

Kevin Wu
ECP1 Final Project

Description:

The point of my project is to flash LEDs in an interesting manner when the exact phrase “go” (followed by a carriage return) is entered into the internal UART port of the 8031. Also, whatever is typed will be echoed at the terminal. Therefore, my project consists of three basic parts. The first part is character input/output, which was the core of Homework #4. This allows the user to type anything into the 8031 UART port, and have it be echoed at the terminal when the ENTER key is pressed. The second part is character string recognition, where several checks are done to see if “go” and only “go” was entered. Lastly, the third part is the LED lighting. I have three LEDs lined up on the general purpose I/O pins of port P1, and the interesting manner in which they flash is essentially back and forth from left to right, over and over again, where each LED stays lit for about 500ms.

I use two timers, timer1 and timer0. I use timer1 for the serial port, and timer0 for timing the on and off states of the LEDs. The main functions are WaitLoop, CheckReturn, SendStr, ProcessChar, TurnOnLEDsMaybe, LEDsFTW, and wait500ms. WaitLoop polls for characters input into the UART port. When a character is received, CheckReturn checks to see if it is either a 'g' if it's the first character received, or if it's an 'o' if it's the second character received, and increments a counter R2 for each case; otherwise, CheckReturn checks to see if it's a carriage return character. In all input cases, except for a carriage return, the character is processed with ProcessChar. ProcessChar stores the received character into external memory space. Once a carriage return is found by CheckReturn, the program calls SendStr, which sends all stored characters out the UART port. After SendStr, the function TurnOnLEDsMaybe is called. It checks R2 to see if it has a value of 2. If it does, it means that the characters 'g' and 'o' were the first and second characters received, respectively. TurnOnLEDsMaybe then checks the character counter to see how many characters were received. If it is also 2, then that means that the only characters entered were 'g' and 'o'. This is the success case of the string “go” being entered, and the LED-lighting function LEDsFTW is called. LEDsFTW basically sets and clears bits 90h, 93h, and 96h (where LEDs have been inserted on the SDK) to turn the LEDs on and off. After turning on one LED, the wait500ms function is called to add a delay of 500 ms. After the delay, the LED is turned off, and the next one is turned on, and so on.

Block Diagram:

When loading and executing program:

LEDs at p1.0, p1.3, and p1.6 on SDK



serial cable connecting PC to serial port P5 on SDK

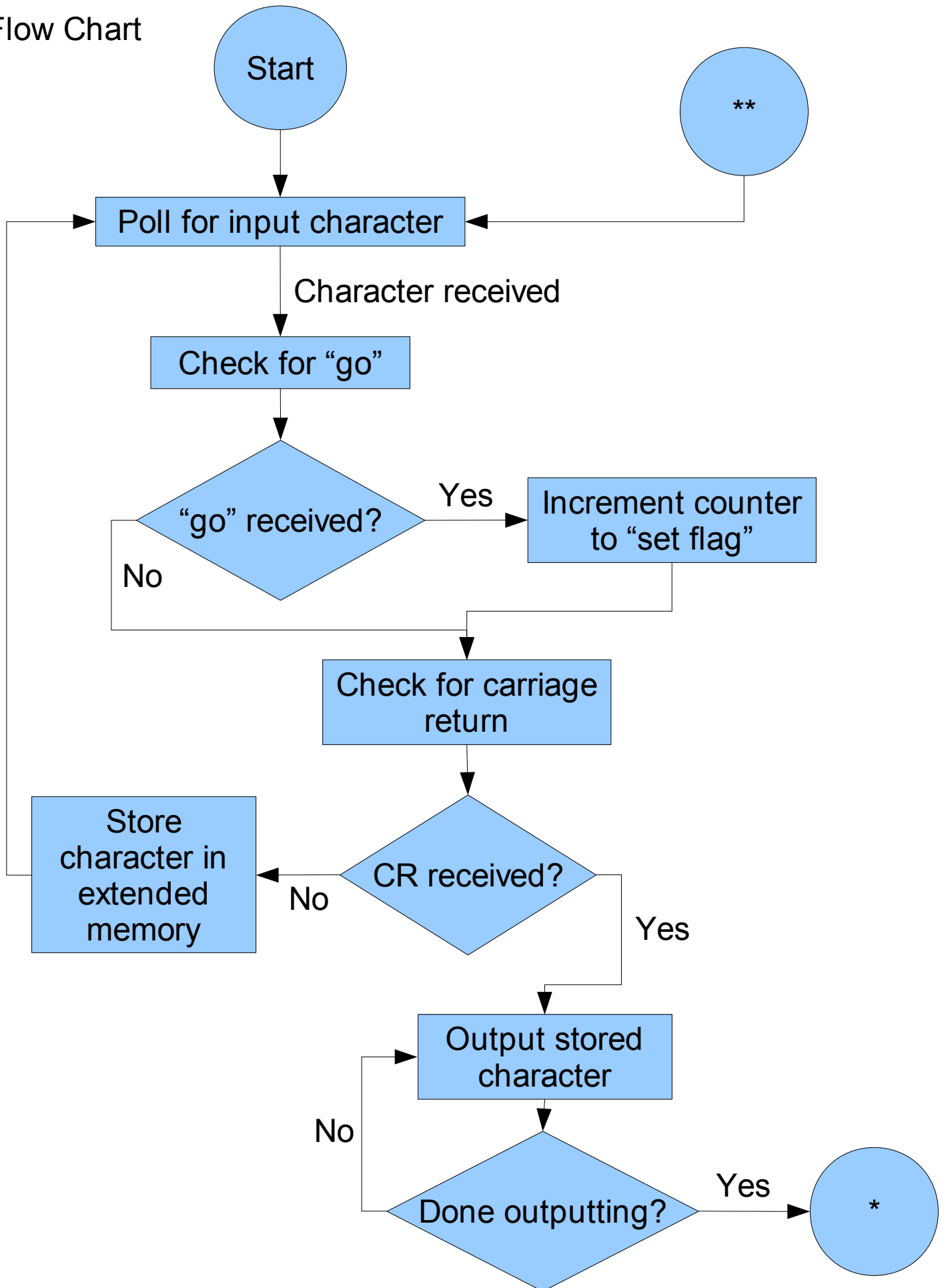
After executing program:

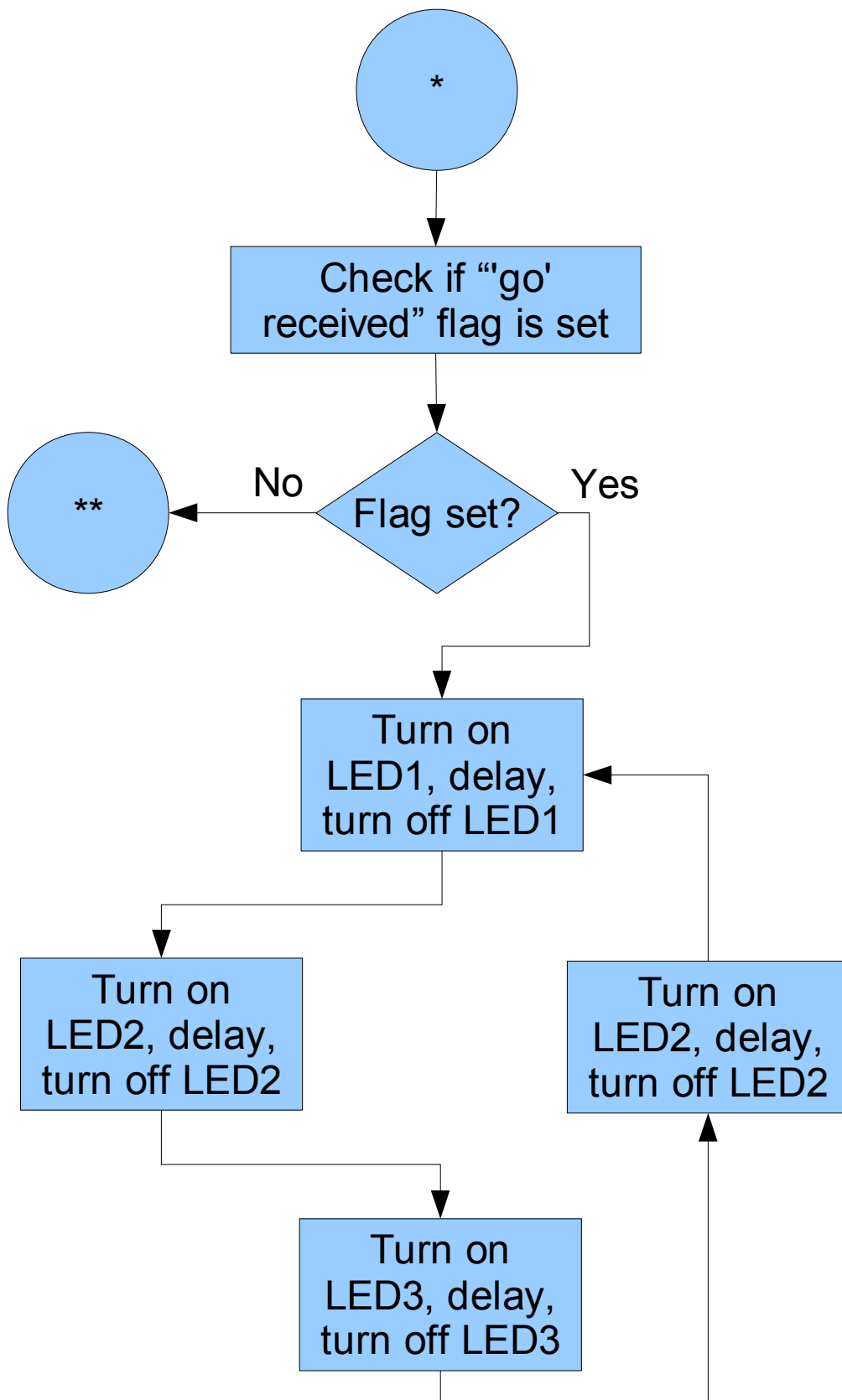
LEDs at p1.0, p1.3, and p1.6 on SDK



serial cable connecting PC to serial port P1 on SDK

Flow Chart





Test:

1. Load program onto SDK.
2. Type “mem” command, then execute.
3. Switch serial cable to P1 connector.
4. Type anything except “go”, then hit ENTER. Check that the input string is echoed at the terminal, and that the LEDs have not changed their initial state.
5. Type a string of characters that includes “go” somewhere in the string (e.g., “i like goats.”), then hit ENTER. Check that the input string is echoed at the terminal, and that the LEDs have not changed their initial state.
6. Repeat step 5 multiple times, with “go” located at different locations in the input string. For at least one of the iterations, put it at the very beginning of the string (e.g., “got milk?” or “go!”).
7. Type either “g” or “o”, then hit ENTER. Check that either “g” or “o” is echoed at the terminal, and that the LEDs have not changed their initial state.
8. Type “go” with different capitalizations: “Go”, “gO”, and “GO”. Check that it is echoed at the terminal accordingly, and that the LEDs have not changed their initial state.
9. Finally, type “go”, then hit ENTER. Check that it is echoed at the terminal, and that the LEDs are now flashing on and off back and forth from left to right, with each light being on for about half a second. Inputting anything else on your keyboard should not affect the program now.

Results:

Program works as expected per test instructions above.