# FTP AND FILE MANAGEMENT:

# SNAP ULTIMATE I/O TRAINING CENTER SUPPLEMENT

Form 1450-041207-December, 2004



43044 Business Park Drive • Temecula • CA 92590-3614 Phone: 800-321-OPTO(6786) or 951-695-3000 Fax: 800-832-OPTO (6786) or 951-695-2712 www.opto22.com

> **Product Support Services** 800-TEK-OPTO (835-6786) or 951-695-3080

> > Fax: 951-695-3017 Email: support@opto22.com

### OptoTutorial: FTP and File Management Form 1450-041207—December, 2004

Copyright © 2003–2004 Opto 22. All rights reserved. Printed in the United States of America.

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

Opto 22 warrants all of its products to be free from defects in material or workmanship for 30 months from the manufacturing date code. This warranty is limited to the original cost of the unit only and does not cover installation, labor, or any other contingent costs. Opto 22 I/O modules and solid-state relays with date codes of 1/96 or later are guaranteed for life. This lifetime warranty excludes reed relay, SNAP serial communication modules, SNAP PID modules, and modules that contain mechanical contacts or switches. Opto 22 does not warrant any product, components, or parts not manufactured by Opto 22; for these items, the warranty from the original manufacturer applies. These products include, but are not limited to, the OptoTerminal-G70, OptoTerminal-G75, and Sony Ericsson GT-48; see the product data sheet for specific warranty information. Refer to Opto 22 form number 1042 for complete warranty information.

All other brand or product names are trademarks or registered trademarks of their respective companies or organizations.

Opto 22 FactoryFloor, Cyrano, Optomux, and Pamux are registered trademarks of Opto 22. Generation 4, ioControl, ioDisplay, ioManager, ioProject, ioUtilities, mistic, Nvio, Nvio.net Web Portal, OptoConnect, OptoControl, OptoDisplay, OptoENETSniff, OptoOPCServer, OptoScript, OptoServer, OptoTerminal, OptoUtilities, SNAP Ethernet I/O, SNAP I/O, SNAP OEM I/O, SNAP Simple I/O, SNAP Ultimate I/O, and SNAP Wireless LAN I/O are trademarks of Opto 22.

ActiveX, JScript, Microsoft, MS-DOS, VBScript, Visual Basic, Visual C++, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries. Linux is a registered trademark of Linus Torvalds. Unicenter is a registered trademark of Computer Associates International, Inc. ARCNET is a registered trademark of Datapoint Corporation. Modbus is a registered trademark of Schneider Electric. Wiegand is a registered trademark of Sensor Engineering Corporation. Nokia, Nokia M2M Platform, Nokia M2M Gateway Software, and Nokia 31 GSM Connectivity Terminal are trademarks or registered trademarks of Nokia Corporation. Sony is a trademark of Sony Corporation. Ericsson is a trademark of Telefonaktiebolaget LM Ericsson.

# Introduction

About This Tutorial	1
Installing the Sample Files	
Feature Overview	2

# Lesson 1: File Management and Logging Data

Overview	5
Concepts	5
File Communication Handle	
LogData Chart	
Activity 1: Logging and Retrieving Data	9
Preparation	
Use a Communication Handle	
Log Data	
View the Data Log	

# Lesson 2: Saving Log Files to Flash

Overview	.21
Concepts	.21
Flash, RAM, and Persistent Memory	. 21
Dynamic Change of a Communication Handle	. 23
LogData Chart	. 24
Activity 2: Dynamically Controlling a Communication Handle and Stor	ing
Data to Flash	.25
Automatically Incrementing the File Name	. 25
Saving a Log File to Flash	. 28
Test Your Changes	. 30

# Lesson 3: FTP Server and Reading Uploaded Data

Overview	35
Concepts	35
How the Sample Chart Works	
Uploading a File to the Brain	
Communication Handle Read Commands	36
Parsing a Text File	
Activity 3: Uploading and Parsing Files	
Reading a File through a Communication Handle	
Parse a Data File	47
Create and Upload a File	48
Test Your Strategy	50



# INTRODUCTION

# **ABOUT THIS TUTORIAL**

This tutorial and its sample files are a supplement to the SNAP Ultimate I/O<sup>™</sup> Learning Center (see Opto 22 form #1408), although the explanations apply to any SNAP Ultimate I/O system that has firmware version R5.0a (or higher) and ioControl<sup>™</sup> version R5.0a (or higher).

The sample ioControl strategy (ULCs1450.idb) is ready for use with a SNAP Ultimate I/O Learning Center. If you are using this tutorial with a different configuration, please note the following:

		Used in	If your SNAP Ultimate I/O isn't connected
Point	Description	Lesson(s)	to the Learning Center
00	Digital Input, Emergency	2, 3	In lesson 2, this point is used with point 01 to implement a write to Flash command. Invoking the Write to Flash feature from this momentary switch prevents accidental, excessive use of the write to flash. For this reason, substituting this switch with a different type of switch is not recommended. In lesson 3, this point initiates a demonstration. Substitute with any integer 32 variable. Set variable to true from Debug mode to simulate switched input.
01	Digital Input, POS	2	This point is used with point 00 to initiate a writing of files to Flash memory. Invoking the Write to Flash feature from this momentary switch prevents accidental, excessive use of the write to flash. For this reason, substituting this switch with a different type of switch is not recommended.
03	Digital Input, Photo Sensor	1, 2	The value of this point is read and written to a log file. Substitute with any digital input, digital output, or integer 32 variable.
04	Digital Output, Outside Light	3	This point is controlled by a recipe file uploaded to the brain. Substitute with any digital output or integer 32 variable.
05	Digital Output, Inside Light	3	Same as point 04.
06	Digital Output, Freezer Door Status	3	Same as point 04.
12	Analog Input, Store Temperature	1, 2	This point is read and written to a log file. Substitute with any analog point or float variable.

# Installing the Sample Files

The sample strategy you will use with this tutorial is provided in before and after form:

- ULCs1450\_before.zip: This is an ioControl strategy that you will complete during this tutorial.
- ULCs1450\_after.zip: This is an example of the completed ioControl strategy from this tutorial.

To prepare for the tutorial, unzip ULCs1450\_before.zip in a directory of your choosing. ULCs1450\_after.zip is provided for reference only.

# Feature Overview

SNAP Ultimate I/O firmware provides FTP support, and ioControl provides file management. These features can be used in a variety of ways:

- A strategy running on a SNAP Ultimate I/O brain can store files in the brain's RAM or Flash memory.
- FTP clients can read files from and write files to the FTP server on the brain.
- The ioControl strategy can read files loaded to the brain from other sources.
- The ioControl strategy can retrieve files from other FTP servers.
- The ioControl strategy can send data in file or string format to an FTP server (e.g., to another SNAP Ultimate I/O brain).

The application profile below shows the various paths of information possible by using the FTP and file management features.



Phone: 951-695-3000 • 800-321-OPTO • Pre-Sales Engineering: 800-452-OPTO • www.opto22.com Form 1450-041207

The above concepts are discussed in the following chapters:

- Lesson 1: File Management and Logging Data Activity 1: Logging and Retrieving Data
- Lesson 2: Saving Log Files to Flash Activity 2: Dynamically Controlling a Communication Handle and Storing Data to Flash
- Lesson 3: FTP Server and Reading Uploaded Data Activity 3: Uploading and Parsing Files



# LESSON 1: FILE MANAGEMENT AND LOGGING DATA

# **OVERVIEW**

The ioControl strategy built in this lesson will demonstrate how to save data from ioControl in a ASCII text file on the brain, where the data can be accessed using FTP client applications.

In addition to using FTP and file management, the strategy will show how to create a permanent file in case of power loss and how to increment the file name to reflect a loss of power.



Any Office or Enterprise Application

Any Equipment, Process, or Asset

# CONCEPTS

# File Communication Handle

Any communication from within ioControl is established through variables called Communication Handles. In ioControl, reading, creating, and appending to files are forms of communication defined within Communication Handles. It is important to know that Communication Handles are used to communicate with other Ethernet and serial devices, but this lesson focuses on using Communication Handles for file management.

The value of a Communication Handle is a text string that provides the communication parameters between ioControl and the file on the brain. For example, a file communication handle may contain the following information:

file: a, logdata. txt

which tells ioControl to use the brain's file management to append information to a text file called logdata.txt. A file communication handle has the following parameters:



A file Communication Handle will begin with file: r, file: w, or file: a and be followed by a filename. Each of these support commands:

# ioControl Commands used with File Communication Handles

Commands to Communication Handle will vary according to the type of action defined in the handle:

Both the action and file name are defined in a Communication Handle. To change either, reassign the value or use a separate Communication Handle.



You can set the initial value of a Communication Handle using ioControl's Configuration mode, or you can use ioControl and OptoScript<sup>™</sup> commands to set the value while the strategy is running. You will do both in this lesson, initially to create a file Communication Handle, and then to automatically increment the log file name.

Using a Communication Handle requires the following:

- Defining the Communication Handle
- Opening the Communication Handle and testing for errors
- Executing a command using the Communication Handle:

File read commands:

- Receive Numeric Table
- Receive Pointer Table
- Receive String
- Receive String Table,
- Transfer N Characters
- Send Communication Handle Command (delete, getpos, Setpos:<position>, Find:<mystring>)

File write or file append commands:

- Send Communication Handle Command (Find:<mystring>)
- Transmit Character
- Transmit New Line
- Transmit Numeric Table
- Transmit Pointer Table, Transmit String
- Transmit String Table
- Closing the Communication Handle

# LogData Chart

The LogData chart contains some basic functions as well as some empty blocks that serve as placeholders for functionality you will add in this and in following lessons. The diagram below describes the key functions this chart will perform when you complete lesson 1.

E	SNAP Ultimate I/O Strategy: ULCs1450
ioC	ontrol Chart: LogData
1	Initialize Store Carriage Return and Line Feed into String Variable
2-	Convert data, create string DateToString DateToString
	sDate, sTime, sTemperature, sState, sCRLF
	String Variable: sData File Storage
3	Open Communication nResult Comm. Action: File:A Handle File Name: DataLog.txt
4	Did File Open? Y Transmit N Pause
	In this block you will create a string variable used to terminate the string sent to the data log. Your instructions will put ASCII codes 13 and 10, which correspond to a carriage return and a line feed, into the variable sCRLF.
2	You will use OptoScript to convert numerical, date, and time variables to strings and join the strings into one variable to send to the Communication Handle.
	This block opens the Communication Handle. This action will return a value used to determine if the attempt to establish communication was successful. The result is stored in the integer variable nResult.
4	You will use the condition block to evaluate nResult. If it equals 0, then communica- tion is established and the chart can begin writing to the log file.
e	The Transmit String command sends the contents of sData to the communication handle. The communication handle determines the action (appending) and the file name (DataLog.txt), which were defined as the initial value of the Communication Handle. The Append action will append the string or create a new file if the file doesn't exist.

# ACTIVITY 1: LOGGING AND RETRIEVING DATA

# Preparation

The sample ioControl strategy is ready to use with SNAP Ultimate I/O Learning Centers. To use this strategy, you will need to modify the Control Engine.

# 1. Open Strategy ULCs1450.idb in ioControl.

# 2. Add a Control Engine

- a. In the strategy tree, right-click the Control Engines folder.
- b. Click Add.

<b>() ioControl - [Powerup]</b> C Elle Edit Configure Chart Subroutine C	iompile M	<u>1</u> ode	T <u>o</u> ols	⊻iew	₩ir	ndow	He	p	-	-		-	-
● # #   × • • ×   # \$   •	) () () () () ()	) ?   ~		口 所	• •	6 0 12		A Đ	() ()	2 C	16 0×16		>
JLCs1450 Control Engines Control Engines Control Engines Control Engines Add Configure Delete Polete Polete Control Variables Configure Control String Variables	nfigure C	ii.	   	   		•	· · · · · · · · · · · · · · · · · · ·						
Pointer Variables     Communication Handles     Pointer Tables     Pointer Tables     Pointer Tables     I/O Units	Acti <u>v</u> e Eng	ine: :sociati	ed with	Strate	egy:			-		<u>5</u>	et Ac Add	tive	
									[	1 ) ownl	Dele	te <u>O</u> ption	8

c. In the Configure Control Engines dialog box, click Add.

Acti <u>v</u> e Engine:		-
Select Control Engine	×	Set Active
UID Station 10	<u>A</u> dd	<u>A</u> dd
1	Modify	<u>M</u> odify
	Delete	Delete
		Download Options
OK Cancel		Č.
UK Cancel	Cancel Help	]

All control engines you have configured on your computer are shown in the Select Control Engines dialog box. If you do not have a control engine, see Creating a Control Engine below.

d. Select your Control Engine and click OK.

Configure Control Engin	25	The Control Engin
Configure Configured Control Engines: Configure Configured Control Engines: Configure Ethernet Connection Configure control engine name and parameters: Control Engine Name: Uio Station 10 Settings IP Address: 10 . 0 . 4 . 10 Port: 22001 Retries: 0 Imeout (msec): 5000 Cancel	Add Modify Delete M Downlo	Active Add. Add. Delete Dad Options

# Use a Communication Handle

Communication Handles can be assigned an initial value and their value can be reassigned using chart commands or OptoScript. Here you will assign the initial value. Later, you will use OptoScript to reassign the value.

# 1. Define the value of a Communication Handle.

- a. Right-click Communication Handles.
- b. Click Add.
- c. In the Add Variable dialog box, type the following:
  In the Name field, type ch\_AppendData
  (Throughout this tutorial, ch\_ will be used to name Communication Handles.)
- d. In the Initial Value field, type file: a, datalog.txt NOTE: Type the text as show here and use all lowercase letters.

(ioControl - [Powerup]	omile Mode Tools View Window Help
ULCS1450* OF ULCS1450 Control Engines Subroutines Included Charts Variables	Add Variable
Only Commerce Variables     Only Communication Har gles     Only Configure     Only Configure     Only Configure     Only Configure     Only Configure	Description:         Lype:       Communication Handle         Initialization         Initialize on strategy tun         Initialize on strategy download
ig- 🛅 I/O Units	OK  Cancel

e. Click OK.

# 2. Open the LogData chart.

- a. In the Strategy Tree, expand the Charts folder.
- b. Under Charts, double-click LogData.

# 3. Convert the Data to Strings.

Before any data can be written to an ASCII file, the data must be converted to string format. ioControl provides several commands for converting data, several of which are shown in the Script block Convert Data, Create String. If you are using a SNAP Ultimate I/O Learning Center, you will not need to make any changes to this block, but converting data is an essential step, so it is recommended that you examine how this is done. a. Double click the block Convert data, create string.

	Initial	data string	
	Na	ntoScript - LooData - Couvert data, create string	
Γ	0	影 暋 麅   ユ ニ   <b>鍋</b> 猛   <b>∧ 兆</b> % %   a+b 存 存   +R0 ×C0 +V   ダ	
	commu	OptoScript Code:	
- -	F. Addree	<pre>// Convert data to strings DateToString(MMDDYYY(sDate); TimeToString(Photo_Sensor, sPointState); NumberToString(Choto_Temperature, 8, 3, sTemperature); // Used in Activity 2 to increment file name // NumberToString(nRun, sRun); // Create the string to be written to the file sData = sDate + ", " + sTime + ", PhotoSensor, " + sPointState +</pre>	• ", Tempera
		Output:	
		OK Cancel Help Command Help	Ln 8, Col 1

This script block converts the date, time, digital input point 3 (Photo\_Sensor) and analog input point 12 (Store\_Temp) to string values. The conversion of nRun to a string is commented out and will be used in Activity 2.

If you have a SNAP Ultimate I/O Learning Center, make no changes to this block. If you are using this strategy with different points, see "Modifying the Sample Strategy" on page 13, for how to modify these lines of OptoScript:

NumberToString(Photo\_Sensor, sPointState);

FloatToString(Store\_Temperature, 8, 3, sTemperature);

The following line joins the contents of variables sData, sTime, sPointState, sTemperature, sCRLF into one variable:

```
sData = sDate + "," + sTime + ", PhotoSensor," + sPointState + ", Temperature," + sTemperature +
sCRLF;
```

In addition, the optional descriptions PhotoSensor and Temperature are included. Being text, these descriptions appear in red.

- b. If you are using a non-Learning Center configuration, you may wish to edit the red text to describe your points; otherwise, make no changes to the script.
- c. Click OK to close the OptoScript editor.



## 4. Open the Communication Handle.

- a. Double-click the block Open Communication.
- b. In the Instructions dialog box, click Add.

c. In the Add Instruction dialog box, click Select.

Initialize	🔊 Instructions - LogData - (	Open communication	×	
Convert data, create string			Add <u>M</u> odify	
Change F e Name	Add Instruction			×
Open communication	Instruction: Absolute Value			Select
Did	Lomment:			
File Open?	Groups:	Instructions:	×	
Address Error	Analog Point Chart Communication Control Engine	Accept Incoming Communication     Clear Communication Receive Buffer     Clear Receive Buffer     Close Communication	-	- -
	Digital Point Error Handling I/O Unit VO Unit - Event Message	Get Communication Handle Value Get End-Of-Message Terminator Get Number of Characters Waiting Lister for Incoming Communication		
Pause	I/O Unit - Memory Map I/O Unit - Scratch Pad Logical	Open Outgoing Communication Receive Character Receive N Characters		
	Mathematical Miscellaneous PID Pointers	Receive Numeric 1 able Receive Pointer Table Receive String Receive String Table		
	Simulation String	Send Communication Handle Comman Set Communication Handle Value	id 🗾	
	OK Cancel	Help Command Help		

The Select Instruction dialog box shows all the commands you would use with a Communication Handle.

- d. In the Select Instruction dialog box, select *Communication* in the Groups list and then select the instruction *Open Outgoing Communication*.
- e. Click OK to close the Select Instruction dialog box.

Review: Three ways to locate an instruction	
Type the first few letters.	Click here to see the alphabetized list.
Edit 1 istruction         Instruction:         Instruction:	Click here to open a categorized list of commands.

 f. In the Add Instruction dialog box, select the following: Communication Handle: *ch\_AppendData* Put Result In: *nResult*

hange File Name	Add Instruction	
Open	Open Dutgoing Communication	<u>S</u> elect
	Comment:	
Did T		
ile Open?	Type Name	
F	Communication Handle Communication Handle Ch_AppendData	
Idress Error	Put Result in Integer 32 Variable   Result	_
Pauro		
rause		

The variable nResult is an Integer 32 variable that has already been created. By putting the result into this variable, you can test the success of this command by reading this variable.

- g. Click OK to close the Add Instruction dialog box.
- h. Click Close to close the Instructions dialog box.

## 5. Test the result of Open Communication.

The Did File Open? block reads the value of nResult. If the value is 0, you can assume that the communication handle was opened successfully and you can write to the Communication Handle.



Make no changes to this block, because the command you need has already been created.

### **Possible Errors**

The sample strategy shows how to check the success of the Open Communication command before transmitting data. In your application, you would create actions suitable to the type of errors you may encounter. Here are a few examples: 0 =Success.

-47 = Invalid connection. Make sure the same IP address is not already open.

-49 = No more connections are available. Maximum number of connections already in use.

-50 = Open connection timeout. Could not establish connection within the timeout period.

-78 = No destination given. When sending a file via FTP, use Send Communication Handle Command to specify the name of the file on the remote server.

-446 = FTP: Login failed. Check user name, password, and maximum number of logins on server.

-447 = FTP: Connection failed. Check IP address and port.

-448 = FTP: Could not create session. Check IP address and port.

### 6. Write data to the Communication Handle.

You will add two commands to the Transmit String block. At this point, the Communication Handle is open, so you can transmit data using a Transmit String command. Then, you will close the Communication Handle.

a. Double-click the block Transmit String.

Initialize	Add Instruction	Instructions - LogData	- Transmit String	X Add Modify Delete Next Block X Select
	From Communication Handle Put Status in	Type String Variable Communication Handle Integer 32 Variable Lelp Comman	Name  SData  Ch_AppendData  nResult	¥ ¥

b. In the Instructions dialog box, click Add.

- c. In the Add Instruction dialog box, select the following: Instruction: *Transmit String*From: *sData* (NOTE: Make sure you choose *sData* and not *sDate*.)
  Communication Handle: *ch\_AppendData*Put Status In: *nResult*
- d. Click OK to close the Add Instruction dialog box.
- e. Keep the Instructions dialog box open.

## 7. Close the Communication Handle.

At this point, you do not need to keep the Communication Handle open, so you will add an instruction to close it.

a. In the Instructions window, click Add.

	From	sData	Add
	Communication Handle	ch_AppendData	Modify
	Put Status In	nkesuk	Delete
Transmit String			<u>N</u> ext Block
dd Instruction			
nstruction:			
Close Communication			▼ <u>S</u> elect
Comment:			
	Туре	Name	
Communication Handle	Type Communication Handle	Name ch_AppendData	<u>×</u>
Communication Handle Put Status in	Type Communication Handle	Name ch_AppendData	2
Communication Handle Put Status in	Type Communication Handle • Integer 32 Variable •	Name   ch_AppendData   nResult	2
Communication Handle Put Status in	Type Communication Handle • Integer 32 Variable •	Name   ch_AppendData   nResult	
Communication Handle Put Status in	Type Communication Handle • Integer 32 Variable •	Name   ch_AppendData   nResult	
Communication Handle Put Status in	Type Communication Handle • Integer 32 Variable •	Name   ch_AppendData   nResult	
Communication Handle <sup>9</sup> ut Status in	Type Communication Handle	Name   ch_AppendData   nResult	<u>,</u>
Communication Handle <sup>2</sup> ut Status in	Type Communication Handle	Name   ch_AppendData   nResult	
Communication Handle Put Status in	Type Communication Handle • Integer 32 Variable •	Name Ch_AppendData InResult	

 b. In the Add Instruction dialog box, select the following: Instruction: Close Communication
 Communication Handle: ch\_AppendData
 Put Result in: nResult

c.	Click OK to close the Add Instruction	Instructions - LogData - Transmit Strin	g X
	dialog box	From eData	<u>Add</u>
d.	Click <i>Close</i> to close the Instructions	Communication Handle ch_Appe Put Status in nResult	ndData <u>M</u> odify Delete
	dialog box.	Close Communication Communication Handle ch_Appe Put Status in nResult	ndData <u>N</u> ext Block
			Previous Block
		Close Help Comman	d Help

Your strategy is ready to be downloaded and compiled.

# 8. Compile your strategy.

Choose Compile → Compile All: ULCs1450

# Log Data

The strategy creates a text file and appends the date, time, temperature and point state every five seconds.

# 1. Make sure your SNAP Ultimate I/O Learning Center is turned on and connected to the network.

# 2. Download and run the strategy.

- a. Choose menu command *Mode*  $\rightarrow$  *Debug*.
- b. Acknowledge all download messages.
- c. Choose *Debug*  $\rightarrow$  *Run*.

## 3. Allow the strategy to run at least 30 seconds to log data.

While the strategy is running, flip the Photo Sensor switch and hold the temperature probe so the data will vary.

## 4. Stop the strategy.

a. Choose *Debug*  $\rightarrow$  *Stop*.

# View the Data Log

The data logs created on the SNAP Ultimate I/O brain can easily be opened using any FTP client application, such as Notepad or Excel. The following shows how to open the log file in Notepad:

## 1. Start Notepad.

Choose Start  $\rightarrow$  Programs  $\rightarrow$  Accessories  $\rightarrow$  Notepad.

## 2. Open the data log.

- a. From the File menu, choose Open.
- In the File name field, type the FTP address of your brain as follows:
   ftp: //10. 0. 4. 10

NOTE: Substitute the 10.0.4.10 with the IP address of your brain.

c. Click Open.

This will display all files on the brain.

Open					?×
Look jn:	10.0.4.10		•	🗢 🗈 💣 🎫	
History Desktop My Documents	E DataLog.txt				
My Computer	File name:	ftp://10.0.4.10		<b>•</b>	Open
My Network P	Files of <u>type</u> :	Text Documents (*.txt)		-	Cancel

NOTE: Accessing the FTP site from Notepad copies the files to your workstation. Therefore, to ensure that you are retrieving the latest files, it is important to re-enter the FTP address when you wish to open a file.

d. Select a data log file and click Open.

<mark>∕⊴datalog[1].txt</mark> - N File Edit Format H	Notepad Help			_ 🗆 X
09/15/2003, 14 09/15/2003, 14 09/15/2003, 14 09/15/2003, 14 09/15/2003, 14 09/15/2003, 14 09/15/2003, 14 09/15/2003, 14	19:47, PhotoSensor, 19:52, PhotoSensor, 20:02, PhotoSensor, 20:07, PhotoSensor, 20:12, PhotoSensor, 20:12, PhotoSensor, 20:22, PhotoSensor, 20:27, PhotoSensor, 20:27, PhotoSensor,	1, Temperature, 1, Temperature, 0, Temperature, 0, Temperature, 1, Temperature, 1, Temperature, 1, Temperature, 1, Temperature, 1, Temperature,	75.346 75.316 75.377 75.346 75.346 75.346 75.316 75.346 75.346	×

The data is now ready for use in your application.

In Lesson 2, you will continue working with the LogData chart to add some features that will backup up your DataLog in the event of a power failure.

# LESSON 2: SAVING LOG FILES TO FLASH

# **OVERVIEW**

In Lesson 1, you used a Communication Handle to write data to a text file on the brain. The Communication Handle was given an initial value that described the file management operation and the filename to act upon. Assigning an initial value is suitable in many cases. However, you may wish to have ioControl make decisions that result in the need to change the function of the Communication Handle. For example, if your system is saving log files, and the power was interrupted, you could increment a number in the file name to indicate that DataLog 2 represents a break in operation from DataLog 1.

In this lesson you will complete the LogData chart by adding the following:

- A persistent variable that is incrementally increased each time the strategy starts.
- OptoScript code that dynamically assigns a file name using the value of the persistent variable.
- A command that sets the value of your communication handle.
- A command that saves files in RAM to Flash.

# CONCEPTS

# Flash, RAM, and Persistent Memory

The SNAP Ultimate I/O brain provides permanent and temporary memory. The permanent area (Flash) typically is used to save the brain's firmware, configuration settings, application-ready strategy, and custom files. The RAM memory is the brain's normal operating memory and will contain much of the same information as the Flash, but the RAM memory is not saved when the brain is turned off or reset. In addition to Flash and RAM, the brain supplies non-volatile, battery-backed RAM, of which 320 MB is available for persistent variables (ioControl variables, such as integers, floats, tables, explicitly configured as persistent) and for variables that are initialized on download.

If you have information in your strategy that is not part of the configuration information already in permanent storage and must be saved in the event of a power loss (assuming an uninterruptable power supply is not an option) you have two options:

- Put data into the appropriate type of persistent variable. This method has a 256 KB limit, which is adequate for many applications; a persistent float or integer table could hold 65,536 elements.
- Store data in a file on the brain and then implement a Store To Flash command from ioControl. This method must be used with caution. Flash memory has an inherent limitation on how many times it can be written to. It is written to every time you save a strategy or configuration settings to Flash, and in normal use, you would never reach the limit. However, it's possible to design a programming loop

that repeats a Save to Flash command on a short interval. Left running, such a loop could conceivably exhaust the Flash.

A logical scenario in which you would use a Save to Flash command from ioControl is monitoring for an unusual condition. For example, you could have a point monitor a current draw on your primary power source and sense a shift in load to an uninterruptable power supply. Upon sensing that main power is lost, your strategy could write data to Flash and send a variety of warning messages via email, ioDisplay alarm, FTP, etc.



In this lesson, you create a persistent variable that will count power cycles and implement a command that will save your log file to Flash.

The following commands can be implemented from ioControl:

Erase Files in Permanent Storage

- Save Files to Permanent Storage (NOTE: Subdirectories and files in them are not stored to Flash.)
- Load Files From Permanent Storage

# Dynamic Change of a Communication Handle

In the previous lesson you saw that a Communication Handle is a variable containing communication parameters as text strings. For example, the Communication Handle ch\_AppendFile contains the following:

file: a, datal og. txt

The fill e parameter tells ioControl to use the I/O Unit's file management and is followed by an append instruction and a filename. You can use ioControl instructions to set the value of any Communication Handle. In this lesson, the persistent variable nRun is stored in non-volatile RAM and is incremented each time the strategy starts. By converting this variable to a string and inserting it into a file name, you can dynamically change the file name within a Communication Handle.



The result is that each time the brain is started, a new log file with a numerically incremented file name will be created.

# LogData Chart

In the previous lesson, you implemented the core functions of the LogData chart. In this lesson, you will add a few features so that the chart will function as described below.



# ACTIVITY 2: DYNAMICALLY CONTROLLING A COMMUNICATION HANDLE AND STORING DATA TO FLASH

# Automatically Incrementing the File Name

# 1. Return to Configure mode.

a. Choose Mode  $\rightarrow$  Configure.

# 2. Create a Persistent variable.

- a. In the Strategy Tree, right-click the Numeric Variables folder.
- b. Click Configure.
- c. In the Configure Variables dialog box, select *Persistent* from the Scope list box.



- d. Click Add to open the Add Variable dialog box.
- e. In the Name field, type nRun.
- f. In the Type field, select Integer 32.g. In the Initial Value
- field, type 0. h. Click *OK* to close the Add Variable dialog box.

Add Variable		
<u>N</u> ame:	nBun	
Description:		
<u>T</u> ype:	Integer 32	
<ul> <li>Initialization</li> <li>Initializ</li> <li>Initializ</li> </ul>	e on strategy run Initial ⊻alue: 0 e on strategy download	
	CancelHelp	

Close the Configure Variables dialog box. i.

#### 3. Add Increment Variable command to Initialize block.

a. Double-click the Initialize block.

You will add one instruction. It does not matter which sequence the instructions appear in this block, but this example adds the new instruction to the end of the list.

Convert dat	ta,			
Create sti	Instructions - LogData -	Initialize		×
Change	Append Character to Stri	ng	sed	
Name	Append	13		
×	To	sCRLF	Modity	
Open	Append Character to Stri	na	Delete	
communic	Append	10		
	То	SCRLF	Next Block	< 1 I
Did			The second s	
			Proviens Ris	all i
File Ope	n		<u>Previous Blo</u>	ck
Eile Ope	n		<u>Previous Blo</u>	ck
File Ope Add Instruction Instruction:	ble		Erevioùs Blo	<u>sk</u>
File Ope	ble		<u>Previous Blo</u>	<u>select</u>
Eile Ope Add Instruction Instruction: Increment Varia Comment:	n ble		Erevious Blo	<u>S</u> elect
Eile Ope	h n ble		Erevious Bla	<u>S</u> elect
File Ope	n ble Type	Name	Erevious Blo	<u>S</u> elect
Eile Ope	n ble Type Integer 32 Variable	Name : I I InRun	Erevious Blo	Select
Add Instruction	n ble Type Integer 32 Variable	Name ; I nRun	Erevious Blo	<u>S</u> elect
Add Instruction	ble Type Integer 32 Variable	Name : I InRun		<u>S</u> elect
Eile Ope         Add Instruction:         Instruction:         Increment Varia         Comment:	n ble Type [Integer 32 Variable	Name P I Run		<u>S</u> elect
Add Instruction	n ble Type [Integer 32 Variable	Name		<u>S</u> elect
Eile Ope         Add Instruction:         Instruction:         Increment Varia         Comment:	ble Type Integer 32 Variable	Name > T [nRun	Previous Blo	<u>S</u> elect
Add Instruction: Instruction: Increment Varia Comment:	ble Type Integer 32 Variable	Name P I Run	Previous Blo	<u>S</u> elect
Eile Ope	ble Type Integer 32 Variable	Name 2 V  nRun	Erevious Blo	Select

d. In the add instruction dialog box, choose the following:

Instruction: Increment Variable

Name: nRun

- e. Click OK to close the Add Instruction dialog box.
- Click Close. f.

Each time the strategy is started (presumably, when the brain is restarted and Auto Run is active) the LogData chart will run and increase the value of nRun by 1.

# 4. Increment File Name in a Variable.

To create a new Communication Handle containing a dynamically incremented file name, you will need to join the strings. This operation is easy in OptoScript and needs to happen before the Change File Name block. Therefore, it will be simplest to add the new commands to the existing script block Convert Data, Create String. (Note: You could also create a new OptoScript block and place it between the Convert data and Change File Name blocks, but the code you need to add has already been written for you in the Convert data, create string block.)

a. Double-click the script block Convert Data, Create String.



### b. Delete the slashes that precede the following lines

// NumberToString(nRun, sRun);

```
// sFileName = "file:a,log_" + sRun + ".txt";
```

### Your script should appear as follows:

```
// Convert data to strings
DateToStringMMDDYYYY(sDate);
TimeToString(sTime);
NumberToString(Photo_Sensor, sPointState);
FloatToString(Store_Temperature, 8, 3, sTemperature);
// Used in Activity 2 to increment file name
NumberToString(nRun, sRun);
```

```
// Create the string to be written to the file
sData = sDate + ", " + sTime + ", PhotoSensor, " + sPointState + ", Temperature, " +
sTemperature + sCRLF;
// Create the file name and communication handle value
sFileName = "file:a,DataLog_" + sRun + ".txt";
```

c. Click *OK* to close the OptoScript editor.

### 5. Assign new value to Communication Handle.

- a. Open Change File Name block.
- b. Click Add.
- c. Select the following: Instruction: Set Communication Handle Value From: String Variable – sFileName To: Communication Handle – ch\_AppendData

	<b>(</b> ]Instruction	ns - LogData - Change File N	ame		×
Initialize	>				<u> </u>
*				Modi	ly.
Convert da a,	)			Dele	te
				-	
Changy File	Add Instruction			Next BI	ock
Name Name					
Open	Instruction:				
communication	Set Communication Han	dle Value			▼ <u>S</u> elect
	Comment:				
Did	1				
File Upen?					
F↓		Туре	Nam	ne	
	From	String Variable	▼ sFile	eName	•
Address Error	To	Communication Handle	▼ ch	AppendData	-
Pause					
		. 10 os a 100	1		
	OK Cance	el <u>H</u> elp Comm	and Help		

- d. Click OK to close the Add Instruction dialog box.
- e. Click Close to close the Instructions dialog box.

# Saving a Log File to Flash

If desired, you can save a log file to Flash where it will be preserved should the brain lose power. The OptoScript implementing this feature has been commented out. Implementing an automated Write to Flash should be done cautiously: Flash memory has a limit to how many times it can be written to. Though the limit is high, an automatic loop repeating at a short interval could reach this limit if left running.

To write your log file to flash, do the following:

# 1. Save Files to Flash.

a. Open the bloc b. Click <i>Add</i>	k Write to Flash.	o Flash	Add Modify	
Add Instruction Instruction: Save Files To Permanent Storage Comment: Tune	Name	Select	Block s Block	$\frac{trin}{2}$ ints 0 $\frac{1}{1}$
Put Status in Integer	32 Variable <u>Help</u> <u>Command Help</u>		-	Write to Flash

- c. In the Add Instruction dialog box, select the following: Instruction: Save Files to Permanent Storage
   Put Status In: Integer 32 Variable – nResult
- d. Click OK to close the Add Instruction dialog box.
- e. Click Close.

## 2. Enable the Save to Flash and AutoRun options.

This step will ensure that your strategy is saved across a restart.

a. Select File  $\rightarrow$  Strategy Options.

b. Click the Download tab.



- c. Under Flash Memory, first select *Save strategy to flash memory after download*. Selecting this option enables the Set AutoRun option.
- d. Click Set AutoRun flag after download.
- e. Click OK to close the Strategy Options dialog box.

# **Test Your Changes**

### 1. Download and run your strategy.

- a. Choose  $Mode \rightarrow Debug$ .
- b. Choose *Debug*  $\rightarrow$  *Run*.

Allow the strategy to run for a few minutes. (Note: AutoRun takes effect after the strategy is running.)

## 2. Allow your strategy to log data.

Your strategy is logging data from the Photo Sensor switch and the temperature probe.

## 3. Turn on the Emergency and POS switches.

Simultaneously, hold both switches down for a few seconds.

## 4. Restart your strategy.

a. Turn off your SNAP Ultimate I/O Learning Center. The power switch is below the power cord connection. ioControl will provide a timeout warning.

ioControl	×
	Timeout. No response from device. Check hardware connection, address, power, and jumpers.
	Last command sent:
	ANY.TASKS?
	Abort Retry Ignore

- b. Select Abort; this will return ioControl to Configure mode.
- c. Turn your Learning Center back on.

Since you set the AutoRun flag, the strategy should restart after the brain is restarted.

Accessing files that have been saved to Flash is no different from accessing the files on RAM, the only difference is that log files will be preserved when the power to the brain is turned off.

# 5. Start Notepad.

## 6. Open the data log.

Note that files may be shown from previous sessions. These files may have been downloaded to your computer during the previous FTP session and may not be current. It is recommended that you reenter the connection to the brain's FTP to ensure that you are offered the most recent files residing on the brain.

- a. From the File menu, choose Open.
- b. In the Open dialog box, type the following:

ftp: //10. 0. 4. 10

NOTE: Substitute 10.0.4.10 with the IP address of your brain.

c. Click Open.

This will display all files on the brain.

Open					? ×
Look jn:	2 10.0.4.10		💌 🗢 🔁 (	·	
	≡ DataLog_1.txt ≡ DataLog_2.txt				
Desktop					
My Documents					
My Computer					
Mu Network P	File <u>n</u> ame:	ftp://10.0.4.10			<u>O</u> pen
	Files of type:	Text Documents (*.txt)			Cancel

The files on the brain reflect the restarts. In the example above, DataLog\_1.txt was created when the strategy was downloaded and represents the file that was

copied to Flash before the loss of power. DataLog\_2.txt was created after the brain was restarted.

# Disable AutoRun and Saving to Flash

You will not need to save to flash to complete the following lessons.

## 1. Disable AutoRun.

- a. In Configure mode, choose File  $\rightarrow$  Strategy Options.
- b. Click Download.
- c. Deselect the options under Flash Memory.

## 2. Disable the Save To Flash command.

- a. Open the block Write To Flash.
- b. Select the Save Files to Permanent storage command.
- c. Click *Delete*.
- d. Click Yes.
- e. Close the Instructions dialog box.



# 3. Download and run your strategy.

This will replace the previous strategy.

## 4. Reset nRun.

a. Make sure the strategy is running.



b. In the Strategy Tree under Numerical Variables, double-click nRun.

- c. Expand the nRun Scanning dialog box.
- d. In the Value field, type 0.

🕵 "nR	un" (NOT SCANNING)	_ 🗆 X
Name: Type:	nRun Integer 32 Variable Initialized: Not Applicabl	е
⊻alue:	0	
CI	lose Apply Add Watch H	elp

- e. Click Apply.
- f. Click Close.
- 5. Return to Configure mode.



# LESSON 3: FTP SERVER AND READING UPLOADED DATA

# **OVERVIEW**

In this lesson, you will modify the chart LEDsFromRecipe so it will parse a commadelimited ASCII file (recipe.txt) uploaded to the brain. The values contained in recipe.txt are converted to integers and then written to digital output points 5 (Outside\_Light), 6 (Inside\_Light), and 7 (Freezer\_Door\_Status). This example converts strings to integers but the same methods can easily be adapted to convert strings to float variables to control analog output points.

Any Office or Enterprise Application FTP client Send file to brain and create directory Create delimited Ethernet **SNAP Ultimate I/O** text file FTP Server ioControl strategy uio E -- recipe.txt Read data from file Write to points Ultimate I/O Learning Center LEDS

Any Equipment, Process or Asset

# **CONCEPTS**

# How the Sample Chart Works

The LEDsFromRecipe chart will respond to the SNAP Ultimate I/O Learning Center's Emergency switch to initiate the commands that will read a text file, parse it, and

control the LEDs (digital output points 4, 5, and 6). The diagram below describes the key functions this chart will perform when you complete lesson 3.



1	A text file containing three values, each followed by a comma delimiter, is loaded to the brain using the brain's FTP server.
2	When the Emergency button is held down, ioControl sets the value of the Commu- nication Handle, establishes communication, and sets the end-of-message termi- nator.
3	The Communication Handle buffer contains the contents of the file. A receive string table command puts the contents into a string table using the end-of-mes-sage terminator to separate table elements.
4	The values in the string table are converted to integers and then moved to the dig- ital output points.

# Uploading a File to the Brain

Any FTP client can upload a file to the Ultimate I/O brain. Files can be uploaded to the brain's root directory or to subdirectories.

In this lesson, you will use ioManager as your FTP client application to create a directory on the brain and upload a text file to this directory.

# **Communication Handle Read Commands**

A Communication Handle is used to read a file from the brain. The Communication Handle contains the *file*, *r* parameters to indicate the read function, followed by the file

name. If the file resides in a subdirectory, the subdirectories are included in the file name. For example:

file,r:recipe.txt
file,r:\directory\recipe.txt

When using a read Communication Handle:

- Pay careful attention to case; the Communication Handle is case-sensitive.
- Set End-of-Message (EOM) Terminator after the Communication Handle is opened. The character is represented by an ASCII value (see the ASCII table under "String Commands" in Chapter 10 of the *ioControl User's Guide*, form #1300). Common EOMs include a comma (character 44) and a colon (character 58). The default EOM is a carriage return (character 13).
- Close communication when finished.

# Parsing a Text File

The illustration below shows four ioControl instructions used to move data from the Communication Handle buffer to a point.



This lesson also presents a simpler method using OptoScript:

```
for nCounter = 0 to 2 step 1
nt_recipe[nCounter] = StringToFloat(st_recipe[nCounter]);
```

next

```
Outside_Light = nt_recipe[0];
Inside_Light = nt_recipe[1];
Freezer_Door_Status = nt_recipe[2];
```

The first three lines replace 6 of the ioControl instructions by creating a loop that converts each string to a float while putting the value into a table.

The last three lines move individual values from the Integer 32 table to digital points.

# **ACTIVITY 3: UPLOADING AND PARSING FILES**

# Reading a File through a Communication Handle

1. Make sure that you are in Configure mode.

### 2. Create Communication Handle.

- a. In the strategy tree, right-click the Communication Handles folder.
- b. Click Add.

⊡ 🔁 ULCs1450	
🗄 🖳 🛅 Control Engines	
Subroutines Included	
🗄 💼 Charts	
Powerup	
🛱 📶 Variables	
🕀 💼 Numeric Variables	
🕀 💼 String Variables	
Pointer Variables	
🔁 🔄 Communication Handles	
Mumeric Tables     Add	
String Tables     Configure	
Pointer Tables	
E I/O Units	_
Add Variable	
Name: ch_ReadFile	1
Description:	1
	1
Lype: Communication Handle	1
- Initialization	1
Initialize on strategy run     Initial Value:	1
O Initialize on strategy download	1
	_

c. In the name field, type ch\_ReadFile.

NOTE: You will use a Set Value of Communication Handle command to assign the value, so you don't need to configure an initial value here.

d. Click OK to close the Add Variable dialog box.

- 3. Create variable for End-of-Message Terminator.
  - a. In the strategy tree, right-click the Numeric Variables folder.
  - b. Click Add.

□ (III) ULCs1450		
🕀 📄 Control Engines		
📋 🛅 Charts		
LEDsFromRecip	P	
- 📌 LogData		
- 🖓 Powerup		
🖻 💼 Variables		
😥 🏐 Numeric Variabl		
庄 💼 String Varia 👘	dd	
💼 Pointer Var 🔍	ionfigure	
🚊 💼 Communica 🦷	elete:	
ch_Append	Data	
ch_ReadF	dd Yariable	Y
🕀 💼 Numeric Table		
🕀 💼 String Tables	Name: nEOM	1
- Pointer Tables		-
	Description:	
	Type: Integer 32	
	_ Initialization	
	Initialize on strategy run Initial ⊻alue: 44	
	Initialize on strategy download	
		7
	Cancel Help	

- c. In the Add Variable dialog box, provide the following information:
  - Name: nEOM
  - Type: Integer 32
  - Initial Value: 44

(44 is the ASCII character for a comma.)

d. Click OK to close the Add Variable dialog box.

# 4. Open the LEDsFromRecipe chart.

- a. In the Strategy Tree, expand the Charts folder.
- b. Double-click LEDsFromRecipe.

# 5. Add Instructions to Open Communication block.

a. Double-click the Open Comm block.

Furn on point 0 (Emergen o initiate reading of file.	Block 0 cy) Set Recipe T T pen Comm		
Read Recipe.txt Write to LEDs	Instructions - LEDsFrom	nRecipe - Open Comm	Add Modify Delete
Instruction: Set Communication Handle Comment:	s Value	Y	Select
From	Type String Literal	Name file:r,\uio\recipe.txt	•
To	Communication Handle	ch_ReadFile	<b>_</b>
OK Cancel	Help Command	Help	

- c. In the Add instruction dialog box, select the following information: Instruction: Set Communication Handle Value
   From: String Literal – file:r, \uio\recipe.txt
   (NOTE: Type the text exactly as shown, using all lowercase letters.)
   To: Communication handle – ch\_ReadFile
- d. Click OK.
- e. Leave the Instructions dialog box open.

## 6. Open Outgoing Communication.

a. In the Instructions dialog box, click below the Set Communication Handle Value.

From	Handle Value "file:r,\uio\recipe.txt"	Add		
То	ch_ReadFile	Modify		
		Delete	<b>¥</b>	
	Add Instruction		•	
	Instruction:			
	Open Outgoing Comm	unication		▼ <u>S</u> elect
	Comment:			1.549
	Comment:			
	Comment:	Туре	Name	
Close 1	Lelp Communication Handle	Type © Communication Handle 💌	Name Ch_ReadFile	
Close 1	Comment:	Type <sup>2</sup> Communication Handle ▼ Integer 32 Variable ▼	Name ch_ReadFile nCommStatus	× ×
Close 1	delp Communication Handle Put Result in	Type 9 Communication Handle ▼ Integer 32 Variable ▼	Name ch_ReadFile nCommStatus	*
Close 1	Comment:	Type Communication Handle	Name ch_ReadFile nCommStatus	× ×

- c. In the Add Instruction dialog box, select the following information: Instruction: Open Outgoing Communication
   Communication Handle: ch\_ReadFile
   Put Result In: Integer 32 Variable – nCommStatus
- d. Click OK.
- e. Leave the Instructions dialog box open.

# 7. Set End-of-Message Terminator.

a. In the Instructions dialog box, select below the Open Outgoing Communication instruction.

et Communication Handle Valu	ie "filer: \uio\recine txt"	Add		
То	ch_ReadFile	Modify		
		Database		
en Outgoing Communication	ch DeadFile	Delece		
Put Result in	nCommStatus	Next Block		
	Add Instruction	Provincius Plank I		
	Instruction:			
	Instruction: Set End-Of-Message Terr	minator		Select
Close Help	Instruction: Set End-Of-Message Terr Comment:	minator		Select
Close	Instruction: Set End-Of-Message Terr <u>C</u> omment:	minator		Select
Close Help	Instruction: Set End-Of-Message Terr Comment:	minator		▼ <u>S</u> elect
Close Help	Instruction: Set End-Of-Message Terr Comment:	minator Type	Name	Select
Close Help	Instruction: Set End-Of-Message Terr Comment: Communication Handle	minator Type Communication Handle	Name [ch_ReadFile	Select
Close Help	Instruction: Set End-Of-Message Terr Comment:	minator Type Communication Handle	Name ch_ReadFile	Select
Close Help	Instruction: Set End-Of-Message Terr Comment: Communication Handle To Character	minator Type Communication Handle	Name ch_ReadFile nEDM	Select
Close Help	Instruction: Set End-Of-Message Terr Comment: Communication Handle To Character	Type Communication Handle	Name Ch_ReadFile NEDM	<u>Select</u>
Ciose <u>H</u> elp	Instruction: Set End-Of-Message Terr Comment: Communication Handle To Character	Type Communication Handle	Name ch_ReadFile nEOM	<u>Select</u>
Ciose <u>H</u> elp	Instruction: Set End-Of-Message Terr Comment: Communication Handle To Character	Type Communication Handle	Name ch_ReadFile nEOM	V Select

- c. Select the following information: Instruction: Set End-Of-Message Terminator Communication Handle: ch\_ReadFile Put Status In: Integer 32 Variable – nEOM
- d. Click OK.



When you are finished, you should have the following instructions in the order shown below.

e. Click Close.

## 8. Test communication.

a. Double-click the Did File Open? condition block.

b. In the Instructions dialog box, click Add.



- c. In the Add Instructions dialog box, provide the following information: Instruction: Equal?
  Is: Integer32 Variable – nCommStatus
  To: Integer 32 Literal – 0
- d. Click OK.

Instructions - LED	FromRecipe - Did file open?	×
ls Equal?	nCommStatus	Add
То	0	Modify
		<u>D</u> elete
		<u>N</u> ext Block
		Previous Block
		<u>Operator</u>
		© AND © OR
Close	Help Command Help	

e. Click Close.

# 9. Receive String table.

- a. Double-click the Read Recipe block.
- b. In the Instructions dialog box, click Add.

	Block 0			
<b>()</b> Inst	ructions - LEDsFromRec	ipe - Read Recipe.txt	×	
			Add	
_			200	
			Modry	
	Add Instruction		Delete	X
Riad L	Instruction:			
Recipe.txt	Receive String Table			▼ <u>S</u> elect
1				
Write to LEDs				
		Туре	Name	<u> </u>
	Length	Integer 32 Literal	3	•
_	Start at Index	Integer 32 Literal	0	
	0(7-1)-			
	Uriable	String Table	st_Recipe	<u> </u>
	Communication Handle	Communication Handle	ch_ReadFile	<b>•</b>
	Put Status in	Integer 32 Variable	nCommStatus	
I				<b>_</b>
[	OK Cancel	Help Command He	elp	

- c. In the Add Instruction dialog box, provide the following information: Length: Integer 32 Literal – 3
  Start at Index: Integer 32 Literal – 0
  Of Table: String Table – st\_Recipe
  Communication Handle: ch\_ReadFile
  Put Status in: nCommStatus
  - (NOTE: You may need to scroll the window to see all the options.)
- d. Click OK.
- e. Leave the Instructions dialog box open.

## 10. Close communication.

a. In the Instructions dialog box, select below the Receive String Table instruction.



KOU10	structions - LEDsFromRec	ipe - Read Recipe.txt	×	
R	eceive String Table	868	Add	
	Length	3		
	Start at Index	0	Modify	
	Or Table Communication Handle	st_Recipe	Delete	
	Put Status in	nCommStatus		
	T dt otdtdo in		Next Block	
d L	A		Descience Disabil	
1 EDe	Instruction: Close Communication			✓ <u>S</u> elect.
LEDs	Instruction: Close Communication <u>C</u> omment:			<u>S</u> elect.
LEDs	Instruction: Close Communication Comment:	Туре	Name	▼ <u>S</u> elect.
LEDs	Instruction: Close Communication Communication Handle	Type Communication Handle	Name ch_ReadFile	Select.
LEDs	Instruction: Close Communication Comment: Communication Handle	Type Communication Handle	Name ch_ReadFile	Select.
LEDs	Instruction: Close Communication Comment: Communication Handle Put Status in	Type Communication Handle 💌 Integer 32 Variable 💌	Name ch_ReadFile nCommStatus	Select.

- c. Select the following information: Instruction: Close Communication
   Communication Handle: ch\_ReadFile
   Put Status in: Integer 32 Variable – nCommStatus
- d. Click OK.

Your Instructions should appear as follows:

Receive String Table		Add
Length	3	800
Start at Index	0	Modify
Of Table	st_Recipe	
Communication Handle	ch_ReadFile	<u>D</u> elete
Put Status in	nCommStatus	
		<u>N</u> ext Block
Close Communication		
Communication Handle	ch_ReadFile	Previous Block
Put Status in	nCommStatus	

e. Click Close.

# Parse a Data File

The chart shows two ways to extract the data from the Communication Handle buffer. In each case, the following must occur:

- The string data residing sequentially (e.g., "1,0,1,") in the Communication Handle must be put into a table.
- A specific string value must be converted from its string format to a numerical value.
- The numerical value is applied (moved) to the point.

## 1. Examine ioControl commands.

a. Double-click the Write To LEDs block.

The instructions in this block are ready to use; you do not need to make changes. Three instructions are used in three sets, as shown here.



NOTE: If you do not have a Learning Center, you'll need to make sure that the Move commands cite digital output points configured on your SNAP Ultimate I/O. If you do not have three points to control, remove all but the first three commands.

b. Click Close.

## 2. Examine OptoScript.

As an alternative to ioControl instructions, you can use the OptoScript sample that is provided (expand the chart window if the OptoScript block isn't shown). The advantage of the OptoScript is that is uses fewer steps.

a. Double-click Write to LED (Alternative).

The first three lines (not counting the comment in line 1) use a repeating loop that puts the text into the string table and coverts it to a number.



Setting the digital points to the value of the elements in nt\_recipe, becomes a simple action of assigning the desired table value to each point:

```
Outside_Light = nt_recipe[0];
Inside_Light = nt_recipe[1];
Freezer_Door_Status = nt_recipe[2];
```

b. Close the OptoScript editor.

## 3. Choose block

- a. If you wish to use the ioControl instructions, make no changes.
- b. If you wish to use the OptoScript, redraw the chart so that the OptoScript block is used instead of the instruction block (both blocks are named Write to LEDs).

# Create and Upload a File

## 1. Create a comma-delimited file.

- a. Open Notepad.
- b. Type the following in a new file:
  - 1, 0, 1,

Make sure you include a comma after each number, and do not use spaces between numbers and commas.

c. Save your file as recipe.txt.

## 2. Upload recipe.txt using ioManager.

a. Start ioManager.

io	Manager				
File	Tools <u>V</u> iew <u>H</u> elp				
D	Inspect Maintenance	Ctrl+I Ctrl+M			
	Listen for BootP Reques Change IP Settings Download New Control I	t Engine Firmware			
	Import/Copy I/O Unit				
		🔄 I/O Unit Maintenance			×
		Command			
		Install Firmware Install Module Firmware	Filename: C:\Program Files\Opto22\UIO Learni	IP Addresses 10.0.4.10	Add
		Upload File From I/U Unit Upload File To I/O Unit Delete File On I/O Unit	Destination:		Modify
		Read Filenames On I/O Unit Save Files To Flash	\uio\recipe.txt		Delete
		Clear Flash Files			Save
		Execute			Load
		Results			
					Сору
					Select All
					Liear Completed Hesults
		Close			

b. From the ioManager application window, chose *Tools*  $\rightarrow$  *Maintenance*.

- c. In the Command Field, select Upload File to I/O Unit.
- d. In the File Name field, type the path and file name of recipe.txt you just created, or use the browse button to locate the file recipe.txt.
- e. In the destination field, type the following:

\ui o\reci pe. txt

(NOTE: The path and file name must correspond to the path and file name cited within the Communication Handle.)

- f. Select the IP address of your brain. (NOTE: if your brain's IP address is not listed, click *Add* and type the IP address in the Add IP Address dialog box.)
- g. Click Execute.

# Test Your Strategy

- 1. Download and run your strategy.
- 2. Hold down the emergency switch for at least a second.



Notice that your LEDs are now on or off depending on the values contained in recipe.txt.

To change the states of the points, edit recipt.txt on your computer, upload it to the brain, and then hold down the emergency switch again.