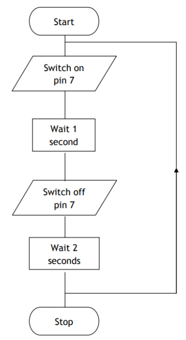
Flowcharts & Programming

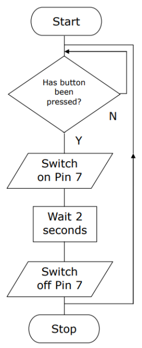
Terminator symbol		Used for the start and end of a main program or sub-procedure.
Line symbol		Shows the direction of program flow. For flow down or to the right, an arrow is not needed. For flow upwards or to the left, arrows are added.
Input/Output		Used to control outputs or to show that data is being received.
Process symbol		Used for operations which take place within the microcontroller, for example a delay.
Decision symbol		Program flow is determined by a "yes" or "no" answer to the question in the box.
Sub procedure symbol		Used to call a sub-procedure.

Switching on outputs




```
symbol red = 7          'rename output 7 'red'

main:
    high red            'set pin 7 on
    pause 1000          'keep pin 7 on for 1 second
    low red             'set pin 7 off
    pause 2000          'keep pin 7 off for 2 seconds
    goto main           'jump back to main
```

Making decisions




```
main:
    if pin0 = 1 then light
    goto main
light:
    high 7
    pause 2000
    low 7
    goto main
```

Variables

```
symbol counter = b0         'rename variable b0 'counter'
symbol green = 5            'rename output 5 'green'

flash:
    for counter = 1 to 10   'start a for... next loop
        high 5              'switch green on
        pause 1000          'wait 1 second
        low 5               'switch green off
        pause 1000          'wait 1 second
    next counter            'add 1 to counter
    end                     'ends program
```

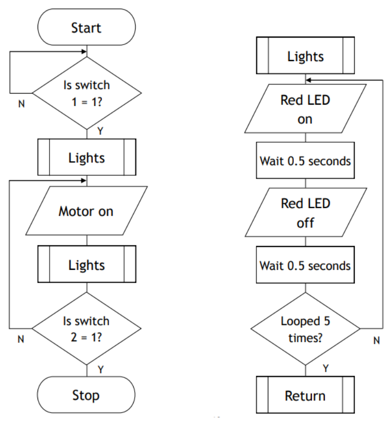
To test an analogue sensor (such as a thermistor or LDR), which can show you the actual value as it changes on screen, you are going to use the following test program.

```
test:
    readadc 0, b0      'read channel 0 into variable b0
    debug b0           'transmit value to computer screen
    pause 100          'short delay
    goto test          'jump back to the start
```

Testing sensor inputs

```
symbol counter = b1
symbol eyes = 7
symbol buzzer = 6

main:
    debug b0
    pause 100
    readadc 0, b0
    if b0 > 180 then buzz
    low buzzer
    pause 100
    if b0 < 175 then flash
    low eyes
    pause 100
    goto main

flash:

    high eyes
    pause 100
    goto main

buzz:
    high buzzer
    pause 100
    goto main
```

Subprocedures




```
symbol red = 7
symbol fan = 5
symbol counter = b0

main:
    if pin1 = 0 then main       'check to see if start switch is pressed
    gosub lights                'jump to the subroutine lights
    gosub motor                 'jump to subprocedure motor
    gosub lights                'jump to subprocedure lights
    end                         'ends program

lights:
    for counter = 1 to 5        'start a for...next loop
        high red                'red LED on
        pause 500               'wait for 0.5 seconds
        low red                 'red LED off
        next counter            'repeat LED flashing 5 times
    return                      'return to main program

motor:
    high fan                    'switch fan on
    if pin2 = 1 then motor_off  'test pin 2
    goto motor                  'keep fan on

motor_off:
    low 5                       'switch fan off
    return                      'return to main program
```

Using motors & Pulse Width Modulation

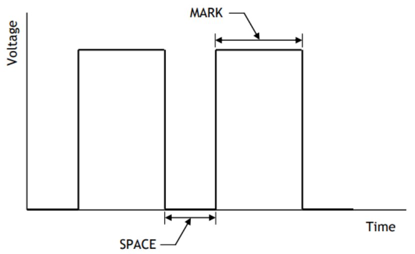
There are three main ways that you can slow down or change the output speed of a motor:

A) You can do it mechanically by using a belt drive system or gear system. For example, the speed of a motor car can be changed simply by changing gear.

B) An electrical method of slowing down the speed of an electric motor is to change the supply voltage across it.

C) The speed of a motor can also be controlled by switching the power supply on an off very quickly. This makes it an ideal method for a microcontroller which can use Pulse Width Modulation.

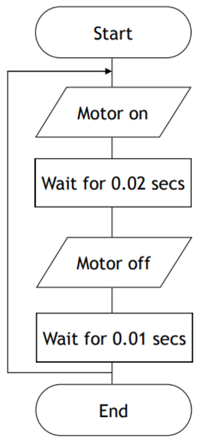
Pin 4	Pin 5	Direction
Off	Off	Off
Off	On	Clockwise
On	Off	Anti-clockwise
On	On	Off



In the graph shown above, the "mark" is twice the "space" time. So the mark to space ratio is 2:1. this will produce an average output voltage of two thirds of the supply.

$$\frac{2}{3} \times 5V = 3.33V$$

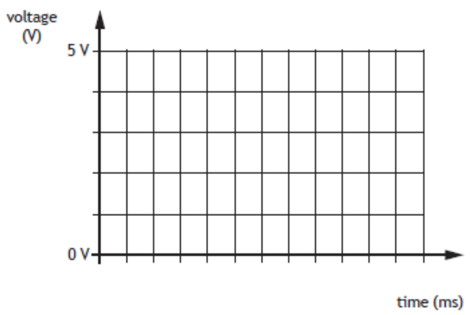
The flowchart for the Pulse Width Modulation to control the speed of a motor is shown. Notice that the length of the "mark" and "space" time has been kept very short to prevent the motor turning in a jerky manner. With this mark-to-space ratio the motor should rotate at two-thirds of it's full speed. Develop a PBasic program for the flowchart. Try altering the mark-to-space ratio and note the effect on the motor.



4. The speed at which an automatic garage door opens is controlled by a motor using pulse-width modulation (PWM).

(a) Complete the graph below to show how PWM could be used to make the motor rotate at half speed. 2

You should include at least three pulses.



(b) Describe how the speed of this motor could be decreased using PWM. 1

____

____

____

____

4. (continued)

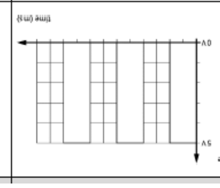
An alternative method of speed control involves varying the size of the DC voltage supplied to the motor.

(c) Describe one advantage of using PWM in comparison to varying the size of the voltage supply. 1

____

____

____

4.	(a)			2	1 mark for correct ratio - on for the same length of time as off 1 mark for digital signal - all pulses must be the same height
	(b)		Decrease mark time, space time stays same Increase space time, mark time stays same Decrease mark time, increase space time Decrease of mark:space ratio or duty cycle	1	1 mark for suitable response - must refer to both mark and space
	(c)		Torque remains the same at all speeds	1	1 mark for suitable response

Fault finding

4. An air pressure supply system in a car garage will be operated by programmable control. The system must meet the following specification.

- The system is activated when a user presses a start button
- The pumping system then switches on
- The air pressure is monitored by an analogue sensor
- When the air pressure rises above a set level (100), the pump switches off
- A light must flash five times to indicate that the air pressure has reached the set level
- The system then resets, ready for the next user

Input	Pin	Output
	7	Pump
	6	Light
Start button	1	
Pressure sensor	0	

Part of the same test program is shown below in **PBASIC** and **ARDUINO** code.

```
let dirs = %11110000
symbol pressure = b4

main: if pin1 = 1 then main

      high 7

check: readadc 0, pressure

if pressure ≤ 200 then check

      low 7

      for b3 = 0 to 5
            high 6
            pause 200
            low 6
            pause 200
      next b3

     goto main
```

```
int Pressure = 0;
int Pressuresensor = 0;
int Pump = 7;
int Light = 6;
int Startbutton = 1;

void setup(){
pinMode (Pressuresensor, INPUT);
pinMode (Startbutton, INPUT);
pinMode (Pump, OUTPUT);
pinMode (Light, OUTPUT);
}

void loop(){
 if (Startbutton == LOW) {
     digitalWrite(Pump, HIGH);
 } else {
     digitalWrite(Pump, LOW);
 }
 Pressure = analogRead (Pressuresensor);
 if (Pressure <= 200){
   digitalWrite (Pump, LOW);
 }
 for(int counter=0; counter<=5;
counter=counter+1)
      {digitalWrite(Light, HIGH);
      delay(200);
      digitalWrite(Light, LOW);
      delay(200);
 }
}
```

4. **(continued)**

There are three faults in the program shown opposite.

Identify the lines where the faults are and write the corrected code.

Complete for **either** PBASIC or ARDUINO. 2

The first fault is shown below.

Correction 1 PBASIC *main: if pin1 = 0 then main*

Correction 1 ARDUINO *if (Startbutton == HIGH) {*

Correction 2

Correction 3

10. (continued)

The program shown below in PBASIC and ARDUINO code is known to have faults.

```
main: if pin3 = 1 then jump
      if pin2 = 1 then main
      if pin1 = 0 then main

jump: high 7
      pause 1500
      high 6
      pause 200
      low 6

     goto jump
```

```
void loop(){
  if (digitalRead(3)==HIGH){jump();}
  else if (digitalRead(2)==HIGH){loop();}
  else if (digitalRead(1)==LOW){loop();}
  else{loop();}
}

void jump(){
 digitalWrite(7,HIGH);
 delay(1500);

 digitalWrite(6,HIGH);
 delay(200);
 digitalWrite(6,LOW);
}
```

(b) Describe three different faults in the program shown above. 3

Fault 1

Fault 2

Fault 3

Flowcharts

12. (continued)

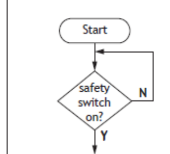
A second prototype using a microcontroller instead of a logic circuit is to be tested. The system must meet the following criteria:

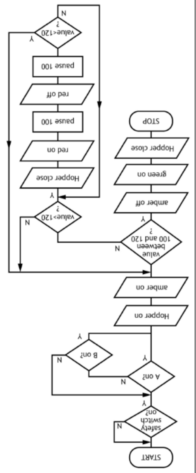
- the system must wait until a safety switch has been activated
- when either of the manual start switches A or B are pressed a signal must be sent to open the hopper and the amber LED must come on
- if the weight reading is between a value of 100 and 120 the green LED alone must come on and the hopper must close
- if the reading is greater than 120 the hopper must close and the red LED must flash 5 times per second until the reading drops to less than 120 (material will be removed manually by an operator to reduce the weight).

Input	Pin	Output
	7	hopper open (1)/close (0)
	6	red LED
	5	green LED
	4	amber LED
safety switch	3	
manual start switch A	2	
manual start switch B	1	
weight (analogue value)	0	

12. (continued)

(d) Complete, with reference to the pinout table and the criteria given opposite, a flowchart for the system. 10





8. An anti-lock braking system is used to control the speed of an elevator as it descends. This uses a form of pulse-width modulation operated by a microcontroller.

If the speed of the elevator is too fast, the brakes will increase the proportion of operating time.

The table below identifies the connections to the microcontroller.

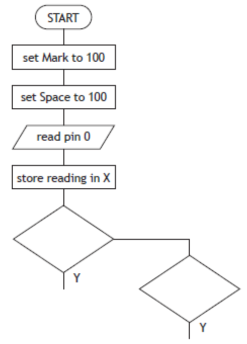
Input	Pin	Output
	7	brake
ground level sensor	1	
speed sensor (analogue)	0	

The system must perform the following steps.

- Values for mark and space both need to be initially set to a value of 100
- A reading must be taken from a speed sensor and its value stored in variable X
- If the value of X is greater than 128 then mark increases by 1 and space decreases by 1
- If the value of X is smaller than 128 then mark decreases by 1 and space increases by 1
- If the value of X is 128 then mark and space do not change
- The brake must be switched on and off for the times specified mark and space (this will be in milliseconds)
- The process must continue until the ground level sensor is activated

8. (continued)

(a) Complete, with reference to the specification and the input/output table, this flow chart for the control of the system. 8

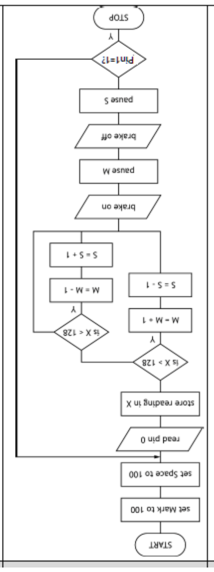


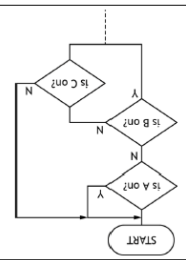
8. (continued)

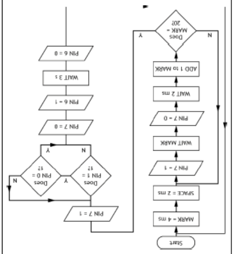
An alarm (Z) is part of the elevator's operating system and needs to be activated under the conditions given by the following Boolean equation.

$$Z = \overline{A} \cdot (B + C)$$

(b) Complete the following flowchart to perform the function described above. 3

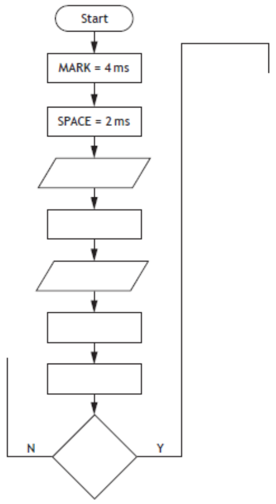






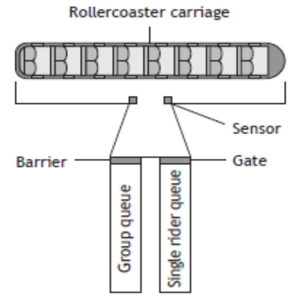
11. (continued) MARK

(b) Complete, with reference to the specification and input/output table shown opposite, the flowchart for the control of the motor. 13



6. A new rollercoaster has been designed for a theme park. The ride has two queues: one for groups, who wait behind a barrier, and the second for single riders, who wait behind a gate.

The system counts how many people get on the carriage as they pass the sensor.



INPUT	PIN	OUTPUT
	7	Barrier (open = high)
	6	Gate (open = high)
	5	Motor
	4	
Launch Switch	3	
Start Switch	2	
Single Rider Switch	1	
Sensor	0	

The control sequence for the rollercoaster has the following steps.

- An operator presses a start switch to begin the boarding process.
- The barrier will open to allow up to 16 people from the group queue to take a seat.
- The sensor will give a high signal each time a person passes through, allowing the number of people to be counted.
- If the carriage is not filled from the group queue, a single rider switch is pressed by the operator.
- The barrier should then close and the gate should open to allow single riders on to occupy all remaining seats.
- Once riders are seated the operator presses a launch switch which closes the gate and the motor is then on for 30 seconds.
- The system should loop.

(a) Draw a flowchart to show how the ride is operated. 9