4 Bit LED Adder With Carry
By: Alpha

Overview: As the name expresses, this is a four bit LED adder which preserves its carry. There are three groups of LEDs (*RESULT*, *FIRST NUMBER* and *SECOND NUMBER*) to show the numbers and three switches. Two of them (*NUMBER-1* and *NUMBER-2*) are to adjust the input numbers and the other one (*RESULT*) to tell the program to show the result.

Circuits and Registers Structure:

- *FIRST NUMBER*'s LEDs are connected with register *B* (in increasing order of the LED numbers and Register pins).
- *SECOND NUMBER*'s LEDs are connected with register *C* (in increasing order of the LED numbers and Register pins).
- *RESULT LED*s are connected with register *D* (in increasing order of the LED numbers and Register pins) (in this case, there are five LEDs. Rightmost one is the carry)
- Leftmost LEDs are representing the LSB for each group.
- The input is being collected using the fifth Pin for each register.
- One common ground for all the devices.

Code Structure:

- Code mainly consist the main function and two libraries for AVR.
- In the main function, at the beginning, all three registers' pins are being initialised as per the needs.
- Next few lines consist few variables to output the numbers and result.
- Then, there is a while loop which keeps the program alive whole time.
- Inside the while loop, there are mainly three *if-else* condition blocks - each of them checks which switch is being pressed and executes the appropriate lines.

Variables:

- There are two 8-bit Integers and one 16-bit integer. The 8-bit integers, *num1* and *num2* are being used to display and keep track of the input numbers. The 16-bit Integer, namely *res*, is used to store the result of the addition operation. Initially all of them are set to zero.

Subroutines:

- During the programme's main loop, it continuously checks which button is being pressed.
- If the button is pressed to adjust any number of the two, it first waits a few milliseconds to prevent taking one switch press as multiple presses. In the next line it makes sure the same button is pressed. If the condition meets, It increases the appropriate number by one. It also checks if the number crossed fifteen, as we are working only with 4-bits. If the number is greater than fifteen, the next line makes it zero again. In the last line of the these *if* blocks, we just load respective register with the new number.
- The first three lines does the same thing like the previous two *if* and *else if* blocks as described above. The main task of this block is the add the 8-bit Integers. As the *res* variable is 16-bit, it stores the addition result in it including the carry. The following line sets the *D* register with the result to display. The rest of the code re-initializes all the variables those are related to the *FIRST NUMBER* and the *SECOND NUMBER*..