**Hands-on Lab**

**Lego Programming – BricxCC File Handling**

NxC provides the ability to save data to files. This provision is important; sensors can be sampled and the resulting data can be saved for future plotting of performance. In other words, the NXT Brick can act as a datalogger.

# **Concept 1 – File Saving:**

# Let’s write a program that computes a function, saves the results to a file, so that we can plot it. We’ll use the function $y=x^{2}$ where we know, when plotted is a parabola..

**Step 1:** Create a new file called **x^2File1.0.nxc**

// File: x^2File1.0.nxc - Works!

// Date: 10/22/22 10:40

// Desc: Calculate x^2 and save to file to show parabola plot

// Vers: 1.0: ME 425 release

// Refs: squareAndSqrtToFile0\_2a.nxc

task main () {

 // Declare variables -------------------------------------------------------

 int x; // integers from -5 to +5

 int xSquared; // square of x

 string strX, strXSquared; // string versions of x and xSquared

 // File related variables

 unsigned int result; // flag returned when handling files

 byte fileHandle; // handle to the data file

 short bytesWritten; // number of bytes written to the file

 string fileName, fileHeader, text; // name, header and text to write to file

 // Algorithm starts here ---------------------------------------------------

 // (1) Set up the file

 fileName = "parabola.csv" ; // <---- file name you want data saved to

 result=CreateFile(fileName, 1024, fileHandle);

 // (1A) Check if filename already exists, and overwrite it

 while (result==LDR\_FILEEXISTS) {

 CloseFile(fileHandle);

 DeleteFile(fileName);

 result=CreateFile(fileName, 1024, fileHandle);

 } // end while

 // (1B) write column header to file

 fileHeader = "X, X^2" ; // <---- header for columns in your data

 WriteLnString(fileHandle, fileHeader, bytesWritten);

 // (2) Process that generates data

 for (x = -5; x <=5; x++) {

 xSquared = x\*x;

 TextOut (10, LCD\_LINE4, FormatNum("x = %d" , x));

 TextOut (10, LCD\_LINE5, FormatNum("xSquared = %d" , xSquared));

 Wait (SEC\_1); // wait 1 sec so user can see calculations displayed

 // Create string version of calculated values

 strX = FormatNum("%d" , x);

 strXSquared = FormatNum("%d" , xSquared);

 // Concatenate the 2 strings into a single one.

 // Write resulting string to file. The text will end with an EOL character

 text=StrCat(strX, "," , strXSquared);

 // (2A) Write the string version of numerical data to file

 result=WriteLnString(fileHandle, text, bytesWritten);

 // Safety check: if the end of file is reached, close the file

 if (result==LDR\_EOFEXPECTED) CloseFile(fileHandle);

 } // end of for loop

 // (3) Finished algorithm. Clean up and quit gracefully

 ClearScreen();

 TextOut(0, LCD\_LINE2, "Quitting", false);

 // (3A) Close the file

 CloseFile(fileHandle);

 PlaySound(SOUND\_LOW\_BEEP); // Beep to signal quitting

 Wait(SEC\_2);

 StopAllTasks();

} // end of main

**Step 2:** Save, compile and execute the resulting program. The program iterates from -5 to +5, displaying the integers and their square. Additionally, in the background, the Brick stores the data to file named: **parabola.csv**.

To view this data file, after the program completes, select Tools – NXT Explorer (see **Figure 1A**). A pop-up box should display the files stored within your NXT Brick (as shown **in Figure 1B**). Click-and-drag the file parabola.csv from the left pane (i.e. Brick’s directory) to the right one (your PC’s drive).



**Figure 1B:** Click-and-drag the file parabola.csv to your PC.



**Figure 1A:** Launch the NXT Explorer to view Brick’s files

parabola.csv

**Step 3:** Double-click on the version of parabola.csv that is saved on your PC. Excel should already be configured to open CSV (comma-separated files), resulting in **Figure 1C.**



**Figure 1C:** Excel opens the resulting parabola.csv file. One can then select data for a scatter plot.

**Code Explanation:**

As the NXC function CreateFile suggests, it creates a file. Highlighted in orange one sees:

result=CreateFile(fileName, 1024, fileHandle);

The 3 inputs are: the file name (a string); the maximum size of the file; and the memory location (a byte) of the file. Respectively the variables declared in this program are called fileName; 1024 (number of bytes i.e. characters), and fileHandle.

Above the call to CreateFile, highlighted in yellow, one sees the string “parabola.cvs” assigned to the variable fileName. The 2nd yellow-highlight writes string data to the file:

fileHeader = "X, X^2" ; // <---- header for columns in your data

WriteLnString(fileHandle, fileHeader, bytesWritten);

The NXC function WriteLnString is used to write the string (saved in fileHeader) to the memory location of the file (fileHandle). It stores the number of alphanumeric characters that was successfully written to the file, in the variable bytesWritten. The effect is that first line of data in the file parabola.csv will be “X, X^2”, namely column titles for our data.

Now the for-loop simply iterates from -5 to +5. But because the file requires strings, FormatNum is used (see aqua highlight) to convert the integer values of x and xSquared into strings, respectively strX and strXSquared. StrCat is then used to combine these strings into a single one called text. Finally, before the loop returns, WriteLnString is used to write text to the file.

Lastly, the file must be closed. Thus, when the for-loop is done, the NXC function CloseFile is used:

CloseFile(fileHandle);

The program waits 2 seconds, StopAllTasks, and exits.

**Exercise 1:** In NxC create programs for the following:

* 1. Iterate integers from -10 to +10 incrementally by 1. Compute the cube and save to a file named “cubic.csv”. Export the data file and plot the curves in Excel.
	2. Define a frequency $ω=10 rad/s$. Compute the function$ y=\sin(x)$. Save the data for 2 periods worth of data and plot the curve in Excel.
	3. Recall your Domabot. One lab configured it with an ultrasonic sensor. The task was to dock 15 cm from a wall. In another lab, the Domabot was configured with a light sensor. That task had the robot follow a black line. Chose one configuration and capture sensor data for each iteration of the loop.