



# **Training Manual for Cscape and XLe/XLt**



## PREFACE

This manual provides introductory level training for Cscape Software users using XLe or XLt.

Copyright (C) 2009 Horner APG, LLC., 59 South State Street, Indianapolis, Indiana 46201. All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior agreement and written permission of Horner APG, Inc.

All software described in this document or media is also copyrighted material subject to the terms and conditions of the Horner Software License Agreement.

Information in this document is subject to change without notice and does not represent a commitment on the part of Horner APG.

Cscape, SmartStack, SmartStix and CsCAN are trademarks of Horner APG.

DeviceNet is a trademark of the Open DeviceNet Vendor Association (OVDA), Inc.

Ethernet is a trademark of Xerox Corporation.

CompactFlash is a registered trademark of SanDisk Corporation.

***For user manual updates, contact Technical Support:***

North America:

(317) 916-4274

[www.heapg.com](http://www.heapg.com)

email: [techsppt@heapg.com](mailto:techsppt@heapg.com)

Europe:

(+) 353-21-4321-266

[www.horner-apg.com](http://www.horner-apg.com)

email: [techsupport@hornerirl.ie](mailto: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.**

## TABLE OF CONTENTS

<i>Introduction to Cscape</i> .....	7
<b>Quick Start Guide</b> .....	7
Objective:.....	9
Equipment Needed:.....	9
Notes: .....	14
<b>LAB 1</b> .....	15
<b>Basic OCS Configuration and Programming</b> .....	15
Objective:.....	17
Procedure: .....	17
Notes: .....	26
<b>LAB 2</b> .....	27
<b>Screen Manipulation</b> .....	27
Objective:.....	29
Screen Overview .....	29
Part 1 – Switching and Forcing.....	29
Switching and Forcing Lab.....	30
Part 2 – Changing the System Registers .....	31
Part 3 – Screen Jumps .....	31
Screen Jumps Lab .....	32
Part 4 – Menu Object Lab .....	34
Extra Credit #1 .....	36
Extra Credit #2.....	36
Extra Credit Solutions:.....	37
Notes: .....	38
<b>LAB 3</b> .....	39
<b>Timers and Counters</b> .....	39
Objective:.....	41
Timers Overview: .....	41
Part 1 – TON Timers Lab: .....	42
Part 2 – Retentive TON Timers Lab .....	43
Part 3 – TOF Timers Lab .....	45
Counters Overview: .....	46
Part 4 – Counters Lab .....	46
TOF Solution: .....	48
Counter Lab Solution:.....	48
Notes: .....	49

<b>LAB 4.....</b>	<b>51</b>
<b>Move Operations.....</b>	<b>51</b>
Objective:.....	53
Overview:.....	53
Part 1 – Move Lab.....	55
Part 2 – Block Move Lab.....	55
Part 3 – Block Fill Lab.....	56
Part 4 – Constant and Indirect Moves Lab.....	57
Move Lab Solution: .....	58
Notes: .....	59
<b>LAB 5.....</b>	<b>61</b>
<b>Basic CsCAN Networking.....</b>	<b>61</b>
Objective:.....	63
Overview:.....	63
Part 1 - Analog Data Over CsCAN.....	63
Part 2 – Digital Data Over CsCAN.....	67
Part 3 – SmartStix .....	69
Notes: .....	70
<b>LAB 6.....</b>	<b>71</b>
<b>Graphics Editor.....</b>	<b>71</b>
Objective:.....	73
Overview:.....	73
Graphics Objects:.....	74
Graphics Editor Lab.....	78
Part 2 – Extended Graphic Alarms Lab .....	79
Notes: .....	81
<b>LAB 7.....</b>	<b>83</b>
<b>Removable Media Functions.....</b>	<b>83</b>
Objective:.....	85
Ladder Logic Functions .....	85
Procedure: .....	85
Solution:.....	88
Automatic Data Logging.....	89
Removable Media File Naming.....	89
Symbol Description Example .....	89
Removable Media File Counters .....	90
Removable Media Program Downloads .....	91
Removable Media Screen Captures .....	91
Procedure: .....	92
Notes: .....	93
<b>CHEAT SHEET.....</b>	<b>94</b>
<b>HORNER APG CONTACTS.....</b>	<b>96</b>

# **Introduction to Cscape**

## Quick Start Guide

## **Quick-Start Guide**



## **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:**

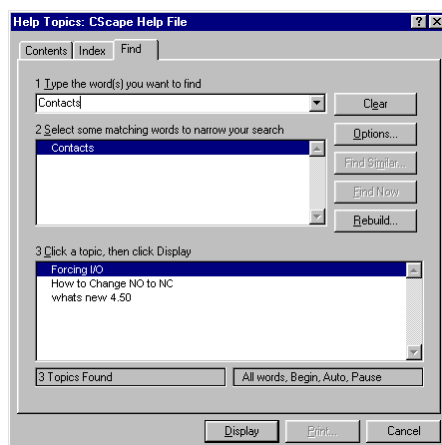
A PC with Cscape loaded.

#### **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.

## **Quick-Start Guide**

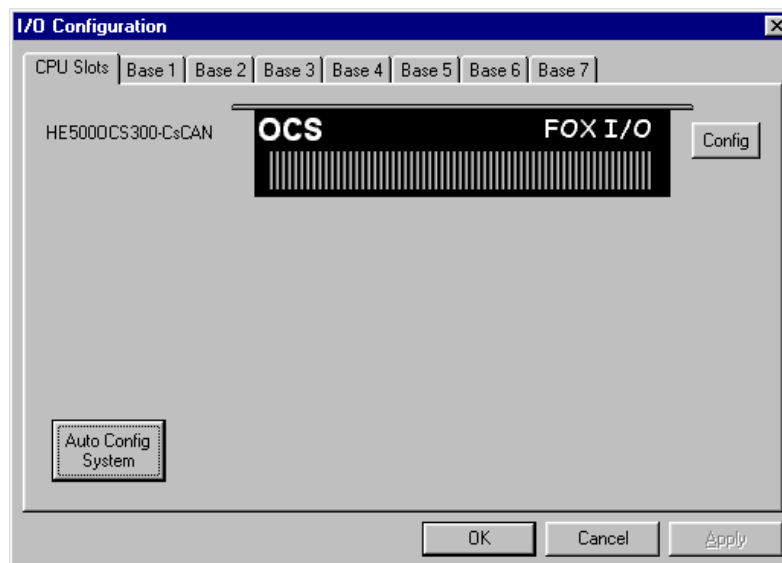
- 1.4 Searches can be done through selecting **F**ind from the top of the screen.
  - 1.4.1 Upon opening the Find portion of the help file, type in “Contacts” and the following will be shown on the screen.



- 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 **F**ile 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 **F**ile 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 **F**ile 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 by clicking the **C**ontroller 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.

## **Quick-Start Guide**

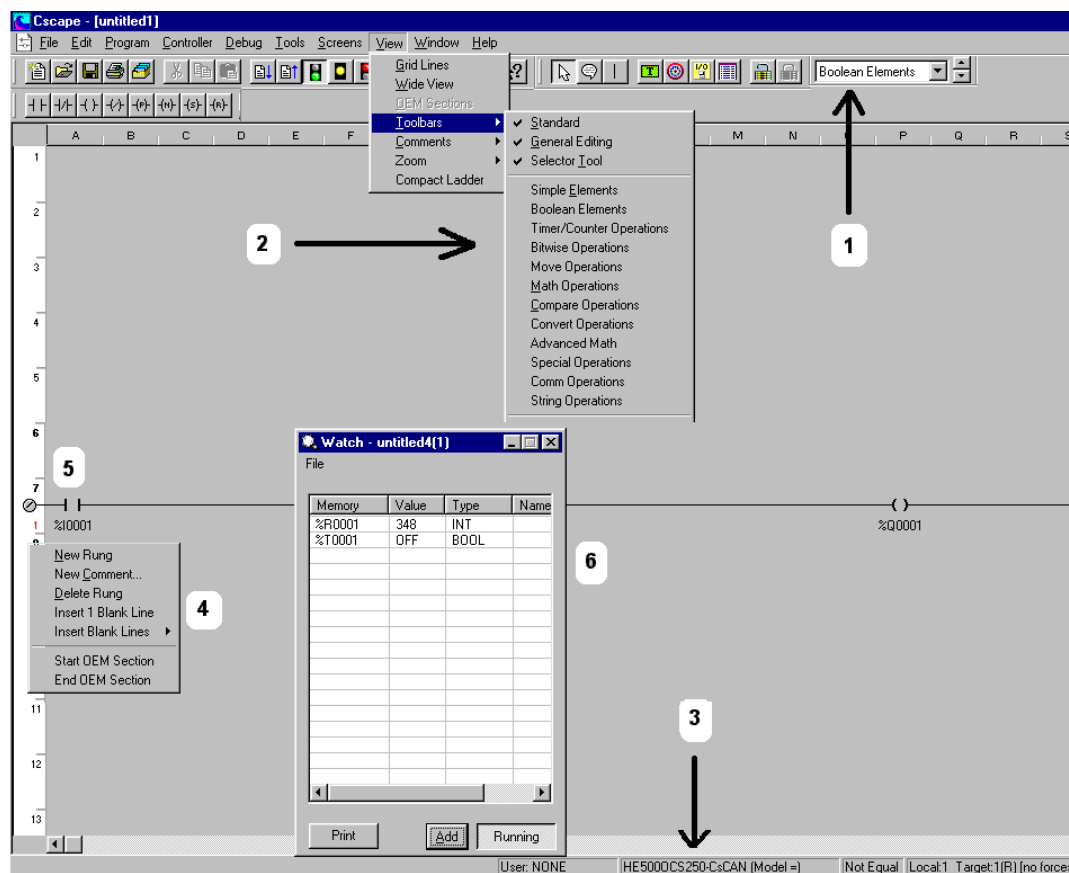
- 2.4.2 Configure the controller from the Auto-Configure option. Keep in mind on existing programs that Auto-Configure will erase I/O configurations that deviate from the default parameters. An example of this would be in an application with a High Speed



- 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

## Quick-Start Guide

Cscape program 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

## **Quick-Start Guide**

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.

## **Quick-Start Guide**

**Notes:**

# **LAB 1**

## Basic OCS Configuration and Programming

## **Lab 1: Basic OCS Configuration and Programming**



## **Lab 1: Basic OCS Configuration and Programming**

### **Objective:**

The objective of this lab is to provide the knowledge to use Cscape to create a basic program including hardware configuration, logic design, and screen development. This foundation will then be used to help expand skills in the use of Cscape and the XLe/XLt OCS.

### **Basic Configuration and Programming Overview:**

When starting a new Cscape program, the first thing that should always be done is to configure the correct controller and I/O. This gives the basis for such things as where the inputs and outputs will be mapped for use in the program and will tell Cscape which screen size to edit when screens are created. Incorrect I/O configuration can also cause all I/O to stop working.

After configuring the controller, programming can commence with I/O point naming, ladder logic editing and screen creation.

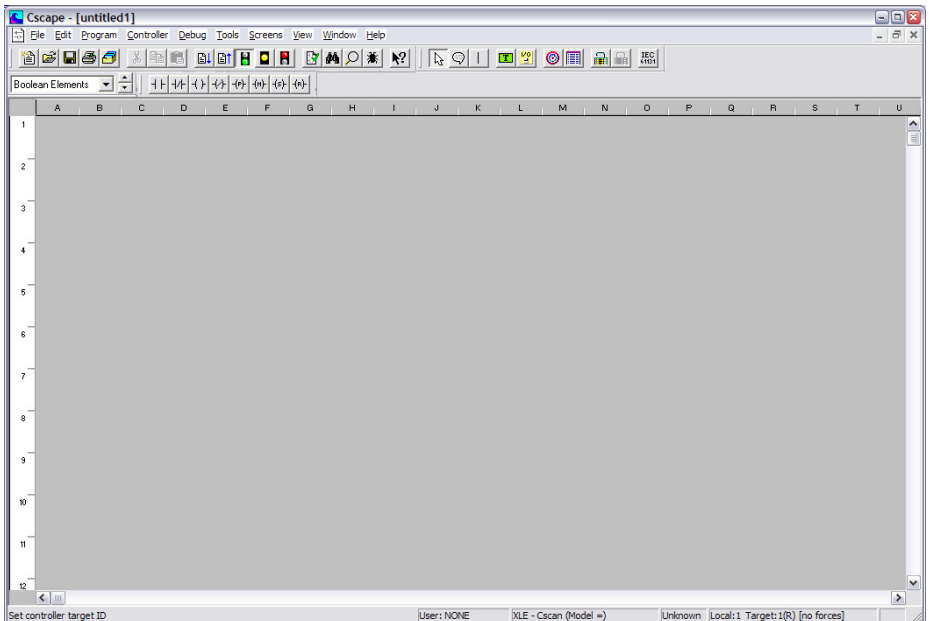
### **Procedure:**

#### **Step 1**

- **Connect the OCS to a PC.** Use the controller's programming cable to connect the controller's programming port to a PC COM port. A USB-to-RS232 adapter may be needed since most modern laptops do not come with a COM port.

#### **Step 2**

- **Power up the OCS and start Cscape on the PC.** Connect the power supply to the XLe/XLt. Open the Cscape program on the PC. A new, blank program called "untitled1" is automatically opened. The default controller for any new Cscape program is an XE102 XLe model. If the controller being programmed is anything different, the I/O configuration must be changed to match, covered here in a following step.

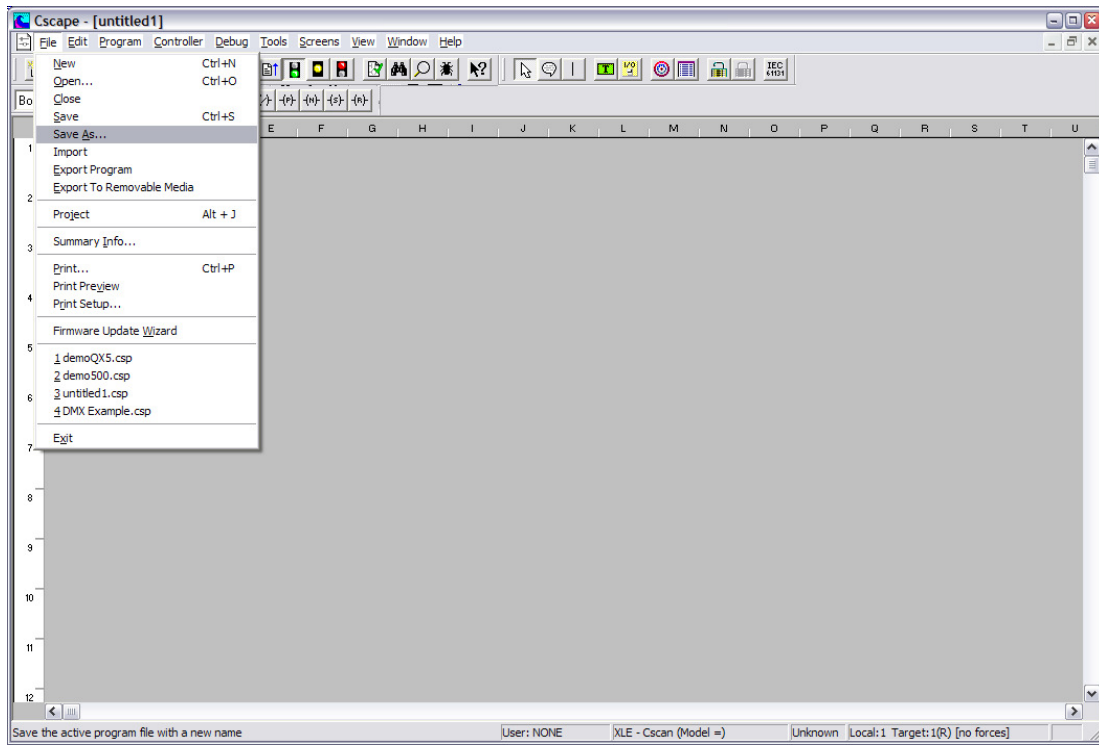


## **Lab 1: Basic OCS Configuration and Programming**

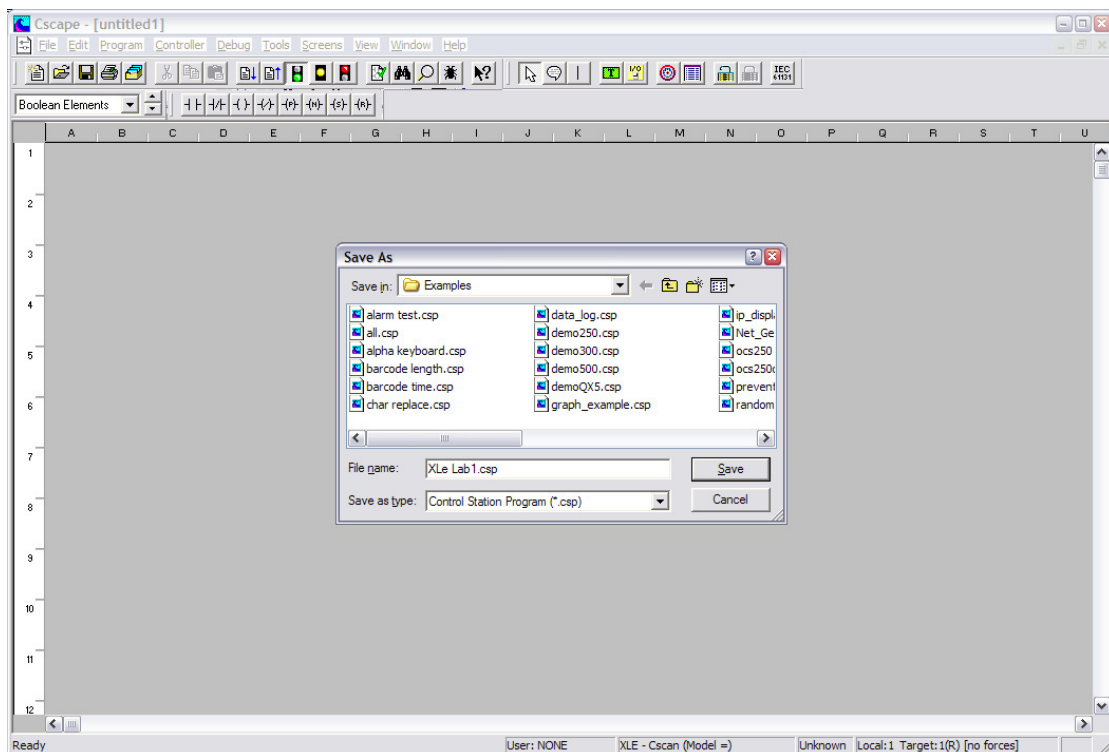
### Step 3

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

Click on the **F**ile menu and select **S**ave **A**s...



Type the program name, such as 'XLe Lab1.csp', in the File Name dialog box and click the Save button.

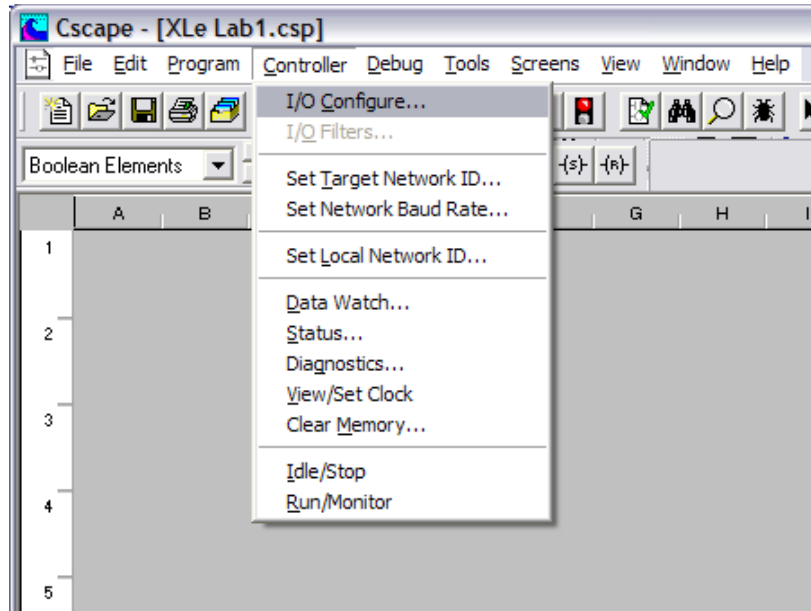


## **Lab 1: Basic OCS Configuration and Programming**

### Step 4

#### ➤ **Configure the OCS Controller**

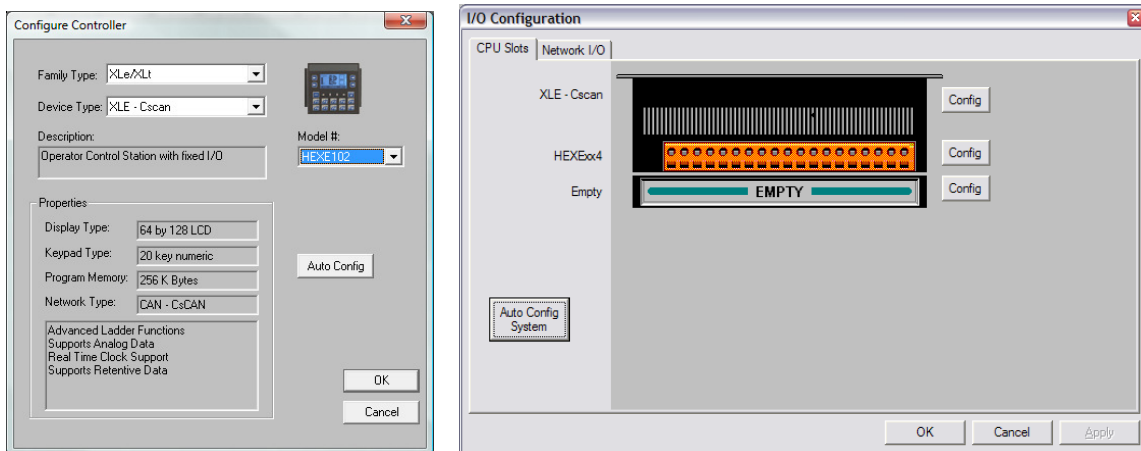
Either Click the 'I/O' button in the toolbar , or click on the **C**ontroller menu and select **I/O Configure**:



If online with the OCS, use the 'Auto Config System' button. Clicking it will automatically configure the controller and any attached I/O if connected to the OCS properly. If this is the case, Skip to Step 5 after Auto Config is done.

Otherwise, to do it manually: **(Skip to Step 5 if Auto Config has been done)**

1. Double click on the controller picture or click the 'Config' button next to it.
2. Select the 'XLe/XLt' Family Type and select the correct entry from the Device Type list depending on the model being used. Then select the Model # to match the model. For this example, the HEXE102 is used.
3. Click OK to see the overall I/O configuration.



4. Click OK again to exit the I/O configuration.

## **Lab 1: Basic OCS Configuration and Programming**

### Step 5

#### ➤ **Save the program.**

Either click the 'Save' button on the toolbar , or click on the **F**ile menu and select **S**ave.

### Step 6

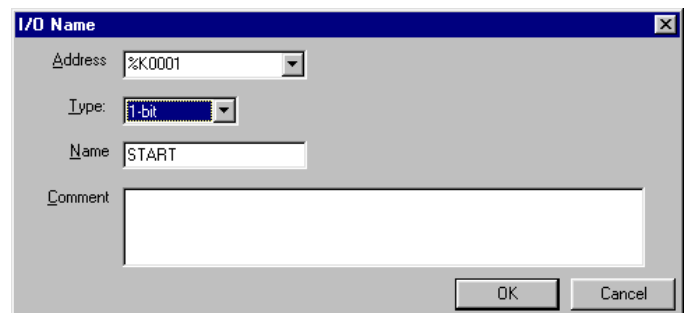
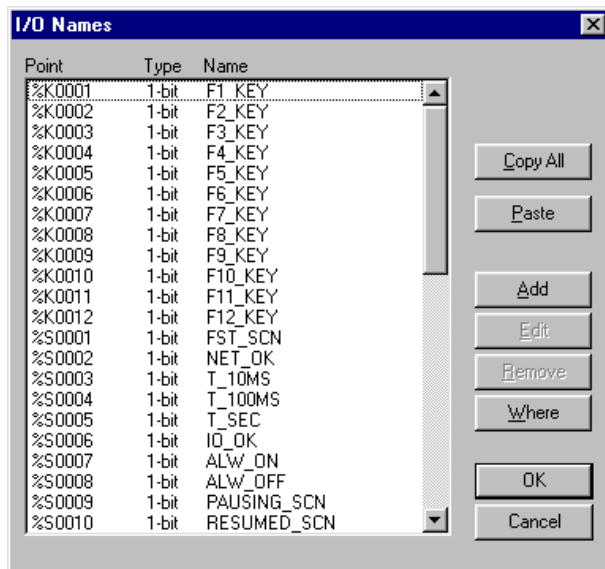
#### ➤ **Name some I/O points.**

Click on the **P**rogram 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

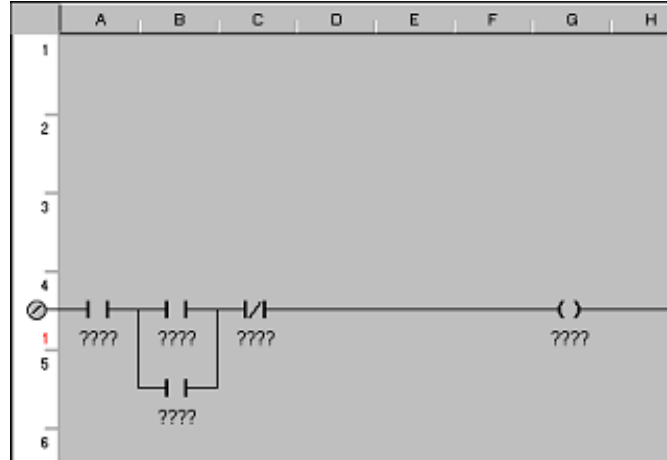


## **Lab 1: Basic OCS Configuration and Programming**

### Step 7

#### ➤ **Program the following rung:**

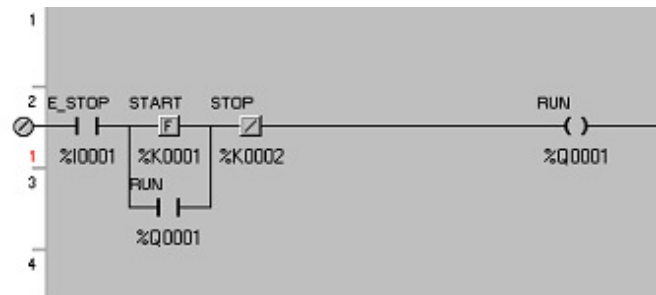
1. Select and drop the three normally open contacts.
2. Select and drop the normally closed contact.
3. Add the vertical connecting lines.
4. Select and drop a normally open coil.



### Step 8



#### ➤ **Add the element names.**

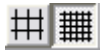
1. Double click on each element in the rung.
2. Select the name or address from the drop down list.
3. Click OK



### Step 9


#### ➤ **Create a screen.**

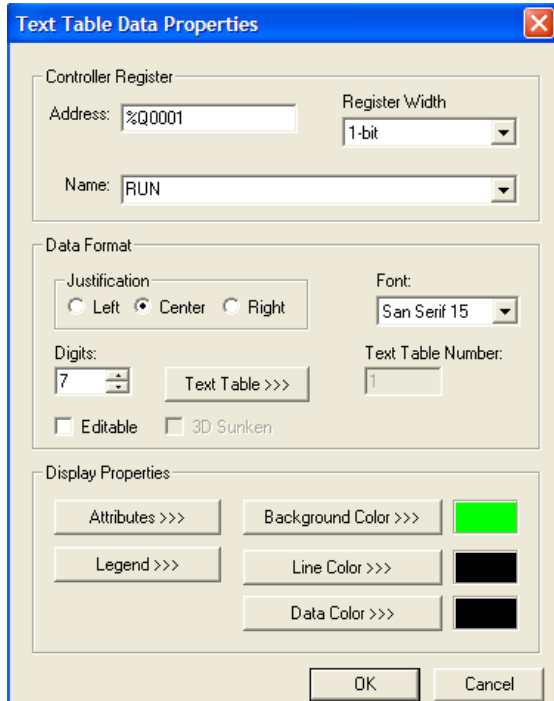
1. Open the Graphics Editor by clicking the Green 'T' button  in the toolbar or by clicking on the **Screens** menu and selecting **View / Edit Screens....** This will open a separate window in which OCS screens may be edited.
2. Using the Static Text object in the Graphics Editor, insert Static Text (a label) at the top center of the screen. To do this, click and release once on the 'Static Text' button  in the Graphics Editor toolbar, bring the mouse down to the screen, click AND HOLD at the top-left of where the text is wanted, drag the text box down and to the right to the size wanted, then release the mouse button. Objects can always be resized and moved later.

**Note:** The 'Snap To Grid' buttons  on the Graphics Editor toolbar may have to be toggled off in order to move the static text where desired.

3. Double-click the Static Text box that was just drawn to configure it. Edit the text to display MACHINE.
4. Change the Justification, Font Type and Colors if desired. Click 'OK' when finished. Move or resize the Static Text box so that all of the text is shown and so that it looks satisfactory. Aside from any grid lines, this is exactly what the screen will look like on the OCS once the program is downloaded.

## **Lab 1: Basic OCS Configuration and Programming**

5. Now find the Text Table button on the toolbar  and draw a text table on the screen under the Static Text box that is already there. Note that it comes default with a border and a legend that says "Text Table". Double-click on it to configure it...



The dialog box is titled "Text Table Data Properties". It contains three main sections: "Controller Register", "Data Format", and "Display Properties".

- Controller Register:** Address: %Q0001, Register Width: 1-bit, Name: RUN.
- Data Format:** Justification: Center, Font: San Serif 15, Digits: 7, Text Table Number: 1. There are checkboxes for "Editable" and "3D Sunken".
- Display Properties:** Buttons for "Attributes >>>", "Background Color >>>" (green), "Legend >>>", "Line Color >>>" (black), and "Data Color >>>" (black).

Buttons at the bottom: OK, Cancel.

6. A Text Table is just a data field that can link less meaningful number data to more meaningful text. For this lab, it will be linked to the RUN coil in the ladder program, which is a 1-bit register, %Q1. Select a Register Width of 1-bit.
7. Link this text table field to the RUN output, %Q1, in the Controller Register Address or select "RUN" from the Name pull-down list. Note that the correct Register Width must be selected first if the Controller Register is to be selected by name. Only I/O names associated with registers of the same Register Width will be available.

**Note:** There are 200 text tables to use per program. A text table is not linked to only one register... the registers are linked to a text table. It's just a lookup table for any register to reference. Therefore, one text table can be linked to by as many different registers as needed. For example, a text table containing "OFF" and "ON" for the values 0 and 1 can be linked to, multiple times, by different registers on the same or different screens, all of which can be displayed as "OFF" or "ON" for the values 0 and 1.

8. Click the 'Text Table' button about halfway down the configuration window to bring up the text table editor.
9. Note in the upper right that table 1 is being edited. Click the 'Add' button and link the value of 0 to the text "STOPPED". Click 'OK' and note the entry in the text table.
10. Click the 'Add' button again and link the value of 1 to the text "RUNNING". Click 'OK'. Click 'OK' again to exit the text table editor. Note that whichever text table is visible when 'OK' is clicked in the text table editor is the one that will be linked to this Text Table data field.



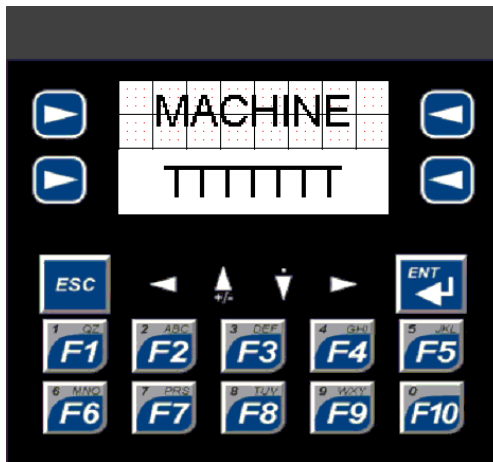
The dialog box is titled "Edit/View Text Tables". It features a table with two columns: "Value" and "Text".

Value	Text
0	STOPPED
1	RUNNING

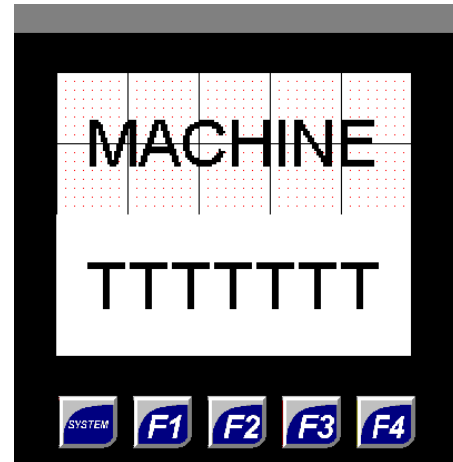
On the right side, there is a "Table Number:" field set to 1, and buttons for "Add", "Edit", and "Remove". At the bottom right, it says "Bytes Used: 32" and has an "OK" button.

## **Lab 1: Basic OCS Configuration and Programming**

11. Back in the Text Table Data Properties window, in the Digits area, make sure there are at least 7 “digits” specified (‘RUNNING’ and ‘STOPPED’ both have 7 letters) or the text will get cut off.
12. Change the Font to match the static text box if desired.
13. Click the ‘Attributes’ button towards the bottom. Uncheck the ‘Border’ box and click ‘OK’.
14. Click the ‘Legend’ button at the bottom. Delete the legend text and click ‘OK’
15. Click ‘OK’ again to exit the configuration window. Resize and move the text table data field if needed to show all of the data. The screen might look something like the ones shown here but remember... it’s up to the designer.



**XLe**



**XLt**

16. Once satisfied with the look of the screen, close the Graphics Editor just like any other program by clicking on the X at the top right of the window... changes are automatically saved.


Step 10

➤ **Save the program.**

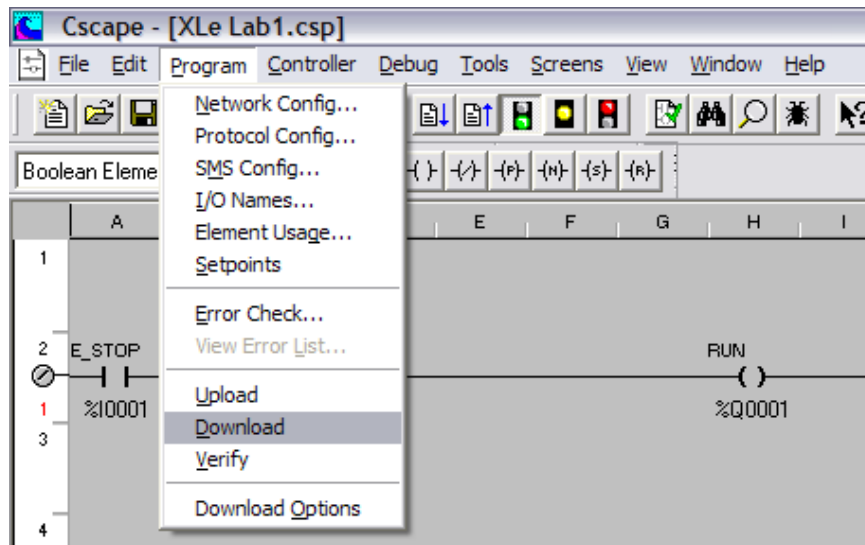
## **Lab 1: Basic OCS Configuration and Programming**

### Step 11

#### ➤ **Download the program to the OCS.**

1. Click  in the Cscape toolbar, or select the **Program** menu and click **Download**.

**Note:** It is not required to take the OCS out of RUN mode before downloading a program. The download will place the controller in STOP mode, download the program, and then return the controller to the mode it was in before the download.




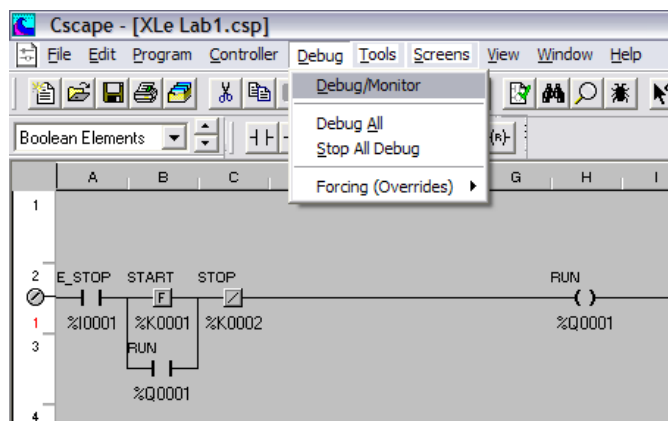
2. Use the SmartLoad function when the Download dialog box appears.
3. Click 'OK'.

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

### Step 12

#### ➤ **Debug the program**

Click  in the Cscape toolbar, or click on the **Debug** menu and select **Debug/Monitor**





## **Lab 1: Basic OCS Configuration and Programming**

### **Close switch 1 on the Input Simulator.**

Switch 1 is connected to the first digital input, which is addressed to %I01.

- ❑ In Cscape, using Debug, E\_STOP & STOP should now be red.
- ❑ On the OCS, the screen should display MACHINE STOPPED.

### **Push the F1 key on the OCS.**

- ❑ In Cscape, using Debug, START should turn red until the F1 key is released.
- ❑ The RUN coil and contact should both turn red.
- ❑ On the OCS, the screen should change to display MACHINE RUNNING.
- ❑ Output 1 should turn ON; you may be able to hear a relay output click.

### **Push F2, the 'STOP' button, on the OCS.**

- ❑ The output should turn OFF
- ❑ The screen should show MACHINE STOPPED.

### **Experiment with the different aspects of the program.**

- ❑ Press F1 to start the circuit again.
- ❑ Use the E\_STOP switch to stop it instead of F2.
- ❑ Try starting the circuit without first closing the E\_STOP switch... does it work?

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

## **Lab 1: Basic OCS Configuration and Programming**

**Notes:**

# **LAB 2**

## Screen Manipulation

## **Lab 2: Screen Manipulation**

## **Lab 2: Screen Manipulation**

### **Objective:**

The objective of this lab is to demonstrate several different methods used to manipulate screens through ladder logic and through the “Screen Jump” object in the Graphics Editor. This is how the programmer will determine when any given screen will be displayed.

### **Screen Overview**

When using Horner APG graphics-based controllers, there are 1023 screens to use in the program. These screens are all there already, just blank and waiting for the programmer to put things on them. 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 objects 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.

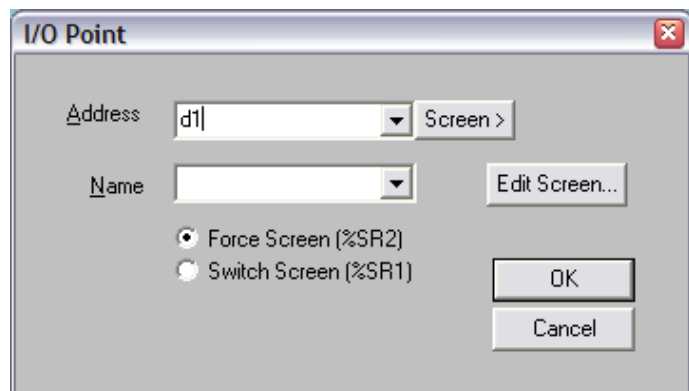
Horner OCS uses dedicated %D bit-length registers that are numbered the same as the screen they represent. %D1 is for screen 1, %D312 is for screen 312, and so on up to %D1023 for screen 1023. They can be addressed to an output coil to switch to or force a screen. They can also be addressed to input contacts to indicate when a particular screen is currently being displayed.

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. %SR1 should always have a number between 1 and 1023 and never be zero. %SR2 and %SR3 normally show zeroes unless a screen is being forced or a system screen is being displayed, in which case they show a non-zero number.

### **Part 1 – Switching and Forcing**

%D registers can be used as output coils to control screens. To do this, simply place a normally open coil in the ladder logic and then configure it for a %D register. This can be done by typing a %D reference directly into the address box or by clicking the ‘Screen >’ button to choose a screen from thumbnail pictures of configured screens.

When configuring a coil for a %D register, the configuration box changes to account for some additional options.




## Lab 2: Screen Manipulation

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

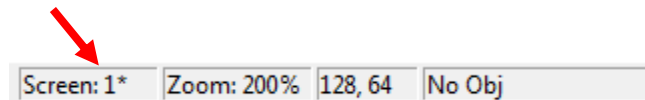
When **forcing** a screen, the screen will be forced to display for as long as the coil has power. The screen number of the screen being forced will also be reflected in %SR2... note the reference to %SR2 next to the Force Screen option. If a screen is forced, the value in %SR1 is not affected and stays 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, %SR2 will have a value of 20 and screen 20 will be displayed. When screen 20 is released from its force, %SR1 will still have a value of 51, %SR2 will have a value of 0 and screen 51 will be displayed.

When **switching** a screen, the screen specified by the %D register will be switched to and will continue to display even after power to the coil has been lost. This change is reflected in %SR1... note the reference to %SR1 next to the Switch Screen option. For example, if screen 51 is being displayed and the screen is switched to screen 30, the value in %SR1 will change from 51 to 30 and screen 30 will be displayed.


### Switching and Forcing Lab

1. Create a new program. Configure the I/O as shown in Lab 1.
2. Using the Graphics Editor (click  on the toolbar... or from the **Screens** menu, select **View/Edit Screens...**), 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".

**Note:** In the Graphics Editor, the screen currently displayed in the editor is indicated in the status bar at the bottom right of the screen...



The asterisk indicates that this is the Initial Screen, the one that shows up when the OCS is first turned on or put into RUN mode.

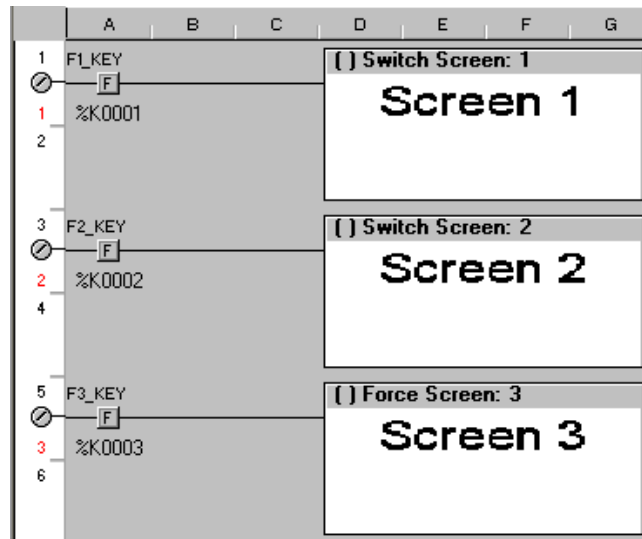
To navigate to different screens in the editor, use the arrow keys or the thumbnail selector in the toolbar... 

3. Once done labeling the screens, exit the Graphics Editor and save the program.
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 configuring the coil, clicking the 'Screen>' button will bring up a screen thumbnail selector. A screen to associate the coil with may be chosen 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. The program should look similar to what is shown on the next page. Note how the coil symbols have


## **Lab 2: Screen Manipulation**

changed into a thumbnail version of what is on that particular screen specified for that particular coil.

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 the button is held down. When the button is released, the screen previously viewed will come back up. Try pressing F3 from both screen 1 and screen 2 to see this.

Open a Data Watch window in Cscope by clicking on the magnifying glass next to the debug button.  Add %SR1 and %SR2 as INT values to watch the system registers and what they do when the buttons are pressed.

**Note:** More than one value may be added at a time to the Data Watch window by clicking the 'Add' button and typing in 'SR1-2' or 'R51-75', for instance. This will work for up to 50 registers at a time.

### **Part 2 – Changing the System Registers**

Though there is no lab portion on this method, another way of displaying a screen is to directly move a value into one of the screen system registers. Using a 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.

Cscope 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 'Force Screen' or the logic Alarm handler can be used.

### **Part 3 – Screen Jumps**

One 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.




## **Lab 2: Screen Manipulation**

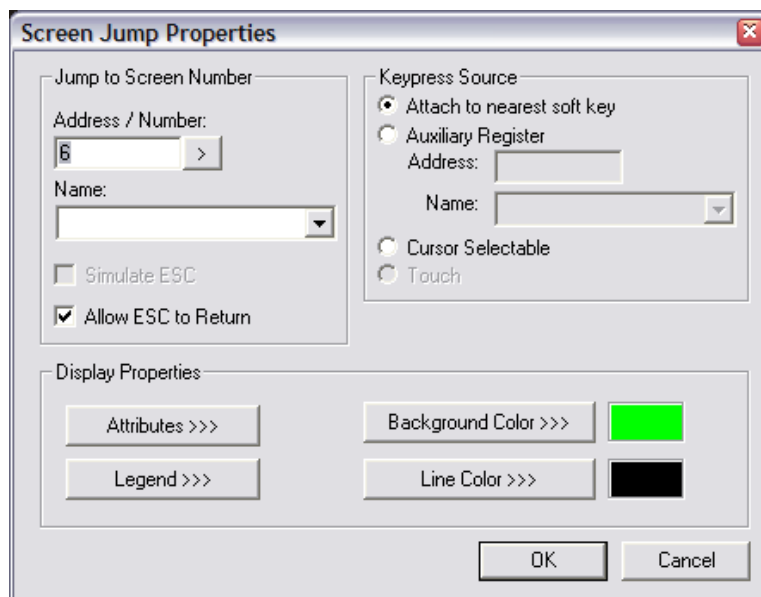
One advantage to using Screen Jumps, in addition to not having to program ladder logic to do it, is that a menu-like structure can be simulated. (See Part 4 of this lab for other menu functionality.) When configuring a Jump Screen, there is an option to “Allow ESC to Return”. With this option checked, using the jump keeps the last page on an internal memory stack so that it can be recalled. Pressing the ESC key will recall the page from which the current page was jumped to. On touch-screen controllers that don’t have an ESC key built in, a screen jump can be configured with the option to “Simulate ESC”. Up to 16 “layers” can be recalled in order to back up through a menu system.

**Note:** When mixing Screen Jumps and Ladder Logic control of screens, the “Allow ESC to Return” memory stack is erased as soon as Ladder Logic switches or forces a screen. Be careful!

### **Screen Jumps Lab**

Adding to the program from Part 1:

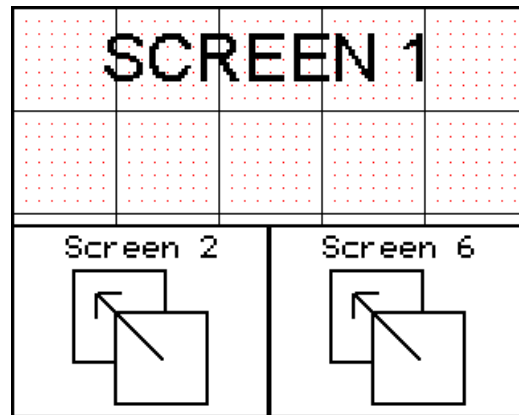
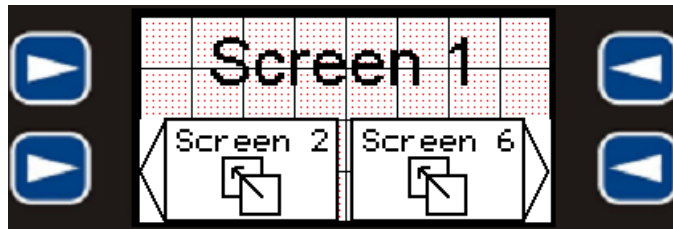
1. Using the graphics editor, configure screen 6 with a Static Text label reading “Screen 6” as done with the other screens.
2. Go to screen 1 and add a Screen Jump . Draw it out in the same fashion used with Static Text or Text Table data. On models without a touch-screen, such as the XLe, the Screen Jump will be linked to the nearest softkey... simply drag it around once it is drawn to attach it to the desired one. The softkeys are the buttons on the side of the screen with arrows on them   and can be linked to on-screen objects. On touch-screen controllers, such as the XLt, 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. Either type “6” into the Address/Number box or click the ‘>’ button to select the desired screen from a screen thumbnail list. Check the “Allow ESC to Return” box. Change the Legend to something meaningful.



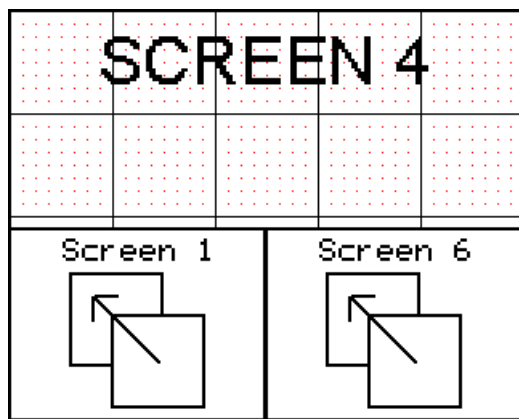
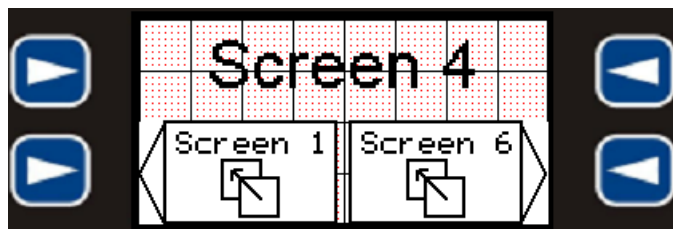
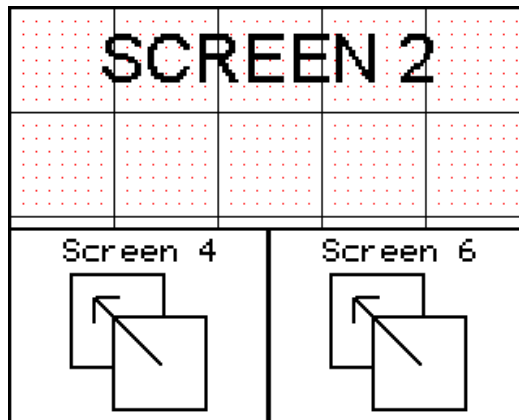
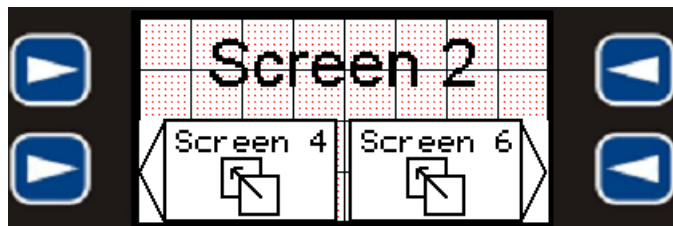


## Lab 2: Screen Manipulation

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

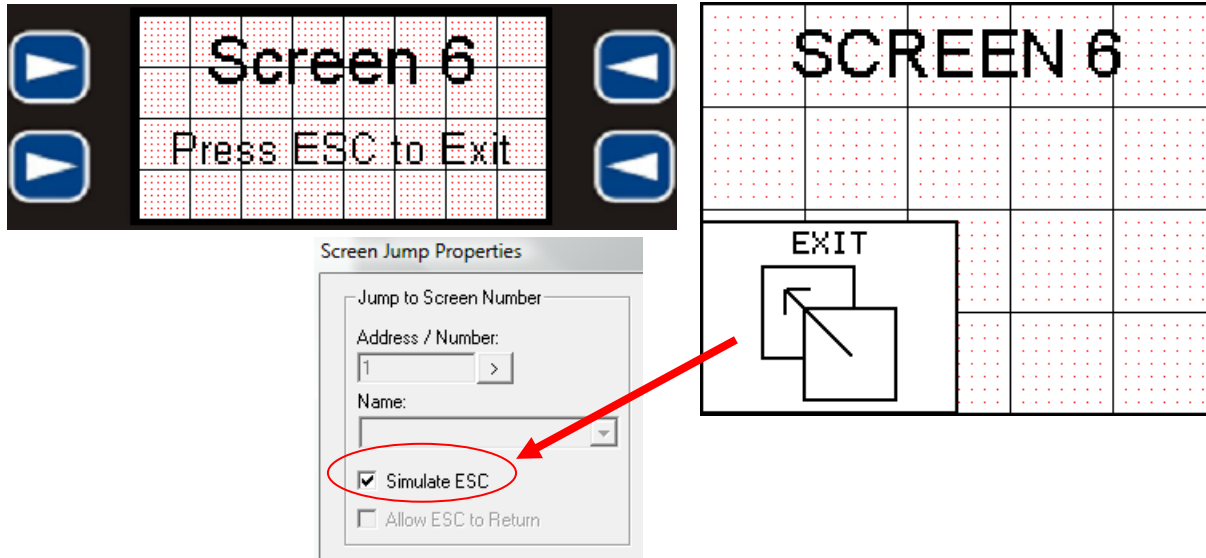


5. On screen 2, configure the same Screen Jump to screen 6 (copy and paste it from screen 1 if desired). Make sure “Allow ESC to Return” is checked.
6. Add another Screen Jump to go to screen 4. Again, do NOT check “Allow ESC to Return” for this jump.
7. Repeat step 5 and 6 for screen 4. Make the first Screen Jump go to screen 1.



## **Lab 2: Screen Manipulation**

8. If using a controller with a built-in ESC key, like the XLe, add a note on the screen to inform the user to press the ESC key to exit, then skip to step 10.
9. For touch-screen controllers without a built-in ESC key, like the XLt, go to screen 6 and put in another Screen Jump. Double-click on it to configure it and check the “Simulate ESC” box. Change the legend to display ‘EXIT’. This is all that is needed for this jump.



10. Exit the Graphics 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 because the controller remembers where it was since the “Allow ESC to return” option was checked.

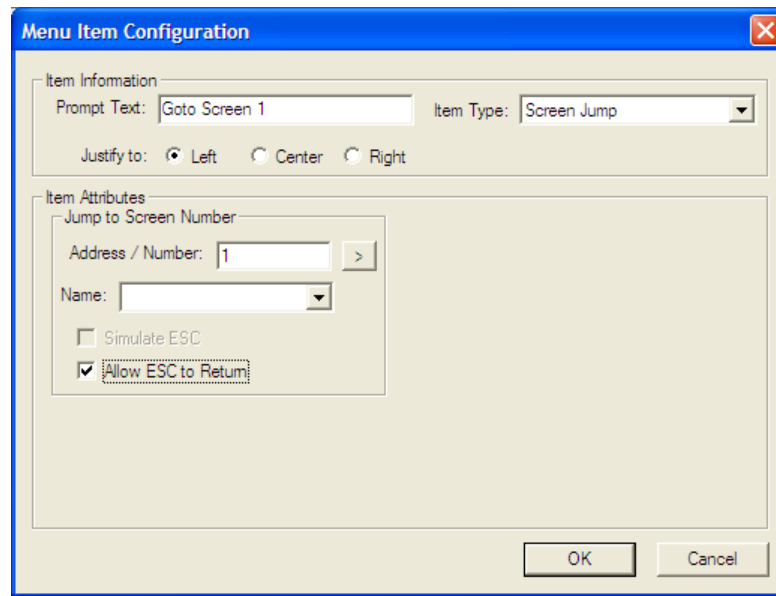
### **Part 4 – Menu Object Lab**

On XLe and NX22x controllers, the display is small (128x64) and 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.

**Note:** The menu object is considered a data field. By default, it is configured for “Exclusive Focus”, which means it is a data field that is always selected for “editing” and that there is no way to get out of editing it. When configured this way and a menu screen is being displayed, the %K registers and any object linked to the soft keys will NOT function. The %SR56 Last Key system register can be used in these cases to monitor for F-Key or soft key presses. By un-checking the “Exclusive Focus”, other objects can be used on the same screen but the menu must be selected for editing before it can be used.

## **Lab 2: Screen Manipulation**

1. Start a new program. Configure the Controller and the I/O as before.
2. Open the Graphics Editor and configure screens 1 – 6 to indicate 'SCREEN #' using the Static Text as 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.
3. If using a touch-screen OCS, also place a screen jump "Exit" button on each screen and check the "Simulate ESC" box in the configuration for each.
4. Place a Menu on screen 7.  Stretch it out to cover the entire screen.
5. Double-click the Menu Object and click on Configure Menu Pages.
6. Press Add.
7. In the Prompt Text, type 'Goto Screen 1', select Screen Jump in the Item Type, put a 1 in the Address/Number, check the 'Allow ESC to Return', and press OK.



The dialog box is titled "Menu Item Configuration". It has a blue title bar with a close button (X) in the top right corner. The dialog is divided into two main sections: "Item Information" and "Item Attributes".

**Item Information:**

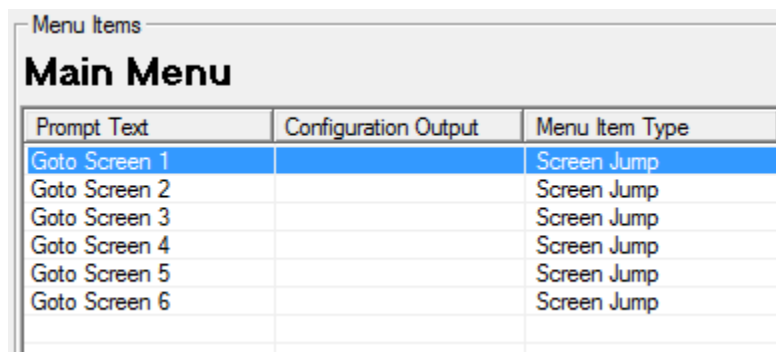
- Prompt Text:** A text field containing "Goto Screen 1".
- Item Type:** A dropdown menu showing "Screen Jump".
- Justify to:** Three radio buttons: "Left" (selected), "Center", and "Right".

**Item Attributes:**

- Jump to Screen Number:** A section containing:
  - Address / Number:** A text field containing "1" with a right-pointing arrow button next to it.
  - Name:** A dropdown menu.
- Simulate ESC:** An unchecked checkbox.
- Allow ESC to Return:** A checked checkbox.

At the bottom right of the dialog are "OK" and "Cancel" buttons.

8. Repeat steps 6 and 7 for screens 2-6, then click 'OK' to return to the main menu configuration page.



The table is titled "Menu Items" and "Main Menu". It has three columns: "Prompt Text", "Configuration Output", and "Menu Item Type".

Prompt Text	Configuration Output	Menu Item Type
Goto Screen 1		Screen Jump
Goto Screen 2		Screen Jump
Goto Screen 3		Screen Jump
Goto Screen 4		Screen Jump
Goto Screen 5		Screen Jump
Goto Screen 6		Screen Jump

9. Change the Legend of the Menu Object to reflect 'Screen Manipulation'. Click 'OK' to complete the configuration of the menu object.

## **Lab 2: Screen Manipulation**

10. Configure screen 7 as the initial screen by clicking on the **Screens** Menu of the Graphics Editor, selecting **Set Initial Screen** and putting a 7 into the First Screen to display box. Click 'OK' when done.
11. Once done, close the Graphics Editor, save the program, and download the program to the controller.
12. 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 the user to navigate back to the Menu Screen.

### **Extra Credit #1**

**Note:** This Extra Credit requires the use of Move and Comparison functions that have not yet been covered in the lecture. The solution is on the next page.

On non-touch-screen controllers, pressing the up and down arrows simultaneously gets into the system menu. Touch-screen controllers have a System key on the keypad.

Add ladder logic to the 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.

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.

### **Extra Credit #2**

**Note:** This Extra Credit requires the use of Move, Comparison and Math functions that have not yet been covered in the lecture. The solution is on the next page.

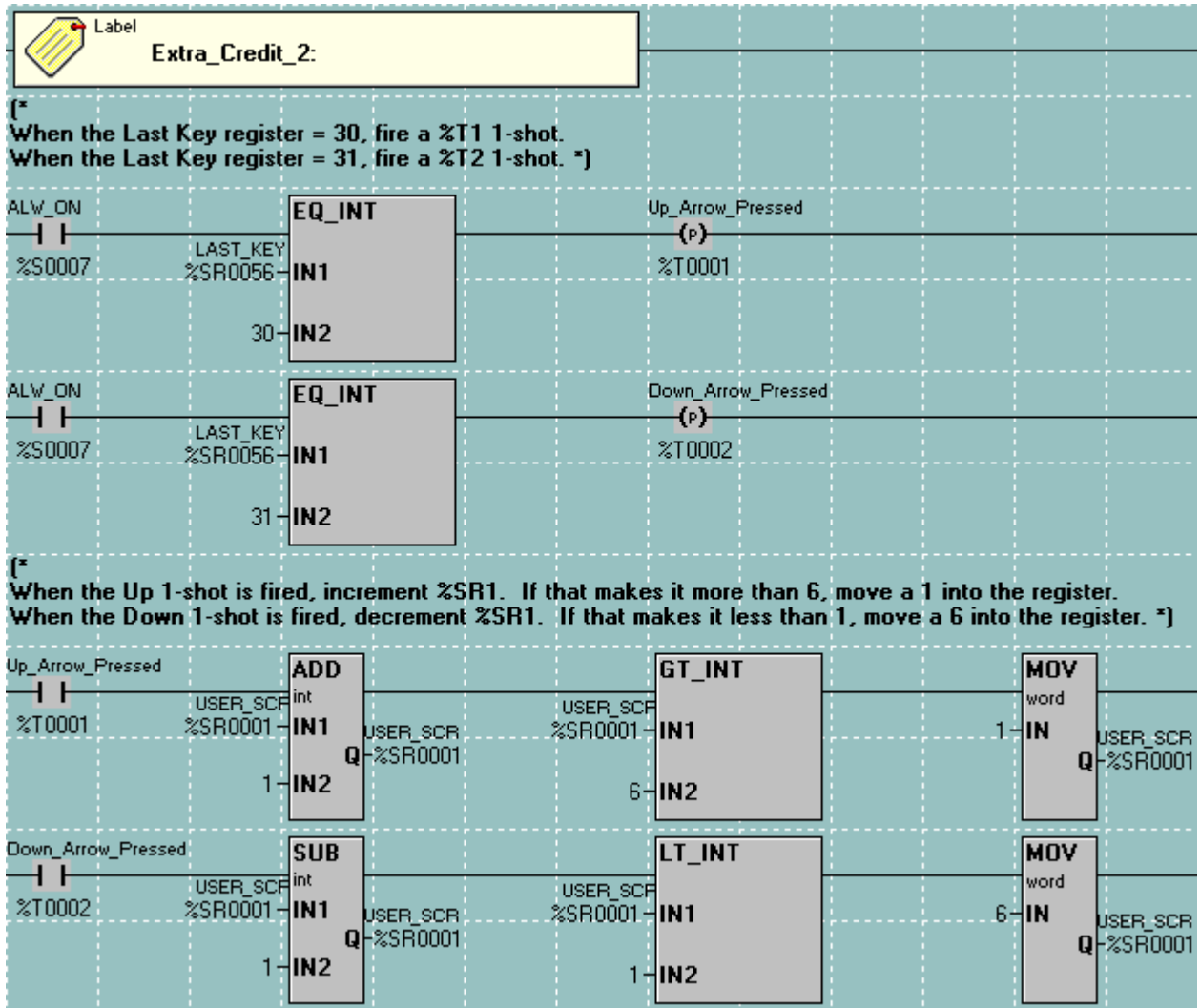
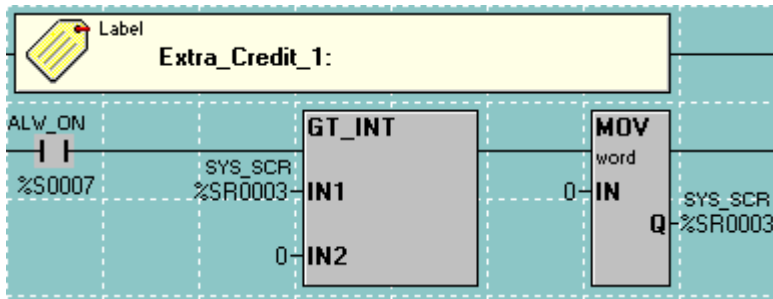
Create ladder logic to scroll through the screens on the controller using the up and down arrow keys on the keypad.

%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. When no key is being pressed, the value in %SR56 is either 0 (no key has been pressed since power-up) or 256 (some key has been pressed since power-up).

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... a value of 6 will have to be moved into %SR1 to "wrap around". Be careful when pressing the Up key from screen 6... a value of 1 will have to be moved into %SR1 to "wrap around". If any screens are to be skipped in between 1 and 6, the code to do that will have to be added as well. 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.

## Lab 2: Screen Manipulation

### Extra Credit Solutions:



## **Lab 2: Screen Manipulation**

**Notes:**

# **LAB 3**

## Timers and Counters

### **Lab 3: Timers and Counters**



## Lab 3: Timers and Counters

### Objective:

Review and understand Timers and Counters and the different types of each.

### Timers Overview:

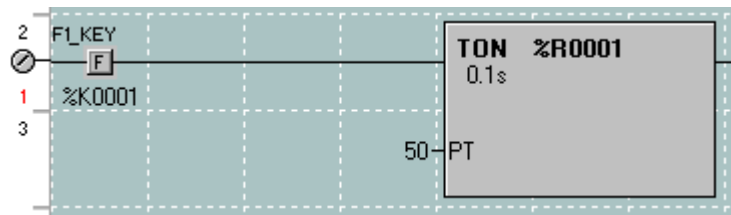
Horner timers consume 2 consecutive word-length registers. Assigning a timer to %R1 means that the timer will consume %R1 and %R2. The first of the two registers contains the accumulated time of the timer. In the second register, the first 14 bits contain housekeeping information and should not be tampered with. The last 2 bits of the second register are status bits for the timer. The 15<sup>th</sup> bit is an “Enabled” status, indicating whether or not the timer currently has power to it. Also, the 15<sup>th</sup> bit is only useful for TON timers. The 16<sup>th</sup> bit of the second word is a “Done” status, indicating when the timer is passing power to the rest of the rung it is on.

A single bit of a word-length (16-bit) register may be addressed in ladder logic as, for example, %R2.15 for the 15<sup>th</sup> bit of %R2 or %R2.16 for the 16<sup>th</sup> bit of %R2.

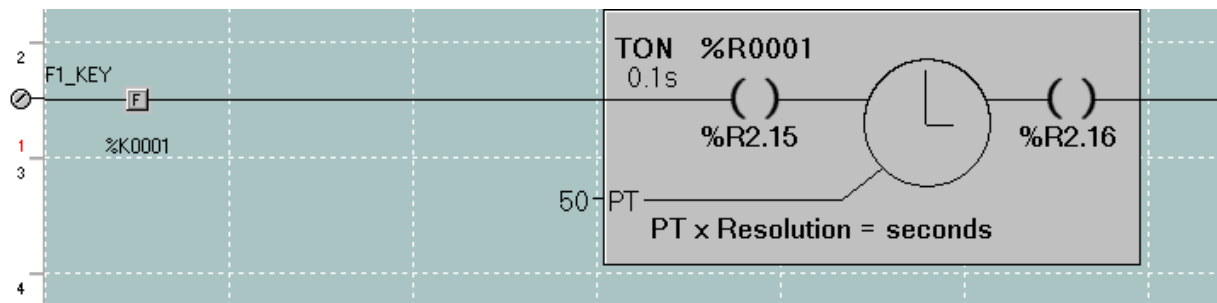
A third word-length register, separately and not necessarily consecutive, can be assigned as the “PT”, or Preset Time. However, the PT can also just be a static value and, therefore, “hard-coded” into place and not consuming any register memory.

**Note:** The standard register type to assign a timer would be %R registers. Don’t let the %T registers be tempting. The ‘T’ stands for Temporary, not Timer.

A timer appears like this in ladder logic:



However, the internal logic of a Timer can be thought of like this:



In the above example, a 100ms-resolution (0.1s) TON timer is configured at %R1 and has a hard-coded PT value of 50.  $PT \times Resolution = \text{time}$ , so  $50 \times 0.1s = 5.0 \text{ seconds}$ ... this timer has a PT of 5.0 seconds.

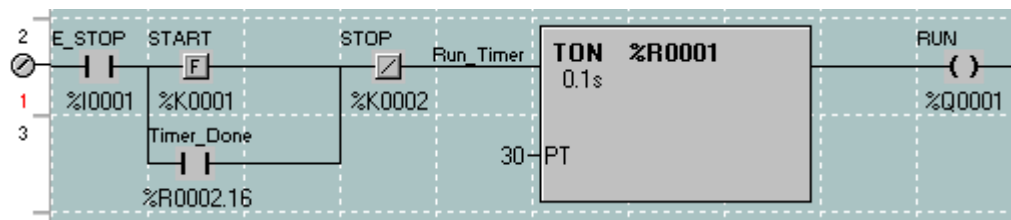
If F1 is not pressed:	%R1 = 0, %R2.15 = OFF, and %R2.16 = OFF
If F1 is pressed for 2.5 seconds:	%R1 = 25, %R2.15 = ON, and %R2.16 = OFF
If F1 is pressed for 5 seconds or more:	%R1 = 50, %R2.15 = ON, and %R2.16 = ON

## **Lab 3: Timers and Counters**

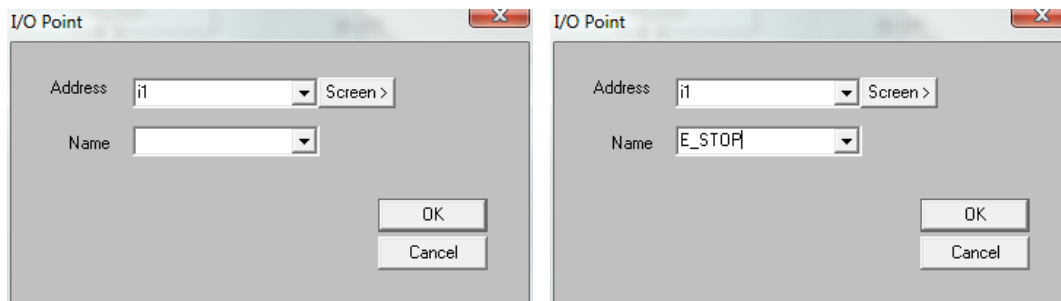
### **Part 1 – TON Timers Lab:**


TON timers start timing when they receive power and pass power to the rest of the rung when they reach their Preset Time. If a TON timer loses power, it resets itself to zero immediately.

1. Create a new program.
2. Title the program “Timers.csp” or some such thing.
3. Set the target ID to match the controller to be programmed.
4. Configure the controller. (Reference Lab 1 for correct procedure)
5. Create a modified version of the Run circuit from Lab 1. Require the user to hold the ‘START’ button for 3 seconds before the circuit is energized. Do so with a 100ms-resolution TON timer and use the “Timer Done” bit to hold the circuit on.



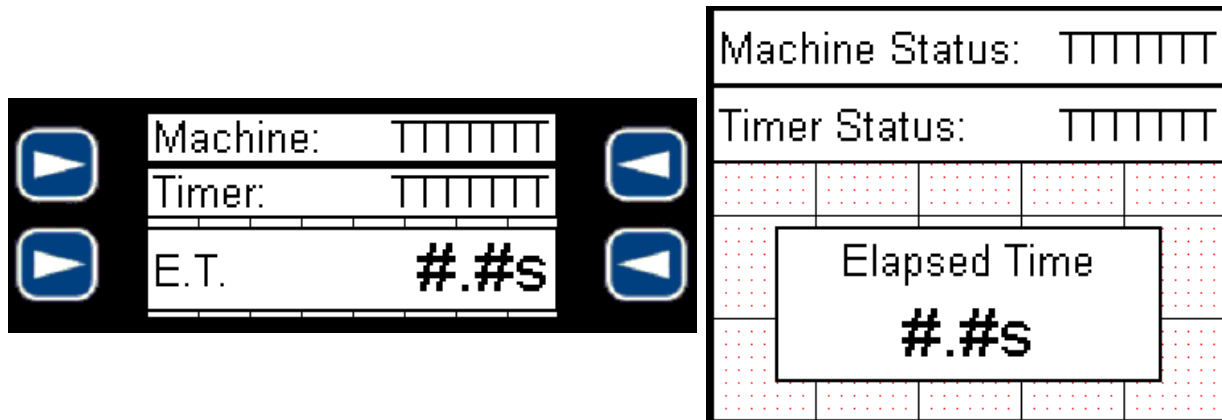
- Since this is a new program without the I/O names entered from Lab 1, try naming the I/O in this lab on the fly. Type the address in, and then give it a name in the ‘Name’ box. Existing names may be overwritten.



6. Open the Graphics Editor and configure a text table on the screen to show ‘STOPPED’ or ‘RUNNING’ (reference Lab 1 if needed). Link it to the %Q1 ‘RUN’ output.
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.
8. 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 2 with 1 decimal place. Add an ‘s’ for the Engineering Units to indicate that the data displayed is in seconds

### Lab 3: Timers and Counters

- Label each field on the screen so they can be told apart. This can be done by modifying the legend for each of the objects placed on the screen. The screen might look something like this in the Graphics Editor:



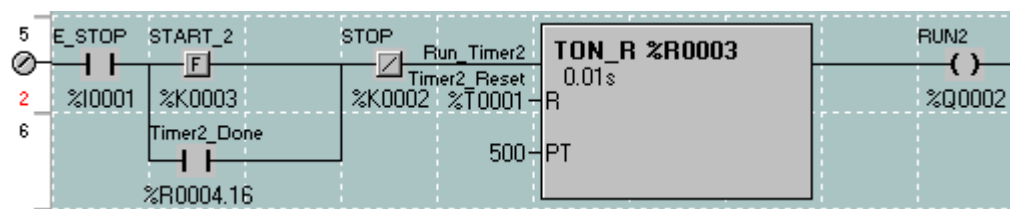
- Save and download the program to the controller and make sure it is in RUN mode.

Operate the START/STOP circuit and take notice of how things work. Watch the accumulated value of the timer counts up as the F1 key is held down... watch how it zeroes out if F1 is let go of before the 3 seconds is up or if the circuit is stopped with the F2 key or the E\_STOP switch. Notice how the %R2.16 "Timer Done" contact holds the circuit on in the same way that the %Q1 contact did in Lab 1.

#### Part 2 – Retentive TON Timers Lab

Retentive TON timers start timing when they receive power and pass power to the rest of the rung when they reach their Preset Time, just like a standard TON timer. However, if a retentive TON timer loses power, it keeps track of where it is and resumes from where it left off when it receives power again. Since it does not reset itself automatically upon losing power, a retentive TON timer requires a Reset Input to be configured in order to tell it when to zero out and start over.

- Add a second "machine" to the program from part 1. This machine will operate using the same E-STOP and STOP buttons, but will have the F3 key as the START button. Use a retentive TON timer with 10ms resolution that requires the user to press and hold the START button for a total of 5 seconds to start this second circuit with a RUN coil at %Q2. Reset this timer using %T1.

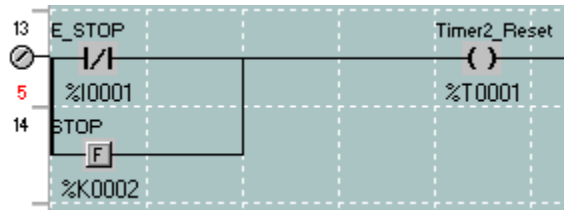


HINT: Since this timer is set for 10ms resolution, 5 seconds is equal to a 'Pt' of 500. 500 10ms pulses equals 5.00 seconds.

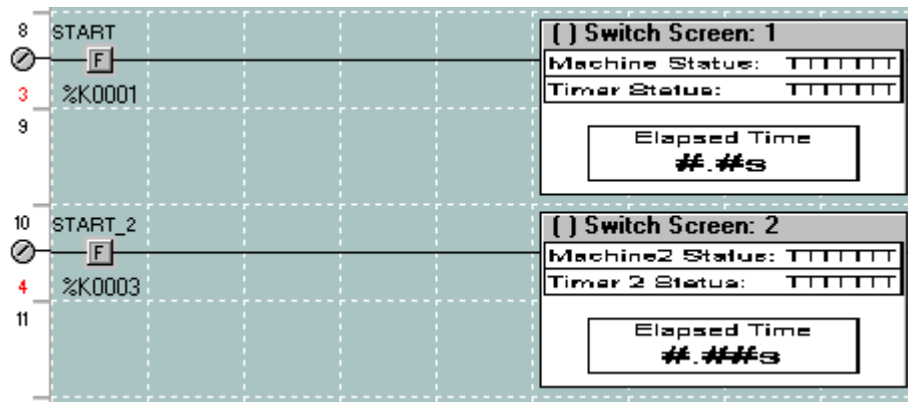
HINT: Remember that each timer takes up 2 %R registers. Make sure this timer doesn't overlap the timer from Part 1... address this one to %R3.

### **Lab 3: Timers and Counters**

2. Create the logic to reset this timer based on the E-STOP switch or the STOP button.



3. Open the Graphics Editor and create a new screen on screen 2 similar to screen 1 from Part 1. This may be done as follows:
  - Go to screen 1
  - Press 'Ctrl-A' to Select All and then press 'Ctrl-C' to Copy it.
  - Go to screen 2 and press 'Ctrl-V' to paste what was copied.
  - Everything pasted is part of one group... to un-group it, right-click anything in the group on screen 2 and select 'Ungroup'. Now they are separated.
4. Change the pasted data fields to be linked to circuit 2:
  - Change the 'Machine Status' text table to be linked to %Q2.
  - Change the 'Timer Status' text table to be linked to %R4.15.
  - Change the 'Elapsed Time' data field to be linked to %R3. Also, add another decimal place and another digit to accommodate this higher-resolution timer.
5. Close the Graphics Editor. Now use the knowledge from Lab 2 to switch to the proper screen based on which machine's START button is being pressed.



6. Save and download the program. With Machine 2, note how the Elapsed Time keeps its place when letting go of the F3 key. Press F2 or flip the E-STOP switch to reset it.

## **Lab 3: Timers and Counters**


### **Part 3 – TOF Timers Lab**

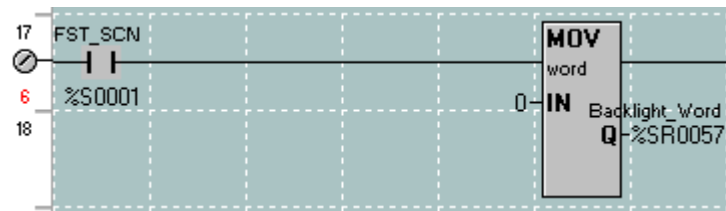
TOF timers are somewhat backwards. A TOF timer will pass power to the rest of the rung as soon as it receives power, also resetting its accumulated value at this time. When the TOF timer loses power, it continues to give power to the rest of the rung to the right of it and starts timing towards the PT. When it reaches its preset time, it will discontinue power to the rung to the right of it. It can be thought of like the operation of a projector with an expensive light bulb... power to the cooling fan is continued for some amount of time after the bulb is turned off in order to cool the bulb down properly and keep it from breaking.

1. To the program from Parts 1 and 2, add a TOF timer to turn off the backlight of the OCS 8 seconds after both machines have stopped. Turn the backlight back on whenever either circuit's START button is pressed.

HINT: Referencing the Cheat Sheet at the back of this manual, look in the System Registers section and find the "LCD Backlight" register... it is %SR57. For non-touch-screen controllers, the backlight is turned off if this register is zero and turned on if it is anything other than zero. For touch-screen controllers, the backlight is dimmable from a value of 0 (off) to 100 (full brightness). On non-touch-screen controllers, if this register is initialized to a value of 0 on first-scan (%S1 system bit), only the first bit of %SR57 needs to be altered to turn the backlight on or off.

HINT: Remember not to overlap the timers! Use registers that haven't been used yet.

Do the programming yourself if possible, but one solution is at the end of the Timers and Counters lab. However, since the Move lab hasn't been done yet, here is how %SR57 can be initialized to a value of 0 on first-scan. The MOV function can be found in the "Move Operations"... . Simply type in a '0' for the source and use %SR57 for the destination, naming it "Backlight\_Word" in the process.



2. After writing the rest of the program, download and make sure the controller is in RUN mode.

Use the Debug function to monitor the TOF timer and take note of how it works. When either START button is pressed (F1 or F3), the backlight should come on. If the circuit is successfully started, the backlight should stay on indefinitely. If the circuit is stopped using the E-STOP or STOP (F2) button, or if the START was unsuccessful, the light should go out after 8 seconds.

Note also how the TOF timer kept the backlight on for 8 seconds when the program was first downloaded. This is because the %R register used for this timer hasn't been used before in this controller and started with a zero in it. It may not work the same in subsequent downloads or power-ups because %R registers are retentive and remember where they were before. This could be remedied by adding another MOV function on the first-scan line to also initialize the first register of the timer to zero.

## **Lab 3: 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 Counter and a Count-Down Counter.

Horner counters also consume 2 consecutive word-length registers. Assigning a counter to %R51 means that the timer will consume %R51 and %R52. The first of the two registers contains the accumulated count of the counter. In the second register, the first 14 bits contain housekeeping information and should not be tampered with. The last 2 bits of the second register are status bits for the counter. The 15<sup>th</sup> bit is an “Enabled” status, indicating whether or not the counter currently has power to it. The 16<sup>th</sup> bit of the second word is a “Done” status, indicating that the counter has reached or exceeded its count and is now passing power to the rest of the rung it is on.

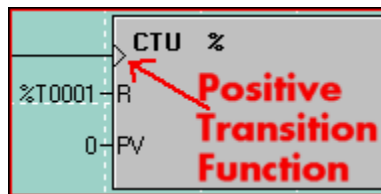
IMPORTANT: The Counter will continue to pass power to the rest of the rung if it has reached or exceeded its count even if it is no longer being powered. It must be reset to stop passing power at that point.

A third word-length register, separately and not necessarily consecutive, can be assigned as the “PV”, or Preset Value. However, the PV can also just be a static value and, therefore, “hard-coded” into place and not consuming any register memory.

**Count-Up counters (CTU)**, when reset, initialize their accumulated value to 0 and count up from there, passing power when they reach their preset value (PV). It will continue to count upwards past the PV with each additional trigger.

**Count-Down counters (CTD)**, when reset, initialize their accumulated value to the preset value (PV) and count down from there, passing power when they reach 0. It will continue to count downwards past 0 with each additional trigger.

Counters increment or decrement only once every time they see power come on from the ladder rung. They must “lose” power and then be powered again for the count to increment. This is what the little triangle at the counter input means:



### **Part 4 – Counters Lab**

1. To the program from the timer labs, add a CTU counter that will count the number of times machine 1 has been successfully started.
2. Add a CTD counter that will only allow machine 2 to be started 3 times before the operator must reset the counter to start the machine again.

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

3. Make the 2<sup>nd</sup> input on the input simulator reset the counter for machine 1.
4. Make the 3<sup>rd</sup> input on the input simulator reset the counter for machine 2.
5. Add a data field to machine 1's screen to indicate the number of times that machine has been started. This data field will display the CTU counter accumulator.

### **Lab 3: Timers and Counters**

6. Add a data field to machine 2's screen to indicate how many more times the machine can be started before requiring a reset. This data field will display the CTD counter accumulator.

The screens may look something like this:

▶	Machine 1: TTTTTT		◀
▶	Timer: TTTTTT		◀
▶	E.T. #.s	STARTS #####	◀

▶	Machine 2: TTTTTT		◀
▶	Timer: TTTTTT		◀
▶	E.T. #.s	RESET IN #####	◀

Machine 1: TTTTTT						
Timer: TTTTTT						
<table border="1"><tr><td> </td><td> </td><td> </td><td> </td><td> </td></tr></table>						
Elapsed Time #.s	Starts #####					

Machine 2: TTTTTT						
Timer: TTTTTT						
<table border="1"><tr><td> </td><td> </td><td> </td><td> </td><td> </td></tr></table>						
Elapsed Time #.s	Reset Req'd in #####					

7. Download and make sure the controller is in RUN mode.

Start each machine and notice how the different counters increment or decrement with each successful start. Make sure Machine 2 cannot be started again if it has been started 3 times and has not been reset. Note how each counter type resets when their respective reset inputs are triggered.

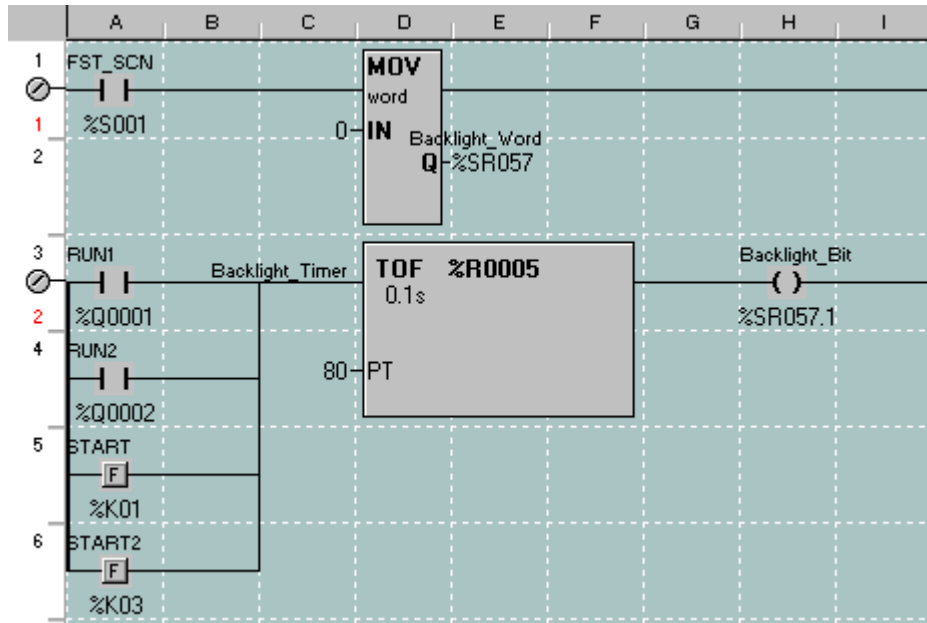
**CONGRATULATIONS, YOU'VE FINISHED TIMERS AND COUNTERS!**

## Lab 3: Timers and Counters

### TOF Solution:

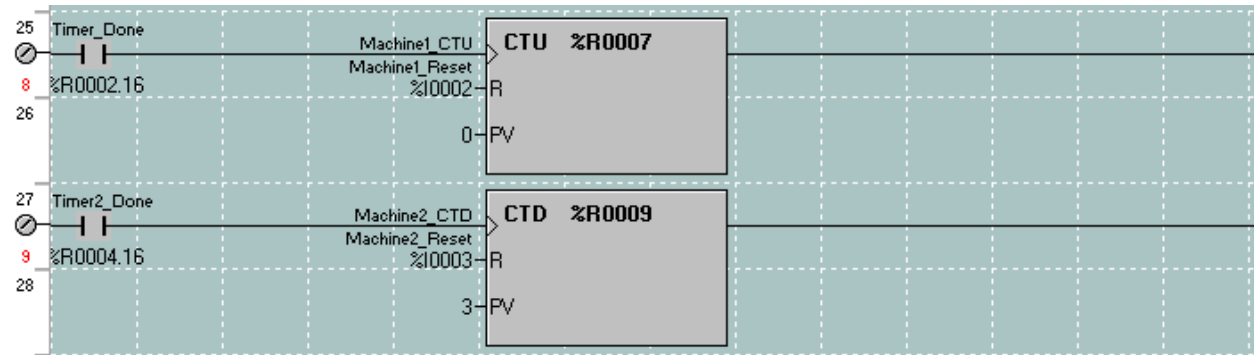
Note how either start button will fire the backlight bit for at least 8 seconds. If either RUN circuit becomes active, it will hold the backlight on until 8 seconds after both RUN circuits are turned off.

**Note:** A different solution must be used if using a touch-screen OCS.

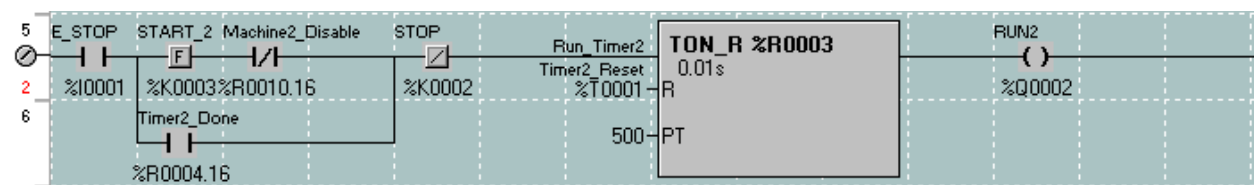


### Counter Lab Solution:

Keep in mind that there are usually many different ways to program a controller, all of which get the task done. Below is one solution to the Counter lab. Note how the CTU counter has a PV of 0. It is only being used for its counting and not really counting towards a goal.



To disable Machine 2 from starting any more if the CTD counter is done, add the "Machine2\_Disable" normally-closed contact to the existing machine 2 start/stop circuit to interrupt the START button from working until the counter is reset:





### **Lab 3: Timers and Counters**

**Notes:**

### **Lab 3: Timers and Counters**

**Notes:**

# **LAB 4**

## Move Operations

## **Lab 4: Move Operations**

## **Lab 4: Move Operations**

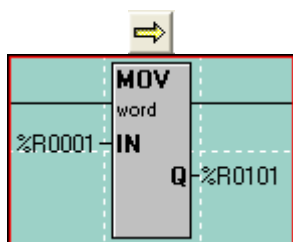
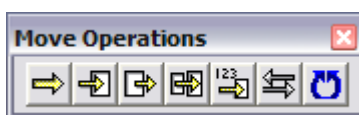
### **Objective:**

Review and understand Move Operations

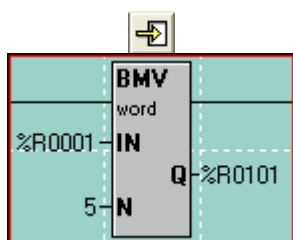
### **Overview:**

There are several types of move functions available for use for several types of different occasions. In general, move functions really act more like copying operations. They will not remove the data from the source in order to move it to the destination, they will just copy it. Also, once a move function has been powered, it will not be undone simply by removing power... the data stays put at the destination after power to the function is removed.

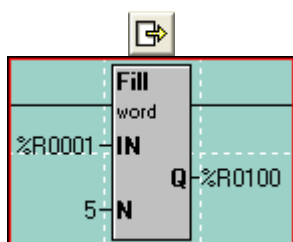
The 'Move Operations' toolbar appears as follows:



The first type of Move is the 'Move Data', 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. This is a good way to initialize a single register on first-scan, like what was done in the timers lab to initialize the backlight system register. The source (IN) can be either a register or a constant value.

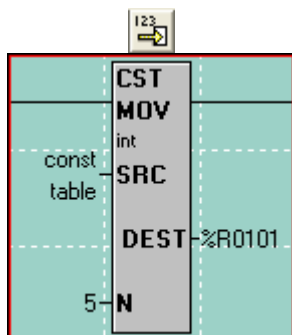


The next type of Move is the 'Block Move', or 'BMV'. It is used to copy a group of bytes, words or double-words to another location. The count (N) determines how many registers are to be copied. In the example to the left, %R1-%R5 are copied into %R101-%R105. Again, this only happens when the ladder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

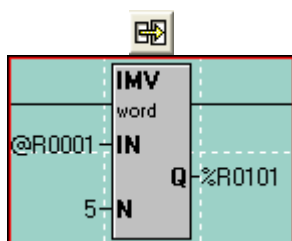


The next type of Move is the "Block Fill", or "Fill". It is used to copy the contents of a single register or value into multiple other registers, thus filling that one value into a group of registers. The count (N) determines how many registers to fill that single value into. In the example to the left, 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 source (IN) can be either a register or a constant value.

## Lab 4: Move Operations

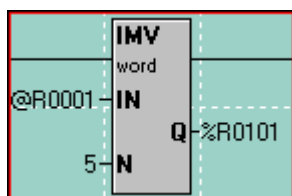


Skipping to the 'Constant Move', or 'CST MOV', it is used to move a group of constant values into a group of consecutive registers. If, for example, the values 1, 2, 3, 4 and 5 were to be moved into %R101, %R102, %R103, %R104 and %R105, respectively, the Constant Move function could be used. The count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

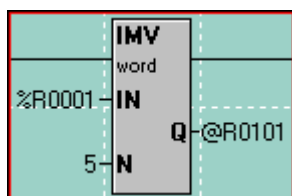


Moving back one to the 'Indirect Move', or 'IMV', it is used to move data from variable positions or to variable positions or both. It functions, for the most part, like the Block Move function. If specified as Indirect, the IN and/or the Q are used as pointers to where in the %R registers to get data 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.

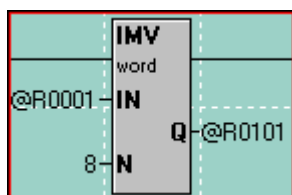
### Indirect Move Examples



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.

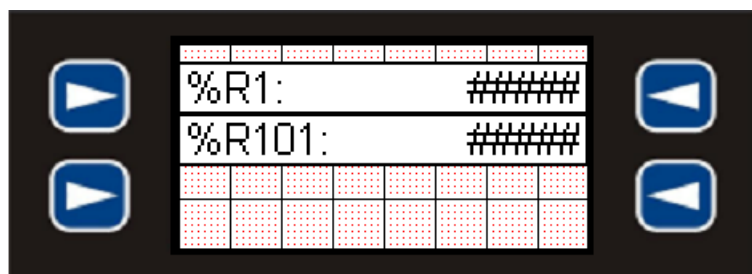


In this example, the Mother of All Confusion, both the IN and the Q are specified as Indirect. This means the controller will look at %R1 and see a value. Let's say it is 241. The controller also looks at the value in %R101. Let's say it is 341. The controller will then take the values in %R241-%R248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

## **Lab 4: Move Operations**

### **Part 1 – Move Lab**

1. Start a new program for the connected controller and call it whatever is desired. Configure the controller and I/O as done before.
2. First things first... the Timers and Counters Lab made use of the backlight register. Since the backlight register is retentive, it will remember where it was when a new program is loaded into the controller. If it was turned off and is not turned back on in this program, it will remain off! Move a value of 100 into %SR57 on first scan to make sure the backlight is on using a MOV function.
3. Move the value in %R1 to %R101 when the F1 key is pressed using a MOV function.
4. Zero out %R101 when the F2 key is pressed by moving a value of 0 into it using a MOV function.
5. 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:



%R1 #####				%R101 #####	

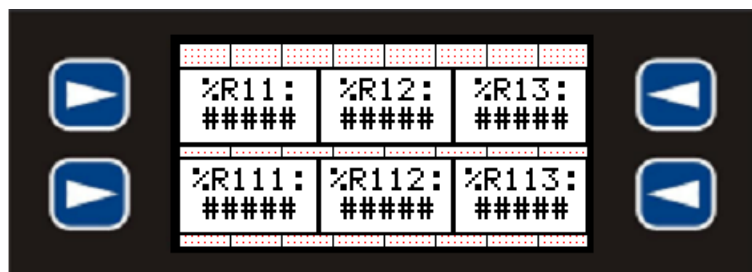
6. Add logic or a Screen Jump to switch to this screen if needed.
7. Download the program and make sure the controller is in RUN mode.
8. Edit the value in %R1 to whatever desired:
  - On XLe, use the up/down/left/right arrow keys to select the field, then press the Enter key to edit, type in a value on the keypad, and then press Enter again.
  - On XLt, simply push the data field on the touch screen like a button, enter a value on the keypad that appears, and then press the Enter button on the screen.
9. Press the F1 key to move the value just edited into %R101.
10. Press the F2 key to zero out %R101.

### **Part 2 – Block Move Lab**

1. Add programming to move the values in %R11-%R13 to %R111-%R113 when the F3 key is pressed using a BMV function.

## **Lab 4: Move Operations**

2. Create another screen with data fields to show the registers. Be sure to make the %R11, %R12 and %R13 data fields editable:

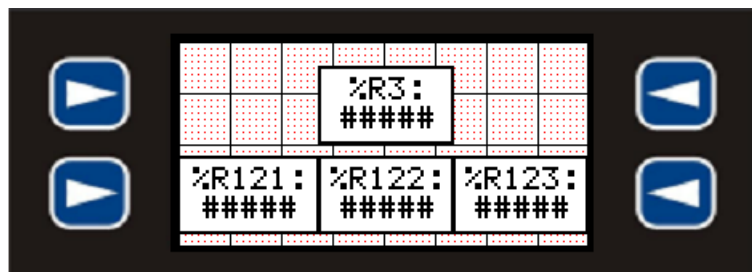


%R11 #####		%R111 #####
%R12 #####		%R112 #####
%R13 #####		%R113 #####

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 desired.
6. Press the F3 key to move all the values just edited in %R11-%R13 to %R111-%R113.

### **Part 3 – Block Fill Lab**

1. Add programming to fill the value contained in %R3 into all the registers from %R121-%R123 when the F4 key is pressed using a FILL function.
2. Fill those same registers with a value of 0 when the F4 key is NOT being pressed using another FILL function.
3. Create another screen with data fields to show the registers. Be sure to make the %R3 data field editable:



		%R121 #####
%R3 #####		%R122 #####
		%R123 #####

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 desired.
7. Press and hold the F4 key to fill the value just edited into %R121-%R123.
8. Let go of the F4 key to zero out the values in %R121-%R123



## Lab 4: Move Operations

### Part 4 – Constant and Indirect Moves Lab

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.

Move Constant Data

Source

Starting reg: 2201  
%R0201 2202  
2203  
Editing reg: 2204  
2205  
2206  
2207  
2208  
Ending reg: 2209  
2210  
%R0210

Number of Items: 10

Destination

Address: %R0201

Name:

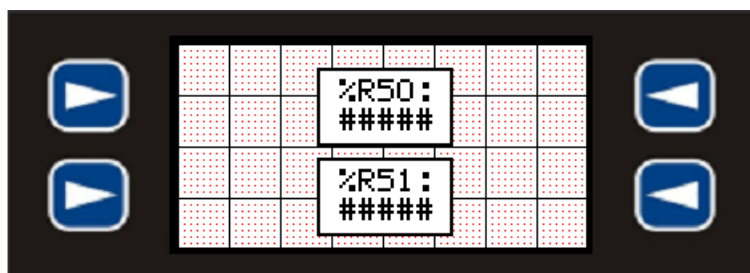
Type: INT

OK Cancel

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

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

3. Use %R50 as the source address and check the 'Indirect' box. Use %R51 as the destination register. Do NOT check the 'Indirect' box for the Destination.
4. Create a screen with data fields showing %R50 (editable) and %R51:



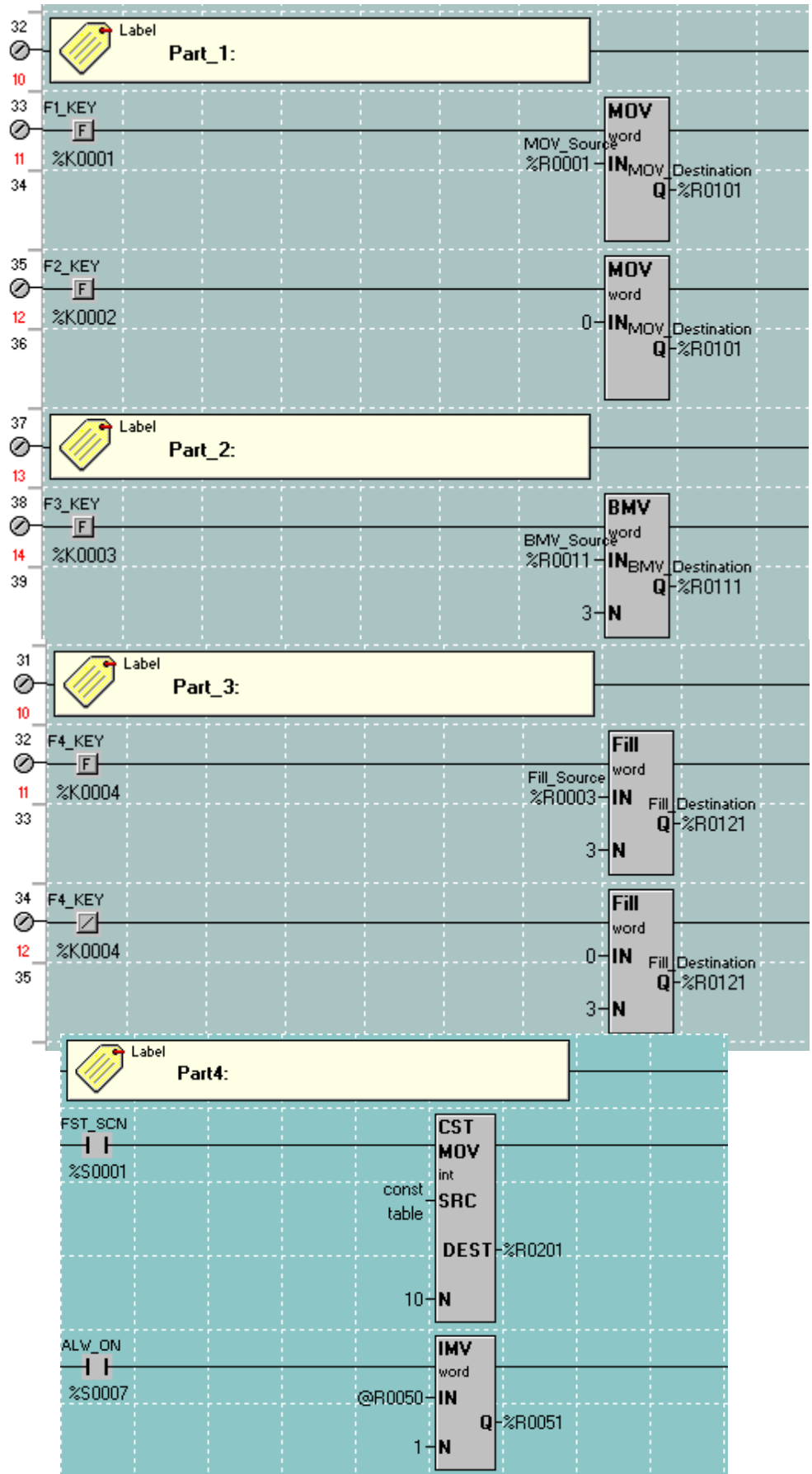
%R50			%R51	
#####			#####	

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.

%R50 is now a "pointer" that tells the indirect move function where to get its source data. Edit the value in %R50 to equal 201. This will point the indirect move function at %R201 for the source of the move. The value that was loaded into %R201 using the Constant Move function, 2201, should now be shown in %R51. Change %R50 to any value between 201 and 210. One of the values in %R201-%R210 are moved to %R51 based on the value in %R50.

## Lab 4: Move Operations

Move Lab Solution:



## **Lab 4: Move Operations**

**Notes:**

## **Lab 4: Move Operations**

**Notes:**

# **LAB 5**

## Basic CsCAN Networking

## **Lab 5: Basic CsCAN Networking**

## **Lab 5: Basic CsCAN Networking**

### **Objective:**

Review and understand global data transfer from OCS-to-OCS and from OCS-to-Network I/O over CsCAN.

### **Overview:**

Horner's CsCAN (Cscape Control Area Network) is a peer-to-peer CAN-based network. Though the physical wiring is nothing new to anyone who has ever used a CAN-based network before (DeviceNet, Honeywell's SDS, J1939, to name a few), Horner APG developed their own protocol in order to offer features not available with other such protocols.

As a peer-to-peer network, there is no master or slave. Each OCS broadcasts (or puts) data onto the network to share with any other OCS that wants it. An OCS that needs data from another OCS on the network specifies the other OCS's ID and what broadcasted data from that OCS is needed in order to retrieve (or get) it. It is also possible for an OCS to broadcast a heartbeat onto the network for other OCSs to monitor in order to determine the online state of this OCS.

A slight departure from the peer-to-peer design is when SmartStix network I/O is used. Despite their name, SmartStix have no intelligence and must be told what to do. Only one OCS should be set up to control a SmartStix, especially in the case of SmartStix with outputs. Bad things will happen if two OCSs try to control the outputs from one SmartStix. However, the inputs of a SmartStix are broadcast to the network for any controller to easily retrieve.

A benefit of Horner's CsCAN is the ability to connect a programming cable to one OCS and program any other OCS on the CsCAN network without having to visit each OCS to load programs. Also, if an OCS goes down, there is no worry that it may have been the "Master" and that none of the network can function. Other OCSs can be programmed to deal with it and even take over if necessary.

For Part 1 and Part 2 of this lab, two controllers will be networked together. Work together with another person, one taking Step 1 and the other taking Step 2, or take turns using the equipment. For Part 3, a SmartStix I/O block will be added to the network.

### **Part 1 - Analog Data Over CsCAN**

Step 1 (Lab partner start on Step 2)

#### **➤ Create a new program.**

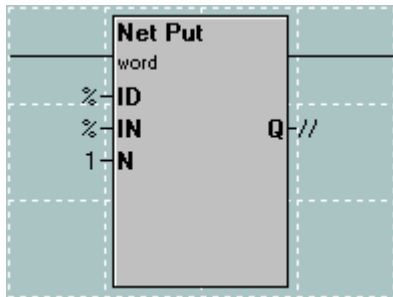
1. Title the program 'Lab 5 Node 1.csp'.
2. Set the target to node id 1. Verify through the system menu that the node address is set to 1 and that the baud rate is 125K.
3. Configure the controller. (Reference Lab 1 for correct procedure)
4. Write a ladder program to increment a counter every 1 second. Assign the counter to %R1. The counter should be configured to count to 200. Use the 16<sup>th</sup> bit of the second word of the counter, %R2.16, to reset the counter upon the

## **Lab 5: Basic CsCAN Networking**

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 pulses every second... Address a Normally Open contact to %S5 to fire the counter once every second.

5. Write a line of code to broadcast the accumulated value of the counter to 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 and then download it to the controller.

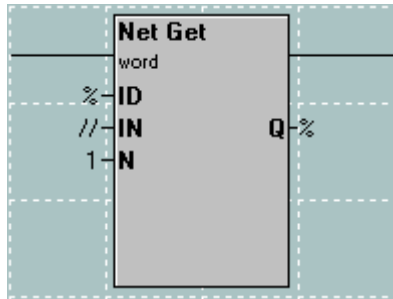
### Step 2

#### ➤ **Create another new program.**

1. Title the program 'Lab 5 Node 2.csp'.
2. Set the target to node id 2. Verify through the system menu of the 2<sup>nd</sup> controller that the node address is set to 2 and that the baud rate is 125K.
3. Configure the controller. (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.



## **Lab 5: Basic CsCAN Networking**



The 'Network Get Words' dialog box is shown. It has a title bar with a close button. The dialog is divided into several sections:

- Network Data:** Contains a dropdown for 'ID' set to '1', a text field for 'Name', and a '16-BIT' indicator.
- Format:** Contains a radio button for 'Global Data'.
- Data:** Contains radio buttons for 'Digital' and 'Analog', and a 'Word Offset' field set to '0'.
- Destination Data:** Contains a dropdown for 'Address' set to '%R0001', a text field for 'Name', and a '16-BIT' indicator.
- Num Words:** A field set to '32' with up and down arrows.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom right.

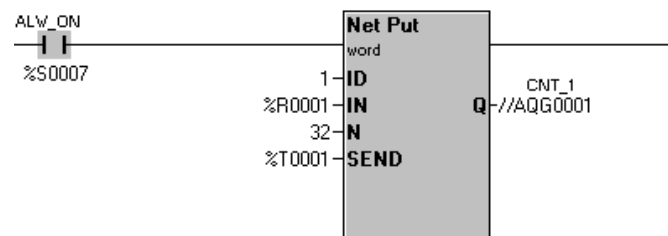
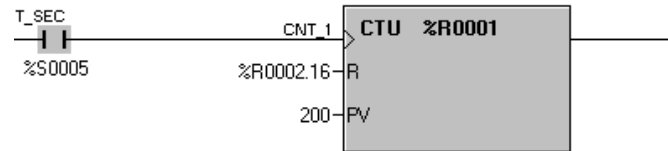
5. Configure Screen 1 to display the “Incoming Data”. This will be data coming from Node 1.
6. Save the program and then download it to the controller.

### Step 3

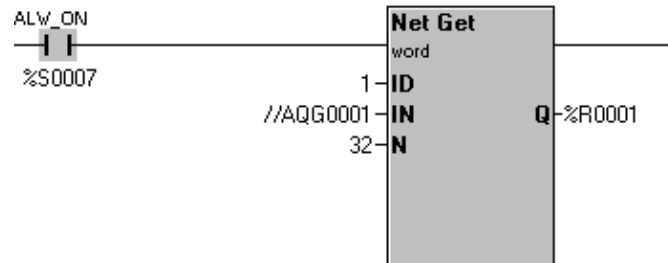
- **Verify the program’s functionality.** Both controllers should display the same value when the value on Node 1 is edited.

## **Lab 5: Basic CsCAN Networking**

### ➤ **Part 1 Solution.**



Node 1 Ladder Logic



Node 2 Ladder Logic

## **Lab 5: Basic CsCAN Networking**

### **Part 2 – Digital Data Over CsCAN**

#### **Step 1**

##### **➤ Modify the Program for Node 2**

1. Broadcast the function keys 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.

The screenshot shows the 'Network Put Words' dialog box. It has a 'Network Data' section with 'ID' set to 2 and 'Name' empty. The 'Format' section has 'Global Data' selected. The 'Data' section has 'Digital' selected and 'Word Offset' set to 0. The 'Source Data' section has 'Address' set to %K0001 and 'Name' empty. There is a checkbox 'Send on Change of State' which is checked. Below it, 'Send' is set to %T0001 and 'Name' is empty. A note says 'This register will force a transmission, ignoring change of state.' At the bottom, 'Num Words' is set to 1. There are 'OK' and 'Cancel' buttons.

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

#### **Step 2**

##### **➤ Modify the Program for Node 1**

1. Configure Node 1 to read the function keys from Node 2 and put them into registers starting at %M1. Write a line of code that uses a NET\_GET instruction configured for discrete from Node 2 with the destination of %M1.

The screenshot shows the 'Network Get Words' dialog box. It has a 'Network Data' section with 'ID' set to 2 and 'Name' empty. The 'Format' section has 'Global Data' selected. The 'Data' section has 'Digital' selected and 'Word Offset' set to 0. The 'Destination Data' section has 'Address' set to %M0001 and 'Name' empty. At the bottom, 'Num Words' is set to 1. There are 'OK' and 'Cancel' buttons.

2. Write additional rungs of logic that will turn on outputs (%Q) when the %M registers from above come on (M1 will turn on Q1, M2 will turn on Q2, etc.). This can either be done using contacts and coils or via a move command.

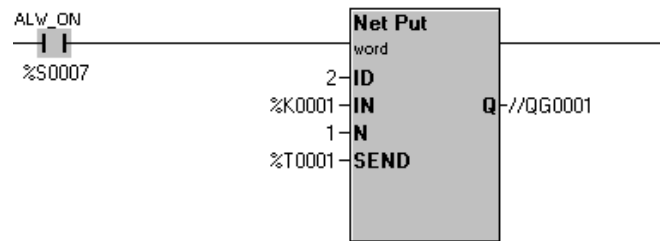
## **Lab 5: Basic CsCAN Networking**

Step 3

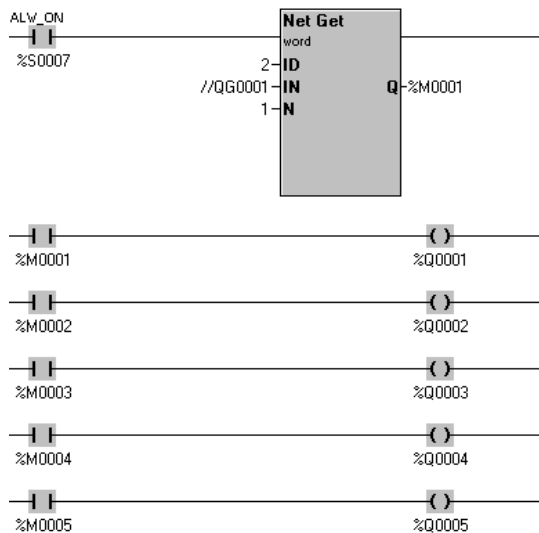
➤ **Verify the program's functionality.**

When F1 on Node 2 is pressed, the 1<sup>st</sup> output on Node 1 should turn on.

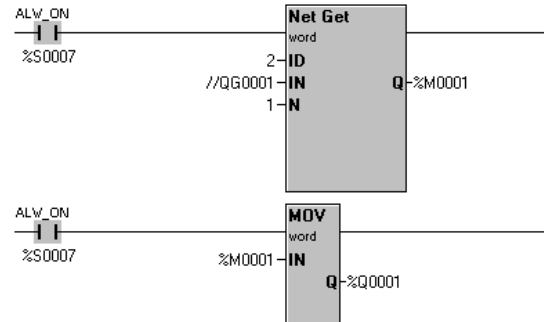
➤ **Part 2 Solution**



Node 2 Program Addition



Option 1



Option 2

Node 1 Program Addition

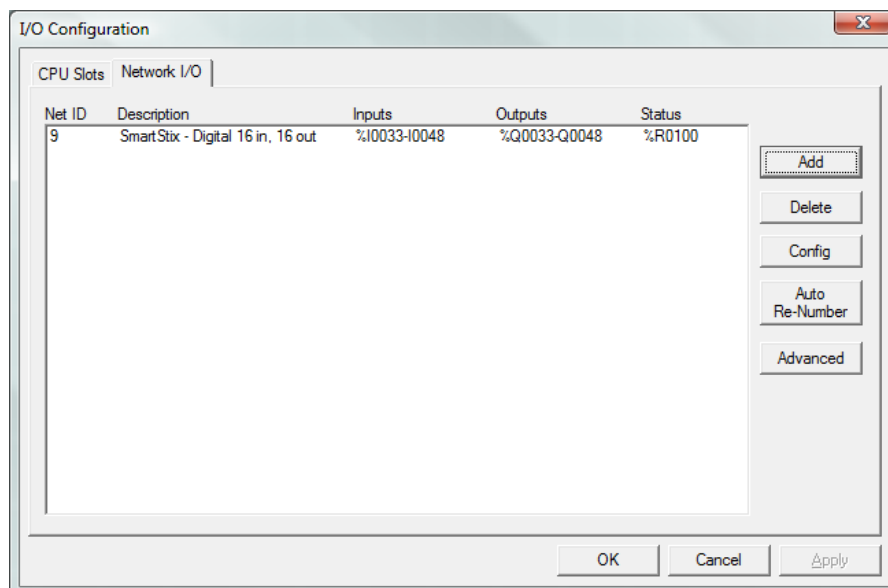
## **Lab 5: Basic CsCAN Networking**

### **Part 3 – SmartStix**

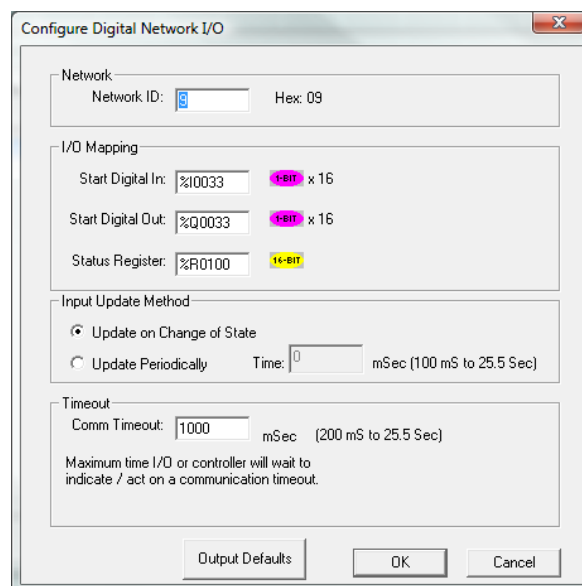
#### **Step 1**

#### ➤ **Modify the Program for Node 1**

1. Configure Node 1 for SmartStix I/O. This is done via the Network I/O tab located in the I/O configuration.



2. Press Add and select “SmartStix – Digital 16in, 16out”.
3. Configure the SmartStix as illustrated in the picture below, making sure no built-in I/O is overwritten:



**Note:** Make sure the Network ID matches the rotary switches on the SmartStix. Also, the rotary switches are in Hexidecimal.

4. Press OK, then OK again.

### **Lab 5: Basic CsCAN Networking**

5. Modify the Node 1 program to turn on outputs starting at %Q33 when the function keys of Node 2 are pressed. Depending on which option chosen in part 2, it will require either changing the address of the coils or changing the destination of the move command.
6. Save program and download changes to Node 1.

#### Step 2

##### ➤ **Verify functionality.**

Press the F1 key on Node 2 and the first output LED on the SmartStix should turn on, F2 should turn on the second, etc.

**CONGRATULATIONS ON COMPLETING THE NETWORKING LAB!**

#### **Notes:**

# **LAB 6**

## Graphics Editor

## **Lab 6: Graphics Editor**



## **Lab 6: Graphics Editor**

### **Objective:**

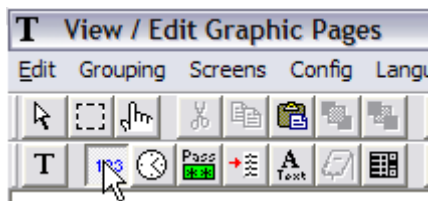
Understand more about how to create screens using the Graphics Editor.

### **Overview:**

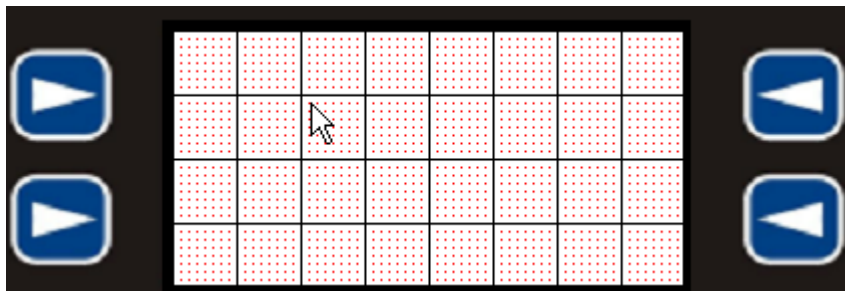
In previous labs, a few things have been learned about screen creation. In this lab, there is more detail on the different objects that can be placed on a screen to either show data in different formats or simply “pretty-up” the display, making it easier to read.

In order to keep from having problems, the recommended way of placing graphics objects on the screen is to click on the desired object on the toolbar and then release the mouse button, just like in the ladder logic part of Cscape. Objects are not dragged from the toolbar onto the screen. Once selected, bring the mouse down into the screen area. Click AND HOLD the left mouse button at the top left corner of where the object is wanted, then drag it out to size. Simply clicking and immediately releasing may make an “invisible” object or may bury the object halfway off the screen where it needs to be deleted and replaced.

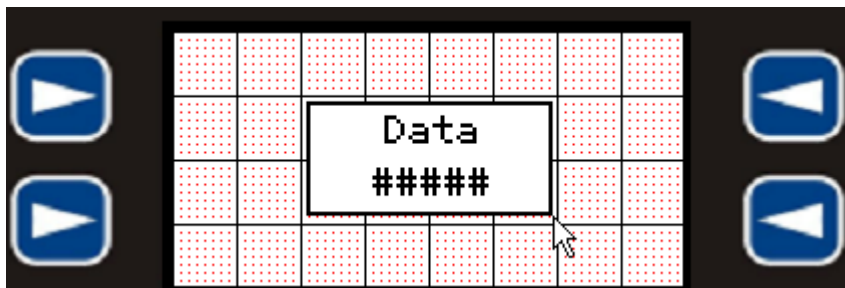
### **Click and release to select an object:**



### **Click and hold at top left of desired position:**



Drag out the object on the screen. Release the mouse button once the object is sized to liking... it can always be moved and resized later:



## **Lab 6: Graphics Editor**

### **Graphics Objects:**

#### *Static Text*



Static Text has been used in previous labs. Static Text is used as a label and is not attached to any register data.

#### *Numeric Data*



Numeric Data has been used in previous labs. Numeric Data is a way to display data in Integer, Double Integer, Floating Point, Hexidecimal, and several other formats. This field is linked to register memory and can be selected as Editable, which means the user will be able to edit the data in this field from the controller, or can be used only as a display that the user cannot change.

#### *Time Data*



Time Data is an automatically formatted data field that shows either the Time or Date. This field is linked to 3 consecutive registers that contain either the time or date in the same order as in the RTC system registers %SR44 - %SR50.

For time: Seconds in the first register, Minutes in the second, Hours in the third.

For date: Day in the first register, Month in the second, Year in the third.

#### *Timer Data*



Timer Data is an automatically formatted data field that shows the elapsed time of a Timer function. Assigned to the timer's address, the resolution of the timer is selected and the format display desired, then data will be displayed formatted with colons and periods as in a digital clock display.

#### *Password Data*



Password Data is a data field linked to register memory that must be 32-bits in length... 2 %R registers, for instance. On the screen, asterisks will display instead of the actual data.

## **Lab 6: Graphics Editor**

### *Text Table*



Text Tables have been used in previous labs. Text Table objects are lookup tables linked to register memory and a single text table. Instead of displaying the actual numeric value in the register, those values will cross-reference to the entries in the text table. Depending on the value in the register, the corresponding text will display. Many different Text Table objects on many different screens linked to many different registers can all access the same text table if it contains the text needed for them all. There are 200 text tables to use, each with twenty 20-character entries, each entry linked to a value.

### *Menu*



The Menu Object has been used in a previous lab. The Menu object makes the creation of a menu system easy. Each entry in the menu can be specified as a screen jump, a link to a sub-menu or a register value to edit.

### *Indicator*



The indicator is used as an OFF/ON indicator for discrete data. It is linked to a 1-bit register. There are several display options to choose from for the desired look.

### *Switch / Button*



The Switch, a.k.a. Button, is used to indicate and control discrete data. It is linked to a 1-bit register in and can be specified to operate as a momentary or toggle switch, as well as “always turn ON” and “always turn OFF”. On controllers with softkeys, the switch will be connected to the nearest softkey by default. On touch-screen controllers, the switch is activated by pressing it directly on the screen. There are several display options to choose from for the desired look.

### *Selector*



The Selector is a method of choosing one of up to 4 items by pressing the selection desired. It is linked to a word-length register that will contain a value depending on the selected item... ‘0’ for Item 1, ‘1’ for item 2, ‘2’ for item 3 and ‘3’ for item 4. The number of items selectable and their names are configurable.

## **Lab 6: Graphics Editor**

### *Screen Jump*



Screen Jumps have been used in several of the previous labs. This is one of the methods of navigating through the screens on the controller.

### *Bar Graph / Meter / 360° Gauge*



These are different ways of displaying the data in a word-length register. The Bar Graph can be configured in either a horizontal or vertical fashion based on its dimensions.

The Meter displays a needle and a sweep from the left to the right.

The 360° Gauge is simply a floating needle that can sweep from any angle to any angle in a 360-degree range... its scale marks or gauge face design must be done separately as a bitmap background or using other methods.

The Bar Graph and Meter objects can be configured to show any number of equally spaced hash marks along their scales. All three objects will automatically scale their sweep to the Min and Max values they are configured for.

### *Static Bitmap / Animation*



These are methods of displaying custom graphics or symbols from the optional Symbols Library software that can be purchased. The Static Bitmap is just one picture that can be loaded from a bitmap file on the hard drive or be specified from the Symbols Library.

The Animation is a group of bitmaps... up to 50, one for each frame of the animation. The Animation is linked to a register and, depending on the value from 0 to 49 in the register, shows that frame number. It is up to the ladder logic to then provide the means of changing that number as desired.

### *Data Trend / X-Y Graph*



These objects are methods of showing a graph of data on the screen.

The Trend object can have up to four pens configured, each linked to a different word-length register. It displays a graph of data as samples over a period of time. The graph fills over a period of time depending on the frequency of the samples. The sample period is configurable, as are many other attributes of this object. The Trend can be configured to either stop filling the graph when it is full or continue to scroll it across the screen as samples are taken.

The X-Y Graph can also have up to four pens configured, each linked to a set of word-length registers. It displays all of its data at once, filling the entire graph, immediately

## **Lab 6: Graphics Editor**

when triggered, using data in consecutive registers following the first one specified for a given pen and as many as are specified in the configuration.

### *Alarm Object*



The Alarm object (gone over in detail later in this lab) is the user interface portion of the Graphic Alarms. It can be displayed as a button or as a list of alarms and can be configured to show either a current summary or a history of alarms.

### *Removable Media Manager*



This object allows access to the directory of the Removable Media card. A directory of files will be shown. Depending on the configuration of this object, the user may or may not be able to delete files, format the card, change the directory, etc. This is a way to give a user limited access to the Removable Media card.

### *Rectangle / Ellipse / Rounded Rectangle / Line*



These simple drawing objects are used solely for the purpose of beautifying the screen. They are not linked to any register. The border and line widths can be altered. On the Rectangle, Ellipse and Rounded Rectangle, the fill color can be specified as transparent or as any of the available colors.

## **Lab 6: Graphics Editor**

### **Graphics Editor Lab**

Start a new program and configure the controller and I/O as has been learned in the past several labs. Save the program.

1. In the Graphics Editor, click on the **Config** menu and select **Alarm** to configure the alarms in general.

Configure the Alarms as follows:

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

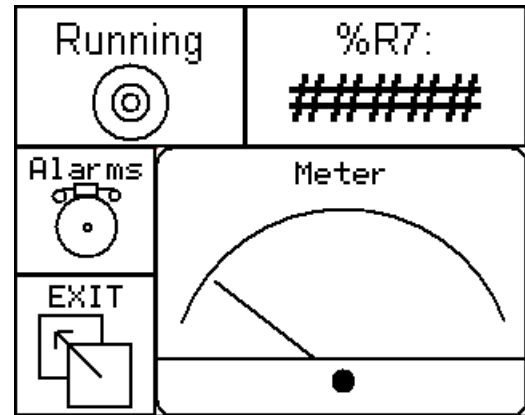
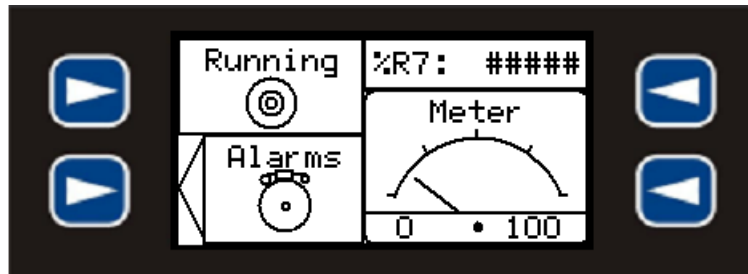
Name the first 4 alarms as follows by double-clicking them in the list:

- ❑ 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

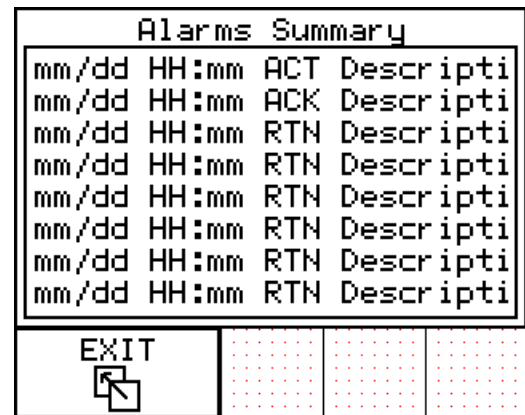
2. Exit the Alarm configuration and the Graphics Editor. In Ladder Logic, using a start/stop circuit like that used in Lab 1, add logic for a Start/Stop holding circuit (F1 Starts, F2 Stops).
3. Add logic to trigger Alarm 1 (%M1601) whenever the value in %R7 is less than 15 AND the circuit is started.
4. Add logic to trigger Alarm 2 (%M1602) whenever %R7 is greater than 90.  
HINT: Compare Functions will need to be used to test the value in %R7.
5. Add logic to trigger Alarm 3 (%M1603) whenever %I1 is off. Add a normally-closed %I1 to the run circuit so that the circuit will not run if %I1 is off. This simulates a Motor Overload Trip.
6. Add logic to trigger Alarm 4 (%M1604) whenever %I2 is off. Add a normally-closed %I2 to the run circuit so that the circuit will not run if %I2 is off. This simulates an E-stop trip.
7. Back in the Graphics Editor on screen 1, create a “Main Menu” with either the menu function or screen jumps. Make menu entries for a “System Monitor” on screen 2, “Current Alarms” on screen 3, and “Alarm History” on screen 4. Make sure to allow some way for the user to get back to the Main Menu.
8. Go to screen 2 and create a system monitor.
  - a. Add an editable data field linked to %R7. Restrict it to a range of values between 0 and 100.
  - b. Add a meter to reflect %R7.
  - c. Add an indicator to tell when the circuit is Started or Stopped.
  - d. Add an Alarm Indicator Button to the Screen. The Alarm Indicator should display an Alarm Summary when pressed, for all Alarm groups.

An example of what this screen might look like:

## Lab 6: Graphics Editor



9. Create screen 3 and add an Alarm Summary Object.



10. Create screen 4 and add an Alarm History Object. Screen 4 will look very much like Screen 3 as they are both lists of alarms. The Summary shows alarms that are happening currently while the History shows alarms happening now as well as in the past.

Download and run the program. 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.

Use the "Alarm Indicator" button on the "System Monitor" screen as an alternate means of viewing the Alarm Summary. Note the conditions that will cause the Alarm Indicator buttons to change color.

### **Part 2 – Extended Graphic Alarms Lab**

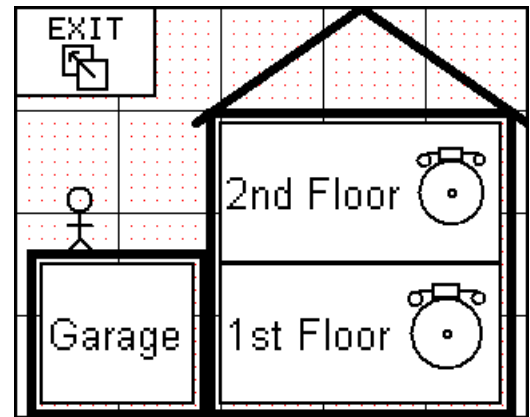
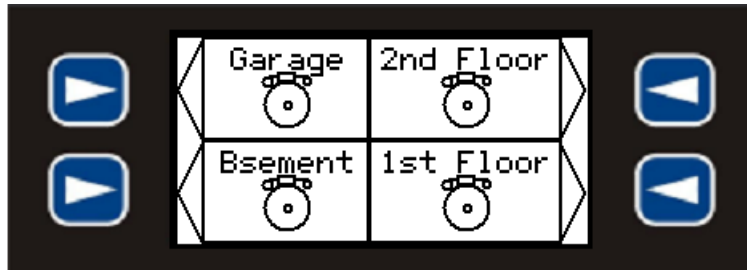
Add a building alarm simulator to the program from Part 1.

1. Add an entry to the Main Menu in order to get to screen 5.
2. On screen 5, represent different areas of a building. Place an alarm indicator button for the First Floor (Group 2), Second Floor (Group 3) and Garage (Group 4). Add more if desired.
3. Add new alarms to the Alarm Configuration, a couple each for groups 2, 3, and 4. For instance, group 2 can contain "First Floor Window" and "First Floor Door", group 3 can contain "Second Floor Window" and "Second Floor Stairs", etc.

### **Lab 6: Graphics Editor**

4. Exit the graphics configuration and add ladder logic to trigger the new alarms based on any unused inputs on the input simulator (%I3 - %I8).

Download the program. Note how the alarms can be separated into manageable groups that can be monitored in separate alarm summary and history logs.





## **Lab 6: Graphics Editor**

**Notes:**

## **Lab 6: Graphics Editor**

**Notes:**

# **LAB 7**

## Removable Media Functions

## **Lab 7: Removable Media Functions**

## **Lab 7: Removable Media Functions**

### **Objective:**

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

### **Overview:**

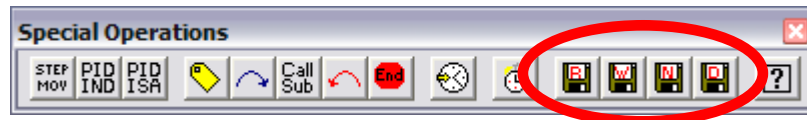
The following controllers support Removable Media: OCS451/551/651, NX and QX controllers and XLe/XLt/XL6 controllers. There are several different ways to use RM:

- ❑ Read/Write/Rename/Delete files in CSV format in Ladder Logic
- ❑ Automatically log data using the Data Logging function
- ❑ Capture screen images as a picture file
- ❑ Load a program into a controller without connecting it to a PC
- ❑ Host a web page that shows controller data over the internet (XLe/XLt does not support this feature)
- ❑ Use FTP to read or write files to the RM card over the internet (XLe/XLt does not support this feature, though there is a utility program that can provide this functionality)

### **Ladder Logic Functions**

Information can be stored to and read from a RM card using functions in Cscape Ladder Logic. This method can be used to datalog information or to read recipes into the controller or to do any number of other tasks.

When writing information to the RM card, 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 on a PC. Conversely, a CSV file could be created from a PC, stored to the RM card in CSV format and then read into the OCS. The Read (R), Write (W), Rename (N) and Delete (D) RM ladder functions are found in the Special Operations toolbar.

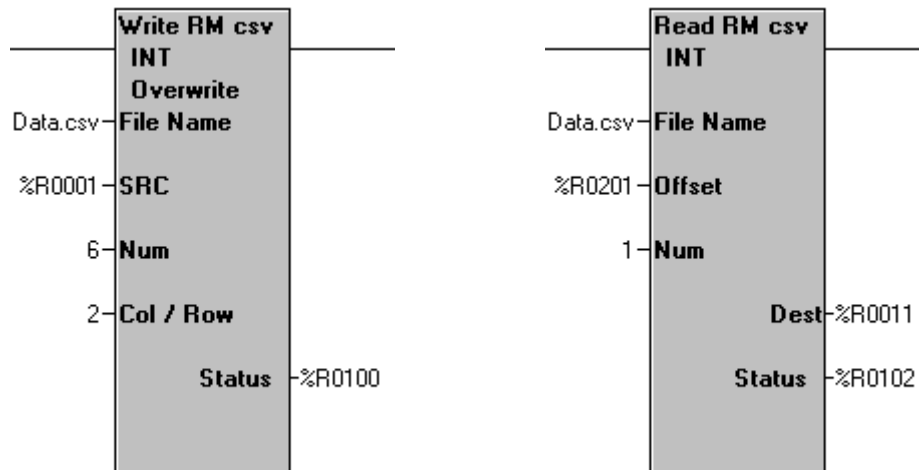



### **Procedure:**

1. Open Cscape and create a ladder program that will **write** 6 registers of information, starting at %R1, when triggered by the F1 key. Use the Write RM function configured for Overwrite to perform this action and call the file Data.csv. Configure the Columns Per Row to be 2 and check the "End of Row Now" box.
2. Create a line of code that will **read** one value from the Data.csv file and store the information in %R11 when the F2 key is pressed. The offset should be configured for %R201. If a constant is used as the offset, the program will always be reading the value from the same place in the file.

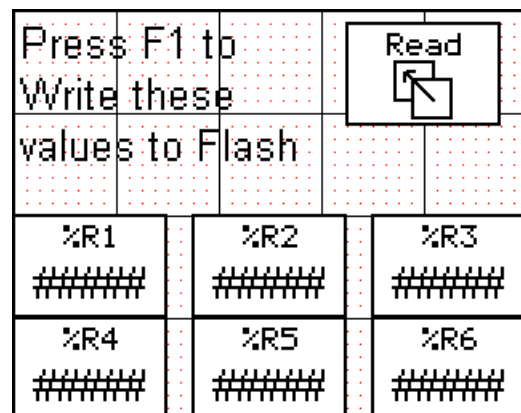
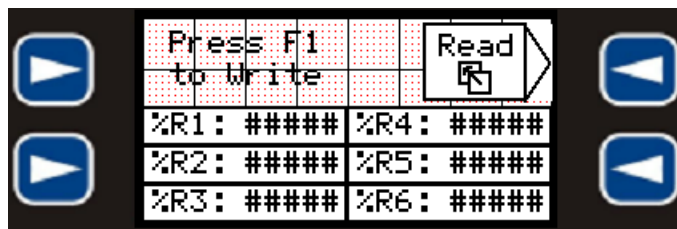
## **Lab 7: Removable Media Functions**

3. Create a line of code that uses the Delete RM block and use the filename of Data.csv. Trigger this Delete function with the F3 key.

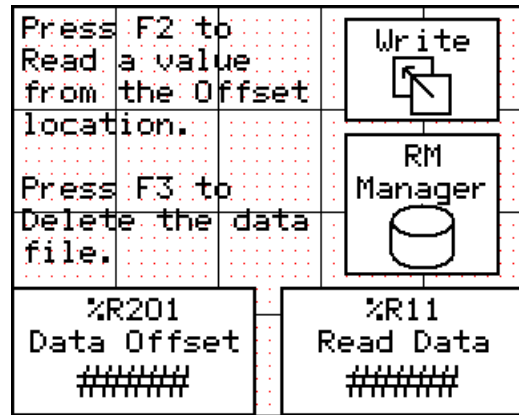
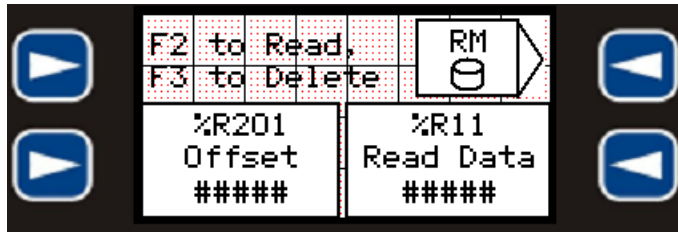


4. Configure Screen 1 to have the following:
  - ❑ Register fields for %R1 - %R6 that are all editable.
  - ❑ Instructions (static text) to "Press F1 to Write".
  - ❑ A screen jump to Screen 2... specify "Allow ESC to Return".
5. Configure Screen 2 to have the following:
  - ❑ Register field for %R201, the offset, that is editable. The valid offset values for this lab as written would be anywhere between 0 and 5. However, make sure this data field is set to a 32-bit Decimal format!!!
  - ❑ Register field for %R11 that is read only
  - ❑ Instructions to "Press F2 to Read, F3 to delete"
  - ❑ Place a RM manager on the screen 

The screens may look something like this:



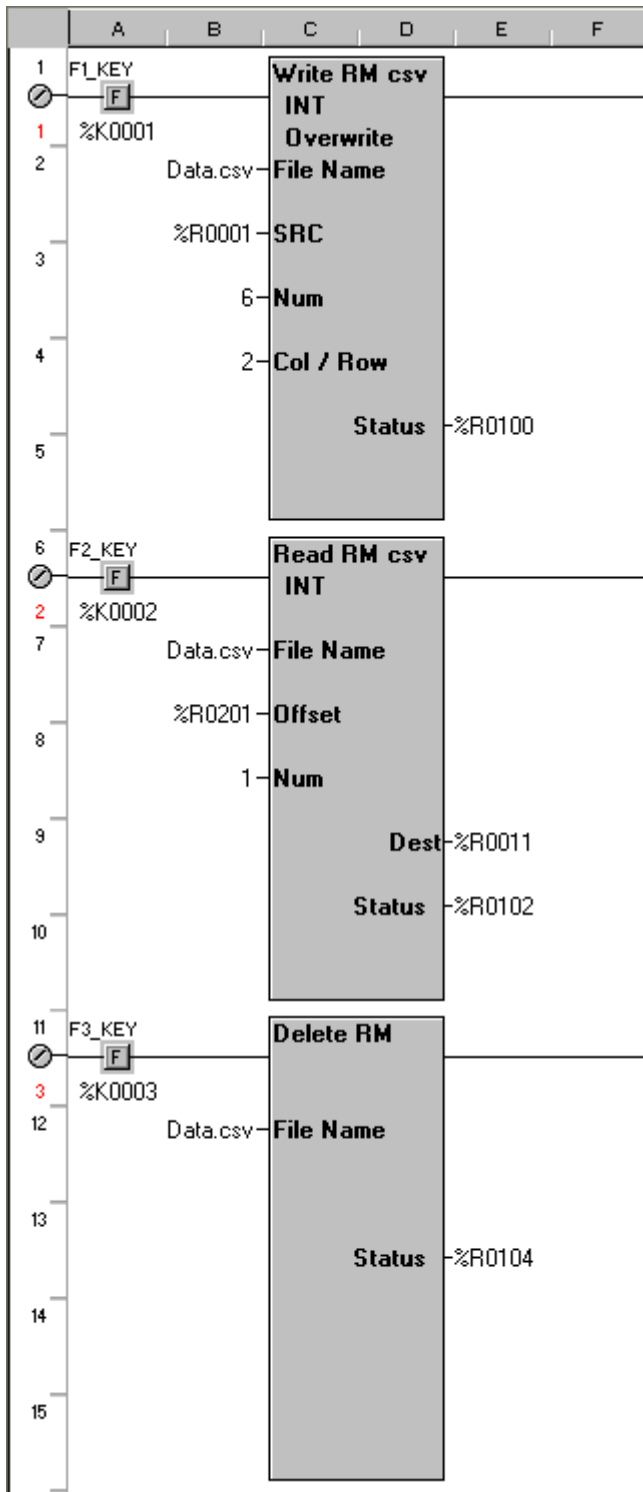
## Lab 7: Removable Media Functions



6. Download the program to the controller.
7. Change the values in %R1-%R6 on Screen 1 to values that will be written to the RM card.
8. Press the F1 key. The information in %R1-%R6 is now stored to the RM card.
9. Go to Screen 2 and make the %R201 offset '0'. Press the F2 key. The first piece of data (what was written from %R1) should now appear in the %R11 data field on the screen.
10. Change %R201 to other offsets from 0-5 and press the F2 key. Depending on the offset, you will read one of the 6 values that was written to the RM card on Screen 1. The offset determines which of those 6 values is read.
11. Use the RM manager to view the contents of the RM card.
12. Press F3 to delete the file. Confirm it is no longer there by viewing the contents of the card using the RM manager.

## Lab 7: Removable Media Functions

*Solution:*





## **Lab 7: Removable Media Functions**

### **Automatic Data Logging**

A newer function of controllers that support Removable Media is the ability to automatically record data to a file at timed intervals. Each entry is time and date stamped and can record up to ten different pieces of data of differing formats if needed. Up to ten different logs can be configured with up to ten pieces of data per log. Additionally, columns are labeled with the I/O name of the register where the data is coming from.

Automatic Data Logging is configured via the 'Program/Datalog Config' menu in Cscape 8.10 and higher.

### **Removable Media File Naming**

The RM 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 7: Removable Media Functions**

### **Removable Media 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

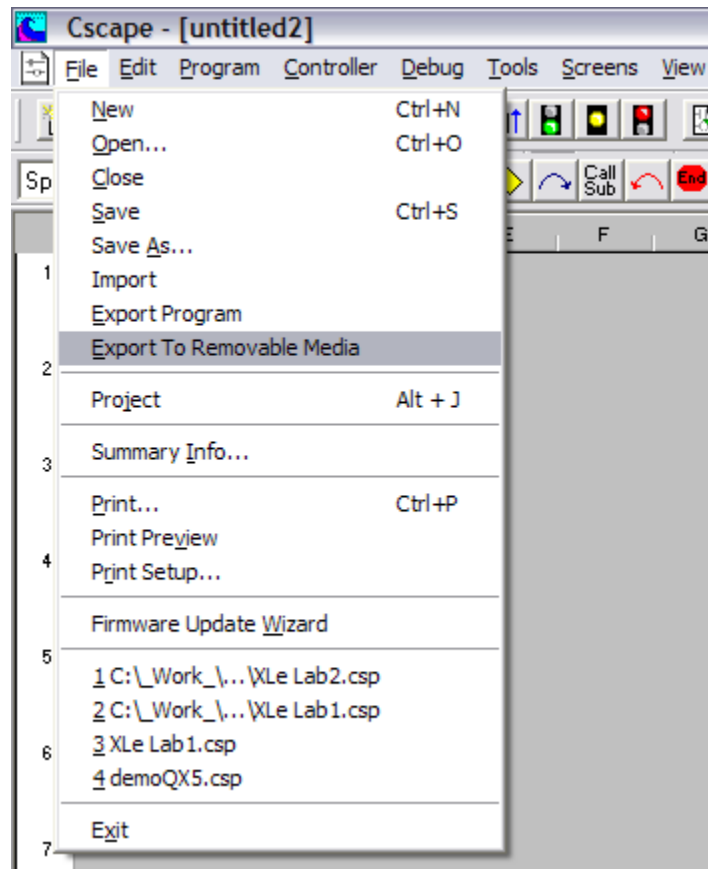
**Note:** A filename extension **MUST** be specified in all cases if there is to be one. It is never automatically added.

## **Lab 7: Removable Media Functions**

### **Removable Media Program Downloads**

Another feature of the Removable Media functionality is the ability to load a controller that has RM capabilities with a program from a RM card instead of through Cscope.

The programmer saves the Cscope 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 Removable Media writer connected to the computer or to anywhere else on the computer to be transferred to Removable Media later. The user will then insert the RM card into the OCS and, through the OCS System Menu, select Removable Media to find the correct file to load. If there is access to a card reader/writer, try it!



### **Removable Media Screen Captures**

Controllers with Removable Media capabilities have the ability to capture a displayed screen to Removable Media as a JPEG or Bitmap file. These images can then be recalled on the unit through the CF manager or viewed on a computer with a graphics viewer.

## **Lab 7: Removable Media Functions**

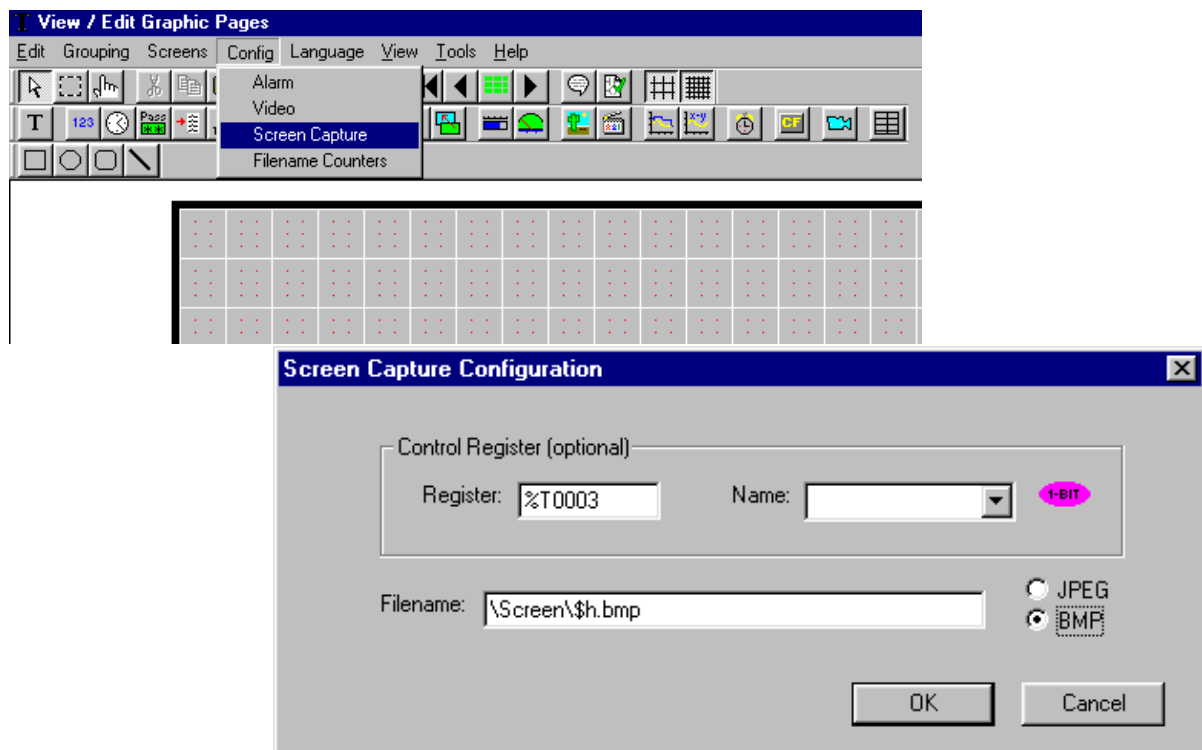
Configuring the Screen Capture function is done through the Graphics Editor by clicking the **Config** menu and selecting **Screen Capture**. A 1-bit register must be configured as a trigger and a filename for the captured graphic file must be specified. The filename date functions and filename counters 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.

**Note:** Since the OCS provides feedback by turning off the trigger bit:

- Do NOT use a 1-shot to trigger a screen capture... setting or toggling on a %T is recommended. When it turns off, the capture is done writing to RM.
- Be careful of using F-keys... Again, the OCS will try to reset it when the capture is done. Using a %T as noted above is recommended instead.

### **Procedure:**

1. Make F10 perform a screen capture. Set %T3 in ladder logic when F10 is pressed.
2. In the Graphics Editor, configure the Screen Capture to trigger off of %T3 and specify a filename.



3. Download to the OCS
4. Press the F10 key.
5. Go to the screen with the RM Manager on it from before, open the RM manager and find the captured graphic.
6. Use the arrow keys to scroll to the file and press Enter to view it.

**CONGRATULATIONS ON FINISHING THE REMOVEABLE MEDIA LAB!**

## **Lab 7: 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	9999
%K	Keypad bits, reflect Function Key status	BOOL	NO	4-12
%D	Display bits, control screens or indicate screen on/off	BOOL	NO	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	1023
2	Alarm Screen Number	0	1023
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 System screens vary from model to model

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



# HORNER

**APG**

1.317.916.4274

1.877.665.5666

[www.hornerocs.com](http://www.hornerocs.com)

## Phil Horner

President

317-492-9080

[phil.horner@heapg.com](mailto:phil.horner@heapg.com)

### Technical Support

Ext. 3

[techspt@heapg.com](mailto:techspt@heapg.com)

### Customer Service

Ext. 1

[APGCustomerService@heapg.com](mailto:APGCustomerService@heapg.com)

## Sales and Marketing

Ext. 2

[apgsales@heapg.com](mailto:apgsales@heapg.com)

### Chuck Ridgeway

*Product Manager*

317-492-9081

Cell: 317-554-9047

[chuck.ridgeway@heapg.com](mailto:chuck.ridgeway@heapg.com)

### Bill Giebel

*Business Development Manager*

317-492-9079

Cell: 317-407-7937

[bill.giebel@heapg.com](mailto:bill.giebel@heapg.com)

### Nate Beachey

*System Design Engineer*

317-492-9118

[nathan.beachey@heapg.com](mailto:nathan.beachey@heapg.com)

### Eric Broyer

*Technical Support Manager*

317-492-9102

[eric.broyer@heapg.com](mailto:eric.broyer@heapg.com)

# HORNER

APG