Stay connected through the *NI LabWindows™/CVI Community* at http://www.ni.com/lwcvi/community/
Johannes Turner
LabWindows/CVI R&D, National Instruments

Dane Stull
Product Support Engineer, National Instruments
#OurGiantsAreFemale

- **Gladys West**
- Contributed to Mathematical model of the shape of Earth
- Mathematician who worked on Satellite geodesy
- Technology eventually paved way for GPS
- Inducted into the Air Force Hall of Fame
- bit.ly/OurGiantsAreFemale
What’s New in LabWindows™/CVI 2019

Get up close with the latest features from LabWindows/CVI 2019 to learn how you can leverage the new source editor productivity tools to write code more efficiently and find out more about how you can easily deploy your applications using packages. Also, see firsthand how you can reduce time to troubleshoot bugs by using the debugging enhancements from LabWindows/CVI.
Here’s what will be covered…

1. New Source Editor Productivity Tools
2. Package Distribution
3. Advanced Debugging Tools
4. Other Notable Features
5. Summary
6. Q&A
New Source Editor Productivity Tools
Better & improved source editor

- Respond to user feature requests
- Align with industry standards by providing better development tooling
- Major step towards us supporting UTF-8 encoding in a future release
- Modern look & feel
Reuse code more easily

Edit code faster

View code better

View build diagnostics directly in your code
Easier coding with reusable snippets

- **Code Snippets** replaces the previous **Edit » Insert Construct** functionality, which didn’t offer user customization
- **On-the-fly integration** of **Code Snippets** with **Show Completions**
- As-you-type functionality to **Show Completions**

![Insert Snippet: if](image)  

*Added in 2019*
Demo

Customizable Code Snippets
and “as-you-type” Show Completions
Code faster with new line operations

- Multi-line edit
- Transpose, move, or duplicate lines
- Remove trailing whitespaces
- Line modification indicators

Added in 2019

```c
int CVICALLBACK PlotShape(int panel, int control, int event, void *callbackData, int eventData1, int eventData2)
{
    int color;
    int color2;
    int shape;
    int fill;
    /* The PlotShape function gets the values of the color, shape, and */
    /* fill controls, and plots shapes on the graph control based on */
    /* the user inputs. This function uses the LabWindows/CVI plotting */
    /* functions for displaying shapes, points, and text on a graph */
    /* control. */
    /* */
    //***************************************************************************

```
Code faster with new line operations

- Effortlessly comment/uncomment code using new keyboard shortcuts:
  - Ctrl+F3 to comment
  - Ctrl+F4 to uncomment

```c
/** Generate some random data first **/

for (i=0; i<100; i++)
{
    datapoints[i] = rand() * 100.0 / (double)RAND_MAX
}

if ((handle = LoadPanelEx (0, "graphs.uir", PANEL, __CVIUserHInst)) < 0)
    return -1;
DisplayPanel (handle);
RunUserInterface ();
DiscardPanel (handle);

return 0;
```
Improved code viewing

- Zooming
- Added support for non-monospaced fonts
- Display whitespaces
- Enable word wrap

```c
int handle;
double datapoints[100];

int main (int argc, char *argv[])
{
    int i;

    if (InitCVRTE (0, argv, 0) == 0) /* Needed if linking in external cor */
    return -1; /* out of m 150% */

    /** Generate some random data first **/

    for (i=0; i<100; i++)
    {
        datapoints[i] = rand() * 100.0 / (double)RAND_MAX;
    }
```
Improved code viewing cont.

- Indent guidelines
- Current line highlighting
- Collapsible regions improvements

```c
int main (int argc, char *argv[])
{
    int i;

    if (InitCRTIE (0, argv, 0) == 0) // Needed if linking in external compiler; harmless otherwise */
        return -1; // out of memory */

    /*** Generate some random data first ***/
    for (i=0; i<100; i++)
    {
        datapoints[i] = rand() ^ 100.0 / (double)RAND_MAX
    }

    if ((handle = LoadPanelEx (0, "graphs.ui", PANEL, CVIUserInst)) < 0)
        return -1;
    DisplayPanel (handle);
    RunUserInterface ();
    DiscardPanel (handle);
    return 0;
}
```

---

*Added in 2019*
Improved code navigation

- Find symbols easier with **Highlighting**:
  - Semantic highlighting
  - Word highlighting

Added in 2017
View build diagnostics in your code

- **Build Diagnostics Indicators** are shown in the gutter to the left of line numbers.
- Provides build information from the Build Output window at the position where the error/warning has occurred.

Added in 2019
Package Distribution
Package Overview

- A Package is an installer technology to deploy files to a target computer
- The NI Packages have the .nipkg extension
- Packages contain:
  - Installation Files (Payload)
  - Metadata
  - Dependencies
- Package Feeds group multiple packages and their dependencies with a manifest containing info
- Easily Distributed with an Installer or through NI Package Manager
- Enables componentized distribution of software
- More information at bit.ly/nipkg
Package Distributions

- LabWindows/CVI 2019 introduces the ability to distribute applications as NI Packages
- New Distribution types
  - Packages
  - Package Installer Patch
- Easily distribute to systems using NI Package Manager
Configuring Packages

- Configure your package using the Edit Package Window
- Select the Output Type
  - Single Package
  - Repository
  - Package Installer
- Set metadata options such as version, description, and maintainer
- Select Dependencies on other Packages
- Configure other advanced settings
Demo
Package Distribution
Advanced Debugging Tools
Debugging complex issues can be difficult

Another effective technique is to explain your code to someone else. This will often cause you to explain the bug to yourself. Sometimes it takes no more than a few sentences, followed by an embarrassed "Never mind, I see what's wrong. Sorry to bother you." This works remarkably well; you can even use non-programmers as listeners. One university computer center kept a teddy bear near the help desk. Students with mysterious bugs were required to explain them to the bear before they could speak to a human counselor.

– Brian Kernighan, Computer Scientist
Debugging complex issues can be difficult

1. **Probing values** of variables at different points during the execution of the application requires frequent interaction with the Watch Window, Array Display Window, etc.

2. **Inserting printfs** introduces the risk of leaving unwanted code in your application

3. **Modifying source code** or **using standard breakpoints** can hinder you from reproducing issues, particularly in multi-threaded applications
An event-based approach to breakpoints

- We’ve introduced the concept of debugging actions/events, that the debugger performs when a breakpoint is hit.

- Use **Tracepoints** and **Thread-Specific Breakpoints** to:
  - Log variables, call stack context, etc.
  - Change the flow of execution of your application

- All of this, *without having to recompile your project.*

Added in 2017
An event-based approach to breakpoints

- Tracepoints can log:
  - $FUNCTION$: Gets the name of the current function.
  - $CALLER$: Gets the name of the caller function.
  - $PID$: Gets the value of the current process ID, as decimal integer.
  - $PNAME$: Gets the name of the current process image.
  - $TID$: Gets the value of the current thread ID, as decimal integer.
  - $CALLSTACK$: Gets the current call stack of the current frame.
  - $TIMESTAMP$: Gets the current time stack.
  - $LINE$: Gets the value of the current source code line in the current file, as decimal integer.
  - $FILE$: Gets the name of the current source code file.
  - {expression}
An event-based approach to breakpoints

"The most effective debugging is the debugging you do not have to do.

– Peter Ilberg, LabWindows/CVI R&D
Demo
Tracepoints and Thread-Specific Breakpoints
Other Notable Features
UI Events Logger

- When using Operate Mode you can use the UI Events Logger to Filter, View, and Save User Events
- Track Exact Steps when you test a user interface
- You can specify which events to record and which to swallow when interacting
- The log panel displays relevant event information and allows you to save the log to a file
Configure Compiler Warnings as Errors

- LabWindows/CVI 2019 allows you to configure Compiler Warnings as Errors.
- This can be configured on a per warning basis with the inclusion of an Error column.
- Compiler warnings can be configured on a per warning basis and by warning level.
- Now there is a column to check for Error in order to configure you’re the compiler to show the warning as an error.

Added in 2019
**Compiler Warnings**

```c
#include "devide.h" /* Needed if linking in external compiler; harmless otherwise */
#include "userint.h"
#include "cassi_i.h"
#include "graph.h"

int handle;
double datapoints[100];

//Zoom in to view main
int main (int argc, char *argv[])
{
    int i;

    if (argc < 1)
        return -1; /* Needed if linking in external compiler; harmless otherwise */
    return -1; /* Out of memory */

    /** Generate some random data first **/

    if ((handle = LoadPanelEx (0, "graphs.wir", PANEL, __CVUserEx(int)) < 0)
        return -1;
    DisplayPanel (handle);
    HandlePanel (handle);
    DiscardPanel (handle);

    return 0;
}

/**************************************************************/
/** The PlotData function gets the values of the color and plotstyle */
/** controls, and displays the random datapoints array on the graph */
/** control based on the user inputs. This function is called by the */
/** Plot Data button, the Line Style Ring Control, and the Trace Color */
/** ring control. */
/**************************************************************/

int CALLBACK PlotData(int panel, int control, int event, void *callbackData, int eventData1, int eventData2)
{
    int color, style;

    if (event == EVENT_SELECT) {
        GetCtrlVal (handle, PANEL_TRACECOLOR, &color);
        GetCtrlVal (handle, PANEL_STYLE, &style);
        DeletedGraphPlot (handle, PANEL_GRAPH, 1, VAL_DELAYED_DRAW);
        Plotxy (handle, PANEL_GRAPH, datapoints, 100, VAL_DOUBLE, style,
                VAL_EMPTY_SQUARE, VAL_DOUBLE, 1, color);
    }
```
TDMS Improvements

- **TDMS_FlushFile** to force any data in the buffer to be written to the .tdms file
- **In Memory TDMS** Functions allow you to open, close, and edit TDMS files in memory
- Addition of new **Virtual Properties**
  - TDMS_NI_DISK_CACHE_SIZE
  - TDMS_NI_DISK_GAP_SIZE
  - TDMS_NI_MINIMUM_BUFFER_SIZE
  - TDMS_NI_IN_MEMORY_WRITE_BLOCK_SIZE
  - For channels and channel groups

Added in 2019
Improved TestStand and LabWindows/CVI Integration

- Easily include Type Information from source code
- Edit Code option now will open the source code directly if available
- Automatically create types from structs and enums
- Specify pointer/array in code for easier imports

Added in 2019
Demo
TestStand and LabWindows/CVI Integration
Key Takeaways

- Deliver new features to help improve productivity
- Enable delivery of more features in the future
- Major step towards UTF-8 support
Stay connected through the *NI LabWindows™/CVI Community* at

http://www.ni.com/lwcvi/community/
Before you go, take the survey.
Download and Login to the NIWeek Mobile App

VIEW WEEKLY SCHEDULE
FIND YOUR SESSION
BUILD YOUR SCHEDULE

More Information at ni.com/niweek
Stay Connected During and After NIWeek

ni.com/niweekcommunity
facebook.com/NationalInstruments
twitter.com/niglobal
youtube.com/nationalinstruments
Questions?
Thank you!