## 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.

### Concept 1 – File Saving:

The program `displaySquareAndSquareRoot1_0.nxc` displayed an integer, its square and square root on the Brick's LCD. This program used the for-loop to iterate the integer from 1 to 10. Building on this example, a program is written to save the values to a file. The file will then be imported into an Excel worksheet. Once one has a worksheet, the data can be manipulated and/or plotted.

**Step 1:** Click `File – Open` and load `displaySquareAndSquareRoot1_0.nxc.` Click `File – Save As` with the name "`displaySquareAndSquareRoot2_0.nxc`".

**Step 2:** Define global variables that serve for file handling. Add the following code to above your `task main` routine.

```
// File: displaySquareAndSquareRoot2_0.nxc
// Date: 10/01/12 15:43
// Desc: Display number, its square and square root save to file
// Vers: 2.0
// Refs: displaySquareAndSquareRoot1_0.nxc

// Global variables (for file writing)
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 fileHeader; // column header for data in the file
int fileNumber, filePart; // integers to split up data file names
string fileName; // name of the file
string strFileNumber; // file number e.g myDataFile 1, 2, 3
string strFilePart; // file part e.g. myDataFile1-1, 1-2, 1-3
string text; // string to be written to file i.e. data values

task main ()
```

Add global variables

**Step 3:** Compose a function to initiate a file. Add the following code above `task main`:

```
string strFilePart; // file part e.g. myDataFile1-1, 1-2, 1-3
string text; // string to be written to file i.e. data values

// Create and initialize a file
void InitWriteToFile() {
  fileNumber = 0;  // set first data file to be zero
  filePart = 0; // set first part of first data file to zero
  fileName = "squareData.csv" ; // name of data file
  result=CreateFile(fileName, 1024, fileHandle);
  // NXT Guide Section 9.100 pg. 1812 and Section 6.59.2.2 pg. 535
  // returns file handle (unsigned int)

  // check if the file already exists
  while (result==LDR_FILEEXISTS) // LDR_FILEEXISTS returns if file pre-exists
     {
      CloseFile(fileHandle);
      fileNumber = fileNumber + 1; // create new file if already exists
      fileName=NumToStr(fileNumber);
      fileName=StrCat("squareData" , fileName, ".csv");
      result=CreateFile(fileName, 1024, fileHandle);
      } // end while

  // play a tone every time a file is created
  PlayTone(TONE_B7, 5);
  fileHeader = "x, x^2, sqrt(x)" ; // header for myData file
  WriteLnString(fileHandle, fileHeader, bytesWritten);
  // NXT Guide Section 6.59.2.43 pg. 554
  // Write string and new line to a file
  // bytesWritten is an unsigned int.  Its value is # of bytes written

} // end InitWriteToFile

task main ()
```

Add this function

**Step 4:** Compose a function to write to file. Add the following code above `task main`:

```
} // end InitWriteToFile

void WriteToFile(string strTempText) {
  // strTempText stores the text (i.e. ticks and motorRpm to be written to file
  // write string to file
  result=WriteLnString(fileHandle, strTempText, bytesWritten);
  // if the end of file is reached, close the file and create a new part
  if (result==LDR_EOFEXPECTED) // LDR_EOFEXPECTED is flagged when end-of-file
    { // close the current file
      CloseFile(fileHandle); // NXT Guide Section 6.59.2.1 pg. 535
    // Closes file associated with file handle

    // create the next file name
    filePart = filePart + 1;
    strFileNumber = NumToStr(fileNumber);
    strFilePart = NumToStr(filePart);
    fileName = StrCat("squareData" , strFileNumber,"-", strFilePart ,".csv");

    // delete the file if it exists
    DeleteFile(fileName); // NXT Guide Section 6.59.2.5 pg. 537
    // Delete the file specified by the string input

    // create a new file
    CreateFile(fileName, 1024, fileHandle);
    // play a tone every time a file is created
    PlayTone(TONE_B7, 5);
    WriteLnString(fileHandle, strTempText, bytesWritten);
    } // end if
} // end WriteToFile

task main ()
```

Add this function

**Step 5:** Next, compose a function that closes the file. Add the following code above `task main`:

```
} // end WriteToFile

// Close the file
void StopWriteToFile() {
  // close the file
  CloseFile(fileHandle);
} // end StopWriteToFile

task main ()
```

Add this function

At this point, save your NxC program. To recap, Step 2 declared the variables needed for file handling and Steps 3 to 5 created functions to respectively initialize (i.e. create), write string data and close a file.

**Step 6:** File data is stored as strings. As such, strings must be declared for each integer and float. Also, to create a file, one must initialize one. Add the following within `task main`:

```
task main ()
{
  int x; // integers from 1 to 10
  int xSquared; // square of x
  float xSquareRoot; // square root of x

  string strX;
  string strXSquared;
  string strXSquareRoot;

  // Create a new file that captures time and motor speed
  InitWriteToFile();

  for (x = 1; x <=10; x++) {
    xSquared = x*x;
    xSquareRoot = sqrt(x);
```

Declare string versions of integers and floats. Also, create a file.

**Step 7:** In the for-loop, the program iterates from 1 to 10, calculating the square and square root. We can use the FormatNum function to create a string version of numbers (i.e. integers and floats). Add the following within the `for-loop`:

```
    TextOut (10, LCD_LINE4, FormatNum("x = %d" , x));
    TextOut (10, LCD_LINE5, FormatNum("xSquared = %d" , xSquared));
    TextOut (10, LCD_LINE6, FormatNum("sqrt(x) = %3.3f" , xSquareRoot));
    Wait (SEC_2);

    // Create string version of calculated values
    strX = FormatNum("%d" , x);
    strXSquared = FormatNum("%d" , xSquared);
    strXSquareRoot = FormatNum("%3.3f" , xSquareRoot);

 } // end of for loop


} // end of main
```

FormatNum is akin to ANSI-C's sprintf() function. It creates strings from numbers.

**Step 8:** Finally, one should write the 3 strings (`strX, strXSquared` and `strXSquareRoot`) to the file. To do so efficiently, one can employ the ANSI-C `strcat` function which concatenates multiple strings into a single one. Finally, write the string to file. Add the following code within the `for-loop`

```
// Create string version of calculated values
    strX = FormatNum("%d" , x);
    strXSquared = FormatNum("%d" , xSquared);
    strXSquareRoot = FormatNum("%3.3f" , xSquareRoot);

    // Concatenate the 3 strings into a single one.
    // Write resulting string to file. The text will be end with a EOL
    text=StrCat(strX, "," , strXSquared, "," , strXSquareRoot, "," );
    WriteToFile(text);

  } // end of for loop

} // end of main
```

Use strcat to combine strings. Write resulting string to file

**Step 9:** After the program has generated the data (i.e. completed the `for-loop`), one terminates the program gracefully by closing the file. One can also add an LCD message and beep to let the user know the program is done. Add the following after the `for-loop` and before the end of `main`.

```
  // Concatenate the 3 strings into a single one.
    // Write resulting string to file. The text will be end with a EOL
    text=StrCat(strX, "," , strXSquared, "," , strXSquareRoot, "," );
    WriteToFile(text);
  } // end of for loop

  // Finished computing square and square root, so clean up and quit
  ClearScreen();
  TextOut(0, LCD_LINE2, "Quitting", false);
  StopWriteToFile();
  PlaySound(SOUND_LOW_BEEP);  // Beep to signal quitting
  Wait(SEC_2);

} // end of main
```

Add this alert user of termination and close file

**Step 10:** Save, compile and execute the resulting program. The program should iterate from 1 to 10, displaying the integers, its square and square root. Additionally, in the background, the Brick stores the data to file named: `squareData.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 `squareData.csv` from the left pane (i.e. Brick's directory) to the right one (your PC's drive).

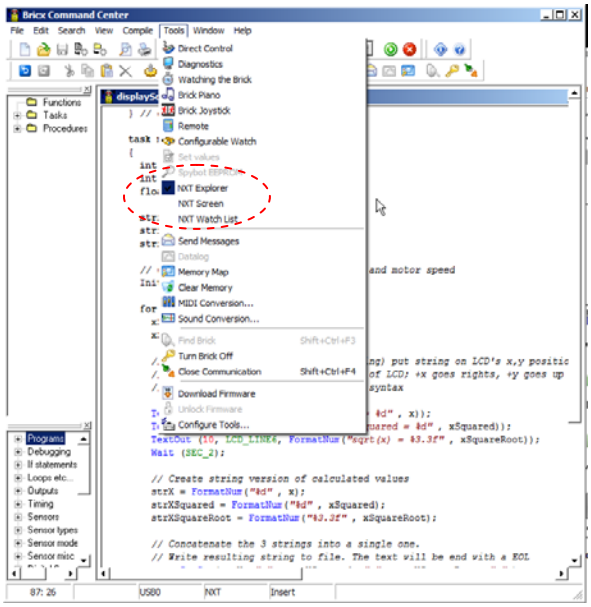


**Figure 1A:** Launch the NXT Explorer to view Brick's files

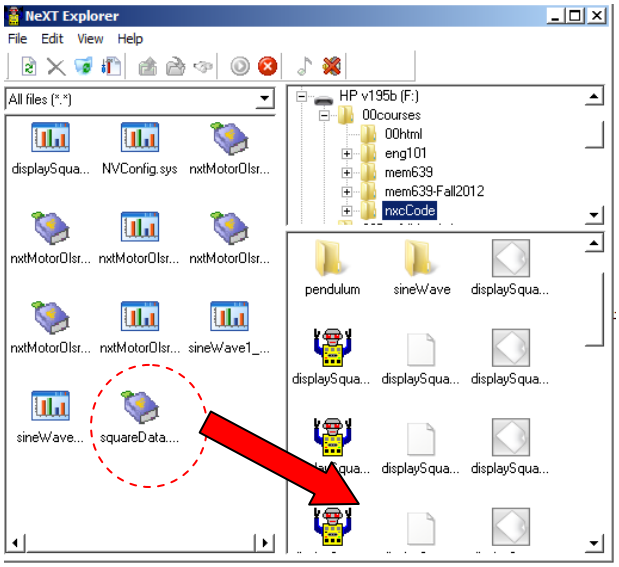


**Figure 1B:** Click-and-drag the data file `squareData.csv` to your PC.

**Step 11:** Double-click on the version of `squareData.csv` that is saved on your PC. Excel should already be configured to open CSV (comma-separated files), resulting in **Figure 1C.** **Figure 1D** shows the resulting scatter plot of the first 2 columns.

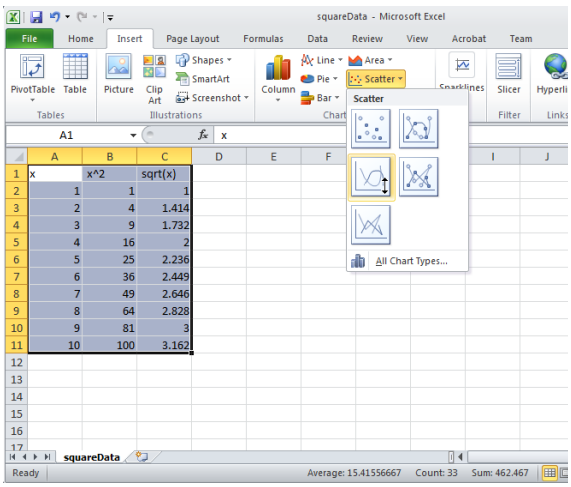


**Figure 1C:** Excel opens the resulting squareData.csv file. One can then select data for a scatter plot.

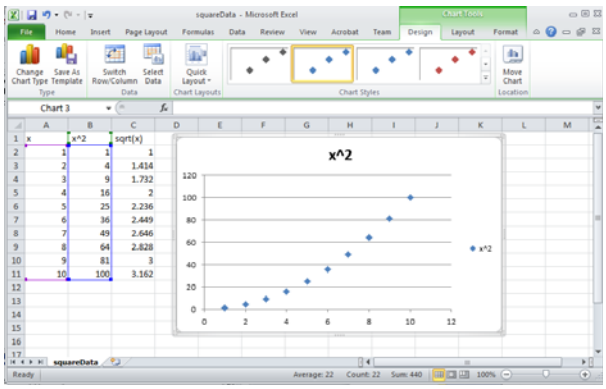


**Figure 1D:** Scatter plot of first 2 columns of data reveal the expected parabolic curve resulting from computing the square of values.

**Code Explanation:** `displaySquareAndSquareRoot2_0.nxc` iterates from 1 to 10 using a for-loop. Within this loop, the square and square root is also computed. To save any values to a file, one must first declare (Step 2) and initialize (Step 3) one. File data is stored as strings (i.e. a collection of alphanumeric characters). As such, string versions of any computation are needed and the strcat function is used (Steps 6 and 7) along with the file writing function created in Step 4. After computations are finished (i.e. for-loop terminates), the file should be closed (Step 9) using the function created in Step 5.

Steps 10 and 11 show the instructions for using NXT Explorer within the BricxCC IDE to export any files saved on the Brick's flash memory, to one's PC.

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

1-1 Iterate integers from -10 to +10 incrementally by 1. Compute the square and cube and save to a file named "squareAndCube.csv". Export the data file and plot the curves in Excel.