Lab01 Introduction to LabVIEW and Data Acquisition

Bill Hung
17508938
EE145M
Lab Time: 9-12pm Wednesday
Lab Partner: Chih-Chieh Wang (Dennis)

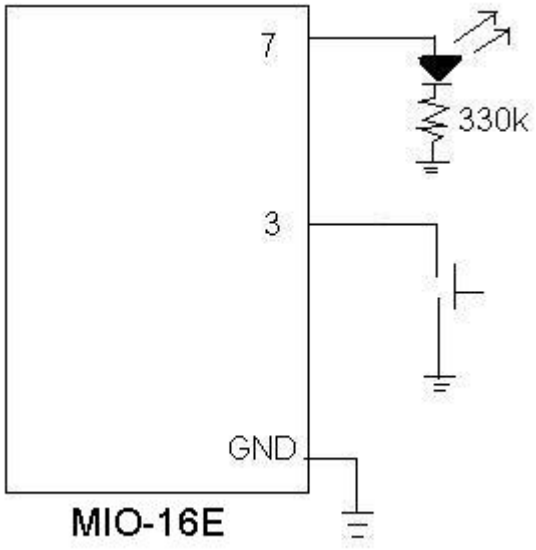
Aim

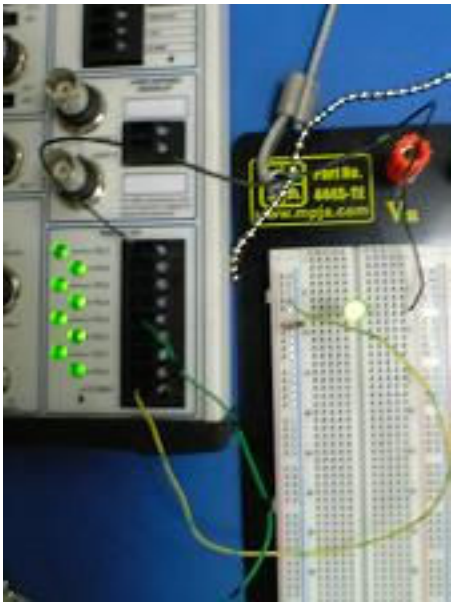
Convert a 16-bit signed integer to 8-bit, 16-bit, or 32-bit numbers, either signed or unsigned. The numbers are represented in decimal, binary, and hex. (Numbers.vi)

Built an interface with a LED through a LabVIEW program and a Data Acquisition Board (DAQ) MIO-16E. Turning the LED on in the external circuit will also turn the LabVEIW indicator on.

1. Setup

The software and equipment used in this lab.





The interface between the MIO-16E board and the circuit is shown above. The MIO-16E board is a data acquisition board connected to the computer. The input pin is pin 3 and the output pin is pin 7. When input pin 3 is connected to ground, the output pin 7 will be high, and current flows through the LED and light is emitted.

This is not the best setup for the experiment because pin 3 is left floating when the switch is open. A better way for the setup is to have a pull-up resistor to pull the potential of pin 3 to VDD when the switch is open.

2. Data summary

First, a LabVIEW program Numbers.vi is built (see the diagram at the end of the report), and a set of input numbers are converted to different representations.

2.1 Tabulate the response of the LabVIEW numbers. Explain your results.

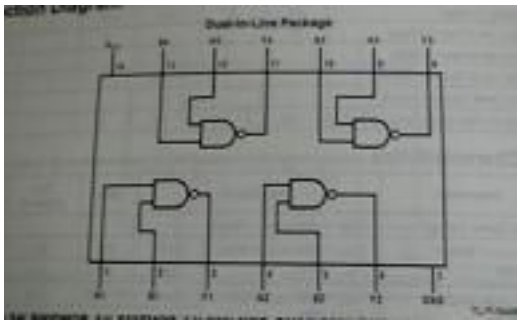
	signed 8b	signed 16b	signed 32b	unsigned 8b	unsigned 16b	unsigned 32b	hex 8b	hex 16b	hex 32b
0	0	0	0	0	0	0	00	0000	00000000
1	1	1	1	1	1	1	01	0001	00000001
2	2	2	2	2	2	2	02	0002	00000002
3	3	3	3	3	3	3	03	0003	00000003
4	4	4	4	4	4	4	04	0004	00000004
5	5	5	5	5	5	5	05	0005	00000005
125	125	125	125	125	125	125	7D	007D	0000007D
126	126	126	126	126	126	126	7E	007E	0000007E
127	127	127	127	127	127	127	7F	007F	0000007F
128	128	128	128	128	128	128	80	0080	00000080
129	129	129	129	129	129	129	81	0081	00000081
130	130	130	130	130	130	130	82	0082	00000082
-130	126	-130	-130	126	65406	4294967166	7E	FF7E	FFFFFF7E
-129	127	-129	-129	127	65407	4294967167	7F	FF7F	FFFFFF7F
-128	-128	-128	-128	128	65408	4294967168	80	FF80	FFFFFF80
-127	-127	-127	-127	129	65409	4294967169	81	FF81	FFFFFF81
-126	-126	-126	-126	130	65410	4294967170	82	FF82	FFFFFF82
-125	-125	-125	-125	131	65411	4294967171	83	FF83	FFFFFF83

For a signed 8-bit integer, the largest representable number is 127(0111 1111), and the smallest is -128 (1000 0000). Therefore, for -129 (if 9 bits, would be 1 0111 1111), the number representation as an 8-bit number is shown as positive 127 (0111 1111) because only the lowest 8 bits were obtained. More detail explanation is in section 3.1.

2.2 Tabulate the logic high and logic low level for the MIO-16E interface when running the InputOutput VI. What logic techno is used (CMOS? TTL?) and why?

MIO-16E

74F00 Logic High


74F00 NAND Gates (National Semiconductor)

	Logic Low (V)	Logic High (V)
MIO-16E	0.180	5.073
74F00	0.18	4.40
TTL (from textbook p. 9)	0.2	3.2

The technology used in the MIO-16E interface board is CMOS technology. According to the textbook by Stephen Derenzo on page 9, if the technology is TTL, the logic high should be around 3.2V. However, the logic high for the MIO-16E board is around 5V, which suggested the technology is CMOS.

Our group did a control experiment using 74F00 NAND gate chip (a CMOS technology) to generate a logic high, and the output of that is 4.4V. The result is similar to that of MIO-16E, because they are both CMOS technology.

3. Discussion

3.1 *Discuss the reasons why 0 and small positive integers have the same representation, whereas a negative number such as -129 has six different representations. Use examples to support your discussion.*

For a signed 8-bit integer, the largest representable number is 127 (0111 1111), and the smallest is -128 (1000 0000). Therefore, for -129 (if 9 bits, would be 1 0111 1111), the number representation as an 8-bit number is shown as positive 127 (0111 1111) because only the lowest 8 bits were obtained from a 16-bit signed integer. For a signed negative 16-bit integer (like -129), when converted to a signed 16-bit integer or 32-bit integer the number appears to be the right number (-129) because no bit needed to be cut.

When converting to an unsigned 8-bit integer, the sign bit is truncated like a signed integer -129. Therefore the number appeared as 127(0111 1111). When 16-bit signed integer is converted to an unsigned 16-bit integer or an unsigned 32-bit integer, the left-most bit is sign extended. For example, if the number is negative (like -129) the extra bit will be filled with 1 (like 111111… 0111 1111). So the 16-bit number -129 appeared to be an unsigned numbers 65407 and 4294967167.

When the signed 16-bit number is converted into a 8-bit Hex, the negative number -129 is truncated to be 127(0111 1111) and appeared as 7F. Similar to unsigned numbers, sign extension happens to 16-bit and 32-bit hex numbers as well. Therefore, -129 appeared as FF7F and FFFFFF7F in hex.

3.2 *What is the main difference between the MIO-16E and the PCI-7831R FPGA board? We will need a high degree of timing accuracy of the FPGA for the DSP labs later on. Why is the FPGA preferred over the DAQ board in timing sensitive application?*

The MIO-16E board is simply a Data Acquisition (DAQ) board, without any timing unit built-in. In other words, when the MIO-16E board receives an instruction to fetch data, the data will be read. The computer is responsible for the timing control.

The PCI-7831R FPGA board handles the timing within the board. Since the timing is performed by a more low-level unit on the PCI-78331R FPGA board, the timing will be more accurate.

The FPGA is preferred over the DAQ board because during high frequency sampling, accurate timing is necessary. The DAQ board timing would be less accurate because high level computer applications control the timing. High level application can be interrupted by other applications, and the clock is less accurate because of longer wire length from the computer to the board to deliver the clock signal.

4. Questions

4.1 *Given an 8-bit binary number where all bits are one, what is the numerical value when it is interpreted as:*

(a) 2's complement integer
(b) an unsigned integer
(c) hexadecimal

(a) -1
(b) 255
(c) FF

Given an 16-bit binary number where all bits are one, what is the numerical value when it is interpreted as:

(a) 2's complement integer
(b) an unsigned integer
(c) hexadecimal

(a) -1
(b) 65535
(c) FFFF

4.2 *In 2's complement arithmetic, the sign of a number is changed (a -> -a) or (-a -> a) by taking the complement of each bit and adding1. Using the 8-bit binary and 2's complement representations given in Table 1.1, show explicitly that:*

(a) The number 0 and the 2's complement of 0 have the same bit pattern (this is expected, since 0=-0).

(b) The number -127 and the 2's complement of 127 have the same bit pattern (this is expected, since -127=-127) .

(c) The number -128 and the 2's complement of -128 have the same bit pattern (this is not expected, since -128≠ -(-128)).

(a) 0000 0000, flip 0->1
1111 1111, +1
0000 0000, which is the same as the original bit pattern

(b) start with 127,
0111 1111, flip bits
1000 0000, add 1
1000 0001, which is the same as the bit pattern of -127

(c) start with -128
1000 0000, flip bits
0111 1111, add 1
1000 0000, which is the same as the bit pattern of -128

This result is not expected, because the two's complement of -128 does not equal to -128.

4.3 *Using the 8-bit binary and 2's complement representations given in Table 1.1, show explicitly that for the numbers 1, -1, 16, and 127, taking the 2's complement twice results in the original number: a = (-(-a))*.

Number 1

1st Pass	0000 0001	2nd Pass	1111 1111
	1111 1110		0000 0000
	1111 1111		0000 0001

Number -1

1st Pass	1111 1111	2nd Pass	0000 0001
	0000 0000		1111 1110
	0000 0001		1111 1111

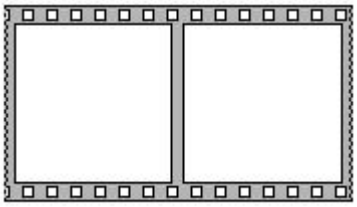
Number 16

1st Pass	0001 0000	2nd Pass	1111 0000
	1110 1111		0000 1111
	1111 0000		0001 0000

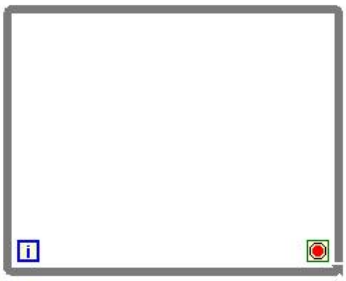
Number 127

1st Pass	0111 1111	2nd Pass	1000 0001
	1000 0000		0111 1110
	1000 0001		0111 1111

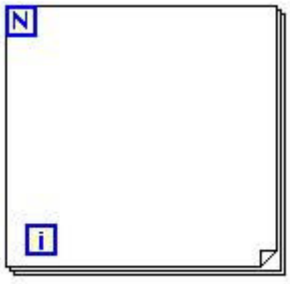
4.4 *Explain the function of the Sequence Frame, the While Loop and the For Loop.*



LabVIEW is a dataflow language. Multiple instructions can be executed in parallel, so in order to execute commands in sequence, a sequence frame is needed. The sequence frame executes operations a frame at a time. The operation within a frame is executed whenever the input data is available, and after all operations within a sequence frame are completed, the next sequence frame will be executed. This sequence frame structure guarantees operations will execute step-by-step.



When the while-loop is set to “Stop If True”, like the diagram above, the while-loop executes the operations within the loop until the condition statement is no longer true. In other words, if the condition statement connected to the red dot is false, the while loop will continues to execute the operations within the while loop again and again. The while loop will stop when the condition statement is true. When the while loop is set as “Continue If True”, then the operations in the while-loop will be executed if the condition is true.



The for-loop executes the operation within the for-loop for a fix number of times. The number of iteration is determined by the number connected to the ‘N’ at the corner. The constant for the iteration number ‘N’ should be placed outside the for-loop. For example, if the for-loop was instructed to be executed 10 times, the operations within the for-loop will be executed 10 times. The ‘i’ indicates the current number of iteration, such as round 1, round 2, round 3…

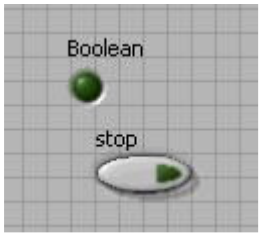
4.5 *Explain the purpose of the inner while loop in Figure 2 (The one inside the sequence frame). Why must the pushbutton control be configured to “Latch when pressed”?*

The inner while-loop checks if the button is being press or not continuously. When the button is press, only one pulse is desired. Since the human response time is much slower than the computer operating speed. If the button is not configured to be “Latch when pressed” the number may increment very quickly, and very soon the number will increment from the smallest number to the largest number when the button is only pressed once.

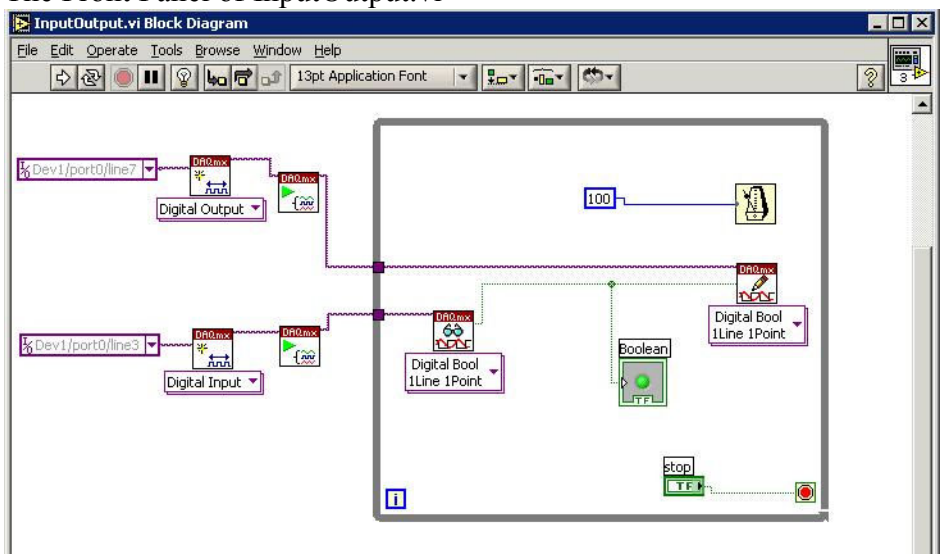
The “Latch when pressed” button makes sure the number is only incremented by one each time the button is pressed.

5. Laboratory Data Sheets

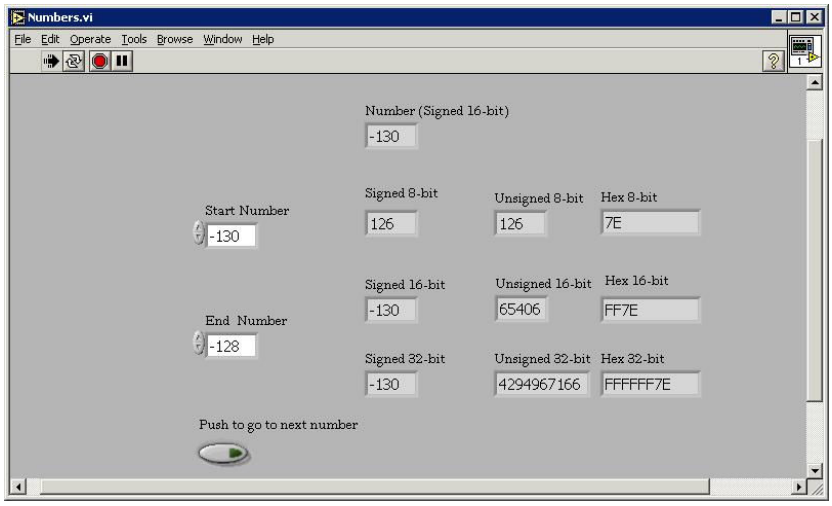
i. InputOutput.vi



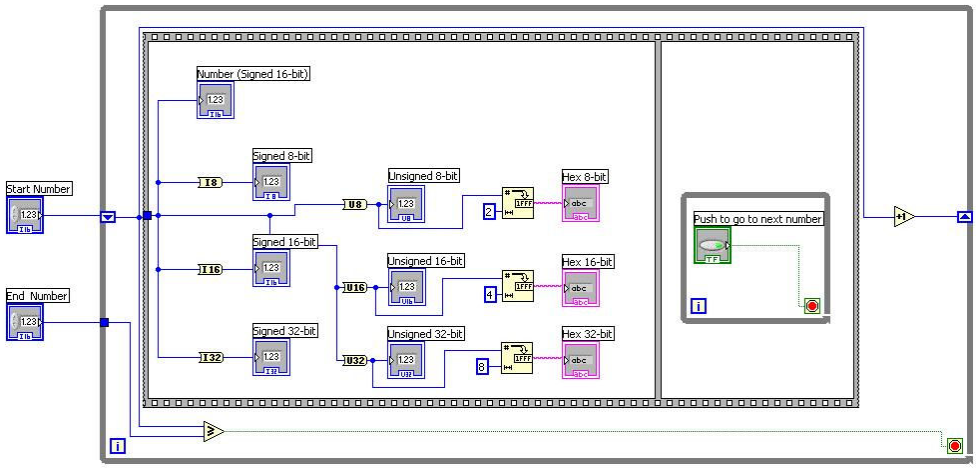
The Front Panel of InputOutput.vi



The Block Diagram of InputOutput.vi



Number.vi Front Panel



Number.vi Block Diagram