Electronics microcontroller (picaxe) project essay

Aim: To program a PICAXE microcontroller to mimic a traffic light sequence, complete with pedestrian crossing lights (and trigger switch).

This could be used to control a set of pedestrian lights. Output PortsBinary2726252423222120Connected Outputn/an/an/aPe. GPe. RTr.

RTr. ATr. GKey: “ Tr.” represents traffic, “ Pe.

” represents pedestrian. For example, the denary number “ 31” would translate to the byte “ 00011101” and would then transfer to the eight outputs, so all connected outputs would be high (and therefore lit) apart from the amber light for traffic. The binary number represents one of two logic states – 0 or 1. The three most significant bits of these bytes can be ignored for this program as they are not used. Circuit OperationThis circuit emulates a traffic light sequence.- The “ let pins = 9” command transfers the byte “ 00001001” to the 8 output pins, which lights the LEDs traffic green, and pedestrian red.

– The “ wait 60” command makes the program wait 60 seconds before moving on to the next instruction. This is so after the program has run through, the pedestrians cannot cross again in quick succession, thus avoiding too much interruption to traffic flow.- “ pin0= 1” decision has two outputs: “ no” loops back to the top of the decision diamond, “ yes” links to the rest of program. This checks if an input switch (a button for pedestrians) has been pressed.

– “ wait 12” lets 12 seconds elapse until the next command, this means that the lights do not change the instant the pedestrian presses the switch.- “ let

pins = 10” transfers the byte “ 00001010” to the outputs, which keeps the pedestrian red light on, but changes the traffic lights from red to amber.- “ wait 3” waits 3 seconds before executing the next instruction.- “ let pins = 12” transfers the byte “ 00001100” to the output pins, which again keeps the pedestrian red on, but changes the traffic light to red.- “ wait 1” waits one second before executing “ let pins = 20”, this is for safety as the traffic lights should be red before the pedestrian light turns green. 20 is the denary number that represents “ 00010100”, which assigns the appropriate logic level to each of the 8 output pins – in this case traffic red and pedestrian green.

– “ wait 8” allows 8 seconds for any pedestrians waiting to cross safely.- “ let b0 = 5” assigns the value of 5 to the variable b0 (one of thirteen variable spaces in the PICAXE’s RAM, each capable of storing one byte).- The “ b0 = 0” decision diamond checks if the value stored in the register b0 is 0. If it is, then it carries on with the rest of the program, if not then the following occurs: 1. “ let pins = 4” transfers the byte “ 00000100” to the outputs, i. e.

all are off bar the traffic red. 2. “ pause 600” waits 600 milliseconds before continuing. 3.

“ let pins = 20” transfers the byte “ 00010100” to the outputs again, turning the pedestrian green back on. 4. “ pause 600” then waits another 600 milliseconds before continuing. 5. “ let b0 = b0 – 1” subtracts one from the value stored in the register b0 each time it is executed. This then loops back up to the above decision diamond, and the loop is executed until b0 = 0, i.

e. when the green pedestrian light has flashed on and off 5 times – this is a safety precaution as it warns anybody on the crossing to hurry, and people approaching the crossing not to start doing so.- Now that b0 = 0, the instruction “ let pins = 12” will be carried out, which again transfers the byte “ 00001100” to the outputs, keeping the traffic red but also turning the pedestrian red.- “ wait 2” waits 2 seconds before running “ let pins = 14”, this is a safety precaution in case anybody did happen to be on the crossing. The latter command transfers the byte “ 00001110” to the eight outputs, which keeps the pedestrian light red, and turns on the traffic amber whilst traffic red remains on.- “ wait 3” then creates a 3 second delay before the program loops back to the beginning, and the lights resort to their default state courtesy of the first command.

Note: “ logic 1” means that the potential at pin 14 (VIN for the PICAXE-18 IC, always 5V) will be transferred to the particular output pin. Testing ProcedureTo test this circuit, all I need is myself and a watch (with a multimeter as backup). I will study by observation whether the correct LEDs are on at the correct time, and in the correct sequence (comparing it to my program at all times). I can do this as if all the LEDs work as expected, the circuit works. However, if LEDs are dim, or do not light then I will have to test the potential at the output pin (by comparing it to 0V) at the time when it is not correct. If there is no potential difference between the pin and 0V, then I will need to check my program, or try an alternative PICAXE IC as a last resort, in case mine has been rendered dysfunctional.

To test the timing, I will only need a regular watch. This is because the clock speed of the PICAXE (by default, when not over-clocked) is ~4MHz, which

means any tiny difference in time to the specified value will be much smaller than the human error caused by me timing it, no matter the sensitivity of the instrument. Testing ResultsI tested the circuit using the above qualitative measures and it worked perfectly, following the exact steps I expected it to. I then took quantitative measurements using a multimeter, so that I would be eligible for the further 2 marks on the mark scheme.

The tables are done in terms of logic levels for simplicity and ease of comparison: “ 1” representing anything ; 4. 9V, whereas “ 0” represents anything ; 0. 1V. Here are my results: