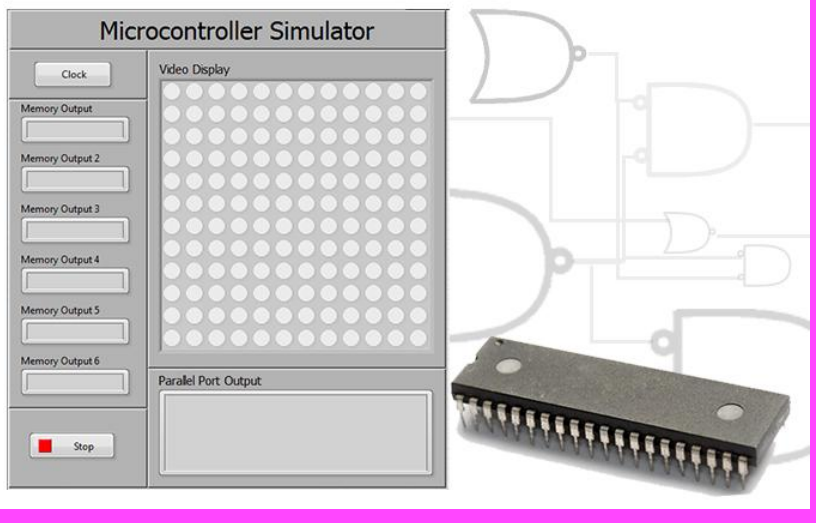
Microcontroller Simulator

Core Concept Instructor Set



Microcontroller example

Materials:

LabVIEW

Learn It!

It is often necessary to convert data from one form to another. Information is typically sent and received using binary. That information would then be interpreted and converted to different data types like Boolean or ASCII to be used in word processors or to make decisions in various systems. Data is converted to different forms based on a standard or algorithm that both the message sender and receiver agree to use.

Encoders and decoders are specific types of data converters, which can be used in microcontrollers to process instructions and communicate with peripheral hardware. Microcontrollers are a central part of electronics that span hobby projects to industrial condition monitoring systems.

In this module you will explore how data conversion can be used in a simulated microcontroller. You will design a decoder to convert address data into device selections and an encoder to convert ASCII data into pixels for a visual display.

"Microcontrollers are a central part of electronics that span hobby projects to industrial condition monitoring systems."

Build It!

Microcontrollers take in instructions, interpret them and interact with different devices based on the instructions. The microcontroller has a main data bus that is connected to all peripheral hardware were commands and data can be transferred between the Central Processing Unit (CPU) and things like RAM and input and output ports. Every microcontroller is made to handle a set of instructions, this simulated version can only handle three, the transfer of data to a location in RAM memory the transfer of data to the simulated VGA port (displayed on the front panel) and the command to send data from RAM to the simulated parallel output port (displayed on the front panel). The first portion of this module is to build a decoder that will choose which simulated hardware device to enable in your simulated microcontroller based on three bits of address data provided by the CPU.

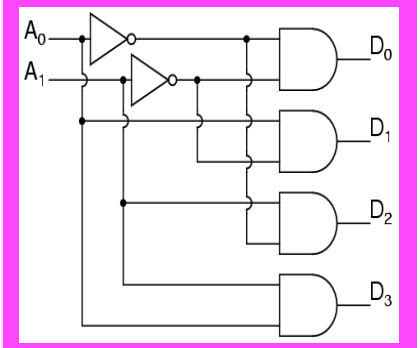


Figure 1 2-bit Decoder logic

Decoder Program

Step 1: Open the Decoder_Student.vi from the "Student" folder found in the "Microcontroller Simulation" Folder and switch to the block diagram.

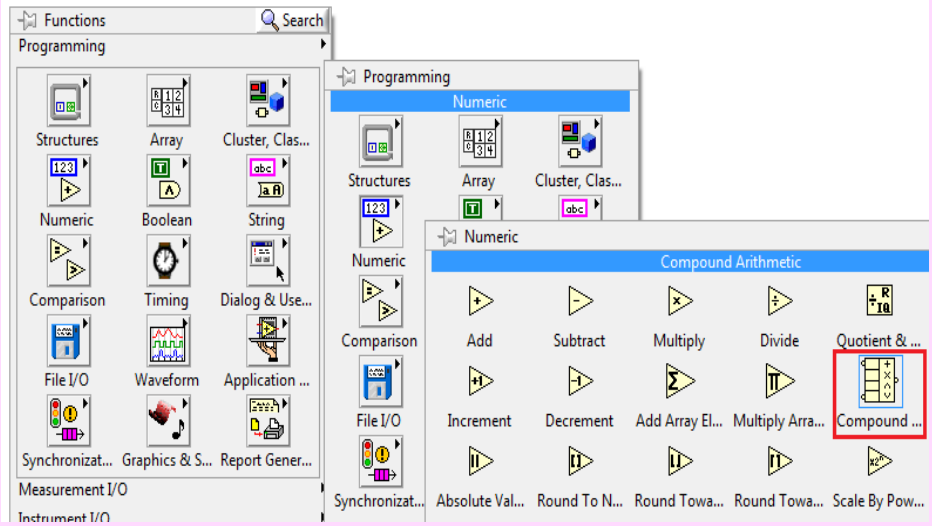


Figure 2 Navigate to Compound Arithmetic function

Place the "Compound Arithmetic" function by right clicking and navigating to **Programming>>Numeric>>Compound Arithmetic.** Make seven more Compound Arithmetic functions either by placing them or copying and pasting them on the block diagram.

Step 2: Place a "Not" function on the block diagram by right clicking and navigating to **Programming>>Boolean>>Not**. Make two more Not functions by either placing them or copying and pasting them on the block diagram.

Step 3: The decoder you are building is giving a single output for each combination of the three Boolean inputs. This is acting as a binary selector for a 3-bit input. There are eight possible combinations for the three bits and those combinations are made using three input AND gates. Hover over each of the Compound Arithmetic functions until blue squares appear on the top and bottom of the function. Click and drag down on the function to expand the inputs to three.

Guiding Questions:

- Research and explain how the OR function works. Explain how you can get the functionality of the AND function using only OR and NOT functions.
- Research and explain how to convert decimal to binary and hexadecimal. Discuss how you can mathematically convert between all of them.
- Find other examples of converting data from one form to another. Find a way to implement your address decoder using a different conversion method.

Step 4: Click on the front end of the Compound Arithmetic function and navigate to **Change Mode>>AND**, do this for the remaining seven. Follow the truth equations provided on the block diagram along with Figure 4 for reference to create your 3-bit selector. Once the wiring is completed, plug in test values for the three Boolean inputs to verify your outputs match the truth equations.

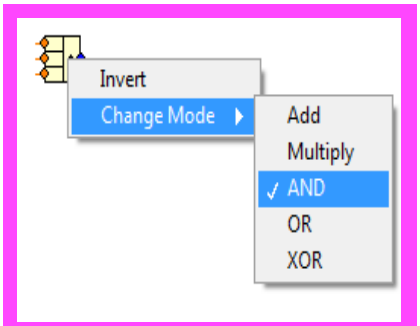


Figure 3 Choosing the AND functionality

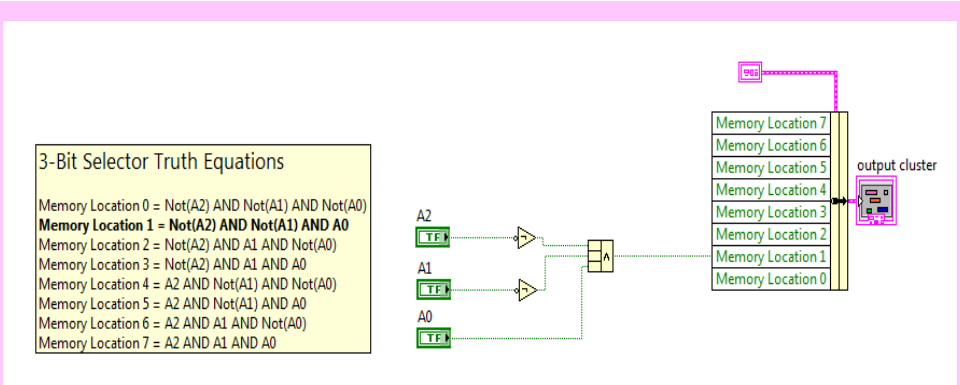


Figure 4 Example code for Memory Location 1

Step 5: Open the Microcontroller Simulator.vi from the "Student" folder and run it. Choose an instruction file to run from the "Instruction Set" folder, the files indicate what should appear at the Parallel Port Output indicator on the front panel once you have clocked through the program. Click on the button labeled "Clock" multiple times and watch as the "Memory Output" indicators update. The code will automatically end after the final instruction has executed, if the decoder has been implemented correctly the Parallel Port Output will display the word indicated by the instruction file.

Guiding Questions:

- How would you translate color in each pixel? Can the VGA encoder be updated to display in full color?
- How can you improve the speed of a VGA display like this?
- Research the architecture of a microcontroller, examine how other hardware devices are connected to the system. Find where the encoders and decoders are used.

VGA Encoder Program

The previous section discussed one way to convert data into a different form; this section will touch on another way. Television screens show pictures based on arrays of pixel data. Three numbers between 0 and 255 are given that represent the amount of Red Green and Blue that should be used to produce the color of the respective pixel. In this section you will create a picture display that will convert the string character from the CPU into a visual representation of that character.

Step 1: Open the VGA Display_Student.vi and switch to the block diagram. This code either allows its output to be updated by the data input when the VGA enable Boolean goes high or maintains its output. Switch to the True case.

Step 2: Place a new case structure by right clicking and navigating to **Programming>>Structures>>While Loop**. This will turn your cursor into the icon that indicates you can select the area you would like to have occupied by the case structure. Click and drag to select the area inside of the True case of the current case structure. Make sure you include the array that is already there.

Step 3: Wire the String data into the case selector of the case structure you just created. Your block diagram should now look like Figure 5.

Step 4: Double click on the Selector label of your case structure to edit the true parameter for this case. Change the text to "A". This will make it so that when this structure sees that character it will go to this case. Click on the lower right hand corner and drag on the Boolean array constant to expand it until you have a 12x12 box of Booleans.

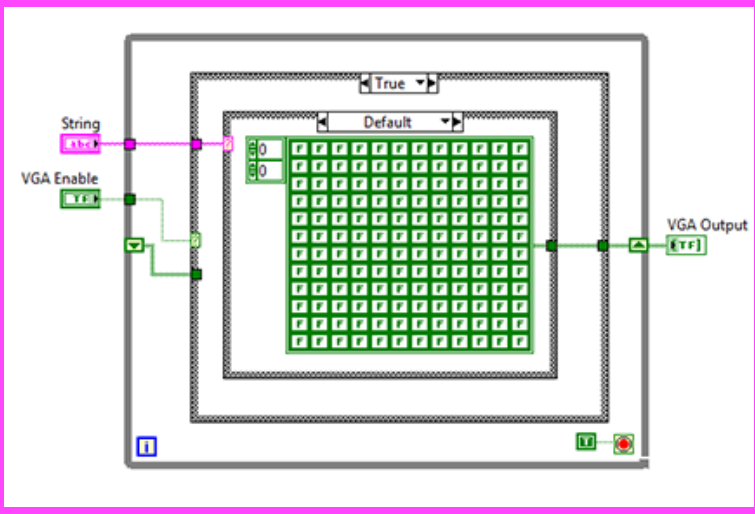


Figure 5 VGA Port Block Diagram

Step 5: Click on the Boolean constants to set them to true in the shape of the letter "A". This 12x12 square represents the video display of characters from you CPU. There are 144 pixels In this simulated display but the only data they have is either on or off.

Step 6: Create another case by right clicking on the Selector Label and choosing "Add Case After". Copy over the Boolean array from the "A" case and reselect the Boolean constants to form the letter "B". Continue this process for the remainder of the letters in the alphabet. You may create extra characters but none of the instruction files require it.

Step 7: Once you have completed all 26 characters you may save your vi and try rerunning the Microcontroller.vi. Now that the VGA port has been updated you should be able to step through entire instruction sets and get the desired data and VGA output. Verify your microcontroller works by running through all of the instruction sets.

Expand it!

- Add code that allow you to treat Boolean data differently, currently the data bus only passes string data but program it so that when a Boolean value appears then memory location 6 gets updated with a question mark.
- Design a way to increase the number of addresses this microcontroller can access. Use this to take in and process larger words.

Research It!

Circuit Implementation using decoders
http://www.ee.surrey.ac.uk/Projects/Labview/Sequential/Course/01-Decoder/decoder1.htm
Decoder
http://www.allaboutcircuits.com/vol_4/chpt_9/4.html
Encoders/Decoders
http://www.cs.umd.edu/class/sum2003/cmsc311/Notes/Comb/encoder.html

Binary, Decimal and Hexadecimal Numbers

http://www.mathsisfun.com/binary-decimal-hexadecimal.html