shows 5.0 seconds in its accumulated time when it is inactive, but the status bit shows 'Off' because power is not being passed.
7. Notice how the accumulated time goes to 0.0 when you press the F3 key and the status bit shows 'Power' immediately.
8. Notice how the accumulated time then starts counting when you let go of the F3 key and how the status bit still shows 'Power' to the rest of the ladder rung, even though power to the rung has been interrupted.
9. Notice how power is discontinued when the timer reaches its 5 seconds.

Lab 4: Timers and Counters

Counters Overview:

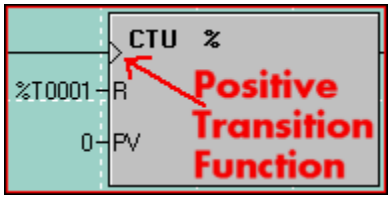
The purpose of the Counters portion of the lab is to demonstrate how Counters work and what the difference is between a Count-Up (CTU) Counter and a Count-Down (CTD) Counter.

Count-Up counters reset to 0 and count up from there, passing power when they reach their preset value (PV).

Count-Down counters reset to their preset value (PV) and count down from there, passing power when they reach 0.

Status bits in the Counter's second register work the same way as the Timer's status bits.

Counters increment or decrement only once every time they see power come on from the ladder rung. This is what the little triangle at the counter input means:

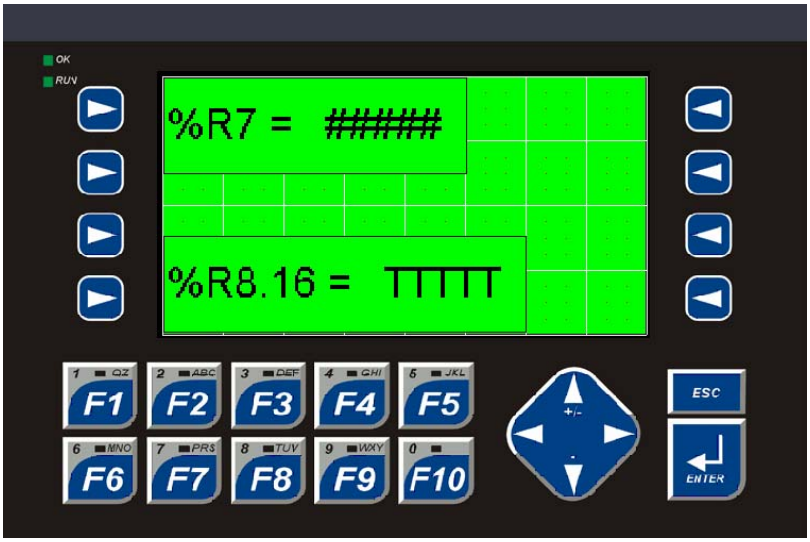


Part 4 – CTU Counters

1. To your program, add a counter that will count the number of times the F4 key has been pressed. If F4 is pressed a total of 4 times or more, power should be passed to the rest of the rung.

HINT: Just like Timers, Counters also take up 2 word-length (%R) registers. Don't step on any of your timers!

2. Make the F10 key reset the counter.
3. Create another screen with a numeric data field and a text table to show the counter's accumulated count and its status bit to let you know whether or not it is passing power. Your screen might look something like this:



4. Add logic or a Screen Jump to switch to this screen if needed.

5. Download and make sure the controller is in RUN mode.
6. Press the F4 key and watch the count increment. When it reaches 4, power should be passed. Pressing F10 will reset the counter regardless of where it is in the count.
7. Notice how the counter continues to count past its preset value if you keep pushing the F4 key. It will continue to count and will also pass power until it is reset.
8. Notice how the counter's status bit acts the same as the timer status bit.

Part 5 – CTD Counters

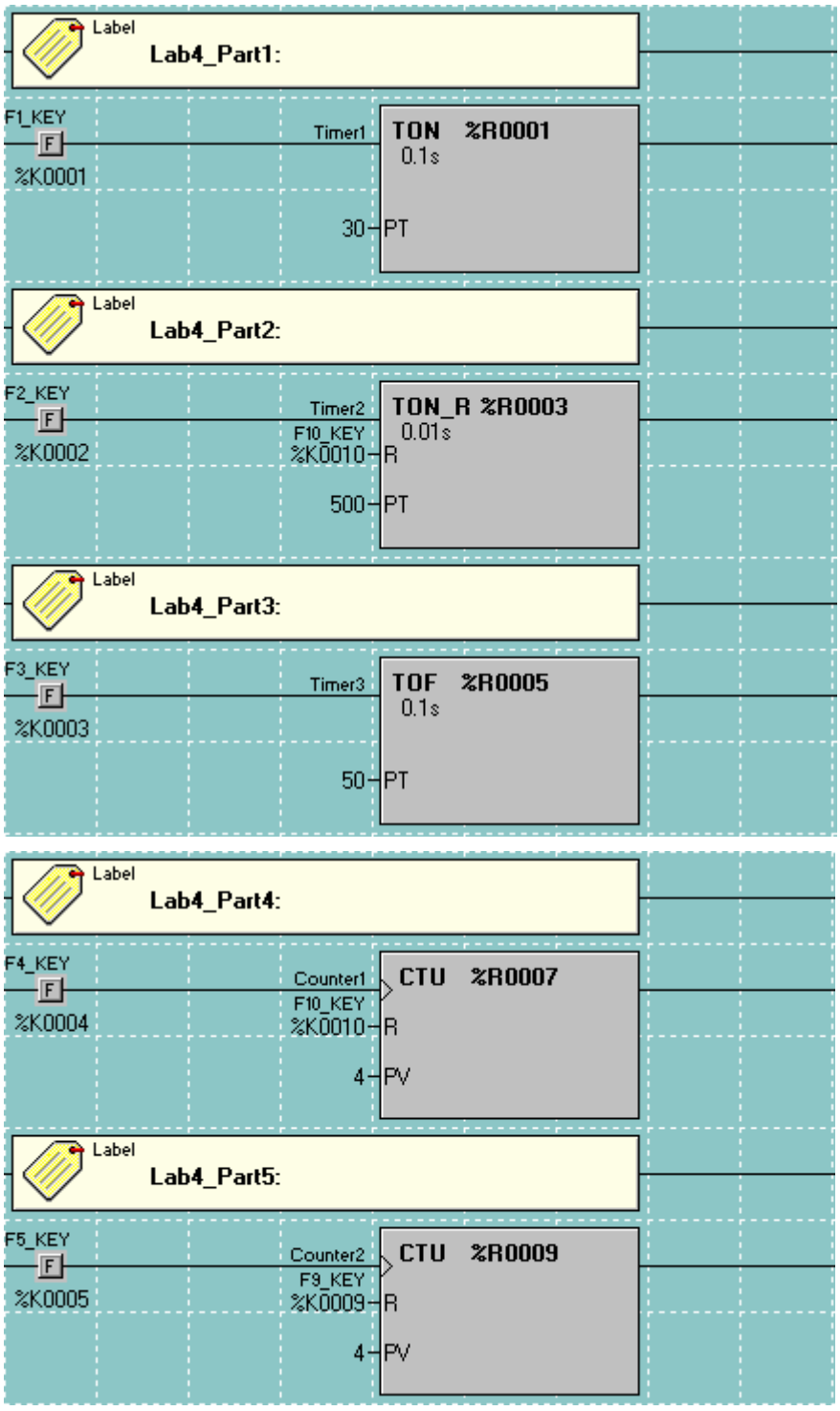
1. To your program, add another counter that will count the number of times the F5 key is pressed. However, use a CTD counter to count down from 4. Use the F9 key to reset the counter.
2. Create another screen to monitor this counter, just like you have for all the other timers and counters. However, make sure the numeric data field for the accumulated count is set up for 6 digits and make sure it is set up for a 'Signed Decimal' display format:

3. Add logic or a Screen Jump to switch to this screen if needed.
4. Download and make sure the controller is in RUN mode.
5. Make sure the counter is reset by pressing the F9 key. Notice how it resets to the preset value.
6. Press the F5 key and watch the count value decrement. When the count reaches 0, power will be passed.
7. Notice how the counter will continue to decrement past 0. Depending on how you have your data field set up, it will either show -1, -2, -3, etc. (Signed Decimal display format), or it will show 65535, 65534, 65533, etc. (Decimal display format, also known as Unsigned Decimal).

CONGRATULATIONS, YOU'VE FINISHED TIMERS AND COUNTERS!

Lab 4: Timers and Counters

Solutions:



Lab 4: Timers and Counters

Notes:

LAB 5

Move Operations

Lab 5: Move Operations

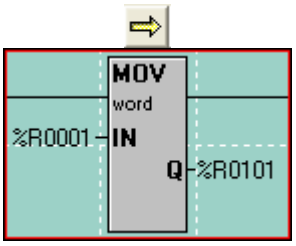
Lab 5: Move Operations

Objective:

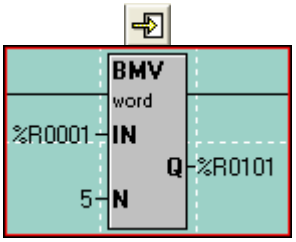
Review and understand Move Operations

Overview:

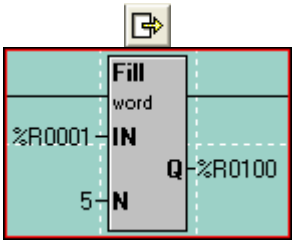
There are several types of Move functions available for use for several types of different occasions. The 'Move Operations' toolbar appears as follows:



The first type of Move is the 'Move Word', or 'MOV'. It is used to copy a single byte, word or double-word from one location to another. The count is locked at 1. In the case of the example to the left, the value in %R1 is copied into %R101. This only happens when the ladder rung receives power. The value in %R101 is NOT taken back out when power is lost to the rung. The IN can be either a register or a constant value.

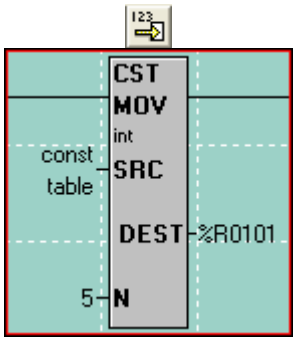


The next type of Move is the 'Move Data Block', or 'BMV'. It is used to copy a group of bytes, words or double-words to another location. The count (N) determines how many registers are to be copied. In the example to the left, %R1-%R5 are copied into %R101-%R105. Again, this only happens when the ladder rung receives power. The IN must be a register reference and constant values are not allowed.

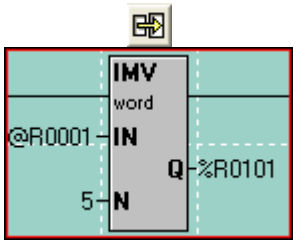


The next type of Move is the "Fill WORD", or "Fill". It is used to copy the contents of a single register or value into multiple other registers, thus filling that one value into a group of registers. The count (N) determines how many registers to fill that single value into. In the example to the left, the value in %R1 is copied into %R101-%R105 so that %R101-%R105 all will have the same value in them. This can be used to zero-out a group of registers. The IN can be either a register or a constant value.

Lab 5: Move Operations

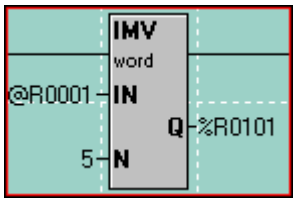


Skipping to the 'Constant Move', or 'CST MOV', it is used to move a group of constant values into a group of consecutive registers. If, for example, you want to move the values 1, 2, 3, 4 and 5 into %R101, %R102, %R103, %R104 and %R105, respectively, then you can use the Constant Move function. The count (N) is automatically determined by how many constant values you enter into the configuration for this function. The source data can ONLY be constant data and cannot be register references.

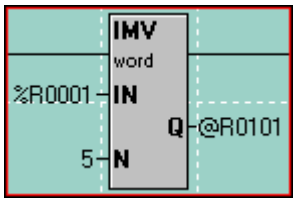


Moving back one to the 'Indirect Move', or 'IMV', it is used to move data from variable positions or to variable positions or both. It functions, for the most part, like the Block Move function. If specified as Indirect, the IN and/or the Q are used as pointers to where in the %R registers to get data from or put data to. When looking at the ladder logic, the @ symbol will appear next to the IN or Q address if it is specified as Indirect. This function can and most likely will get hairy to the uninitiated. It is most handy, though, when data-logging to register memory.

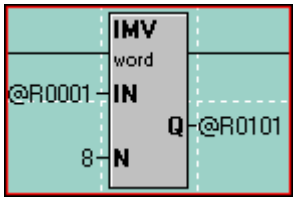
Indirect Move Examples



In this example, the IN is specified as Indirect. This means the controller will look at %R1 and see a value within it. If %R1 has a value of 501 in it, the controller will go to %R501 to get the source data. 5 registers will then be moved from %R501-%R505 to %R101-%R105.



In this example, the Q is specified as Indirect. This means the controller will look at %R101 and see a value within it. If %R101 has a value of 851, the controller will take the data in %R1-%R5 and move it into %R851-%R855.



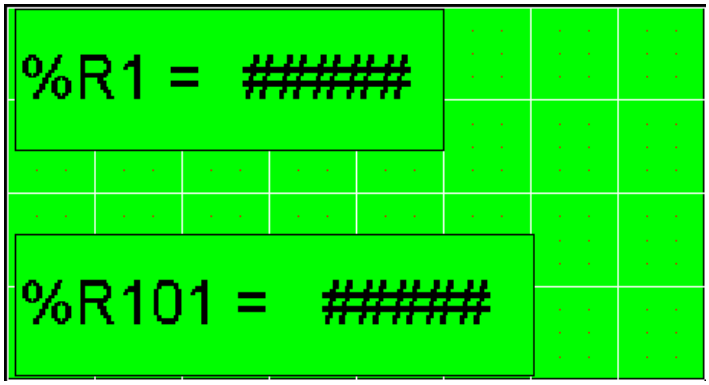
In this example, the Mother of All Confusion, both the IN and the Q are specified as Indirect. This means the controller will look at %R1 and see a value. Let's say it is 241. The controller also looks at the value in %R101. Let's say it is 341. The controller will then take the values in %R241-%R248 and move them into %R341-%R348.

Confused yet? Good, let's get on with the lab.

Lab 5: Move Operations

Part 1 – Move

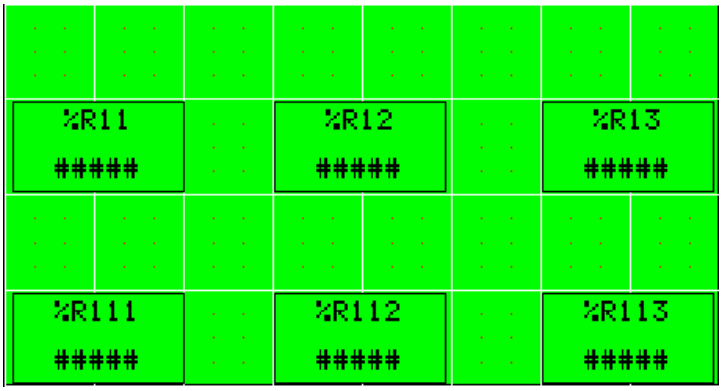
1. Start a new program for the controller you are connected to and call it whatever you want. Configure the controller and I/O as you have done before.
2. Move the value in %R1 to %R101 when the F1 key is pressed.
3. Move the value of 0 into %R101 when the F2 key is pressed.
4. Create a screen with numeric data fields that show %R1 and %R101 and label the fields. Be sure to make the %R1 data field editable:



5. Add logic or a Screen Jump to switch to this screen if needed.
6. Download the program and make sure the controller is in RUN mode.
7. Edit the value in %R1 to whatever you like by pressing the Enter key when the field is outlined, typing in a value on the NX keypad, and then pressing Enter again.
8. Press the F1 key to move the value you just edited into %R101.
9. Press the F2 key to move a value of 0 into %R101.

Part 2 – Block Move

1. Add programming to move the values in %R11-%R13 to %R111-%R113 when the F3 key is pressed.
2. Create another screen with data fields to show the registers. Be sure to make the %R11, %R12 and %R13 data fields editable:

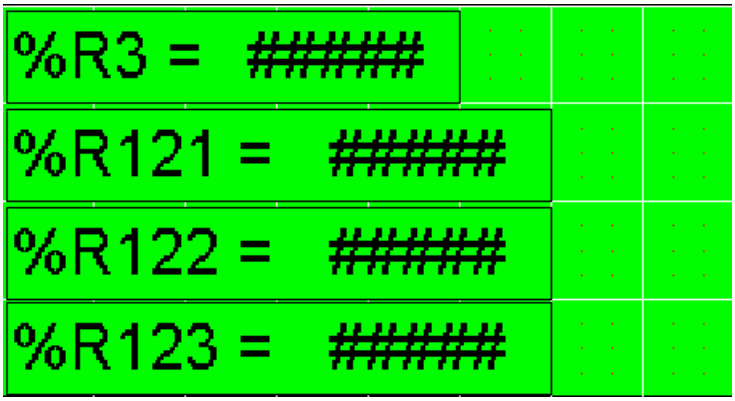


3. Add logic or a Screen Jump to switch to this screen if needed.

4. Download the program and make sure the controller is in RUN mode.
5. Edit the values in %R11-%R13 to whatever you like. Use the arrow keys to select a field, press the Enter key when the field is outlined, type in a value on the NX keypad, and then press Enter again.
6. Press the F3 key to move all the values you just edited in %R11-%R13 to %R111-%R113.

Part 3 – Fill WORD

1. Add programming to fill the value contained in %R3 into all the registers from %R121-%R123 when the F4 key is pressed.
2. Fill those same registers with a value of 0 when the F5 key is pressed.
3. Create another screen with data fields to show the registers. Be sure to make the %R3 data field editable:



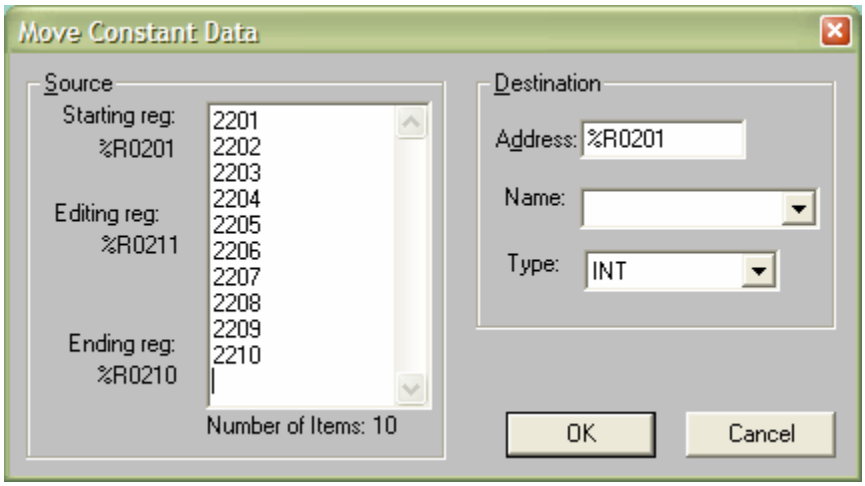
4. Add logic or a Screen Jump to switch to this screen if needed.
5. Download the program and make sure the controller is in RUN mode.
6. Edit the value in %R3 to whatever you like.
7. Press the F4 key to fill the value you just edited into %R121-%R123.
8. Press the F5 key to zero out the values in %R121-%R123

Part 4 – Constant and Indirect Moves

1. Using the Constant Move, add programming that will move the values of 2201-2210 into registers %R201-%R210 on First Scan.

HINT: On your Cheat Sheet, find the %S register that is the system coil for First Scan.

Lab 5: Move Operations

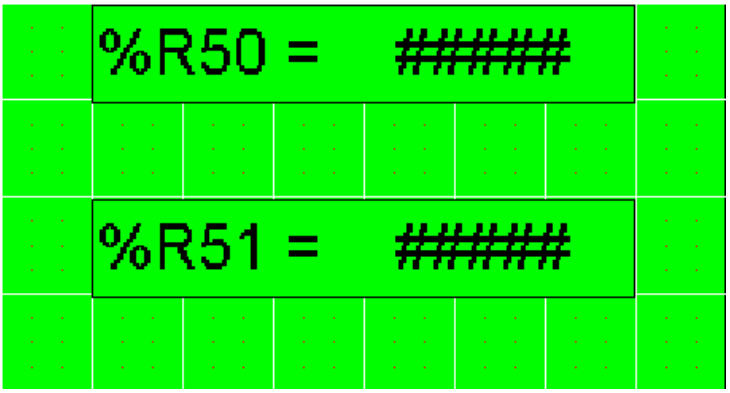


2. Add an Indirect Move to your program that is powered with an Always-On system contact.

HINT: Use the Cheat Sheet to find the Always-On contact!

3. Use the value in %R50 as the "from" address, or pointer. This means you will have to check the Indirect option in the Source area. Use %R51 as the destination register. Do NOT check the Indirect option for the Destination.

4. Create a screen with data fields showing %R50 (editable) and %R51:



5. Add logic or a Screen Jump to switch to this screen if needed.

6. Download the program and make sure the controller is in RUN mode.

7. Edit the value in %R50 to equal something between 201 and 210. You will be able to see the values in %R201-%R210, moved with your Constant Move function, in %R51, based on the value in %R50.

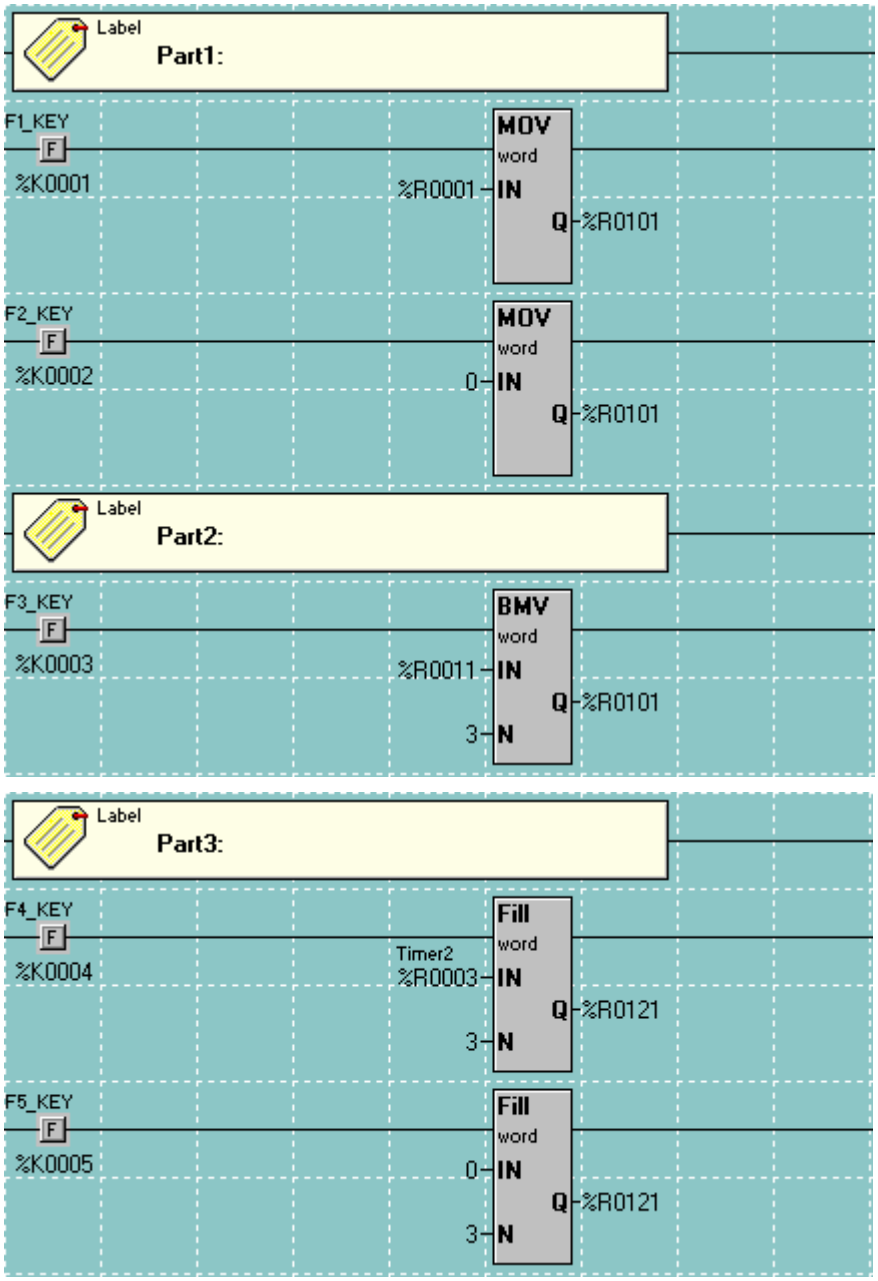
Extra Credit

Use a Move Word function and the F1, F2, F3 and F4 keys to change between your screens in the program. F1 should change to the screen with the Move Word information, F2 should change to the screen with the Block Move information, and so on.

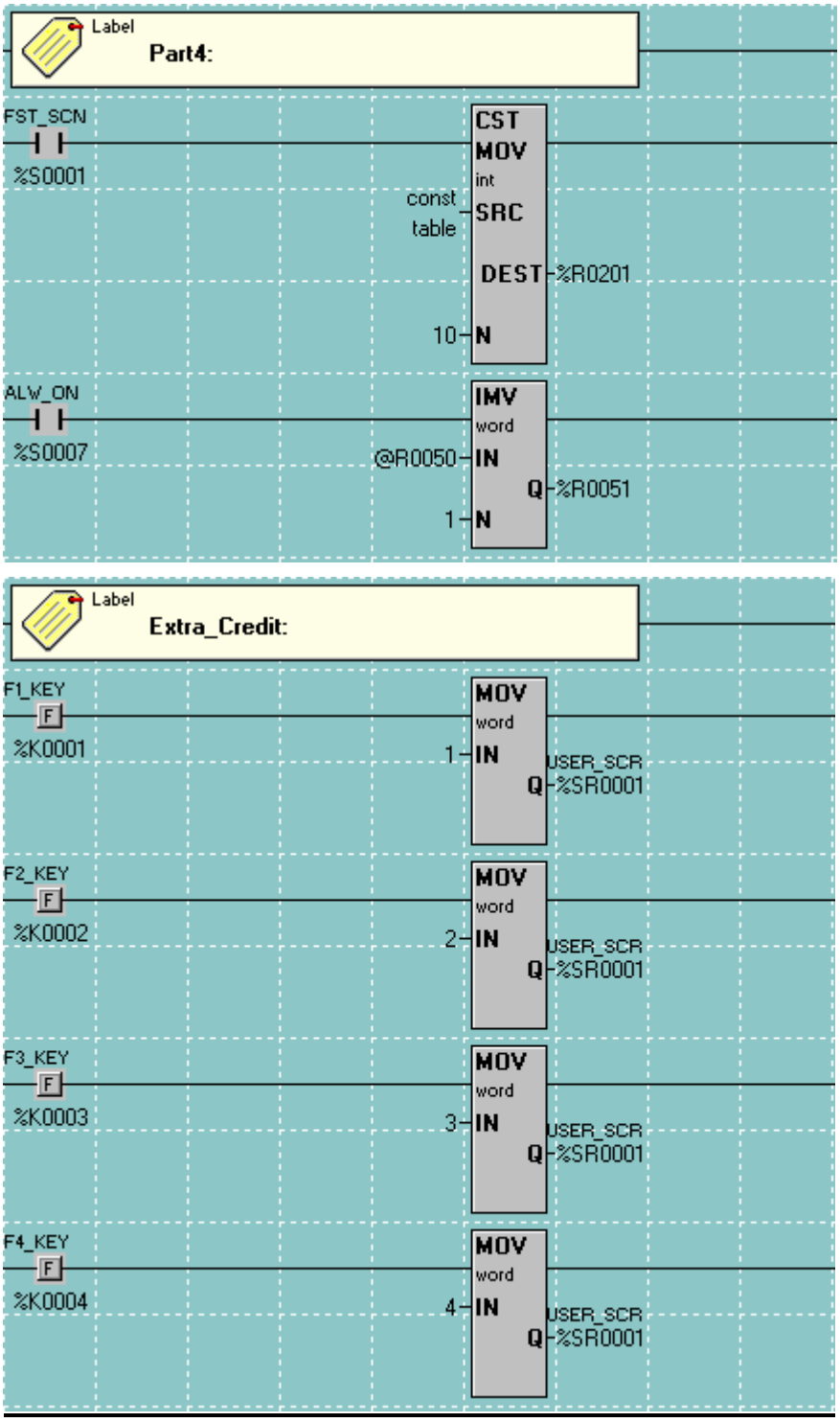
CONGRATULATIONS, YOU'VE FINISHED THE LAB ON MOVE FUNCTIONS!

Lab 5: Move Operations

Solutions:



Lab 5: Move Operations



Lab 5: Move Operations

Notes:

LAB 6

CsCAN Basic Networking

Lab 6: Basic CsCAN Networking

Lab 6: Basic CsCAN Networking

Objective:

Review and Understand global data transfer from OCS-to-OCS over CsCAN.

Procedure:
Part 1 - Analog Data Over CsCAN

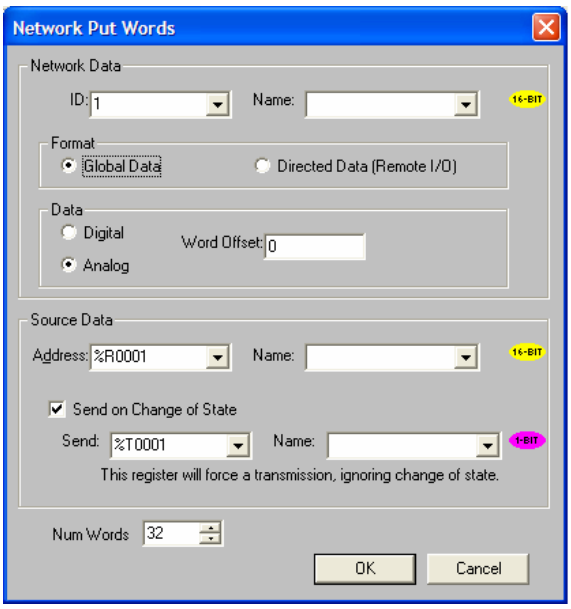
Step 1

- **Create new NX program.**

1. Title the NX program “NX CsCAN”.
2. Set the target to node id 1. Verify through the system menu of the NX251 that the node address is set to 1 and that the baud rate is 125K.
3. Configure the NX251. (Reference Lab 1 for correct procedure)
4. Write a ladder program to increment a counter every 1 second. Assign the counter to %R1. The counter should be configured to count to 200. Use the 16th bit of the second word of the counter, %R2.16, to reset the counter upon the counter reaching the preset value. Remember that the counter will occupy 2 registers so the counter will consume % R1 – R2.

HINT: %S5 is a system register that pules every second

5. Write a line of code to place the accumulated value of the counter out onto the network allowing other nodes on the network to read the information. To perform this task, use an ALW_ON contact, %S7, with a NET_PUT instruction block.



6. Configure a screen to display the accumulated value of the counter. Consult the previous labs for help with this task.
7. Save the program to the PC and then download the program to the NX251.

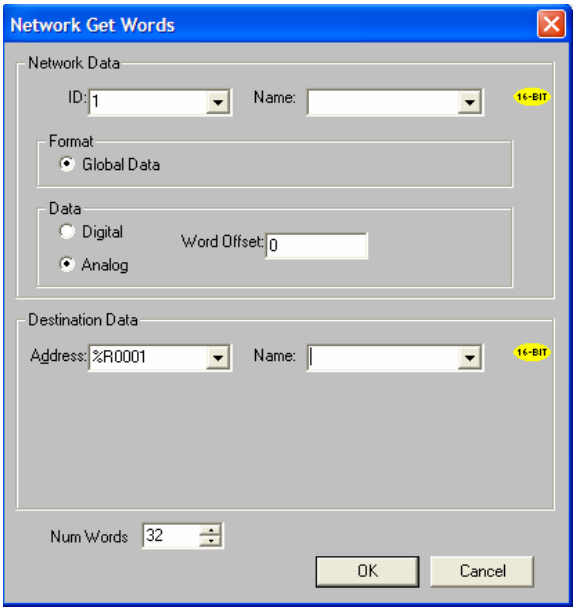
Lab 6: Basic CsCAN Networking

Step 2

- **Create new LX300 program.**

1. Title the LX300 program “LX CsCAN”.
2. Set the target to node id 2. Verify through the system menu of the LX300 that the node address is set to 2 and that the baud rate is 125K.
3. Configure the LX300. (Reference Lab 1 for correct procedure)
4. Configure the network to read the information from node 1 into %R1.

   This will be achieved by using the NET_GET instruction block.



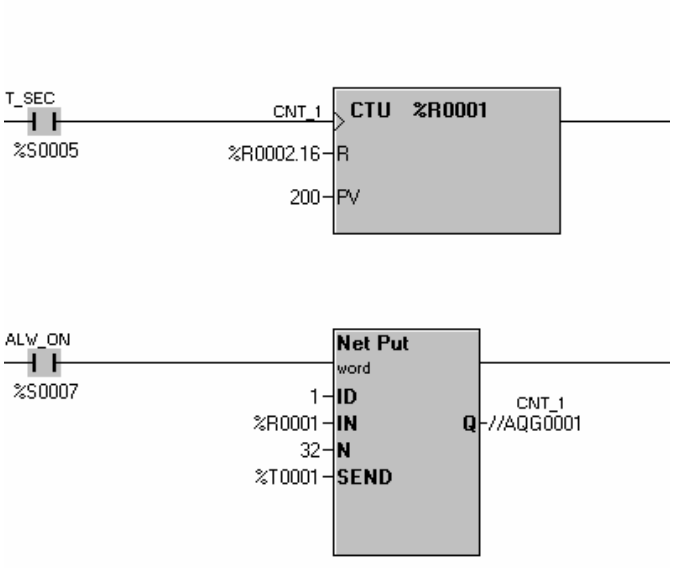
5. Configure Screen 1 to display “Incoming Data” from the NX251.
6. Save the program and then download the program to the NX251.

Step 3

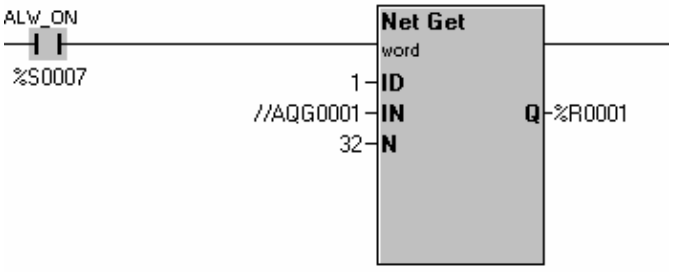
- **Verify the program’s functionality.** The LX300 should display the same value that the NX251 is displaying on the screen.

Lab 6: Basic CsCAN Networking

- **Part 1 Solution.**



NX251 Ladder



LX300 Ladder

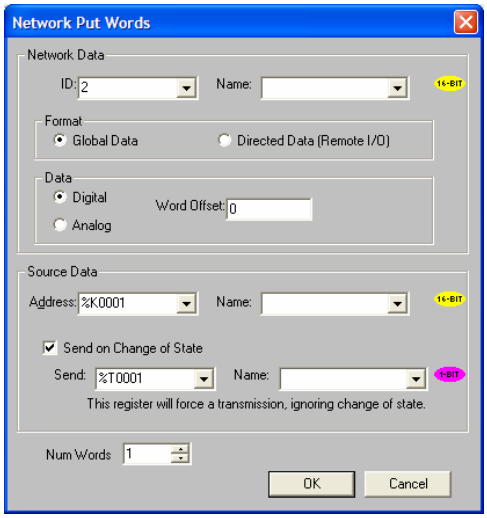
Lab 6: Basic CsCAN Networking

Part 2 – Digital Data Over CsCAN

Step 1

- **Modify LX300 Program**

1. Broadcast the function keys %K1 - %K5 onto the CsCAN network. This will require a NET_PUT instruction block. The instruction block will be configured for node id 2, digital, and the source will be K1 with the number of words equal to 1.

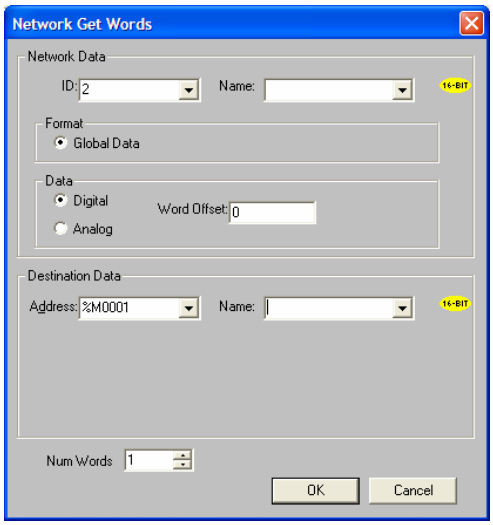


2. Save the program and then download the program to the LX300.

Step 2

- **Modify the NX251 Program**

1. Configure the NX251 to read %K1 - %K5 from node id 2 into %M1 - %M5. Write a line of code that uses a NET_GET instruction configured for discrete from id 2 with the destination of %M1.



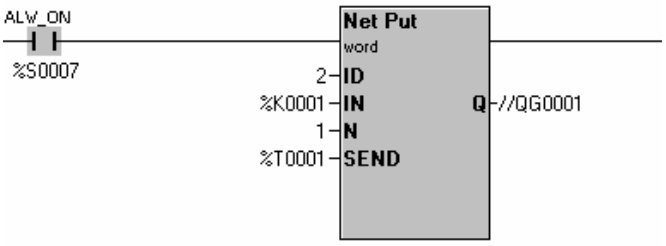
2. Write additional rungs of logic that will turn on %Q1 - %Q5 when %M1 - %M5 (M1 will turn on Q1, M2 will turn on Q2,….etc.) is active. This can either be done through contacts and coils or via a move command.

Step 3

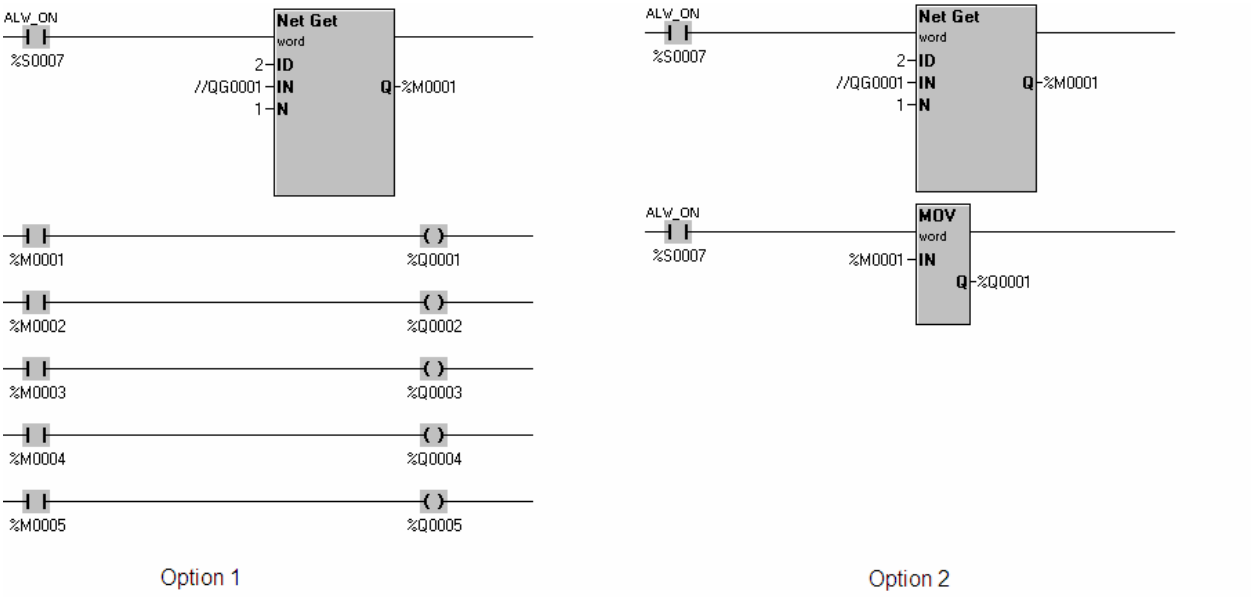
- **Verify the program's functionality.**

When F1 of the LX300 is pressed, LED 1 of the panel, LED 1 that is connected to the NX251 output card, will illuminate.

- **Part 2 Solution**



LX300 Program Addition



NX251 Program Addition

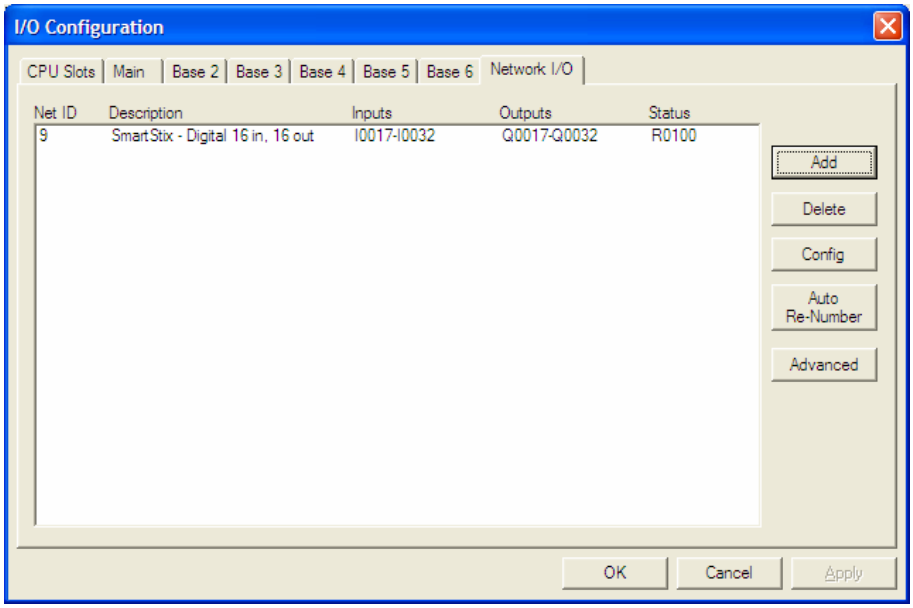
Lab 6: Basic CsCAN Networking

Part 3 – Smart Stix

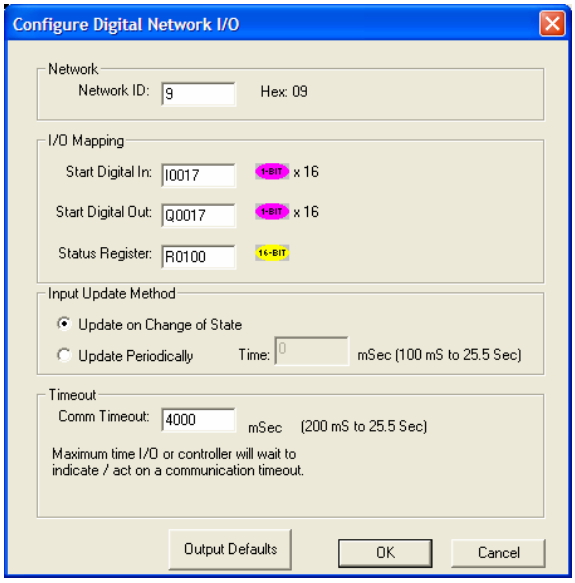
Step 1

- **Modify the NX251 Program**

1. Configure theNX251 for Smart Stix I/O. This is done via the Network I/O tab located in the I/O configuration.



2. Press Add and select “Smart Stix – Digital 16in, 16out.
3. Configure the Stix as illustrated in the picture below:



4. **Press OK**
5. **Press OK**

6. **Modify the NX251 program to turn on %Q17 - %Q25 when the function keys of the LX300 are pressed. Depending on which option you chose in part 2, it will require either changing the address of the coils or changing the destination of the move command.**
7. **Save program and download changes to the NX251.**

Step 2

- Verify functionality**. Press the F1 key on the LX300, the first LED on the Smart Stix should turn on.**

Step 3

- Modify the NX251 Program
  - **Add lines of code that will turn on %Q1 - %Q8 when %I17 - %I24 is active. %Q1 – %Q8 are wired to the LED's on the demo box. The LED's are wired to the Smart Stack module on the back of the NX251. %I17 - %I24 are wired to the Smart Stix at the top right of the demo box.**

Lab 6: Basic CsCAN Networking

Notes:

LAB 7

Color Touch Lab: Screen Creation

Lab 7: Color Touch Screen Configuration

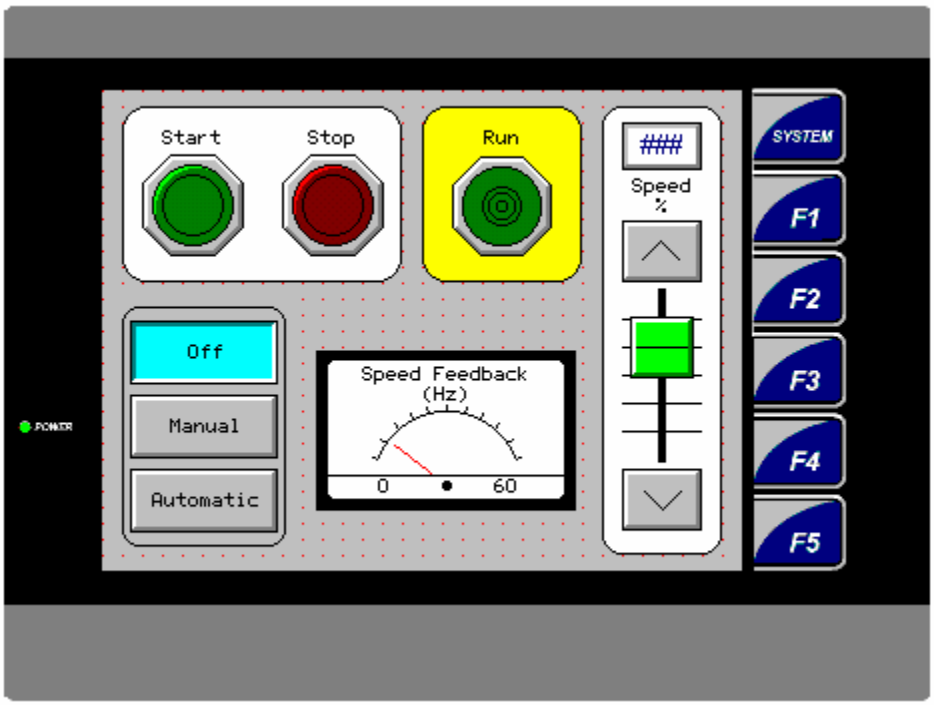
Lab 7: Color Touch Screen Configuration

Objective:

Practice building screens for the LX300.

Connect the serial port of your PC to the LX300. From Cscape, configure the controller using the "Auto Config" function. Remember that the ID of the LX300 is 2.

Create an application screen similar to the following:



Configure the LX300 I/O for the Smart Stix module. This configuration will be similar to the configuration used in the previous lab except use starting addresses as %I1 for the inputs and %Q1 for the outputs.

You will need to change the I/O configuration for the NX251. Only 1 controller can write to the outputs of the Stix at a time, so delete the Stix from I/O configuration used for the NX251 in the last lab and download the change to the NX251.

- Start Pushbutton: Assign to %M1
- Stop Pushbutton: Assign to %M2
- Run Indicator: Assign to %Q1
- Off-Manual-Auto Selector: Assign to %R10
- Speed Feedback (Hz) Indicator: Assign to %R5
- Speed (%) Slider: Map to %R7 (scale 0 to 100)
- Speed (%) Data Box: Assign to %R7

  - Note that the push buttons, slider bar, and the lamp indicators are placed on top of round rectangles. The round rectangles have the fill color modified from the default of none to give the effect shown above.

Lab 7: Color Touch Screen Configuration

Create a simple “Start/Stop” circuit in Ladder with %Q1 (Run) as the Coil.

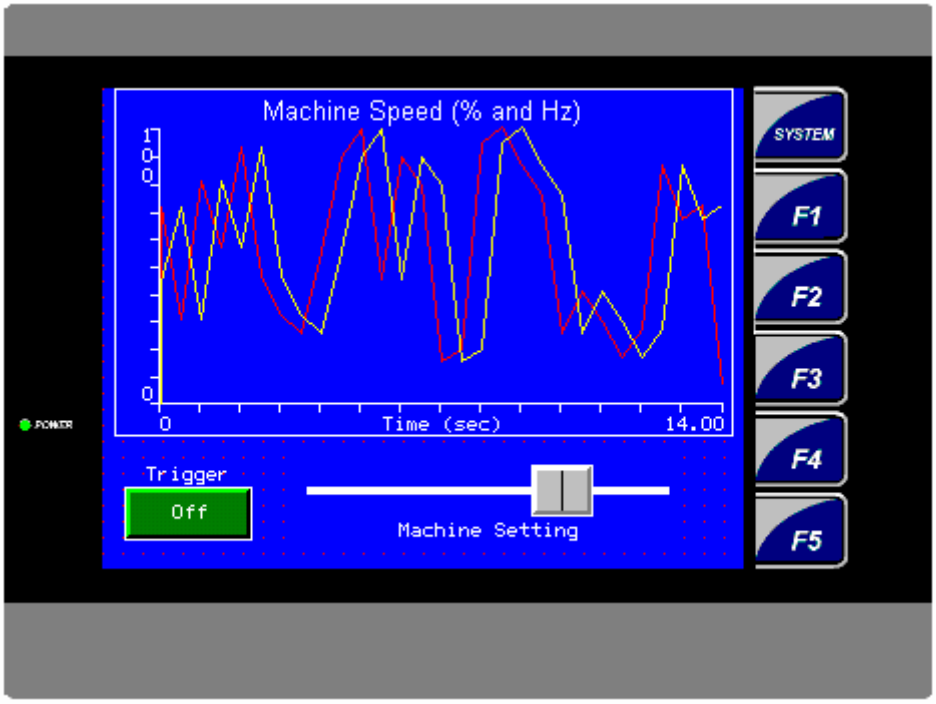
Create a simple rung in Ladder using an Integer Scale Function (look under the “Advanced Math” blocks). The scale function is always on, and should scale input %R7 (0 to 100) to output %R5 (0 to 60).

Run the Ladder Logic, making sure that the logic runs as desired.

Now change your ladder so that the “Run” circuit will not be enabled unless the LX is in “Automatic” mode.

Hint: Automatic Mode is selected when R10 = 2.

Create a new screen, similar to the following:



- Trend Type: 50mS Continuous Scope
- Trend Trigger: Assign to %T1
- Trend Pens: Pen 1 - %R7 scale 0-100. Pen 2 - %R5 scale 0-60.

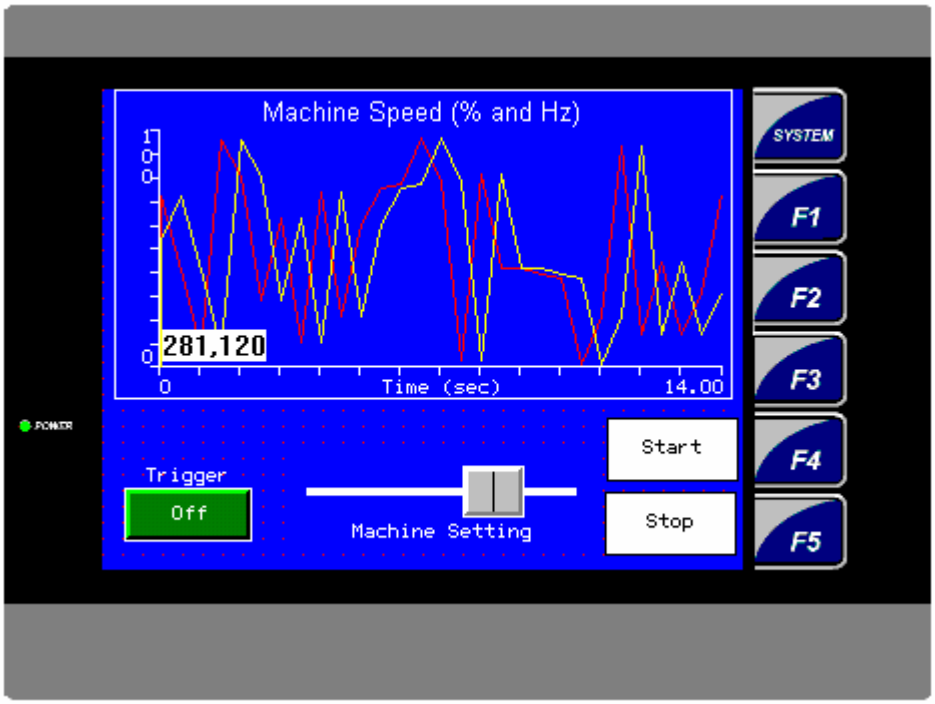
  Don’t forget to Enable Pen 2.
- Pushbutton (“Trigger”): Assign to %T1 – Configure the Pushbutton for Toggle Mode.
- Scale: Assign to %R7 scaled 0-100.

Add logic to your ladder that uses F1 to call up the control panel screen, and F2 to display the trend screen. This should work regardless of which screen is currently being displayed.

Download and test the program.

Lab 7: Color Touch Screen Configuration

Extra Credit: Change your trend screen, similar to the following:



This screen adds "Start" and "Stop" capabilities to this screen. The "Start" and "Stop" Static Text legends on the above screen are only that – legends for the function keys next to them. The user must press F4 to activate the Start Functionality, and F5 to activate the Stop functionality. Use the same Start (%M1) and Stop (%M2) bits previously assigned.

More Extra Credit: Make the "Start" legend visible only when the run circuit is off. Make the "Stop" legend visible only when the run circuit is on. Use the dynamic display properties, accessed through the "Attributes" button when configuring the Static Text legends. Check the 'Visible' box under the Dynamic heading and specify an Override Register. When the first bit of the override register is turned on, this legend will be visible.

Lab 7: Color Touch Screen Configuration

Notes:

LAB 8

Graphic Alarms

Lab 8: Graphic Alarms

Lab 8: Graphic Alarms

Objective:

Understand the powerful Alarm capabilities of the LX.

Procedure:

Build off of your program from the previous lab. Remember that the Smart Stix needs to be configured for the I/O used in this lab.

From the Graphics Editor, click on the **Config** menu and select **Alarm**.

Configure Alarms as following:

- Alarm Trigger: %M1601
- Max Number of Alarms: 32

Name the first 4 alarms as follows by double-clicking them in the list:

- Alarm 1, Group 1 Low-speed Warning
- Alarm 2, Group 1 High-speed Warning
- Alarm 3, Group 1 Motor Overload Trip
- Alarm 4, Group 1 E-stop Trip

Exit the Alarm configuration and the graphics editor.

Add to your logic to trigger Alarm 1 (%M1601) whenever %R7 is less than 15 (and the machine is running)

Add to your logic to trigger Alarm 2 (%M1602) whenever %R7 is greater than 90.

Add to your logic to trigger Alarm 3 (%M1603) whenever %I1 is on. Add a normally-closed %I1 to your run circuit so that the circuit will not run if %I1 is on.

Add to your logic to trigger Alarm 4 (%M1604) whenever %I2 is on. Add a normally closed %I2 to your run circuit so that the circuit will not run if %I2 is on.

Back in the graphics editor, add an Alarm Indicator Button to the Control Panel Screen, as well as to the Trend Screen. An example is shown in Figure 1. The Alarm Indicators should display an Alarm Summary when pressed, for all Alarm groups.

Create a new screen that is called whenever F3 is pressed. This screen should contain an Alarm Summary Object, and an Alarm History Object. An example is also shown in Figure 2. For help in changing to a different screen, consult the screen manipulation lab.

Download and execute the application. Practice triggering alarms, acknowledging them and clearing them. Note the differences between what is displayed in the “Summary” log, and what is displayed in the “History” log.

Lab 8: Graphic Alarms

Use the “Alarm Indicator” button on the first two screens as a means of viewing the Alarm Summary. Note the conditions that will cause the Alarm Indicator buttons to change color.

Extra Credit: Create a screen showing a simple graphical diagram of a house. Place an alarm indicator button on the Basement (Group 1), First Floor (Group 2), Second Floor (Group 3) and Garage (Group 4). Add new alarms to the Alarm Configuration, a couple each for groups 2, 3, and 4. Trigger those new alarms with unused switches on the I/O Simulator (%I3-%I8). Note how groups are a great way to segment alarms into manageable groups that can be monitored in separate alarm summary and history logs, as well as larger groups. (See Figure 3).

Figure 1

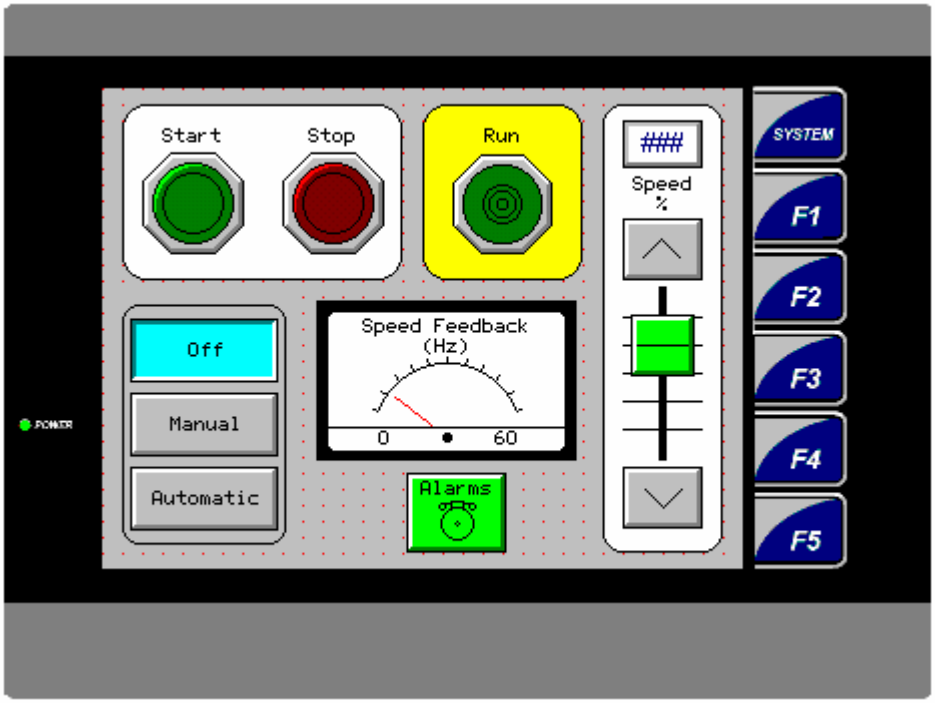


Figure 2

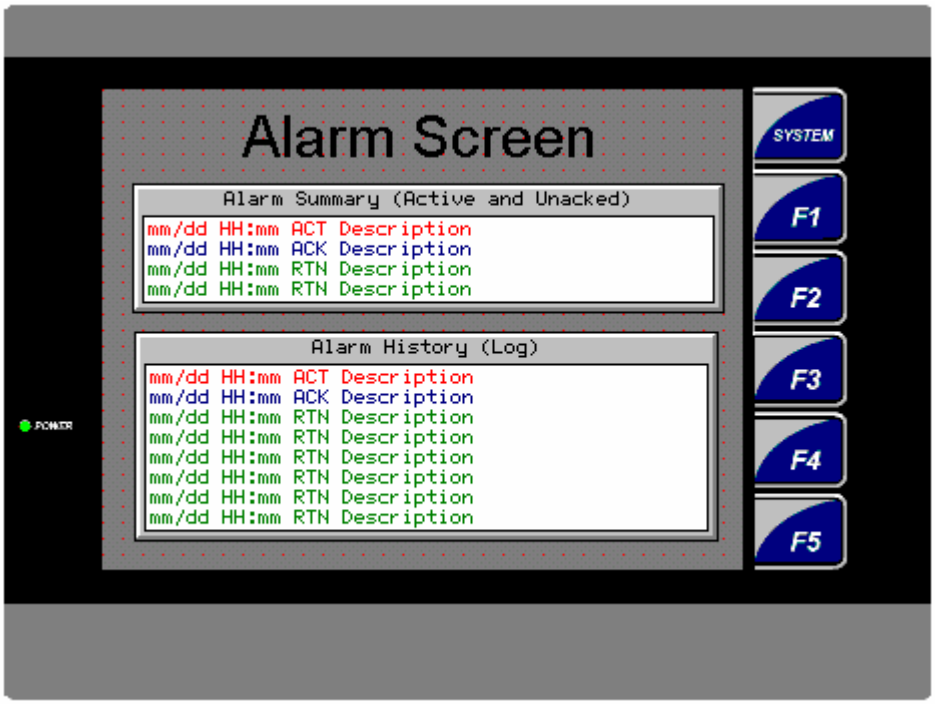
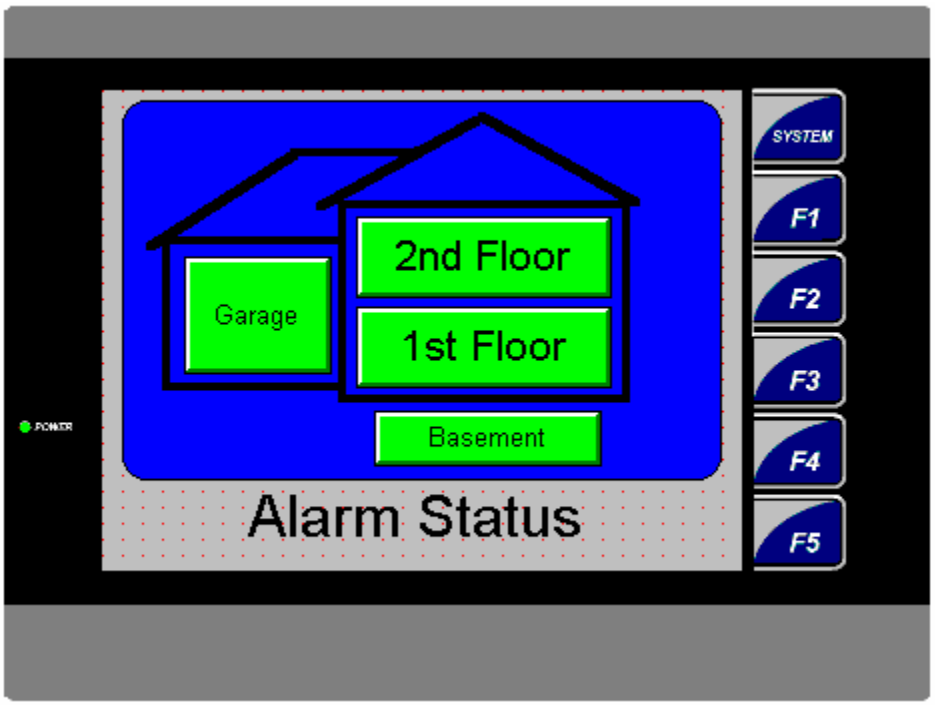


Figure 3



Lab 8: Graphic Alarms

Notes:

LAB 9

CompactFlash Functions

Lab 9: Removable Media Functions

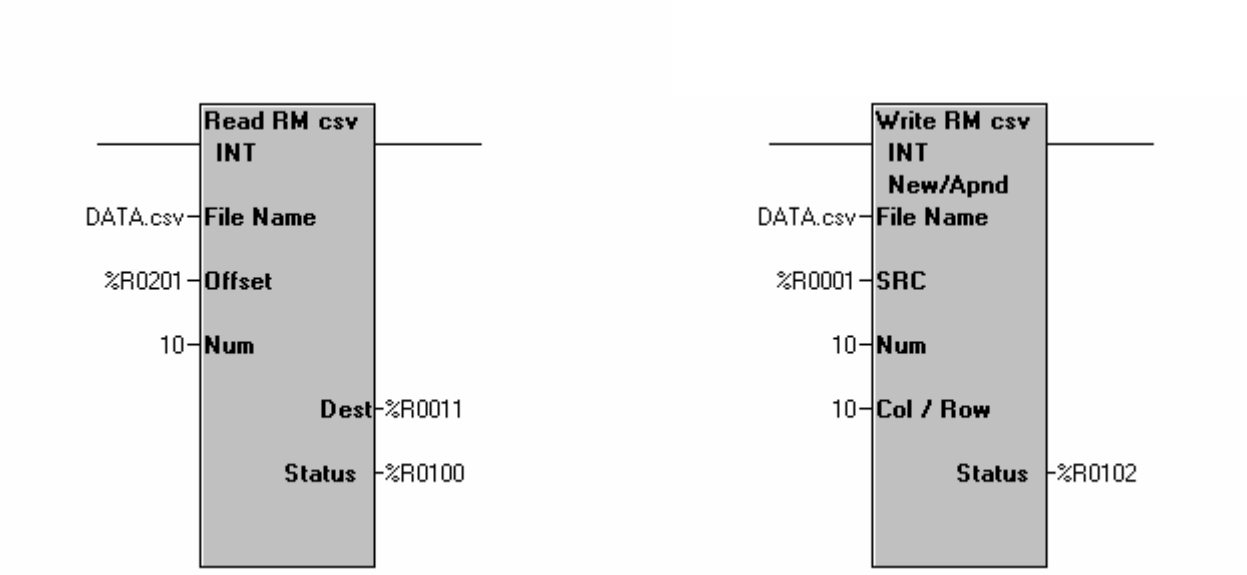
Lab 9: Removable Media Functions

Objective:

Understand the functionality of the Removable Media (RM) on controllers that support it.

CompactFlash Data Files

The following controllers support compact flash, OCS451/551/651, NX22X/25X, XLE. This gives the program the ability to store information to the RM card and also read information back into the program. Since the information is stored in a Comma Separated Value (CSV) format, the RM card can be removed from the unit and then read into a spreadsheet. Conversely, a CSV file could be created from a PC, stored to the flash, and then read into the OCS. RM ladder functions are found in the Special Operations toolbar.



1. Open Cscape and create a ladder program that will write 10 registers of information, starting at %R1, when triggered by %T1. Use the Write RM function configured for Create/Append to perform this action and call the file Data.csv. Place a rest coil after the Write RM function with the address of %T1.
2. Create a line of code that will read the information from the Data.csv file and store the information in %R11 when %T2 is triggered. The offset should be configured for %R201. Place a rest coil after the Read RM function with the address of %T2.
3. Create a line of code that Fills 0 into %R201 - %R202 on a first scan.

4. Now configure a screen 1 with a Menu Object
   - Configure the Legend for RM Functions
   - Configure the first Menu Item for the following:
     - Prompt Text 'Write Data ='
     - Type = Numeric and configured for %R1 editable
   - Configure the second Menu Item for the following:
     - Prompt Text 'Read Data ='
     - Type = Numeric and configured for %R11 un-check editable
   - Configure the third Menu Item for the following:
     - Prompt Text 'Goto RM Manager'
     - Type = Screen Jump to screen 2 with ESC to Return checked
   - Configure the fourth Menu Item for the following:
     - Prompt Text ' RM Actions'
     - Type = Screen Jump to screen 3 with ESC to Return checked
   - Configure Screen 2 with a RM Manager on the screen configured for the option to delete files.
   - Configure Screen 3 with 2 switches configured one for %T1 and the other for %T2 with the action as on for both.
5. Download the program to the XLE. Remember that the controller will need to be configured properly.
6. Use the Menu to navigate through the steps below
7. Change the value in %R1.
8. Press the %T1 button. The information in %R1 is now stored to the RM.
9. Press the %T2 button. The information should now appear in the %R11 data field on the screen.
10. Every time that the T1 button is pressed, the value in %R1 will be stored. So to read different values that have been stored, change the value in %R201 by 10. This can be done through data watch or by placing an editable field on the screen for %R201.

    %R201 = 0 (First write done to the DATA.csv file)

    %R201 = 10 (Second write done to the DATA.csv file)

Lab 9: Removable Media Functions

The next page contains a ladder example of the solution.

CompactFlash File Naming

The OCS/NX CF function blocks support the flash with a DOS/Windows standard FAT16 file system. All names must be limited to the “eight dot three” (8.3) format where the filename contains a maximum of eight characters, a period, and an extension with a maximum of three characters. The entire filename including any path must be less than or equal to 147 characters in length.

When creating filenames and directories, it is sometimes desirable to include parts of the current date or time. There are six special symbols that can be entered into a filename that are replaced by the OCS with current time and date information.

Symbol	Description	Example
$Y	Substitutes the current 2 digit year	2004 = 04
$M	Substitutes the current month with a 2 digit code	March = 03
$D	**Substitutes the current day**	**22nd = 22**
$h	Substitutes the current hour in 24 hour format	4 PM = 16
$m	Substitutes the current minute	
$s	Substitutes the current second	
$p	Substitutes the currently displayed 4-digit screen number (1-1023, Intended mainly for screen capture)	53 = 0053

Note that all the symbols start with the dollar sign ($) character. Date symbols are in upper case; time symbols are in lower case.

The following are examples of the substituted time/date filenames:

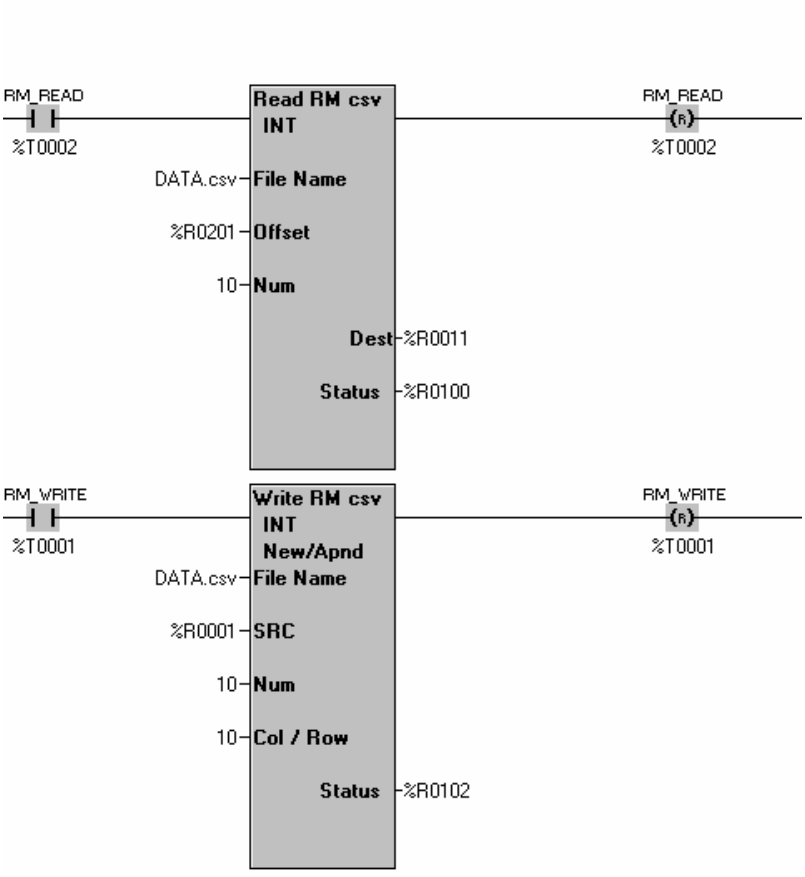
If the current date and time = March 1, 2004 3:45:34 PM

Data$M$D.csv = Data0301.csv

Year$Y\Month$M\aa$D_$h.csv = Year04\Month03\aa01_15.csv

Month_$M\Day_$D\$h_$m_$s.csv = Month_03\Day_01\15_45_34.csv

Lab 9: Removable Media Functions



CompactFlash File Counters

Another tool available for use in naming Removable Media files is the Filename Counter. There are four available Filename Counters that can be separately configured. Configuration is done through the Graphics Editor by clicking **Config** and selecting
Filename Counters.

Each Filename Counter requires a 32-bit register regardless of the maximum values that the counter will see. A maximum value is specified for each counter and also the options to auto-increment and wrap the counter value.

The auto-increment function causes the counter to be automatically incremented by a value of 1 each time the Filename Counter is accessed.

The wrap counter function causes the counter to start over at 0 when the maximum value is exceeded. If the wrap counter function is not activated and the counter reaches the maximum value, the counter will no longer automatically increment and the value will remain at the maximum setting.

Accessing the counters is done similarly to the date and time symbols. The format to access any of them is as follows:

$[counter number]u[# of digits, 1-8]

For example, using counter 1 for a screen capture, if the counter has a Max value of 59, the current value is 35 and the Auto Increment is checked:

$1u4 = 0035

The next time the screen is captured, the value will be 0036, then 0037, etc. This can be implemented into the filename as follows:

Given:

Current date and time = March 1, 2004 3:45:34 PM

Counter 3 Auto Incrementing, Max of 59, currently at 58, Wrap turned ON

Captures\Chan3\$M-$D-$Y\$h$m-$3u2.bmp

= Captures\Chan3\03-01-04\1545-58.bmp

Next screen capture (assuming same time and date)

= Captures\Chan3\03-01-04\1545-59.bmp

Next screen capture (assuming same time and date)

= Captures\Chan3\03-01-04\1545-00.bmp

Note: You MUST specify the filename extension in all cases. It is never automatically added.

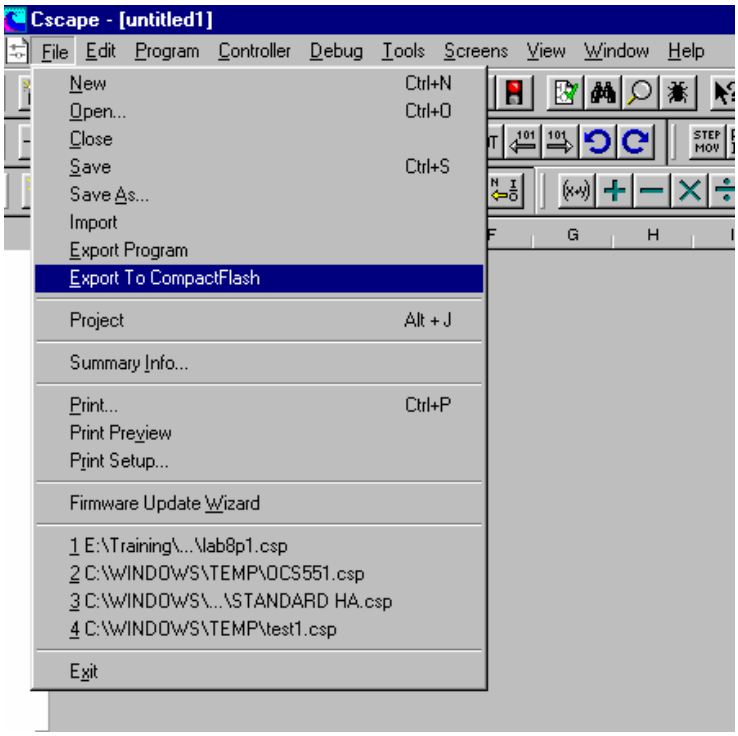
Lab 9: Removable Media Functions

RM Program Downloads

One feature of the Removable Media functionality is the ability to load an OCS/NX/XLE with a program from a RM card instead of through Cscape.

The programmer saves the Cscape program as a special file type with a .pgm extension by clicking the **File** menu and selecting **Export to Removable Media**. It can be exported directly to a Media writer connected to the computer or to anywhere else on the computer to be transferred to RM later. The user will then insert the RM card into the OCS and, through the System Menu, select Removable Media to find the correct file to load.

The screen shot below illustrates where in Cscape the write to Removable Media is done.



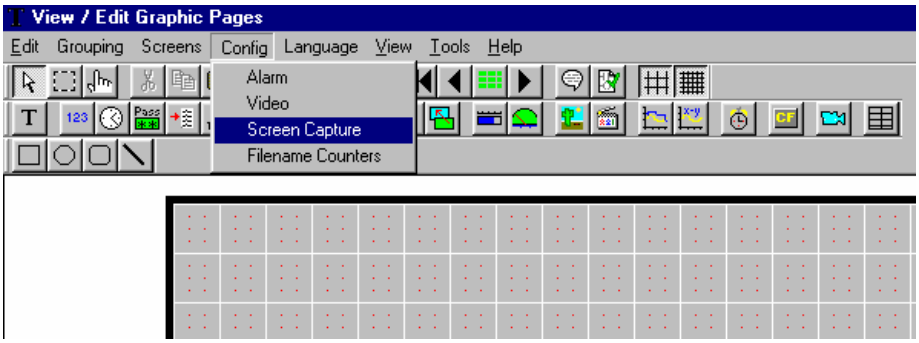
RM Screen Captures

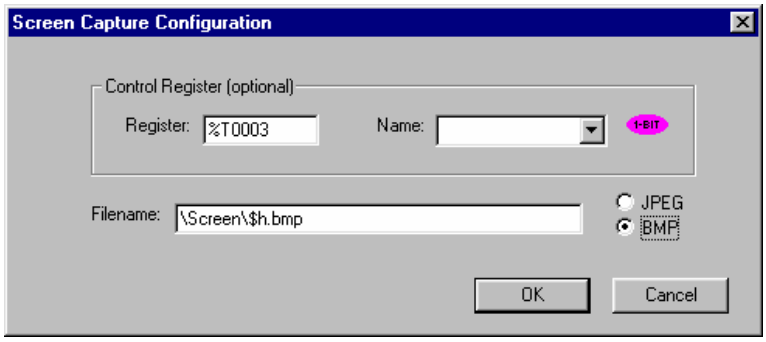
The SVGA ColorTouch OCS, NX, XLE units have the ability to capture a displayed screen to RM as a JPEG or Bitmap file. These images can then be recalled on the unit through the RM manager or viewed on a computer with a graphics viewer.

Lab 9: Removable Media Functions

Configuring the Screen Capture function is done through the Graphics Editor by clicking the **Config** menu and selecting **Screen Capture**. A 1-bit register must be configured as a trigger and a filename for the captured graphic file must be specified. The filename date functions and filename counters can be used for this. The OCS provides feedback that the screen capture is done by resetting the 1-bit register to an OFF state.

1. Add a rung of logic that when pressed %K10 will turn on %T3.
2. In the Graphics Editor, Configure the Screen Capture to trigger off of %T3 and specify a filename.
3. Download to the XLE.
4. Press the F10 Key.
5. Press the Menu to go to the RM Manager and then find the captured graphic.
6. Open the file that was just saved to the RM. Note the dotted edge around the displayed graphic that indicates it is not a live, updating screen.





Lab 9: Removable Media Functions

Notes:

Lab 9: Removable Media Functions

Notes:

CHEAT SHEET

Data Types

BOOL - Boolean; A single bit. It can contain only the values '0' or '1', a.k.a 'FALSE' or 'TRUE'

BYTE - Byte; A string of 8 consecutive bits. Byte format is used more where the value of the data is not as important as the bit patterns (shifts and rotates).

WORD – Word; A string of 16 consecutive bits. Word format is used more where the value of the data is not as important as the bit patterns (shifts and rotates).

DWORD - Double Word; A string of 32 consecutive bits. DWORD format is used where the value of the data is not as important as the bit patterns (shifts and rotates).

INT – Integer; A 16-bit signed value. Integers are used where the value of the data is expected to be in the range of -32,768 to +32,767

SINT - Short Integer; An 8-bit signed value. Short Integers are used where the value of the data is expected to be in the range of -128 to +127.

DINT - Double Integer; A 32-bit signed value. Double Integers are used where the value of the data is expected to be in the range of -2,147,483,648 to +2,147,483,647.

UINT - Unsigned Integer; A 16-bit unsigned value. Unsigned Integers are used where the value of the data is expected to be in the range of -0 (zero) to 65,535.

USINT - Unsigned Short Integer; An 8-bit unsigned value. Unsigned Short Integers are used where the value of the data is expected to be in the range of 0 (zero) to 255

UDINT - Unsigned Double Integer; A 32-bit unsigned value. Unsigned Double Integers are used where the value of the data is expected to be in the range of 0 (zero) to 4,294,967,296.

REAL - Floating Point; A 32-bit value. Values are stored and operated on in IEEE single precision (six digit) format. Values range from -3.40282E+38 to +3.40282E+38.

STRING – String; A variable-length succession of characters. Each character is represented by one byte.

Register Types

Type	Description and example of what might use the type	Format	Retentive	#Available
%I	Discrete Inputs from the field; prox sensors, panel buttons, etc	BOOL	YES	2048
%Q	Discrete Outputs to the field; relays, indicator lamps, etc.	BOOL	NO	2048
%AI	Analog Inputs from the field; Thermocouples, 4-20mA inputs	WORD	YES	512
%AQ	Analog Outputs to the field; 0-10VDC or 4-20mA outputs	WORD	NO	512
%IG	Global Discrete Inputs from the CAN; in from other OCS	BOOL	YES	64 per node
%QG	Global Discrete Outputs to the CAN; out to other OCS	BOOL	NO	64 per node
%AIG	Global Analog Inputs from the CAN; in from other OCS	WORD	YES	32 per node
%AQG	Global Analog Outputs to the CAN; out to other OCS	WORD	NO	32 per node
%T	Internal Temporary bits, use for contacts and coils	BOOL	NO	2048
%M	Internal Temporary bits, use for contacts and coils	BOOL	YES	2048
%R	Internal Registers, use for Timers and Counters and other data	WORD	YES	2048-9999
%K	Keypad bits, reflect Function Key status	BOOL	NO	5-12
%D	Display bits, control screens or indicate screen on/off	BOOL	NO	200-1023
%S	Internal System Bits (See System Registers)	BOOL	---	---
%SR	Internal System Registers (See System Registers)	WORD	---	---

Cheat Sheet

System Bits

Point	Name	Function
%S01	FST_SCN	Indicates First Scan
%S02	NET_OK	Network is OK
%S03	T_10MS	10mS pulse
%S04	T_100MS	100mS pulse
%S05	T_1SEC	1 second pulse
%S06	IO_OK	I/O is OK

Point	Name	Function
%S07	ALW_ON	Always ON
%S08	ALW_OFF	Always OFF
%S09	PAUSING_SCN	Pause 'n Load soon
%S10	RESUMED_SCN	Pause 'n load done
%S11	FORCE	I/O being forced
%S12	FORCE_EN	Forcing is enabled

System Registers

SR #	Name	Min	Max
1	User Screen Number	0	200*
2	Alarm Screen Number	0	200*
3	System Screen Number	0	10*
4	Self Test Result		
5	Controller Mode (RUN..)	0	2
6	Scan Rate Avg		
7	*Reserved*		
8	*Reserved*		
9	Edit Buffer Low		
10	Edit Buffer High		
11	Ladder Size Low		
12	Ladder Size High		
13	User Text Size Low		
14	User Text Size High		
15	System Text Size Low		
16	System Text Size High		
17	I/O Config Size Low		
18	I/O Config Size High		
19	Net Config Size Low		
20	Net Config Size High		
21	Security Data Size Low		
22	Security Data Size High		
23	Ladder CRC		
24	User Text CRC		
25	System Text CRC		
26	I/O Config CRC		
27	Net Config CRC		
28	Security Data CRC		
29	Network ID Low	1	253
30	Network Baud Rate	0	3
31	Network Required	0	1
32	LCD Contrast	1	255
33	Key Toggle Mode	0	1
34	Serial Protocol		
35	Serial Number Low		
36	Serial Number High		
37	Model Number		
38	Engine Version		

SR #	Name	Min	Max
39	BIOS Version		
40	FPGA Version		
41	LCD Columns		
42	LCD Rows		
43	Keypad Type		
44	RTC Seconds	0	59
45	RTC Minutes	0	59
46	RTC Hours	0	23
47	RTC Day of Month	1	31
48	RTC Month	1	12
49	RTC Year	1996	2095
50	RTC Day of Week	1	7
51	Network Error Count		
52-55	*Reserved*		
56	Last Key		
57	LCD Backlight		
58	User Leds		
59-60	*Reserved*		
61	Num Ids		
62-174	*Reserved*		
175	CF Status		
176	CF Free Low		
177	CF Free High		
178	CF Total Low		
179	CF Total High		
180	Reserved		
181	Alarms Unacknowledged		
182	Alarms Active		
183	System Beep	0	1
184	User Beep	0	1
185	Screen Saver	0	1
186	Screen Saver Time	5	1200
187	Network Usage (Avg)	0	1000
188	Network Usage (Min)	0	1000
189	Network Usage (Max)	0	1000
190	Network TX Use (Avg)	0	1000
191	Network TX Use (Min)	0	1000
192	Network TX Use (Max)	0	1000

*Maximum User, Alarm and System screens vary from model to model

Max = 200 for MiniOCS, OCS1x0, OCS2x0… Max = 1023 for NX2xx, OCS3xx, OCS4/5/651

For Details on the functionality of the different SR registers, consult the Cscape help file.

HORNER APG CONTACTS

Phil Horner
President
317-492-9080
phil.horner@heapg.com

Technical Support
Ext. 3
techsppt@heapg.com

Customer Service
Ext. 1
APGCustomerService@heapg.com

Sales and Marketing
Ext. 2
apgsales@heapg.com

Ken Jannotta, Sr.
VP of Sales and Marketing
Office: 434-973-9245
Cell: 434-825-7550
kensr.jannotta@heapg.com

Bill Giebel
Business Development Manager
317-492-9079
Cell: 317-407-7937
bill.giebel@heapg.com

Roy Lowery
Business Development Manager
317-492-9078
Cell: 317-40 7-9506
roy.lowery@heapg.com

Tom Filipek
Business Development Manager
Office: 651-426-2282
Cell: 612-840-6653
tom.filipek@heapg.com

Chuck Ridgeway
Product Manager
317-492-9081
chuck.ridgeway@heapg.com

Eric Broyer
Technical Support Manager
317-492-9102
eric.broyer@heapg.com

Nate Beachey
System Design Engineer
317-492-9118
nathan.beachey@heapg.com