**Introduction**

A broken VI cannot run and this situation is indicated by the RUN button appearing as a broken arrow. When you are creating a new VI and you have not completed wiring some objects in the block diagram the RUN button will show that it is (still) broken, and only after you have completed all wiring correctly you will be able to RUN the program. Broken wires are indicated by dashed black line or dashed line with a red X mark, and broken wires are usually related to wiring of objects that require different data types (the data types are incompatible).

To find out why a VI is broken, click the RUN button and select **SHOW ERROR LIST**. There are two kinds of information displayed in the error list: WARNINGS, and ERRORS. The shortcut key for removing all broken wires is **CRTL-B** (control B). To remove broken wire on at a time, select the wire and press the **DELETE** key.

The following are things to do in debugging: **SINGLE STEPPING**, **EXECUTION HIGHLIGHTING**, **PAUSE** and **RESUME** execution, and **PROBING**.

Debugging tools are accessible in the Block Diagram window and they are shown in a TOOL BAR at the upper side of the block diagram window.

![Debugging Tools](image)

- **Single Step – Step Into**: this button executes the first step of a subVI and then pause at the next step. Click this button again to execute the next step and pause, etc.

- **Single Step – Step Over**: this button a subVI or a portion of a program then pause at the next node.

- **Single Step – Step Out**: this button finishes the execution of the current block of program then pause.

- **Execution Highlighting**: this button highlights the path (wire) that is currently being executed and displays the immediate value of it (like an animation).

- **Break Point**: is a **PAUSE** point in the program that you put on a wire. When the program executes and encounter it the program will pause, allowing you to examine various things like the immediate value in wires, etc.

- **Probe**: is a tag you put on a wire to show the immediate value of data in the wire during execution. Usually you do this in conjunction with single stepping.

This exercise is to give you a hands-on experience with some of the fundamental ideas of LabVIEW by writing a more complex LabVIEW program from scratch. Some of the steps you need to take in order to make things look or do the way you want them will be left up to you to find out.
What to Submit

You will answer a few questions in the end of this handout, put the questions and your answers on the Front Panel and turn in the PDF printout of your VI online including all hidden frames and documentation.

What You Will Do

You will develop a 4-bit Binary Number Conversion program using CASE structures, controls and indicators. Keep your VI organized and clean use different colors, size, shapes, etc.

Document everything with Help Tips and Documentation (right-click the object and select DESCRIPTION and TIP…) so that you can go back to this program in the future and understand what or how things are done. Use Decoration Label in Each Case structure to describe what is done in each case.

The principle of converting a Boolean (base-2) number into a decimal (base-10) value is simple: starting from the right most digit (in Boolean, the word BIT is more appropriate than the word DIGIT, and the positions of the bits are from 0 to n-1, where n is the number of bits, and the position of the right most bit is 0), the decimal (equivalent) value of a binary bit in position n is b * 2^n, where b is the state of the bit (1 or 0), and n is the bit position starting from position 0 (the right-most bit). For example: if the first bit (also called LSB /least significant bit), the right most bit position, has a state of 1 then the decimal value is = 1 because 1 x 2^0 is 1. The next bit to the left of it will be either 0 or 1 x 2^1, etc. The sum of all the values is the decimal equivalent of that Boolean number. For example: a 4-bit Boolean/binary number is 1001, the decimal value will be: starting from LSB to MSB: 2^0 + 0 + 0 + 2^3 = 9. A shortcut way of doing this is by saying that the “weight” of each bit position from LSB to MSB (most significant bit) is 1, 2, 4, 8, 16, 32, 64, 128,… (Multiply previous weight by 2) and the decimal value is the sum of the weights of all bits that are TRUE.

Procedure

1. Create a blank VI with your name, date, and the title of this exercise printed on the Front Panel and on the Block Diagram (Use Decoration objects). Make your VI cosmetically and functionally nice!!
2. Put four Boolean Controls (the controls that can only have True and False values, like a push button or any kind of a switch): On the Front Panel, bring in 4 BOOLEAN controls that look like a vertical toggle switch, name each control BIT0_LSB, BIT1, Bit2, and Bit3_MSB. Feel free to resize and use colors as you see fit. Position the Boolean controls so that Bit0_LSB is on the right of Bit1, Bit1 is on the right of Bit2, etc. Align them horizontally using the ALIGN tool so that the top edge of the controls are aligned with each other. You can move the labels of each object around to make your front panel look pretty.
3. Put four Boolean Indicators:
   On the Front Panel, bring in 4 BOOLEAN indicators that look like a round of other shape LED, name each indicator D0_LSB, D1, D2, and D3_MSB. Feel free to resize and use colors as you see fit. Position the Boolean controls so that D0_LSB is on the right of D1, D1 is on the right of D2, etc. Align them horizontally using the ALIGN tool so that the
Making Simple LabVIEW Program With Data Types and Using Debugging Tools

top edge of the controls are aligned with each other and align each LED so that it is vertically aligned with the corresponding Boolean control.

4. Put One Numeric Indicator in the Front Panel and name is **Decimal Value**.
5. Organize and tidy up your block diagram: In your block diagram organize the objects that you have brought in so far so that the Boolean controls are aligned vertically with Bit3_MSB being the top most object and the Bit0_LSB control being the bottom most object. Also position the corresponding Boolean indicator to the right of the control of the same bit position. Use the alignment tool to align the indicators so that they are aligned horizontally to the control object of corresponding bit position. You can also make the objects on the block diagram look smaller by un-checking the VIEW AS ICON item (right-click on the object), to make your diagram less cluttered.

6. Wire (write) your program: Wire each control to an indicator of the same bit position (Bit0_LSB to D0_LSB, etc.) and test your program if you wish by changing the state of each BOOLEAN control objects on the front panel so that you can see what you have done so far (you will need to click the RUN CONTINUOUSLY button in this case).

7. Adding functionality that converts the binary data into decimal values: In your block diagram, put four case structures one for each Boolean control. Make the controls and indicators located outside the case structures.

8. Wire the Case Select input terminals of the case structures of step 6 to the corresponding Boolean controls.

9. Inside the case structure for Bit3 (msb), in the TRUE case compute 1*2^3, and in the FALSE case put a constant 0

10. Repeat steps 8 for Bit2, Bit1, and Bit0_LSB and change the TRUE case content to 1*2^2 for Bit2, 1*2^1 for Bit1, and 1 for Bit0. Your block diagram should look something as follows:

![Block Diagram](image)

11. Complete the block diagram by wiring the output of each case into a function that adds them and display the result in the numeric indicator (you should already know how to do this). You can use Formula Node or just use the Add functions.
Making Simple LabVIEW Program With Data Types and Using Debugging Tools

12. Test your program by running it with RUN Continuously, change the state of each of the BOOLEAN controls and look at what the LED and DECIMAL indicators show then answer question 4 in the last page of this lab handout.

13. Debugging - Probing: stop your program. Right – click in the wire of Bit 3 and select PROBE. A grey Probe is put on the wire with a number on it and a popup window shows information about where that probe is and its current value. In the figure below, the probe is Probe [1] and it is on Bit 3 and its blue is “Not Executed”, and the Probe Display shows Bit 3 - False.

A probe is a debugging tool for showing the immediate value of data at different point of a program.

14. Instead of Clicking RUN or RUN Continuously, you click Start Single Step and look at your block diagram and see what’s happening, then click the Single Step button again and see what happens in your block diagram. Keep doing this until the control the output of which the probe is monitoring is blinking then click Single Step again to see the value seen by the probe.

15. The object that is blinking in the Single Step mode is the OBJECT TO BE EXECUTED NEXT. To see the immediate value of data when that object is executed you put a probe on the wire of that data path.

16. Stop your program. Right-click on the wire that connect to the Numeric Indicator and select Breakpoint -> Set Breakpoint. This will put a Read Dot on the wire which is called Break Point. You can put as many Breakpoints as necessary in debugging a program.

17. Click RUN (not Run Continuously), and see what happens to your block diagram. Your program stops at the break point and to continue you can either press the CONTINUE button of press the Single Step button.

Questions

1. What is the difference between a BOOLEAN data type and a NUMERIC data type?
2. What is the visual difference between a wire that carries BOOLEAN data and a wire that carries NUMERIC data?
3. What is a Probe, what does it do, and when does it show valid information?
4. Describe a scenario, which you might need to use Breakpoints instead of doing single stepping from the very beginning, and explain why probes are not useful when you just run the VI and not single stepping.