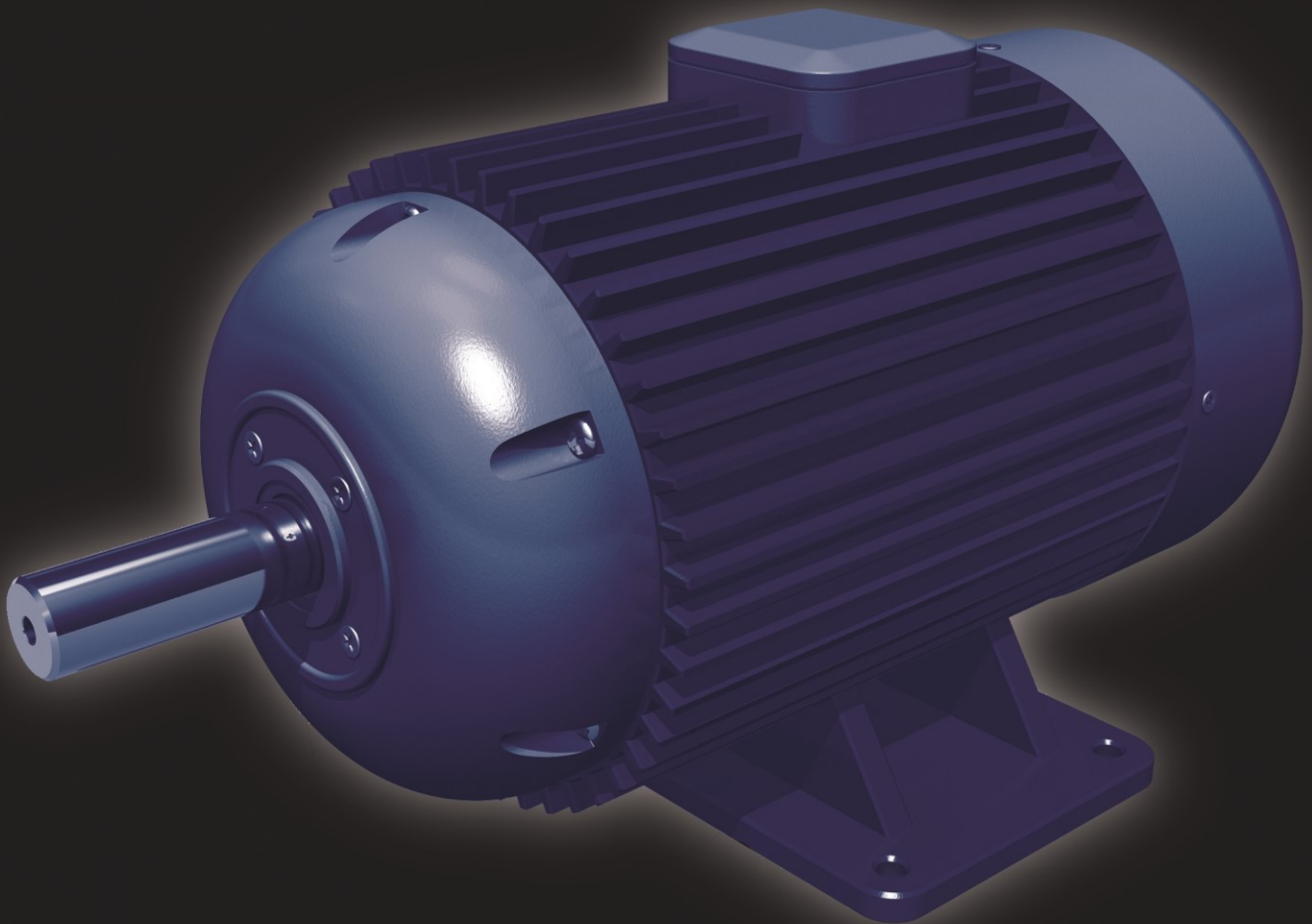


Control of DC motors



WS0247

MATRIX
www.matrixsl.com

Copyright © 2014 Matrix Multimedia Limited

Contents

Introduction to Motor Control - Instructor Guide	1
About this course	4
Learning objectives	5
Scheme of Work	6
1. What is PID Control	6
2. DC Motors	8
3. Matrix DC Motors Solution	10
4. Flowcode Components	11
5. The Programs	14
Introduction to Motor Control - Student Course	16
1. What is PID Control	17
1.1 Control system terminology	17
1.2 PID principles	18
1.3 PID components	19
1.4 PID variations	20
2. DC Motors	22
2.1 Introduction to DC motors	22
2.2 DC motor principles	22
2.2.1 The underlying physics	22
2.2.2 Making it rotate	23
2.2.3 Slip rings	24
2.2.4 Commutators	24
2.2.5 Making a simple motor	24
2.3 Multi-coil DC motors	25
2.4 Driving DC motors	26
2.4.1 Diode protection	26
2.4.2 Controlling the speed of the motor	27
2.4.3 Controlling the direction of rotation	28
2.4.4 Using a full bridge IC	29
3. Matrix DC Motors Solution	30
3.1 Setting up the motors solution hardware	31
3.2 Sensors board controls	31
4. Flowcode Components	32
4.1 The PWM component	32
4.1.1 Configuring the PWM component	33
4.1.2 Controlling motor direction	34
4.2 The ADC potentiometer component	35
4.3 The LCD component	37
4.4 Ghost monitoring	38
4.5 DSP components	40
4.5.1 DSP chain	41

5. The programs	43
5.1 Speed control	
5.1.1 Speed control - Open loop control	44
5.1.2 Speed control with feedback - Closing the loop	46
5.1.3 Speed control - Proportional (P) control	48
5.1.4 Speed control - Proportional and Integral (PI) control	50
5.1.5 Speed control - Proportional, Integral and Derivative (PID) control	52
5.2 Position control	
5.2.1 Position control - Open loop control	54
5.2.2 Position control with feedback - Closing the loop	56
5.2.3 Position control - Proportional (P) control	58
5.2.4 Position control - Proportional and Integral (PI) control	60
5.2.5 Position control - Proportional, Integral and Derivative (PID) control	62
6. Further work	63

About this course

Aims:

The course examines PID techniques used to control the speed and position of an electric motor. It does so through a series of exercises involving programs for the dsPIC microcontroller, using Flowcode (version 7 or later).

What the student will need:

To complete this course the student will need the following equipment:

- Flowcode software (version 7 or later)
- E-blocks including:
 - dsPIC Ghost Programmer (EB091 or BL0032)
 - Sensors board E-block (EB090)
 - 2 x Dual Trim Pot Sensor Modules (EBM006)
 - DC motor speed trainer E-block (EB096)
 - DC servo motor trainer E-block (EB097)
 - FET driver board E-block (EB094)
 - LCD board E-Block (EB005 or BL0169)
 - Universal power supply (HP2666)

Using this course:

This course presents the student with a number of tasks detailed in the following text. All the information needed is contained in the notes.

Before starting any exercise, the student should spend time familiarising him/herself with the course material so that (s)he knows where to look for help.

Time: It will take around twelve hours to complete all exercises on this course.

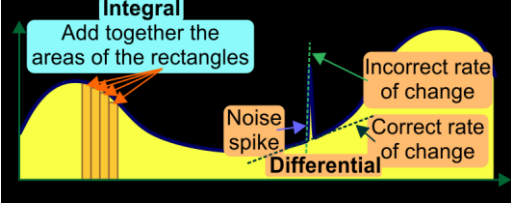
Learning objectives:

On completing this course, the student will be able to:

- name the three elements of a PID control system;
- describe the meaning of the terms 'set point', 'feedback', 'error signal', 'rise time', 'settling time', 'steady-state error', 'overshoot' and 'ringing' applied to a control system;
- draw a labelled block diagram to illustrate the principle of PID control;
- distinguish between the functions of the proportional, integral and derivative elements of the PID control system;
- describe effects of the three elements on the behaviour of the controlled process;
- interpret waveforms showing the set point and feedback signals;
- use Fleming's Left-hand rule to establish the direction of the force acting on a current-carrying wire in a magnetic field;
- describe the need for a commutator in a DC motor;
- state an advantage for a multi-coil over a single coil DC motor;
- explain why a driver circuit is needed to interface a DC motor to a microcontroller;
- explain the need for diode protection when a DC motor is controlled by a microcontroller;
- describe the principles of PWM motor control and contrast it with the use of a series resistor as a means of controlling the speed of a DC motor;
- describe the meaning of the terms 'mark', 'space' and 'duty cycle' applied to PWM control;
- draw a diagram of an 'H' bridge, using mechanical switches, to illustrate the principle of direction control of a DC motor;
- describe what is meant by a 'component' and a 'component macro' in Flowcode;
- configure the Flowcode PWM component to control the speed of a DC motor;
- configure the Flowcode ADC component to generate digital signals from the control potentiometers and sensors;
- configure the Flowcode LCD component to display the values of control signals;
- configure the dsPIC controller to use the 'Ghost' functionality;
- use the Flowcode 'Scope Monitor' function to display signals on the 'Data Scope';
- describe the function of the Flowcode DSP System component;
- describe the purpose of the properties 'Buffer Count', 'Buffer Name', 'Bit Depth', 'Sign' and 'Size' used to configure the Flowcode DSP System component;
- create a Flowcode DSP Chain for a PID control system including 'input', 'control', 'output', PWM and LCD components;
- compile a given Flowcode dsPIC program, transfer it to the dsPIC on the EB091 or BL0032 board and test its performance using 'Ghost' technology to monitor it;
- explain the use of an internal timer in the microcontroller to trigger events in a given macro;
- distinguish between an open-loop and a closed loop control system, giving examples of each;
- create a chart of set point vs feedback signals for a speed control system;
- set up and test 'P', 'PI' and 'PID' speed control systems;
- 'tune' a PID speed control system to meet a given specification;
- create a flowchart to control the direction of rotation of the motor using a signal in the form of a single byte;
- set up and test 'P', 'PI' and 'PID' position control systems;
- 'tune' a PID position control system to meet a given specification.

Scheme of work

Section	Notes for instructors	Timing (minutes)
<p>1. What is PID Control?</p>		
<p>1.1 Control system terminology</p>	<p>Students are introduced to the terms ‘set point’, ‘overshoot’, ‘rise time’, ‘ringing’, ‘settling time’ and ‘steady-state error’. (Other names exist for these terms and the instructor may wish to introduce them at this point).</p> <p>Instructors could introduce a number of processes that use PID control, and discuss:</p> <ul style="list-style-type: none"> • what these terms mean for that process; • the priorities for the control system; • whether these effects would cause a problem. 	<p>10 - 30</p>
<p>1.2 PID principles</p>	<p>This section introduces students to the standard equation for PID control (though there are alternative forms.)</p> <p>The instructor will need to support students with lower levels of mathematical ability in understanding this terminology. (In the end, it does not matter because the software takes care of the ‘number-crunching’.)</p> <p>Again, there is the standard diagram for the PID process, (which also appears in different forms,) showing:</p> <ul style="list-style-type: none"> • how a feedback signal from a sensor is combined with a set point signal to generate an error signal; • how the error signal can be processed within the ‘P’, ‘I’ and ‘D’ components to produce a control signal, ‘C’. <p>It may not be immediately obvious to students that the feedback signal is a measure of the current value of the state of the process, (process variable). Even less obvious is the need to calibrate the feedback signal to match the format of the set point signal. For example, when controlling the temperature of a furnace, the set point ‘pot’ is probably graduated in °C. To allow the PID controller to combine the set point and feedback signals, the feedback signal would have to be converted from volts into °C using thermistor calculations etc. Similarly, the control signal would have to be converted into a suitable PWM or DAC output to drive the system towards the target temperature.</p>	<p>10 - 30</p>
<p>1.3 PID components</p>	<p>This section breaks down PID control into the three components - Proportional (P), Integral (I) and Derivative (D). Once more, the instructor will need to support students with lower levels of mathematical ability in understanding this section.</p> <p>Where two quantities, ‘X’ and ‘Y’ are ‘proportional’, when one quantity is doubled, then so is the other, one quantity quartered, then so is the other etc. Mathematically, their relationship can be written as an equation:</p> $Y = K \times X$ <p>where ‘K’ is a constant (i.e. a fixed number.)</p> <p>‘Integration’ is a calculus-based function that sums a</p>	<p>15 - 35</p>

	<p>consecutive series of values. It is often pictured in terms of calculating the area under the curve in a graph of one quantity plotted against the other, effectively an infinite sum of infinitesimally narrow rectangles under the curve.</p>  <p>The area of one of these rectangles is 'y' x 'x' i.e. 'height' x 'width'. In the PID case, it is 'error' x 'short time elapsed while error was at that value'.</p> <p>The 'Differential' component uses the current rate of change of the error (i.e. the gradient of the graph shown above,) to predict when the error will be zero (i.e. set point signal and feedback signal will be equal). One problem is that noise can appear as sharp spikes on the curve, giving the impression of a very rapidly changing signal, leading to a false prediction of when the error will be zero.</p> <p>The table summarises the effects of increasing the 'P', the 'I' and the 'D' components.</p>	
<p>1.4 PID variations</p>	<p>It is not always necessary or desirable to use all three PID components to control a process. By setting the constant to a value of zero, the corresponding component can be removed.</p> <p>This section looks at various combinations of components and their behaviour. The diagrams are obtained from the Flowcode DataScope and come from the investigations that the students carry out in due course. The instructor will avoid problems later by ensuring that students are able to interpret these diagrams in terms of steady-state error, overshoot, settling time etc.</p>	<p>10 - 30</p>
<p>1.5 Manual 'tuning' of a PID system</p>	<p>There is no simple answer as to how to obtain 'optimal' response from a control system. For industrial processes, the problem of doing so quickly is hugely important, as anything other than 'optimal' can be very costly.</p> <p>There are 'official' algorithms that can be followed - the Ziegler-Nichols methods, the Cohen-Coon method etc... This section gives very general guidelines on how to 'tune' the control system manually. However, as the opening sentence admits, it is an art, not a science.</p>	<p>10 - 30</p>

2. DC Motors		
2.1 Introduction to DC motors	At this point, it may be advantageous to have some cut-away motors available for the students to inspect. They could then identify the principal features.	10 - 30
2.2.1 Underlying physics	<p>Although this section is short, it is important. It should start with a discussion of what we mean by 'magnetic field'. This question is not trivial as it involves forces acting at a distance on objects that are not in contact. The idea of a 'field' is our attempt to feel comfortable with this weird effect.</p> <p>The origin of all magnetic fields is the movement of electrons. Whether the magnetic fields combine to become observable depends on the nature of the material, i.e. is it ferromagnetic. Nevertheless, even though the overall effect from an individual electron is tiny, even modest currents involve the flow of such large numbers of electrons that the magnetic field produced can be considerable.</p> <p>It may be worth reminding students that Victorian scientists got it wrong! Electrons (unknown at the time,) move from the negative terminal of electrical supplies to the positive. We cannot see them so it makes little practical difference to stick with the Victorian notion of current flowing from positive to negative.</p> <p>Another unusual aspect is the direction in which the force acts. Fleming's Left-hand (Motor) rule allows us to predict that direction and students should practice its use if they are to appreciate the requirements for rotation that follow in the next section.</p>	10 - 30
2.2.2 Making it rotate	One problem in understanding this section is the three-dimensional nature of the argument. If models are available, then visualisation will be easier. Students should be encouraged to apply Fleming's rule to identify the directions of the forces on the various limbs of the coil.	10 - 30
2.2.3 Slip rings	For completeness, the treatment includes the use of slip-rings to reverse the current direction in the coil even though this is relevant to AC motors only.	10 - 20
2.2.4 Commutator	<p>Again, it is the three-dimensional nature of the device that adds to the problems in understanding how the commutator works. The diagram shows a simple two-part commutator though, as the photograph in section 2.3 shows, they are usually much more intricate.</p> <p>Here again, a model of a single coil motor will make understanding easier.</p>	10 - 30
2.2.5 Building a simple motor	Students will find it useful to build a simple motor like the one mentioned here to cement ideas about the principles behind them.	15 - 40

<p>2.3 Multi-coil DC motors</p>	<p>The section discusses why multi-coil motors are necessary in most cases. Again, the problem for the student is to visualise the 3-D arrangement.</p> <p>It also includes an explanation of the use of laminations in the core. The idea is to increase electrical resistance to reduce eddy currents induced in the core by the changing magnetic fields. The instructor could remind the students of their GCSE Science formula, ($P = I^2/R$) showing that power dissipated depends on the square of the current and so is reduced considerably by increasing resistance.</p>	<p>10 - 30</p>
<p>2.4 Driving DC motors</p>	<p>Motors draw relatively high currents from the current source. Microcontrollers can deliver relatively small currents. The mismatch is handled by a buffer subsystem, either a bipolar junction transistor, (BJT), or a field-effect transistor such as a MOSFET. Circuits for both subsystems are shown and may be familiar to students from earlier studies.</p> <p>The advantage of the MOSFET is the huge input resistance, meaning that a tiny current is drawn from the source, the microcontroller in this case. Essentially, it is a voltage-controlled device: low input voltage \Rightarrow off, high input voltage \Rightarrow on.</p>	<p>10 - 30</p>
<p>2.4.1 Diode protection</p>	<p>The treatment here assumes that students know that a diode conducts in one direction only, when forward-biased.</p> <p>The concept of 'back e.m.f.' is mentioned briefly and the instructor may wish to expand on this. Whenever a magnetic field moves across a conductor, it generates a voltage. A collapsing magnetic field is essentially a moving field. When the current ceases in the coil of a motor, the magnetic field it produced collapses rapidly and behaves like a fast-moving magnetic field. It generates a very high voltage in the coil and in the opposite direction to that which created it. For that reason it is known as 'back e.m.f.'</p> <p>Many semiconductor devices are sensitive to and can be damaged by high voltages. The role of the protective diode is to start conducting when this 'back e.m.f.' tries to build up and to clamp this voltage at 0.7V, the voltage across a silicon conducting diode. This is too small to damage the controlling device (BJT, MOSFET or microcontroller.)</p>	<p>10 - 30</p>
<p>2.4.2 Controlling the speed of the motor</p>	<p>This section contrasts two methods of speed control - the series variable resistor and the use of pulse-width modulation (PWM).</p> <p>The variable resistor takes part of the driving voltage, leaving less for the motor, which then runs more slowly. However, the current flowing through the resistor is the same as that through the motor and it generates unwanted heating in the resistor, making it an inefficient process.</p> <p>Pulse-width modulation relies on switching the motor on and then off rapidly. When on, all the driving voltage is across the motor and hardly any is dissipated elsewhere, (a little in the connecting wires, perhaps). When the motor is off, no current flows and so no energy is dissipated anywhere.</p>	<p>10 - 30</p>

	<p>The duty cycle for the PWM signal indicate what percentage of each cycle is spent switched on. A duty cycle of 25% means that the motor is switched on for a quarter of the time. The more often the motor is switched on, i.e. the greater the duty cycle, the faster it rotates.</p>	
2.4.3 Controlling the direction of rotation	<p>The direction of rotation of the motor depends on which terminal is positive and which is negative. The way to change the direction, then, is to reverse the polarity of the terminals.</p> <p>The 'Full bridge' subsystem will do this under the control of control signals. This section introduces the principle, using mechanical switches and then extends the treatment to electronic switches using MOSFETs.</p>	10 - 20
2.4.4 Using a full bridge IC	<p>In practice, the subsystem is contained in a single IC such as the L6206 from ST Microelectronics. Usually, the IC also provides diode protection against 'back e.m.f.' and may have other functionality, like overcurrent detection and thermal protection.</p>	10 - 20

3. Matrix DC Motors Solution		
3. Matrix DC Motors Solution	<p>This section outlines the function of each of the E-Blocks boards and shows how they are assembled to make up the DC Motors Solution.</p> <p>The instructor should check the assembly and the wiring connections before the Solution is used.</p>	5 - 15
3.1 Setting up the Motors Solution hardware	<p>This section outlines the checks that should be made prior to powering up the solution. Periodically, the instructor should check that the jumpers are in their correct positions on each board and that the correct voltage is selected on the Sensors board.</p>	5 - 15
3.2 Sensors board controls	<p>The EB090 Sensors board carries two sockets into which various sensor modules can be plugged. For this module, these sockets are occupied by two EBM006 Dual Trimmers. The diagram given here identifies their function in this module.</p>	5 - 10

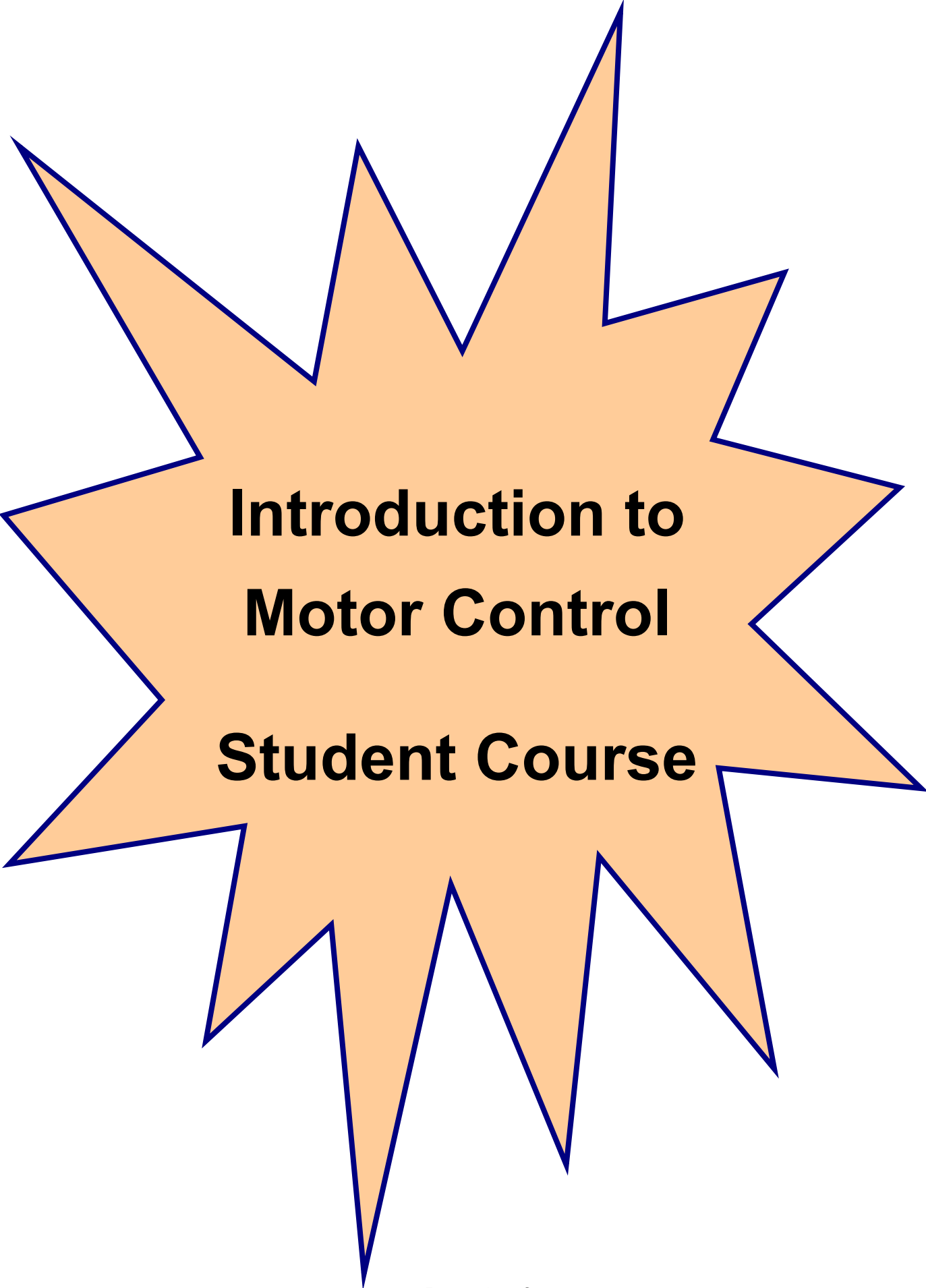
4 Flowcode Components			
4	Flowcode components	<p>The Matrix website, www.matrixtsl.com, contains links to a number of courses on microcontroller programming and Flowcode. Students with little or no programming experience are advised to work through these courses.</p> <p>Flowcode comes with its own vocabulary. For example, a 'component' is a collection of images, macros, even other components, that can be assigned properties. These properties can then be configured and, where appropriate, modified by a program. Some components, such as switches and LEDs, have already been created and these are found in the various Components Toolboxes, 'Inputs', 'Outputs', 'Comms' etc. found across the top of the Flowcode user interface screen.</p> <p>Components can be controlled by 'component macros'. A macro is a section of code that is used repeatedly by a program. For efficiency, it is stored separately and called up when needed. It is sometimes called a subroutine or procedure. Flowcode allows the user to create macros. Programmers at Matrix have already created some for use in controlling components. These are known as component macros and form part of the software suite.</p> <p>Students should be encouraged to visit the Matrix wiki to learn more about Flowcode components.</p>	10 - 20
4.1	The PWM Component	<p>The basis of PWM was described earlier in section 2.4.2. It is characterised by its 'duty cycle' defining for what proportion of each cycle the PWM output is 'on'. Flowcode includes two modes for showing the duty cycle, one analogue - a vertical scale graduated in %, the other digital - a voltage/time plot of the PWM signal, together with timing and frequency information.</p>	5 - 10
4.1.1	Configuring the PWM component	<p>As pointed out above, a Flowcode component has properties that can be configured to define its characteristics. In the case of the PWM component, these include the microcontroller pin on which the signal will appear, its period and frequency.</p> <p>The section gives brief descriptions of the five component macros available within Flowcode to control the PWM component.</p> <p>Students should be reminded that the program is digital, and so the period of the PWM signal is defined as the number of data items making up one cycle of the signal rather than directly as a time period.</p>	10 - 30
4.1.2	Controlling motor direction	<p>This section reminds students that the control system is controlling two motors. To do so, each has its own 'direction' pin, D3 or D7.</p> <p>Furthermore, depending on the logic level assigned to that pin, increasing the duty cycle of the PWM will sometimes increase the speed and sometimes decrease it.</p> <p>The student may wish to 'bookmark' this section for future reference.</p>	5 - 10

<p>4.2 The ADC potentiometer component</p>	<p>The devices that allow the user to control the motors are potentiometers (or ‘pot’s) on the EB090 Sensors board. These output an analogue voltage (one which takes any value within the voltages on the power supply.) The dsPIC is a digital device which processes data in the form of binary numbers. In addition, analogue information is fed back from the speed and position sensors on the motor boards.</p> <p>There must be a means of converting the signals from one form to the other. This is done by the ADC component (ADC = Analogue-to-Digital Converter.)</p> <p>The system must also be able to tell which signal is which. It does so in the same way as that described in the previous section, by allocating each its own pin on the microcontroller. The set point signal appears on channel 1 (pin 15), speed feedback on channel 9 (pin 22), position feedback on channel 10 (pin 23) etc.</p> <p>The same simulation image, a rotary knob, is chosen for all signal sources. (This can be changed in the ADC properties). The section describes some of the component macros available for use with the ADC component. Several of these do the same thing but output the result in a different format.</p>	<p>15 - 30</p>
<p>4.3 The LCD component</p>	<p>The LCD component sends signals in appropriate form to the LCD device. One important requirement is that the signals appear on the correct pin. The ‘Properties’ panel allows the user to specify these pins.</p> <p>There are a large number of component macros associated with the component, but many perform similar functions with different data formats. The ‘Start’ macro must be used before any other to initialise the hardware. The ‘Clear’ macro is used to wipe off all existing characters.</p> <p>The display itself can be thought of as a 16 x 2 array. The ‘Cursor’ macro is used to specify where in this array the first character will be positioned.</p>	<p>10 - 20</p>
<p>4.4 Ghost monitoring</p>	<p>The ‘Ghost’ function allows the Flowcode program to receive and display information from the hardware in real time. This communication takes place via the USB lead from the ‘Ghost’ socket on the EB091 or BL0032 board to the computer.</p> <p>The recommended settings are given in this section. The In-Circuit Debugger (ICD) is not needed and so should be disabled. Monitoring is enabled, as shown. On the ‘Analogue’ tab, the channels of the ADC components of interest are checked. Similarly, any digital signals likely to be monitored on the DataScope, such as PWM signals, must have the corresponding pin selected on the Digital tab.</p> <p>It may be necessary to add a ‘Scope monitor’ component to the dashboard panel to allow signals to be monitored. To do this, click on the ‘Tools’ toolbox and then hover the cursor over the ‘Scope monitor’ icon. A down arrow appears. Click on it and select the ‘Add to dashboard panel’ option. In the Properties panel, select the ‘Scope monitor’. Under ‘Pins’ you can attach up to four signals to the four traces listed.</p> <p>Students will need time to ‘play’ with this facility.</p>	<p>20 - 40</p>

<p>4.5 DSP components</p>	<p>One of the Toolboxes on the Flowcode user interface is labelled 'DSP'. It contains a variety of components for use with a dsPIC microcontroller, such as Filter and Fast Fourier Transform.</p> <p>Every program using DSP is built around a 'DSP System' component. This manages the buffers used in communicating data between other DSP components.</p> <p>The students should be encouraged to read more about these components on the Matrix wiki, using the URLs given.</p>	<p>10 - 30</p>
<p>4.5.1 DSP Chain</p>	<p>At the heart of the DSP program are two macros, the 'TimerTick' and the 'DSPChain' macros. The 'TimerTick' macro is triggered by the interrupt caused when the internal timer 'overflows' - reaches its maximum value and resets. As part of this, the program updates set point and feedback values and the values set on the 'P', 'I' and 'D' pots. It then calls the 'DSPChain' macro. This uses these values to generate the new control signal. Part of this macro scales the control signal and uses it to determine the direction of rotation of the motor.</p> <p>Independently, the 'Main' macro spends most of its time updating the LCD display.</p> <p>Students should be encouraged to study the structure of, and 'play' with, the two macros.</p>	<p>20 - 40</p>

5 The Programs		
5 The programs	<p>The PID principles can be applied to a host of different processes and systems. Here, they are applied to two, speed control and position control of a DC motor.</p> <p>As a result, there are two matching sets of programs, one for speed control and the other for position control.</p>	10 - 20
5.1.1 Speed-control - Open loop control	<p>Where necessary, the instructor should introduce the ideas of open loop and closed loop control.</p> <p>An often-quoted scenario for position control is that of driving a car. The open loop version uses no feedback, i.e. the driver's eyes are closed. Instead s/he is told by how many degrees to turn the steering wheel etc. In the closed loop version, the driver can see the effect of changes made to the steering wheel and adjust the control accordingly to achieve the target position on the road.</p> <p>The focus of this program is the events that take place inside the loop. The signal from the set point 'pot' is used to generate the PWM signal to control the motor speed. Whether it does or not, whether there is a motor or not, is unknown to the program as there is no feedback.</p> <p>The 'Initialise' macro enables the motor driver and PWM component and allocates initial values to direction and speed. These can be adjusted by the events inside the loop.</p> <p>The exercises are designed to familiarise the students with the system and increase confidence in using the system.</p>	10 - 30
5.1.2 Speed control with feedback - Closing the loop	<p>As pointed out in the instructions, this program extends the previous by collecting feedback and using it as a measure of the speed of the motor. However, the feedback signal plays no part in controlling the speed of the motor.</p> <p>The exercises generate a chart of target speed against actual speed for the motor.</p>	10 - 30
5.1.3 Speed control - Proportional (P) control	<p>This is the first of the closed-loop programs. The feedback signal indicates current speed. Combined with the set point signal, (the target speed,) it is used to generate an error signal, indicating how different actual and target speeds are. The control system then weights this value with a number obtained from the 'P' 'pot' and creates a control signal. The bigger the error, the bigger the control signal aimed at achieving the target.</p> <p>The task for the student is to monitor target and actual speeds for a range of values from the 'P' 'pot'. They should encounter overshoot and ringing if they have too high a value of 'P'. Equally, they should notice that the error never falls to zero. There is always a steady-state error.</p>	10 - 30
5.1.4 Speed control - Proportional and Integral (PI) control	<p>The process is the same as that in the previous program except that the Integral term is added to the expression used to create the control signal.</p> <p>Students compare performance of this modified program with the previous one. The aim is to eliminate the steady-state error while preserving a reasonable settling time.</p>	10 - 30

<p>5.1.5 Speed control - Proportional Integral and Derivative (PID) control</p>	<p>This time, all three elements of control, proportional, integral and derivative response are involved. The task is to 'tune' the 'P', 'I' and 'D' constants to produce optimal performance. As 'optimal performance' is a vague term in this context, the instructor may prefer to issue a challenge in terms of producing a result within given parameters of overshoot, ringing and settling time. This could be done as a competition between groups.</p>	<p>10 - 30</p>
<p>5.2.1 Position control Open loop control</p>	<p>This program mirrors that in section 5.1.1 except this time we are looking at position (angle) control. The elements are the same except that an additional macro 'Limits' is used to shield the servo 'pot' from extremes. With the initial position in the centre of the range of possible travel, the program needs to detect whether the target position means rotating clockwise or anti-clockwise. That means setting the motor direction pin to either '0' or '1'. In this program, that is done within the loop. In one direction it is necessary to invert the value assigned to the PWM signal, as described in section 4.1.2. The student task involves making an estimate of settling time using only the set point 'pot'.</p>	<p>10 - 30</p>
<p>5.2.2 Position control with feedback - Closing the loop</p>	<p>This is the position control equivalent of the program in 5.1.2. As in the previous program, the 'Limits' macro is used to prevent excessive speed, and the same modification detects the direction of rotation and significance of the PWM duty cycle. A similar process is applied to the feedback signal to make it compatible with the adjusted set point signal.</p>	<p>10 - 30</p>
<p>5.2.3 Position control - Proportional (P) control</p>	<p>Like 5.1.3, this is a closed-loop program, which uses the position feedback signal to correct the control signal to move the system towards the target position. The control system is weighted by the number obtained from the 'P' 'pot'. The manipulation of the motor direction control signal and the corresponding PWM signal is now done inside the 'DSPChain' macro. The task for the student is to monitor target and actual positions for a range of values from the 'P' 'pot'. Once again, they should be aware of the persistent steady-state error.</p>	<p>10 - 30</p>
<p>5.2.4 Position control - Proportional and Integral (PI) control</p>	<p>The aim is the same as usual except that the control signal depends on the Integral term as well as the Proportional term. Manipulation of motor direction control and PWM signals is again contained within the 'DSPChain' macro. Students again compare performance of this program with the previous one, in terms of the reduction of the steady-state error while maintaining adequate settling time.</p>	<p>10 - 30</p>
<p>5.2.5 Position control - Proportional Integral and Derivative (PID) control</p>	<p>This program involves all three elements of control, proportional, integral and derivative. The task is again to 'tune' the 'P', 'I' and 'D' constants to produce optimal performance, which can be specified by the instructor. Students should be on the look-out for adverse effects caused by the Derivative component.</p>	<p>10 - 30</p>



**Introduction to
Motor Control
Student Course**

1. What is PID Control?

A proportional-integral-derivative controller (PID controller) is a widely used industrial control system.

A system such as an industrial furnace will have a number of associated characteristics - its current temperature, its target temperature etc. A PID controller can be used to control it. It calculates the difference between the target temperature and the current temperature and uses that to drive the heating system for the furnace.

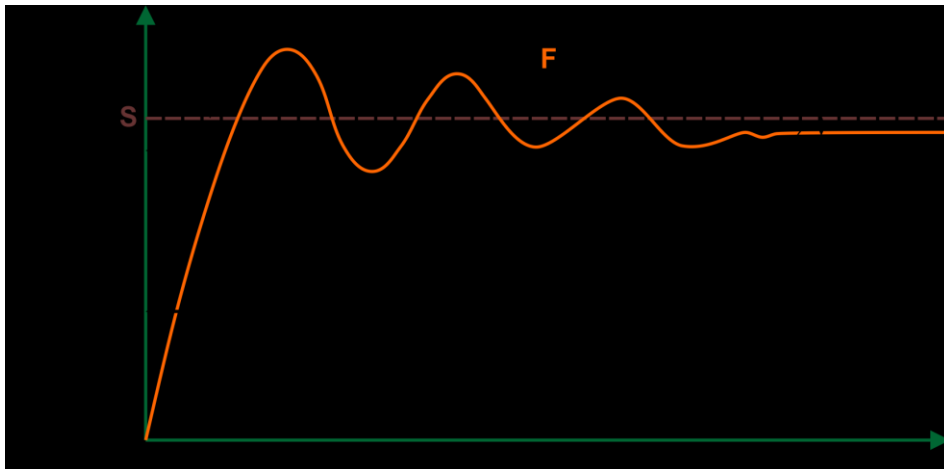
1.1 Control system terminology:

In the jargon of control systems, still using the example of the furnace:

- the target temperature is called the set point **S**;
- the current temperature, measured by a sensor in the furnace, creates a feedback signal **F**;
- the difference between them, i.e. **S-F**, is called the error signal, **E**;
- the controller attempts to minimize the error signal by adjusting the heat energy delivered to the furnace via a control signal **C**.

The aim is to drive the furnace temperature to the target value in the optimal time (not necessarily the shortest time!)

The next diagram illustrates some terms associated with control systems:



- Rise time** - time taken to reach a set percentage (usually 90%) of the target.
- Overshoot** - maximum amount by which the current state exceeds the target, (usually expressed as a percentage of the target).
- Settling time** - time taken to reach a final steady state.
- Steady-state error** - final value of the error signal once steady state is reached.

1.2 PID principles:

The process followed by the PID controller to generate **C** can involve three separate components - the **P**roportional, **I**ntegral and **D**erivative components.

One way to view their effects is:

- the **P**roportional component handles the current value of error signal;
- the **I**ntegral component deals with the effect of past error signals;
- the **D**erivative component predicts the effect of future error signals. based on current rate of change.

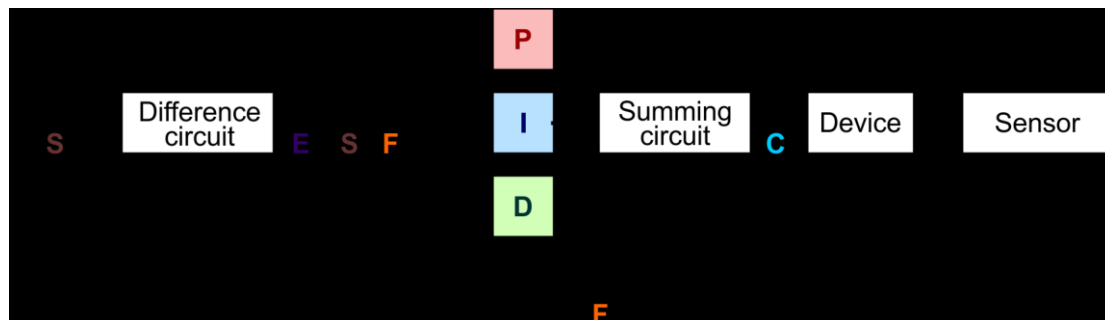
The sum of these generates a control signal, **C**, which adjusts the process by changing the position of a control valve or altering the power setting for a heater, for example. The relative importance of each component is adjusted by changing the value of the constants **P**, **I** and **D** by which each term is multiplied.

The following equation expresses this relationship mathematically.

$$C = (P \times E) + (I \times \int E dt) + (D \times \frac{dE}{dt})$$

Proportional Integral Derivative
↙ ↓ ↘

The next diagram illustrates the PID process:



The system generates the error signal, **E**, by comparing the set point, **S**, with the current condition, indicated by the feedback signal **F**. This error signal can be used to generate the '**P**', '**I**' and '**D**' components, which are then summed to create the control signal **C**. This changes the conditions in the device. The sensor records the new state of the device as a new feedback signal, and so the control process continues.

The microcontroller solution to this control problem uses digital techniques, not differential and integral calculus. However, the result is the same.

1.3 PID components:

The **P**roportional term is the simplest - it uses the current value of the error signal and 'weights' it by multiplying it by the constant **P**. However, it always leads to an offset, called the steady-state error, between the set point and the actual target. The reason for this is mathematical, but can be thought of in terms of the conditions occurring in the particular system.

For example, where an industrial tank of liquid is being heated, there must be heat losses. As the control signal nears the target value, it becomes ever smaller, causing very slow temperature changes. Eventually, the energy supplied is cancelled out by the heating losses. The temperature never reaches the target value. There is a continuing (steady-state) error between them.

Where the controller is filling a tank with liquid, there will be losses due to evaporation. As the liquid level nears the target value, the control signal falls to a low value and the filling operation slows. Eventually, the small amount of liquid added is cancelled out by evaporation losses - another steady-state error.

The **I**ntegral term sums (i.e. integrates,) all previous values of the error signal and 'weights' the result by multiplying it by the constant **I**.

The **D**erivative term checks whether conditions are changing too quickly or too slowly to achieve the target in reasonable time. To do so, it looks at the current rate of change of the error signal and 'weights' the result by multiplying it by the constant **D**.

- Increasing the value of **P** increases the speed at which the response of the controller rises, but too large a value can cause ringing (oscillation around the set point).
- Increasing the value of **I** increases the response time and eliminates the steady-state error. Since it sums all previous error signals, even a small steady-state error present for some time can create a significant response. Once again, too large a value can cause ringing.
- Increasing the value of **D** reduces overshoot and ringing but, in doing so, it slows the response to changing set points. Noise spikes on the error signal can appear as a rapid change in the state of the process and lead to incorrect changes in response.

The table below summarises these effects:

Value increased	Rise time	Overshoot	Settling time	Steady-state error
P	Decrease	Increase	Little effect	Decrease
I	Decrease	Increase	Increase	Decrease
D	Decrease	Decrease	Decrease	No effect

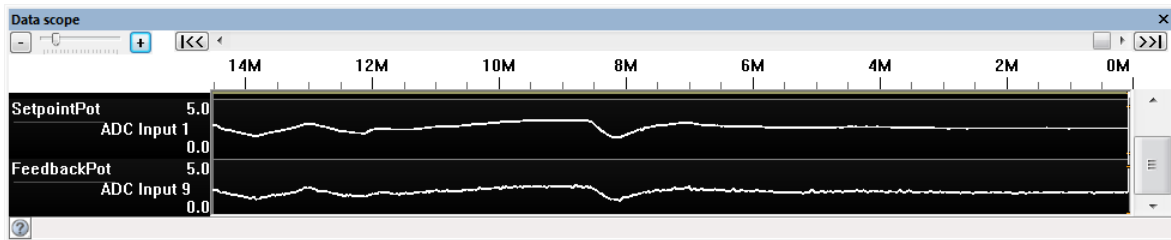
1.4 PID variations:

In some situations, better control is achieved by using only one or two of the components, giving rise to 'P controllers', 'PI controllers' etc. 'PI' controllers, for example, are common, where electrical noise in the error signal might affect a 'D' component.

Each combination has its own characteristic behavior, in terms of the responsiveness of the controller to an error, the rise time, overshoot, settling time, the steady-state error and the degree of ringing (oscillation).

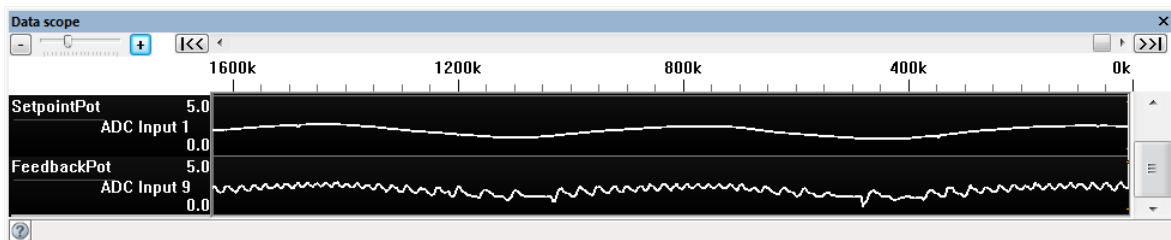
The diagrams that follow, taken from Flowcode PID programs, show some effects of the 'P', 'I' and 'D' components.

Example of P Controller steady-state error:



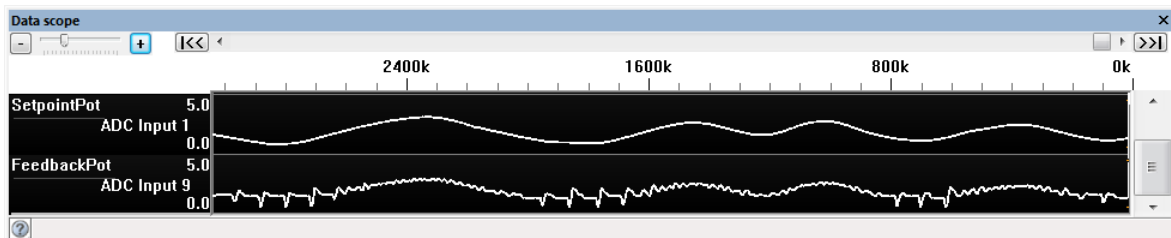
Comparing the waveforms, the set point signal is in the middle of the range whereas the feedback signal is slightly lower. This shows the fixed offset error you get from 'P' control alone.

Example of I Control integral windup



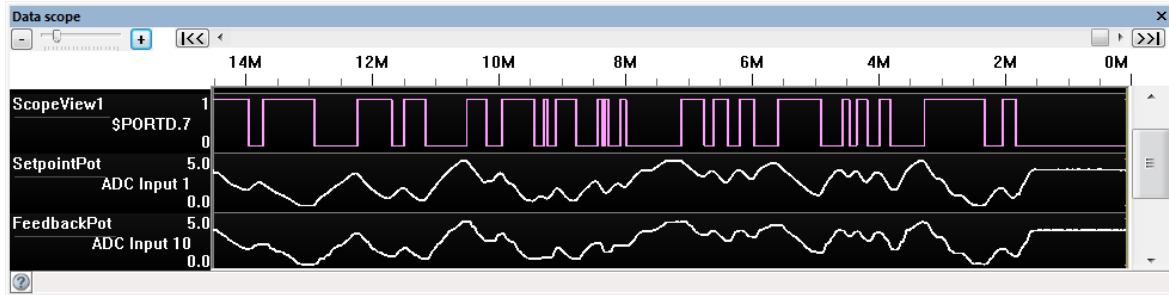
Comparing the waveforms, the set point signal varies smoothly whereas the feedback signal shows some oscillation, the result of too large a contribution from the 'I' component.

Example of D Control instability



Comparing the waveforms, the set point signal again varies smoothly whereas the feedback signal oscillates slightly, a sign of instability.

Example of PID tuned response



Comparing the waveforms, the set point signal varies smoothly. The feedback signal follows it closely. The top signal, 'ScopeView1', shows the signal on the motor direction control pin.

1.5 Manual 'tuning' of a PID system:

The rule to observe is that there are no rules to observe - it is an art not a science!

Typically, the aim is to adjust the '**P**', '**I**' and '**D**' constants so that the system overshoots slightly and then ease back steadily to the target.

In general terms:

- set the '**I**' and '**D**' constants to zero;
- increase '**P**' until the feedback shows some ringing;
- reduce '**P**' to half of this value.

Then:

- increase '**I**' until the steady-state error is eliminated, without causing overshoot or instability;
- increase '**D**' to combat any overshoot and dampen the feedback so that the target is achieved in 'reasonable' time.

2. DC Motors

2.1 Introduction to DC motors

DC motors convert electrical energy into rotational kinetic energy.

They usually consist of a coil rotating inside the magnetic field of either a permanent magnet or

an electromagnet. They rotate because the current

and magnetic field reverse at just the right time.

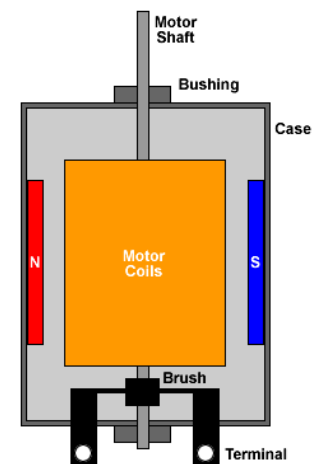
There are two problems to be solved then:

- making electrical connections to a rotating coil;
- reversing the current at the right time.

Two solutions are - use:

- slip rings with an AC supply;
- a commutator and 'brushes' with a DC supply.

The motors provided here use the second of these approaches.



2.2 DC motor principles:

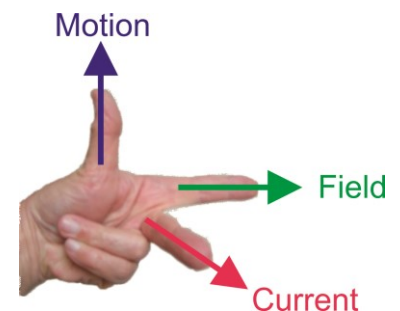
2.2.1 The underlying physics:

- A current is a flow of electrons, tiny 'particles' found in all atoms.
- Whenever electrons move, they generate a magnetic field.
- This interacts with the field of the magnets, causing attraction/repulsion, but at right-angles to the current direction and to the magnetic field.

Fleming's Left-hand (Motor) Rule:

John Ambrose Fleming devised a way to work out the direction a wire will move in:

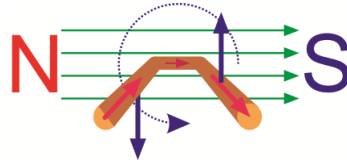
Clamp your **left**-hand to the corner of an imaginary box, so that thumb, fore finger and centre finger are all at right-angles to each other. Then, the **F**ore finger points along the magnetic **F**ield (from North pole to South pole,) the **C**entre finger shows **C**urrent (from positive battery terminal to negative) and the thu**M**b points in the direction of the resulting **M**otion.



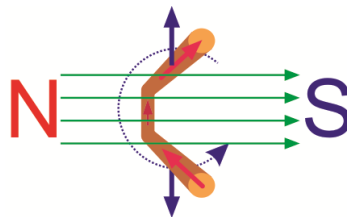
2.2.2 Making it rotate:

Most motors rotate, so how does that happen?

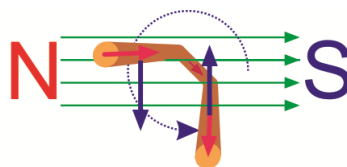
The diagram shows a cutaway coil of wire, disappearing into the sheet of paper. There is a magnetic field from left to right.



Current flows into the paper on the left side of the coil, and out of the paper on the right. Using the Left-hand Rule, the coil sides try to move in the directions shown by the blue arrows and so the coil starts to rotate. (There is no force on the back section of the coil, because the current flows parallel, not at right-angles, to the magnetic field.)



When the coil reaches the position shown in the second diagram, the forces on the two sides are in line, but in opposite directions. There is no longer a twisting effect, but momentum carries the coil past that position.



We now need to reverse the current, so that when the coil reaches the position shown in the third diagram, the forces on the coil keep it rotating. Provided that the current reverses again at the right time, then the rotation will continue.

There are two ways to reverse the current:

1. use **slip-rings** with brushes;
2. use a **commutator** with brushes.

2.2.3 Slip rings:

- Each end of the coil is connected to its own full brass ring.
- Electrical contact is made to each ring with a carbon brush, which moulds to the shape of the ring, reducing contact resistance.
- The coil is supplied with AC, and so the current automatically changes direction in step with the mains supply.

The picture shows one form of slip-ring.

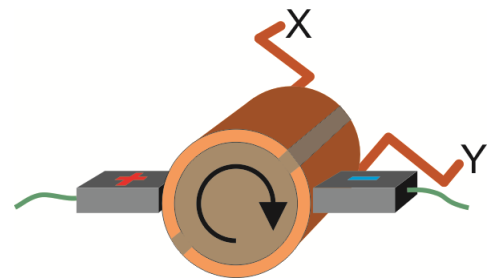


2.2.4 Commutator:

The diagram shows the design of a simple commutator, a brass drum, split into two halves, separated by an insulator.

Electrical contact is made by carbon brushes.

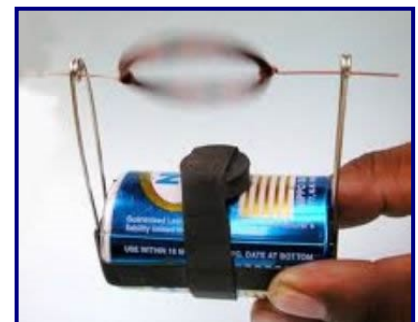
The coil is connected to the two halves at X and Y. As it rotates, X is connected to the positive supply for roughly half of the time, and then to the negative supply. At all times, Y is connected to the opposite supply to X.



2.2.5 Building a simple motor:

Single coil motors can be very easy to build. The following website has a guide on building your own simple single coil DC motor:

<http://www.instructables.com/id/Easy-FAST-DC-MOTOR/>

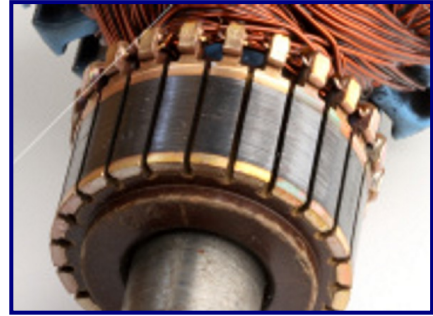


2.3 Multi-coil DC motors:

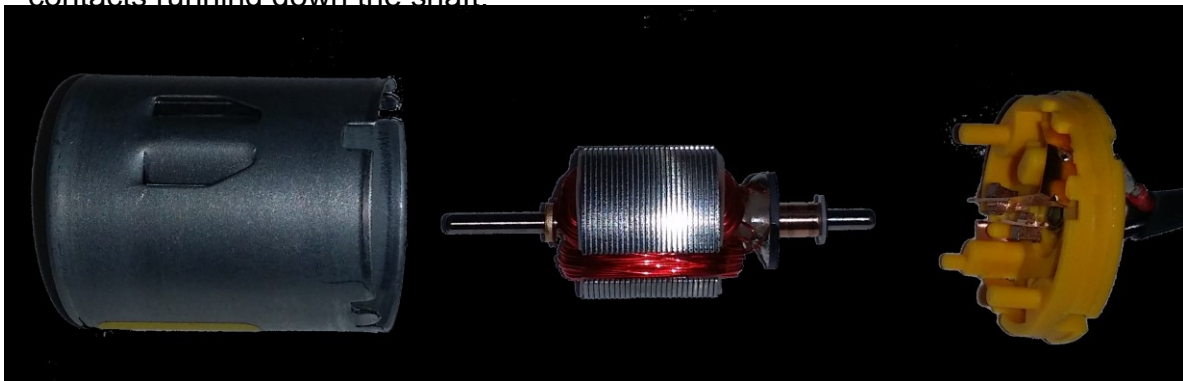
The problem with the single coil motor is that, with only two contacts on the brush, it is possible for the motor to be at rest in the crossover position between contacts. If then energised, it might short-circuit the contacts together, or might not conduct at all. The motor does not rotate!

The solution - use coils in the rotor. Most DC motors have at least three. Using three provides a low cost and efficient way of rotating the motor, ensuring that two contacts are always in contact with the brushes. As only one of the three coils has current passing through it at any given time, the non-active coils have time to cool down. The large number of coils in the rotor requires a more complicated commutator.

The picture shows a typical commutator used in a multi-coil motor. Its surface has been darkened by rubbing on the carbon brushes.



Shown below is a disassembled simple three coil motor. The coils are shown in the middle, made from red insulated wire and wrapped around a core of strips of laminated metal. These increase the efficiency of the motor by increasing the electrical resistance of the core and, hence, reducing eddy currents, induced by the magnetic field. Eddy currents cause unwanted electrical heating, wasting energy. The laminated core also retains the magnetic field within it, intensifying it in the region of the coils, thus, again, increasing the efficiency. The brushes in this example are two springy metal strips, which rest against the three copper contacts running down the shaft.

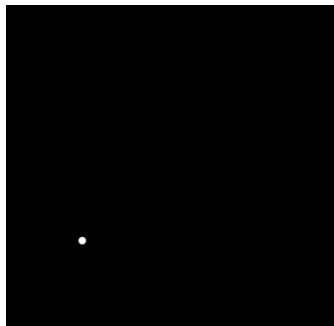
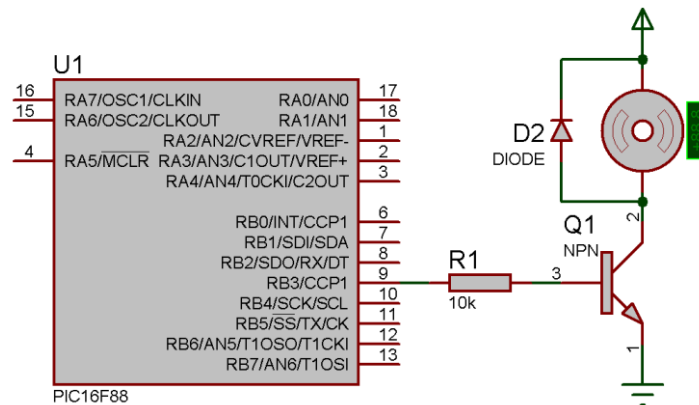


2.4 Driving DC Motors:

A DC motor generally requires an appreciable current, especially when starting the motor, to give it enough torque to start the motor spinning.

A standard microcontroller can output around 20mA on a given pin, which might be sufficient to drive a low power motor but nothing more.

To boost the current we can use a buffer subsystem using either bipolar (BJT) or field-effect (FET) transistors. Examples of each are shown below. The FET circuit uses an n-channel enhancement mode MOSFET. This is a voltage-controlled device which switches into conduction when sufficient voltage is applied at the input.



2.4.1 Diode protection:

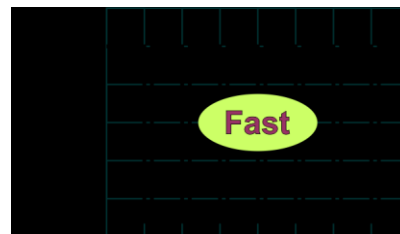
The circuits include a diode, connected in parallel with the motor. Its purpose is to protect the driving circuit and microcontroller from 'back e.m.f.' When the motor is turned off, the magnetic field in its coils collapses and generates a high reverse voltage, called the 'back e.m.f.' The diode is reverse-biased for the system power supply, but conducts when the 'back e.m.f.' occurs, dissipating its energy into the system power supply. This problem occurs with all coil-based devices such as relays, solenoids and inductors.

2.4.2 Controlling the speed of the motor:

One way to control the speed of a motor is to use a variable resistor in series with the motor. However, the variable resistor gets hot - this method is not energy efficient!



A better form of motor control uses **PWM** (pulse-width modulation) instead of a variable resistor. When the motor is switched on, all the supply voltage is across the motor and so none is 'wasted'. When the motor is off, no current flows and so no heat is generated. PWM uses this approach. It turns the motor on and off rapidly, but varies the length of time it is on. The longer this time, the faster the motor runs. The time within each cycle that the motor is on is called the 'mark'. The time for which it is off is called the 'space'. The following diagrams illustrate these ideas:

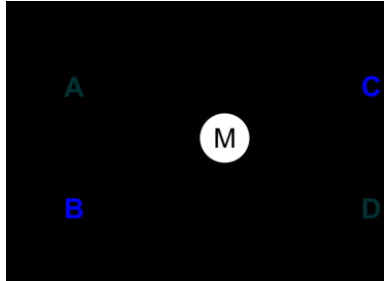


The **duty cycle** for the PWM signal is the proportion of the cycle for which the motor is switched on and is expressed as a percentage. For example, a duty cycle of 50% means that the motor is switched on for half the time.

The microcontroller offers PWM on some of its output pins, marked as 'CCP1', 'CCP2' etc. identifying PWM channel 1, channel 2 etc.

2.4.3 Controlling the direction of rotation:

Full control of a DC motor requires a mechanism for setting the direction of rotation. To reverse the rotation, the power supply to the motor's terminals must be reversed. The 'full bridge' or 'H-bridge' subsystem allows us to do just this. The principle is illustrated in the next diagram:



Closing switches **A** and **D** causes the motor to rotate in one direction. Closing switches **B** and **C** causes it to rotate in the other direction.

To stop the motor:

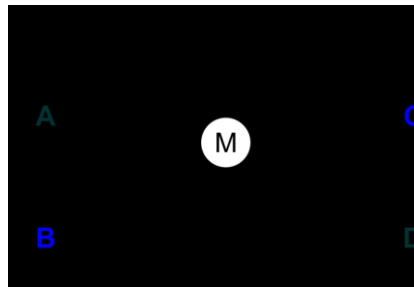
- open one of the closed switches.
- close either **A and C** or **B and D**.

In the first case, the motor will 'free-wheel' to rest.

In the second case, the motor terminals are short-circuited, dissipating the energy stored in the coil's magnetic field, causing the motor to brake to a halt.

It would be unwise to close both switches **A and B** or both **C and D**, as it would short-circuit the power supply. In practice, that is not possible.

In the next diagram, the mechanical switches are replaced by MOSFETs, but the principle is the same. Diodes have been added to protect the semiconductor devices.

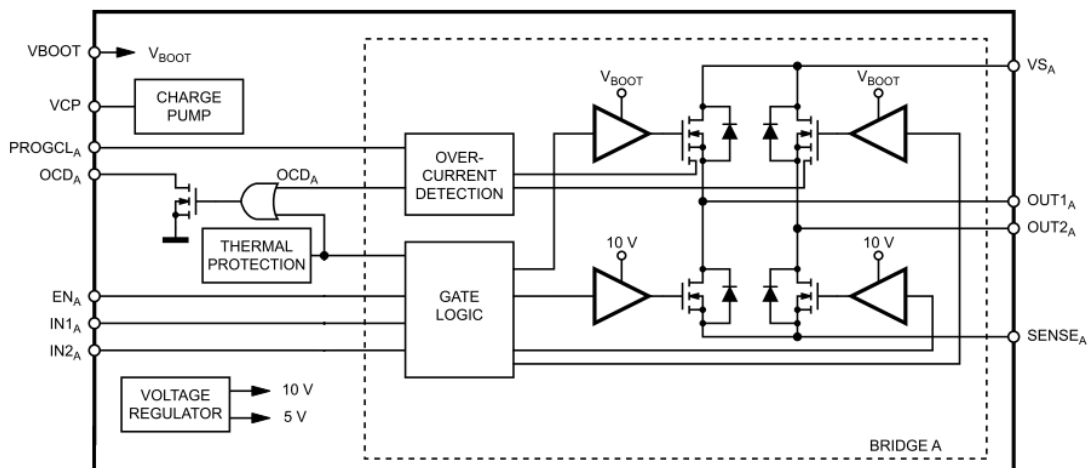


Using a PWM signal with one of the switches in the pair creates speed control.

2.4.4 Using a full bridge IC:

The full bridge circuit is too complex to construct from individual devices.. The Matrix solution uses a single chip, the L6206N driver IC, with two full bridge drivers built into it, to offer this control. MOSFET switches are better than BJT devices at reducing heating effects. At a current of 1A, the power dissipated in a MOSFET is ~ 0.1W.

The block diagram for the driver is shown below. It offers features such as thermal and overcurrent protection. There is an enable input which allows the motor to either brake or free wheel to a halt when the speed signal is turned off.



3. Matrix DC Motors Solution

The Matrix motors solution is made up of several boards each with its own separate functionality within the system.

EB091 or BL0032 dsPIC Programmer:

controls all operations within the system and enables the Ghost functionality.

EB090 Sensors board:

fitted with two Dual Trim Pot Sensor Modules (EBM006), allowing us to specify what we want the system to do.

EB094 FET driver board:

uses a dual H-bridge FET based driver IC to power the motors.

EB096 Motor speed trainer board:

features a DC motor with speed-monitoring capabilities.

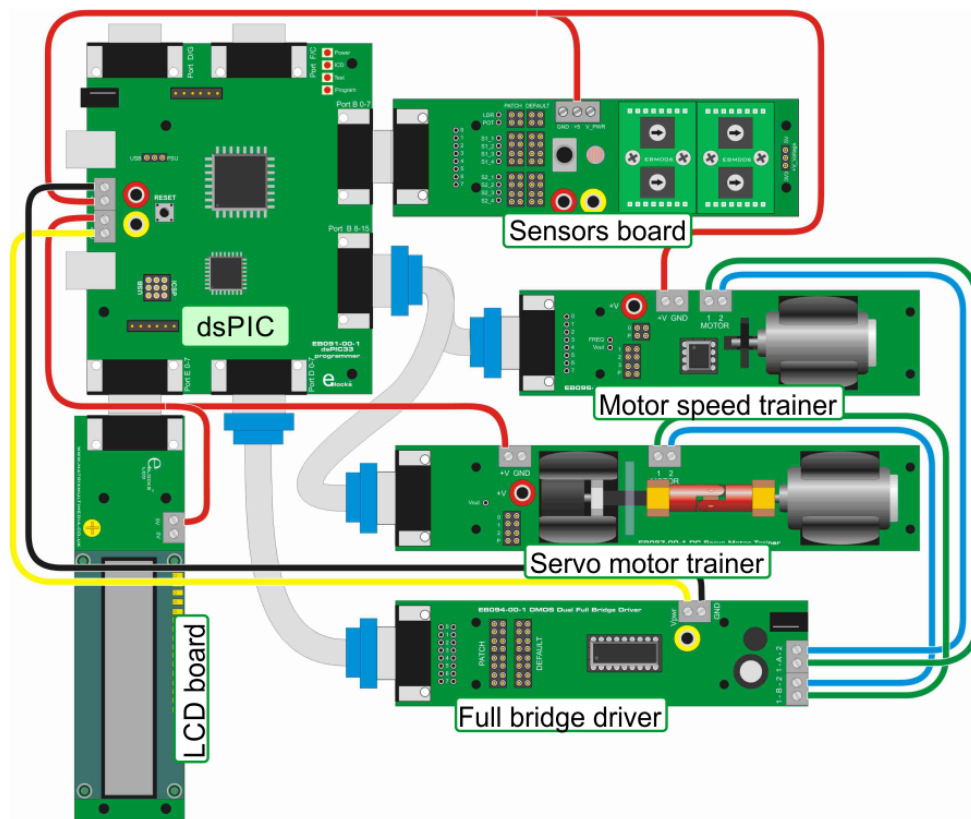
EB097 DC Servo motor trainer board:

features a DC motor with position-monitoring capabilities.

EB005 or BL0169 LCD board:

provides feedback to inform the user what is happening in the system.

The solution layout is shown below. Power leads are shown in red and black.



3.1 Setting up the motors solution hardware:

EB091 or BL0032 dsPIC33 programmer:

Check that:

- the voltage source jumper selects 'PSU';
- the programming source is 'USB' on jumper block J12-J13-J14.

EB090 Sensors board:

Check that:

- the patch selection jumpers are in the default position;
- the processor interface voltage selector is set to 3.3V.

EB096 Motor speed trainer board:

Check that:

- the 'FREQ' jumper is in position '0' (for raw digital output from the encoder);
- the 'VOUT' jumper are in the position '1' (analogue output).

EB097 DC Servo Motor trainer board:

Check that:

- the 'VOUT' jumper is in position '2', the analogue output from the pot.

EB094 FET driver board:

Check that:

- the patch selection jumpers are in the default position.

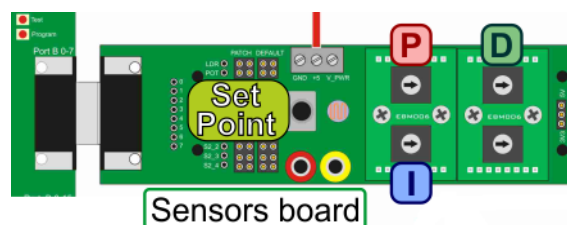
EB005 or BL0169 LCD board:

Check that:

- the patch selection jumpers are in the default position;
- the contrast control is set to make the display clearly legible.

3.2 Sensors board controls:

The diagram identifies the PID controls on the EB090 Sensors board



4. Flowcode Components

A Flowcode component is a combination of images, other components, flowchart macros etc., controlled by the Flowcode program. The diagram shows one such component - a robot arm.



These components are located in the Components toolbox, which is divided into sections such as Inputs and Outputs to help the user find what they want.

For more information about individual components, use the ‘Search’ facility on the Matrix Flowcode wiki <http://www.matrixtsl.com/wiki/>.

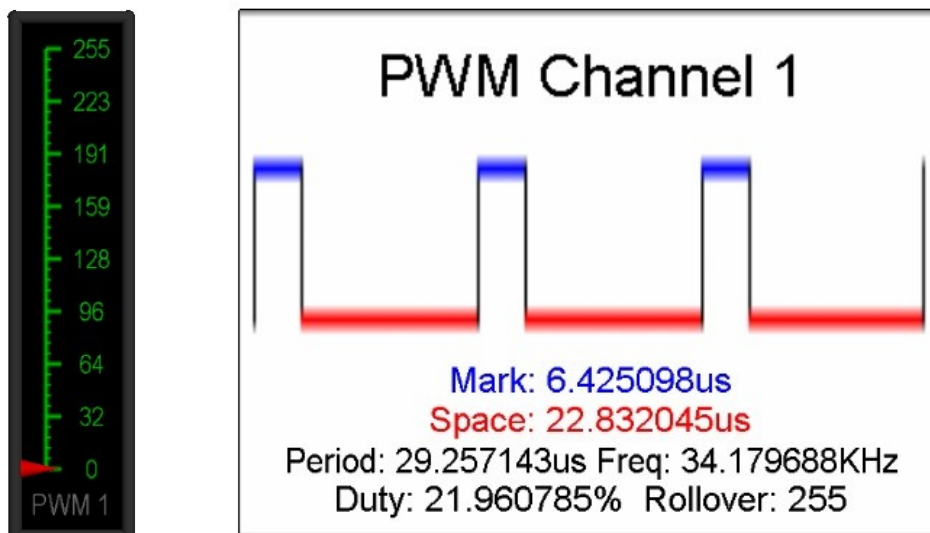
Flowcode components have a set of pre-written ‘Component Macro’s, - sections of code used to define some of its properties.

Within a program, Flowcode senses which components are in use and offers a menu of appropriate component macros when you ‘click’ on the icon.



4.1 The PWM component

The PWM component is used to generate the motor speed control signal for the motor driver IC. It has two graphical modes, analogue and digital. The analogue mode shows the duty cycle on an analogue scale. The digital mode shows the actual waveform and timings that will be produced.



4.1.1 Configuring the PWM component:

The component needs to be configured to correctly map the output signal to the right microcontroller pin to drive the motor. The next two diagrams detail the configuration of the PWM components for the two motor boards.

EB096 - Motor Speed Trainer

Properties

PWM1

Handle	PWM1
Type	PWM
Properties	
Connections	
Channel	Channel 1
Alternative pin	No
PWM Pin	\$PORTD.2
Remap Pin	\$PORTD.2
PWM Frequency	
Period Overflow	255
Prescaler	8
Period (us)	29.257143
Frequency (KHz)	34.179688
Simulation	
Representation	Analogue

Callouts for EB096:

- Selects which PWM channel (microcontroller pin,) the component is connected to.
- Allows an alternative pin to be used if available.
- Specifies the pin assigned to the selected PWM channel.
- Allows the hardware pin to be re-assigned to another pin.

EB097 - DC Servo Motor Trainer

Properties

PWM1

Handle	PWM1
Type	PWM
Properties	
Connections	
Channel	Channel 1
Alternative pin	No
PWM Pin	\$PORTD.6
Remap Pin	\$PORTD.6
PWM Frequency	
Period Overflow	255
Prescaler	8
Period (us)	29.257143
Frequency (KHz)	34.179688
Simulation	
Representation	Analogue

Callouts for EB097:

- Allows the user to specify the number of counts for one PWM period.
- Allows the user to change the number of program cycles per PWM cycle count.
- Displays the time to complete one PWM cycle.
- Displays the frequency of PWM cycles.
- Selects the display mode for the simulation panel.

The following component macros are available to use with the PWM component:

Enable:

activates the particular PWM and overwrites the default properties.

Disable:

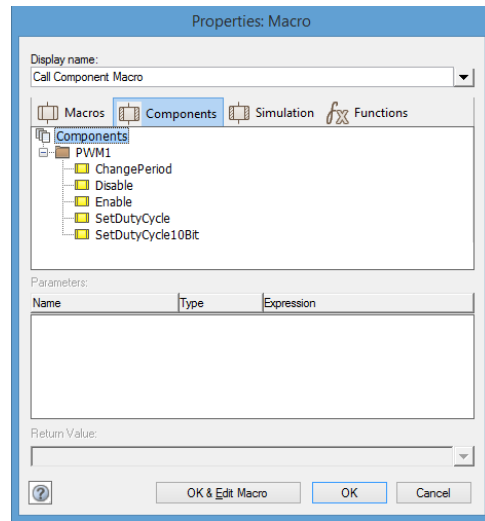
disables the particular PWM.

ChangePeriod:

sets the prescaler (used to reduce the clock speed for the component) and the period (i.e. the number used to represent a duty cycle of 100%).

SetDutyCycle:

sets the duty cycle for the PWM component. For example, if the period overflow value is 200, then a value of 100 produces a duty cycle of 50%. (The *SetDutyCycle10Bit* version uses a ten bit number instead of a byte.)



4.1.2 Controlling motor direction:

The motor direction is set by a output signal on a specific microcontroller pin.

Mode Of Operation	Pin State
Speed Control Trainer Forwards	D3 = 0
Speed Control Trainer Backwards	D3 = 1
Servo Control Trainer Forwards	D7 = 0
Servo Control Trainer Backwards	D7 = 1

When the motor direction pin is logic **0**, a **higher** PWM duty cycle results in faster motor movement.

For example a duty cycle of 0 represents no voltage across the motor while a duty cycle of 255 represents maximum voltage.

When the motor direction pin is logic **1**, a **lower** PWM duty cycle represents faster motor movement.

4.2 The ADC potentiometer component:

The ADC component is used to create and adjust the control signals produced initially on the 'pot's on the EB090 Sensor board. These are the control set point signal, the feedback signal and signals for the **P**, **I** and **D** constants. They are compared with a reference voltage, V_{REF} . There are a range of representations for the ADC component, including rotating knobs or sliders. All the ADC components share the same fundamental properties to set up the parameters for the analogue to digital conversion.

The set point and feedback signals are configured as follows:

Control Set point.

Angle the control points to at its lowest setting, measured counter-clockwise

Angle through which the control sweeps as it turns from minimum to maximum

Select the colour of the potentiometer control cap and pointer.

Analogue input channel - which pin is the analogue input connected to?

Category	Property	Value
Component	Handle	SetpointPot
Component	Type	Potentiometer (Colour)
Simulation	Start Angle	225.000000
	Sweep Angle	270.000000
	Cap Color	0000C0
	Pointer Color	FFFFFF
Connections	Channel	An 1
	Settings	
Settings	VRef voltage	500
	VRef option	VDD
	Conversion speed	FRC
	Aquisition cycles	40

Speed Feedback.

V_{REF} voltage (x 10mV). Default = 500 (=5V)

Defines what is used as the V_{REF} source.

Clock setting to set how fast the ADC peripheral is driven.

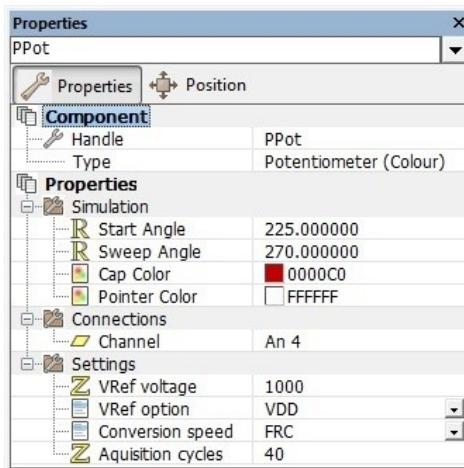
Number of microseconds to wait before starting the ADC sample.

Category	Property	Value
Component	Handle	FeedbackPot
Component	Type	Potentiometer (Colour)
Simulation	Start Angle	225.000000
	Sweep Angle	270.000000
	Cap Color	0000C0
	Pointer Color	FFFFFF
Connections	Channel	An 9
	Settings	
Settings	VRef voltage	500
	VRef option	VDD
	Conversion speed	FRC
	Aquisition cycles	40

The Servo Feedback ADC is configured in the same way as the Speed Feedback ADC except that it selects channel 'An 10' as its connection to the microcontroller.

The ADC components that allow the user to set values for the **P**, the **I** and the **D** constants are configured in a very similar way. The only changes are the reference voltage, V_{REF} , which is given a parameter of '1000' and the channel (pin) used to communicate with the microcontroller.

The 'Properties' of the 'PPot' component are shown below.



The 'IPot' and 'DPot' components are configured in the same way except that the 'IPot' uses channel 'An 5' and the 'DPot' channel 'An 6'.

There are twelve component macros available for use with the ADC component.

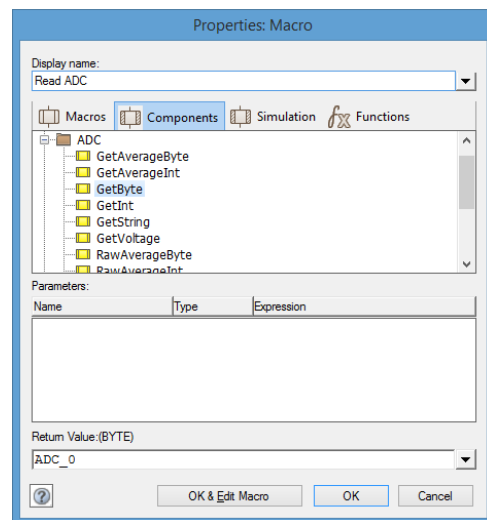
They include:

GetByte, GetInt, GetString, GetVoltage:

reads the value produced by the ADC component as a byte/integer/string or raw voltage and stores the result in the variable in the 'Return Value' box.

GetAverageByte, GetAverageInt:

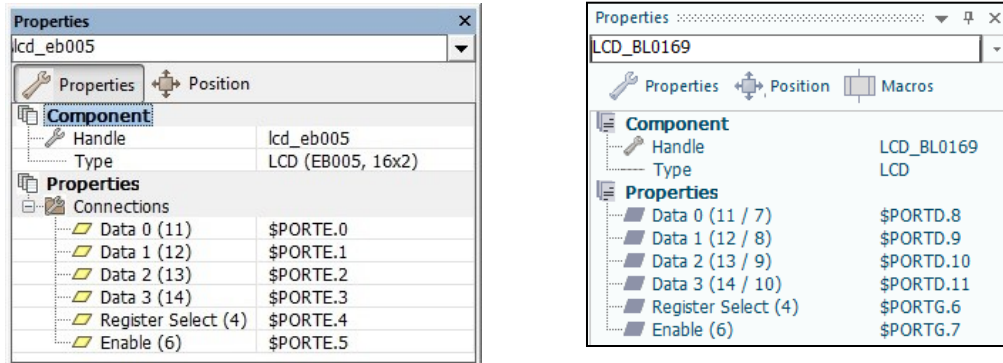
carry out the same function but average the value read over a period of time, specified when configuring the macro.



4.3 The LCD component

The LCD component is used to display the current variables in the system such as the set point, the feedback signal, the control signal and the **P**, **I** and **D** constants.

The following properties specify the pins used to connect the appropriate LCD component to the microcontroller:



There are twelve component macros available for use with the LCD component. They include:

Start:

initialises the hardware and clears the display.

Cursor:

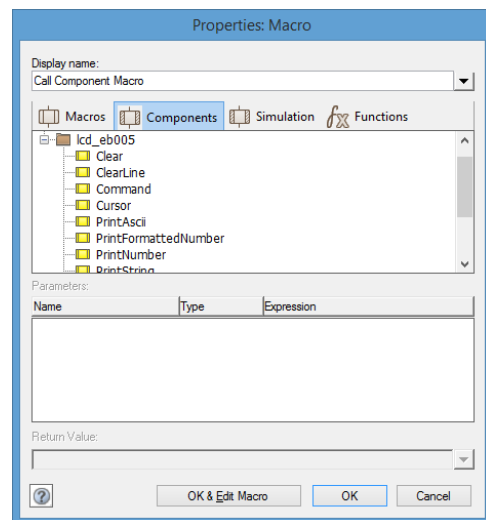
moves the cursor to the position specified by the user.

Clear:

clears the contents currently being displayed.

PrintAscii, PrintFormattedNumber, PrintNumber, PrintString:

specify what format the input to the LCD takes.



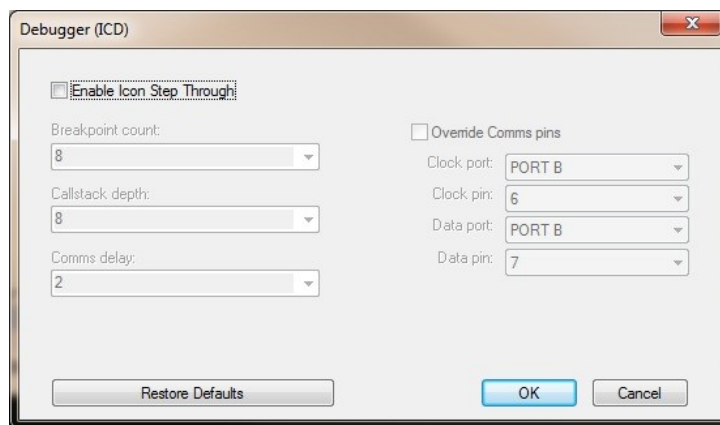
4.4 Ghost monitoring

'Ghost' is a set of technologies provided by the EB091 or BL0032 dsPIC E-block to allow the Flowcode software to monitor hardware activity in real time. In this module, it allows the user to monitor the behaviour of the motors and to measure the effect of changing the control conditions.

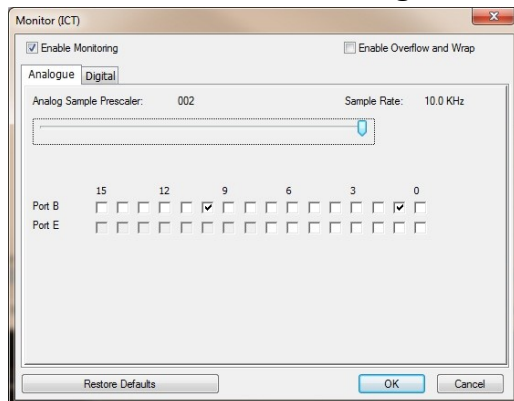
With the EB091 or BL0032 hardware connected via USB to the computer, click the 'Ghost' button to enable 'Ghost' technology.

The settings are accessed via the Debug -> Ghost -> Menu. This provides access to the 'Debugger' (ICD) and 'Monitor' functions. The recommended settings, shown below, disable the 'Debugger' and configure monitoring of a range of both analogue and digital signals:

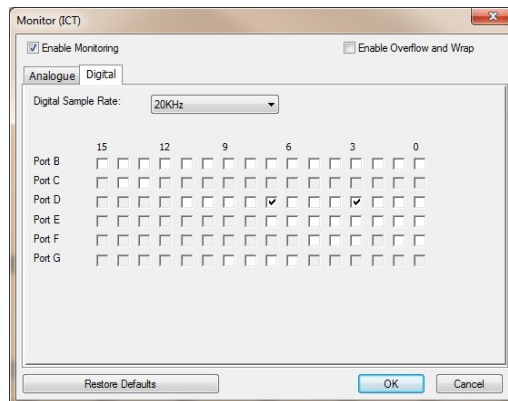
Recommended ICD settings:



Recommended ICT settings:




Analogue

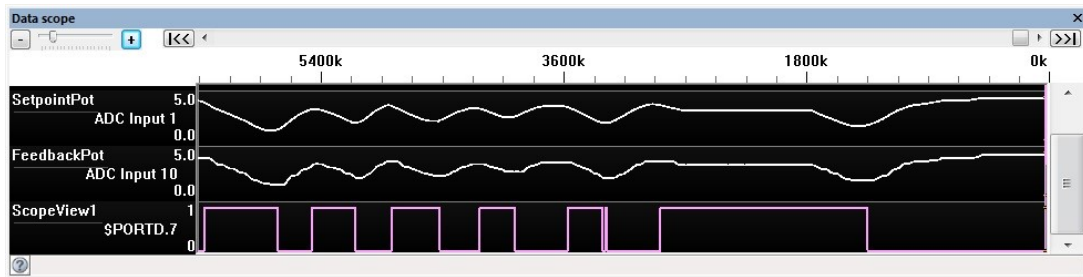


Digital

Once the ICD and ICT settings are configured in this way, you need to add appropriate channels to the 'Data scope' window.

- The set point and feedback signals are added by setting the 'Scope Traces' component property in the Properties panel to 'Yes'.
- The motor direction control pin is added using the 'Scope monitor' component  available from the Tools component category. (Use pin D3 for the speed control motor or D7 for the servo motor.)

The following diagram shows a typical view of the 'Set point', 'Feedback' and 'Motor Direction' signals for the servo motor board. The 'Motor Direction' signal is the 'ScopeView1' trace.



You can add a total of four other signals to the 'Data scope' window using the 'Scope monitor' tool.

To add PWM signals, in the 'ICT Settings - Digital' window, select pins D2 and D6, specified when configuring the PWM components earlier.

4.5 DSP components:

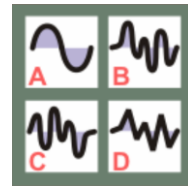
The DSP components in Flowcode can be a bit imposing when you first meet them. The Flowcode wiki site contains a comprehensive guide to the DSP components, how they plug together and operate.

<http://www.matrixtsl.com/wiki/index.php?title=DSP>

There are specific examples for each of the DSP components towards the bottom of this page in the section marked DSP.

<http://www.matrixtsl.com/wiki/index.php?title=Components>

Any DSP system must include a DSP System component, found in the DSP Toolbox. This component creates and manages the buffers (data stores) that link different DSP components and allow communication between them.



It is configured in the usual way using the Properties panel.

Number of buffers required.

Are all buffers same type and length?

Names used to identify the individual buffers.

Maximum number of bits in one data item stored in the buffer.

Is data signed (allowing negative values) or unsigned?

Number of data items stored in the buffer.

Properties	
DSPSystem1	
Properties	Position
Component	
Handle	DSPSystem1
Type	DSP System
Properties	
Buffer Count	3
Simple Mode	Yes
Buffer A Name	Buffer A
Buffer B Name	Buffer B
Buffer C Name	Buffer C
Simple Settings	
Bit Depth	16-bit
Sign	Signed
Size	16

There are six component macros available for use with the component.

They include:

Initialise:

initialises the DSP system.

GetBuff, SetBuff:

extracts / inserts data into the specified buffer.

SetBufferIndex:

manually selects a buffer and position.

TickAllBuffers, TickSpecificBuffer:

moves to the next segment of each / a nominated buffer.

Properties: Macro

Display name: Call Component Macro

Macros Components Simulation Functions

Components

- DSPSystem1
 - GetBuff
 - Initialise
 - SetBuff
 - SetBuffer_Index
 - TickAllBuffers
 - TickSpecificBuffer

Parameters:

Name	Type	Expression

Return Value:

OK & Edit Macro OK Cancel

4.5.1 DSP chain:

The DSP chain is an assembly of DSP components linked together to perform a specific function.

The DSP components in the DSP chain shown here are:

- an input component, to allow us to pass values into the system;
- a control component, to perform the P, PI and PID calculation;
- an output component, to collect values from the system.

In addition, there are:

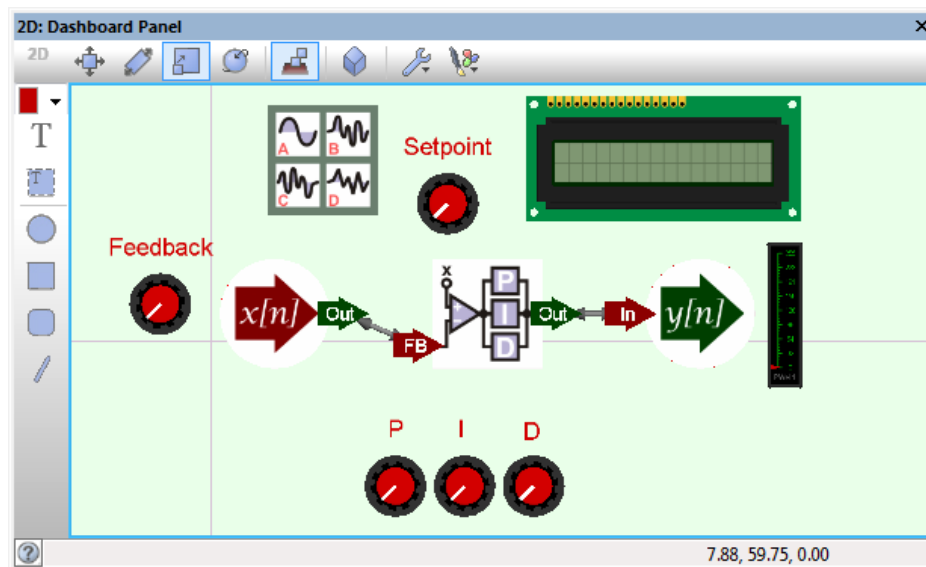
- controls for the Set point and Feedback signals;
- controls for the **P**, **I** and **D** constants;
- images for the output of the LCD and PWM components.

Each DSP component is called in turn by the 'DSPchain' macro.

This macro is part of the 'Timer interrupt' macro allowing the DSP code to be run on a regular cycle.

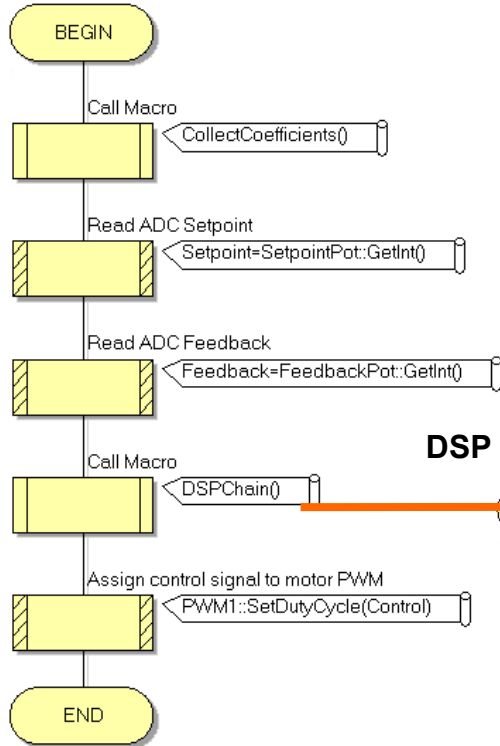
The code in the 'Main' macro runs independently to drive slower devices such as the LCD.

The complete DSP chain is shown below:

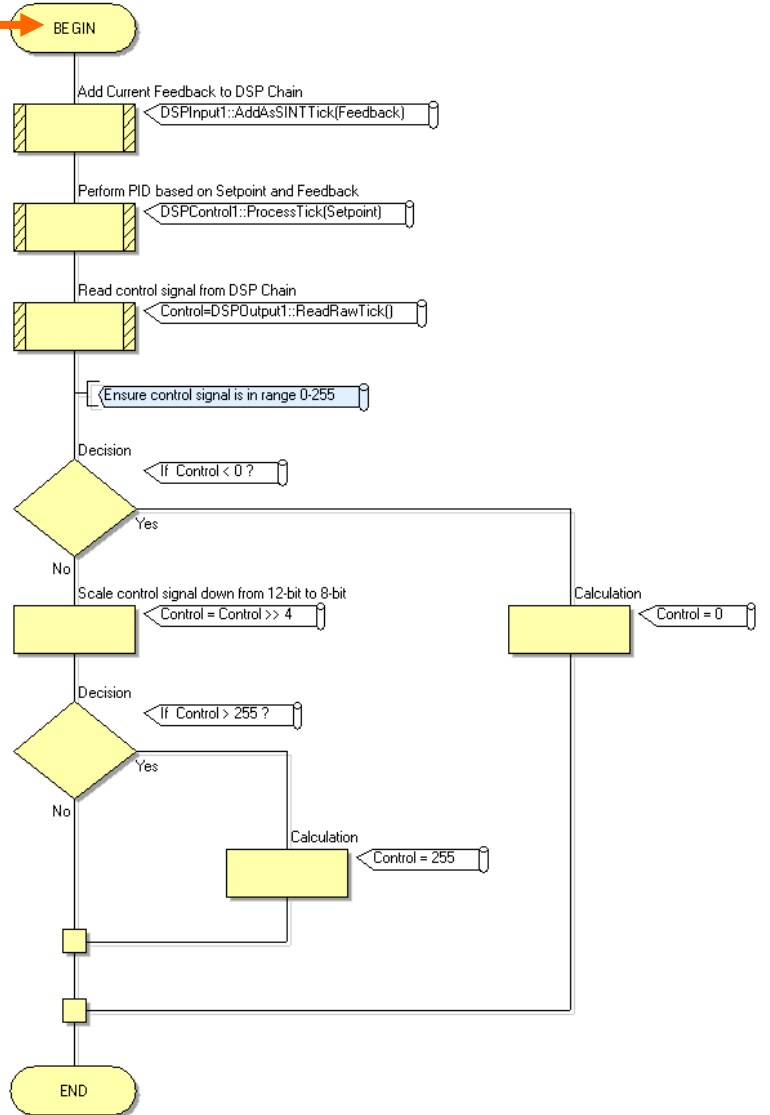


The Flowcode programs for the 'Timer interrupt' and 'DSP chain' macros is detailed below. The word 'Coefficient' refers to the constants **P**, **I** and **D**:

Timer interrupt macro



DSP chain macro



5 The programs:

The programs are divided into two groups:

- 5.1 speed** control;
- 5.2 position** control.

In the **speed** control section, the programs cover:

- 5.1.1 Open loop control;
- 5.1.2 Feedback;
- 5.1.3 Proportional (P) control;
- 5.1.4 Proportional / Integral (PI) control;
- 5.1.5 Proportional / Integral / Derivative (PID) control.

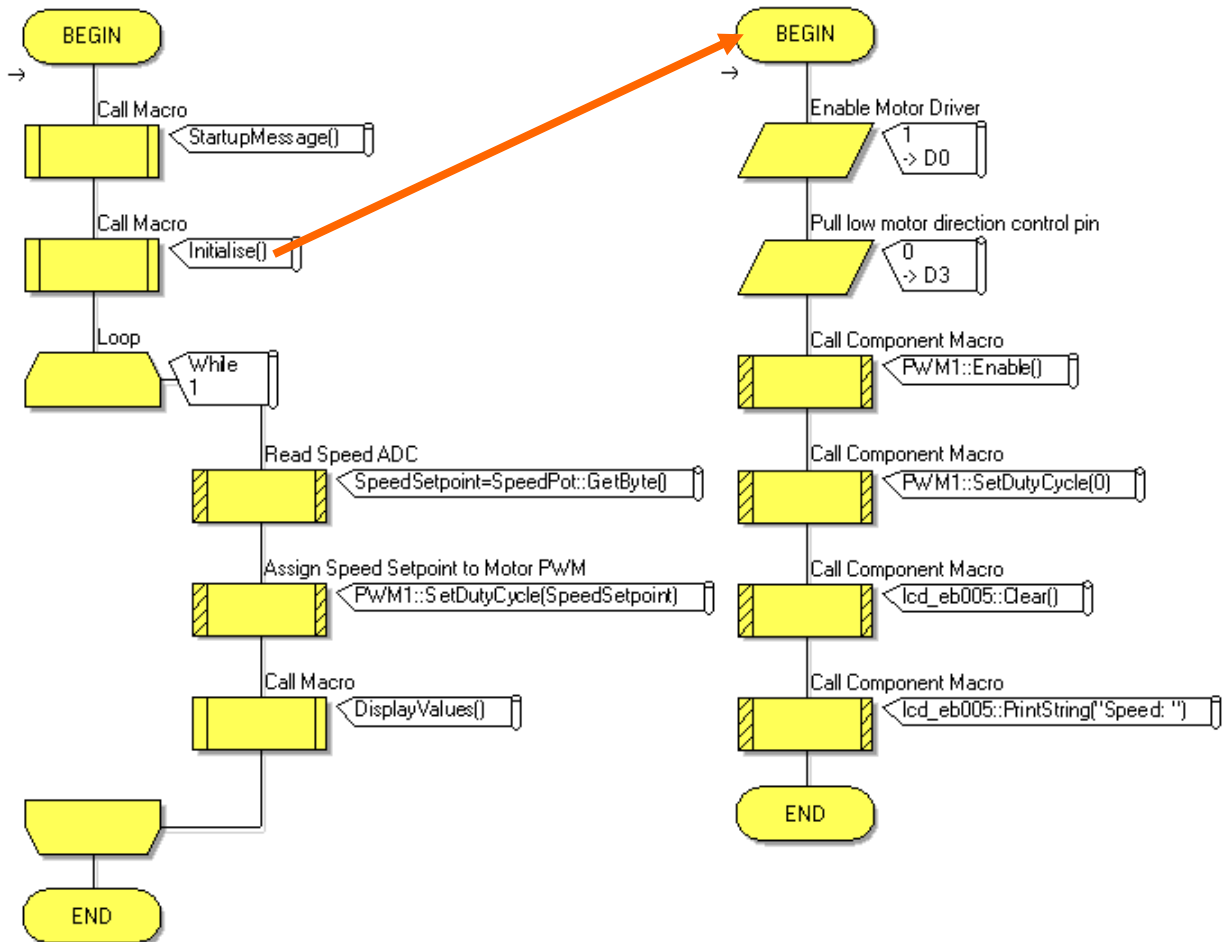
In the **position** control section, the programs cover:

- 5.2.1 Open loop control;
- 5.2.2 Feedback;
- 5.2.3 Proportional (P) control;
- 5.2.4 Proportional / Integral (PI) control;
- 5.2.5 Proportional / Integral / Derivative (PID) control.

5.1.1 Speed control - Open loop control

Program overview:

The program reads the set point potentiometer, uses the value obtained to generate the motor speed PWM signal and displays the value on the LCD. (The duty cycle is specified as a byte - 0 ⇒0%, 255 ⇒100% duty cycle.)



Program in detail:

- The 'StartupMessage' macro activates the LCD and begins with the message "Speed Control 1" for 2 seconds.
- The 'Initialise' macro is shown in the diagram above. Its function is:
 - send an enable signal to the motor driver IC;
 - set the direction of rotation;
 - enable the PWM component;
 - give it an initial duty cycle of 0%;
 - clear the LCD display;
 - print the word "speed" ready for the value to be added.
- The loop then repeatedly reads the value produced by the set point 'pot', uses it to modify the PWM duty cycle and displays it on the LCD.

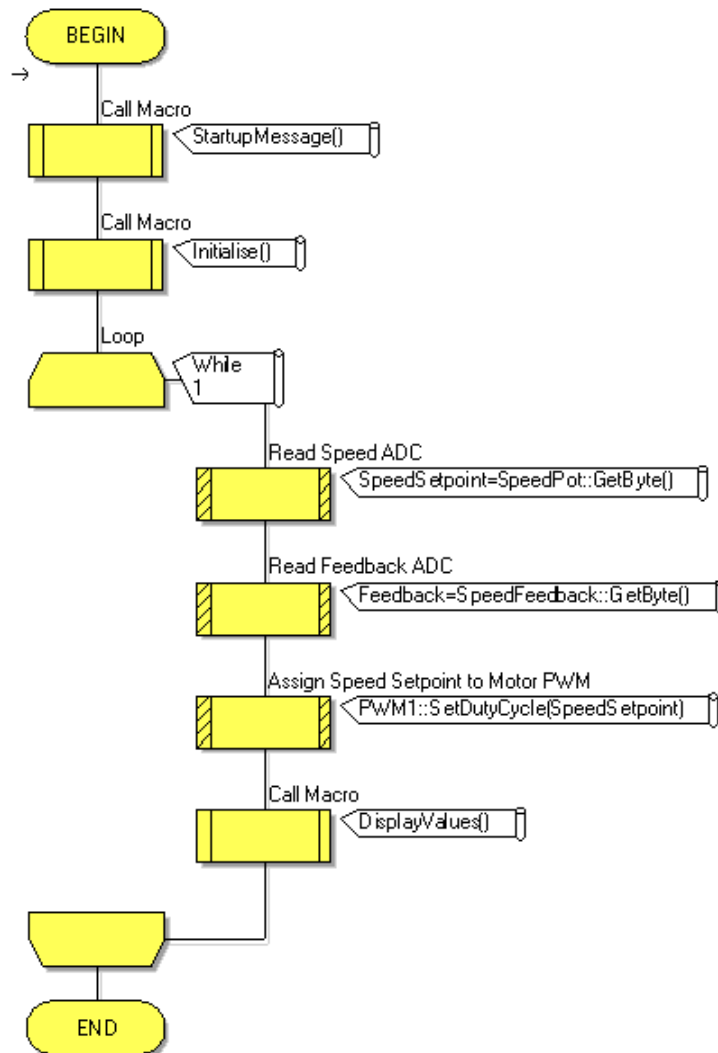
Exercise:

- Adjust the set point potentiometer and record what setting allows the motor to start spinning.
- Once the motor is spinning what value can the set point be lowered to before the motor stops spinning?
- With the motor spinning at a low speed, carefully slow down the motor by sitting your finger on it. Feel the torque driving the motor to rotate. Is the torque constant or does it vary as the motor rotates?

5.1.2 Speed control with feedback – Closing the loop

Program overview:

This program builds on the previous one, using the feedback signal to indicate the speed of the motor on the LCD. The display now shows target speed (from the set point 'pot',) and current speed (from the feedback signal).



Program in detail:

- The 'StartupMessage' macro activates the LCD and now begins with the message "Speed Control 2" for 2 seconds.
 - The 'Initialise' macro is identical to that in the previous program, except that it now displays the value of the feedback signal as well.
 - On the LCD:
 - 'speed' gives the target speed, set by the set point 'pot';
 - 'Fback' gives the current speed given by the feedback signal.

- The loop then repeatedly:
 - reads the value produced by the set point 'pot', modifies the PWM duty cycle and displays it as 'speed' on the LCD;
 - reads the feedback signal and displays it as 'Fback'.

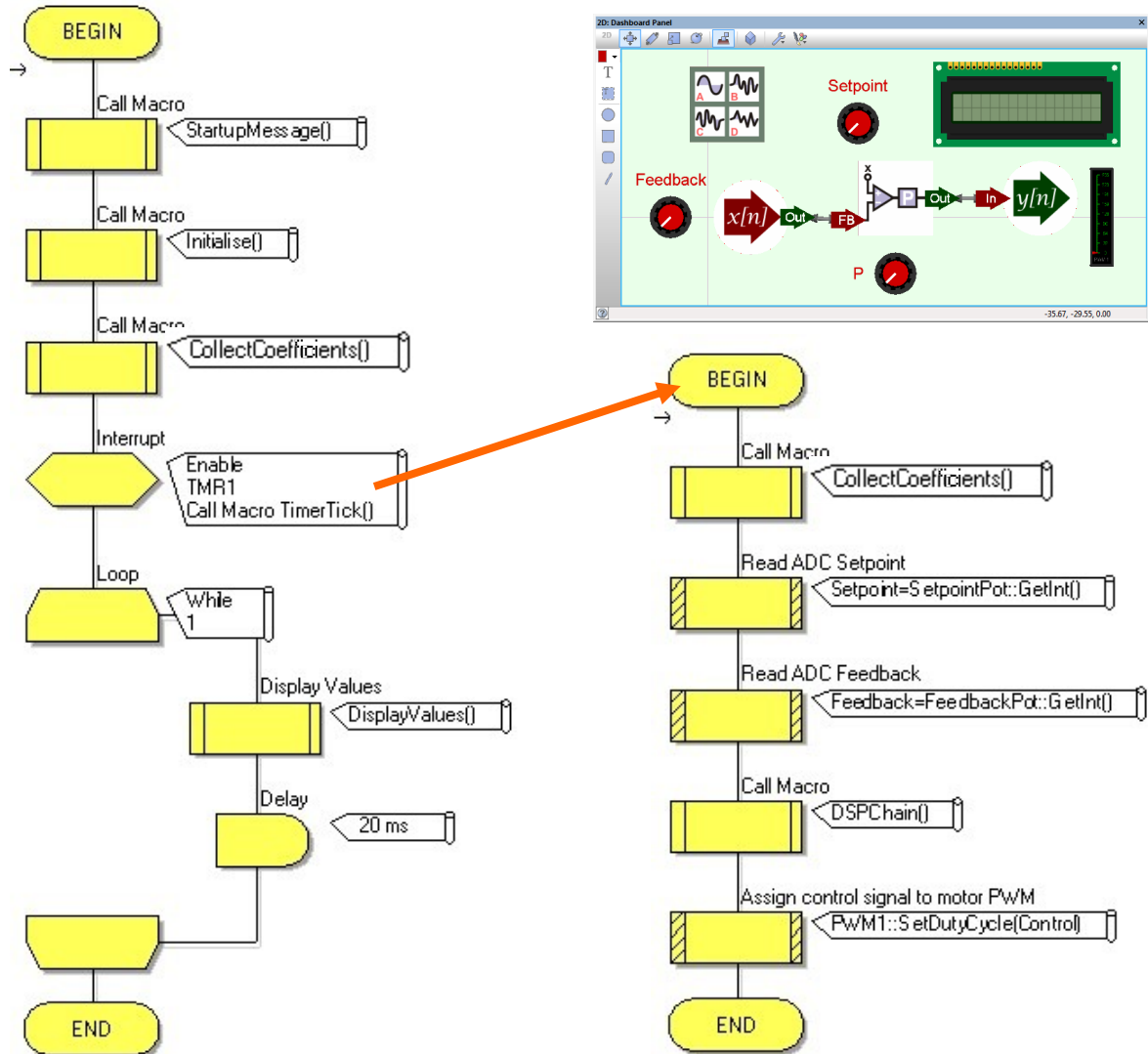
Exercise:

- Adjust the set point 'pot' from min to max in increments of 10 using the LCD as a guide;
- Record the corresponding value of feedback shown on the LCD;
- Use the values to create a chart of set point vs feedback.

5.1.3 Speed control - Proportional (P) control

Program overview:

This program adjusts the PWM output, using the feedback signal to reduce the error between the target and actual speeds, represented by the set point and feedback signals. The timer interrupt macro allows regular timing of samples, unaffected by other operations like updates to the LCD.



Program in detail:

- The 'StartupMessage' macro activates the LCD and now begins with the message "Speed Control 3 P Control" for 2 seconds.
- The 'Initialise' macro does exactly the same as before. It also initialises the DSP system and displays the values of the 'P' signal, the set point signal (as 'SP'), the feedback signal (as 'FB') and the control output signal (as 'C').
- The 'Collect coefficients' macro reads the voltage on the 'P' 'pot' and copies it into the 'P' constant.
- The 'TimerTick' macro is triggered repeatedly, every time the internal timer reaching its maximum value. It reads the current values of the set point and

feedback signals and uses them in the 'DSPChain' macro described earlier to update the value of the 'control' signal, using the relationships:

$$\mathbf{Error = Set\ point - Feedback}$$

$$\mathbf{and\ Control = Set\ point + (P * Error)}$$

It uses the value of 'control' to modify the duty cycle of the PWM signal.

- In the meantime, the loop is repeatedly updating the LCD with current values of the '**P**', '**SP**' '**FB**' and '**C**' signals.

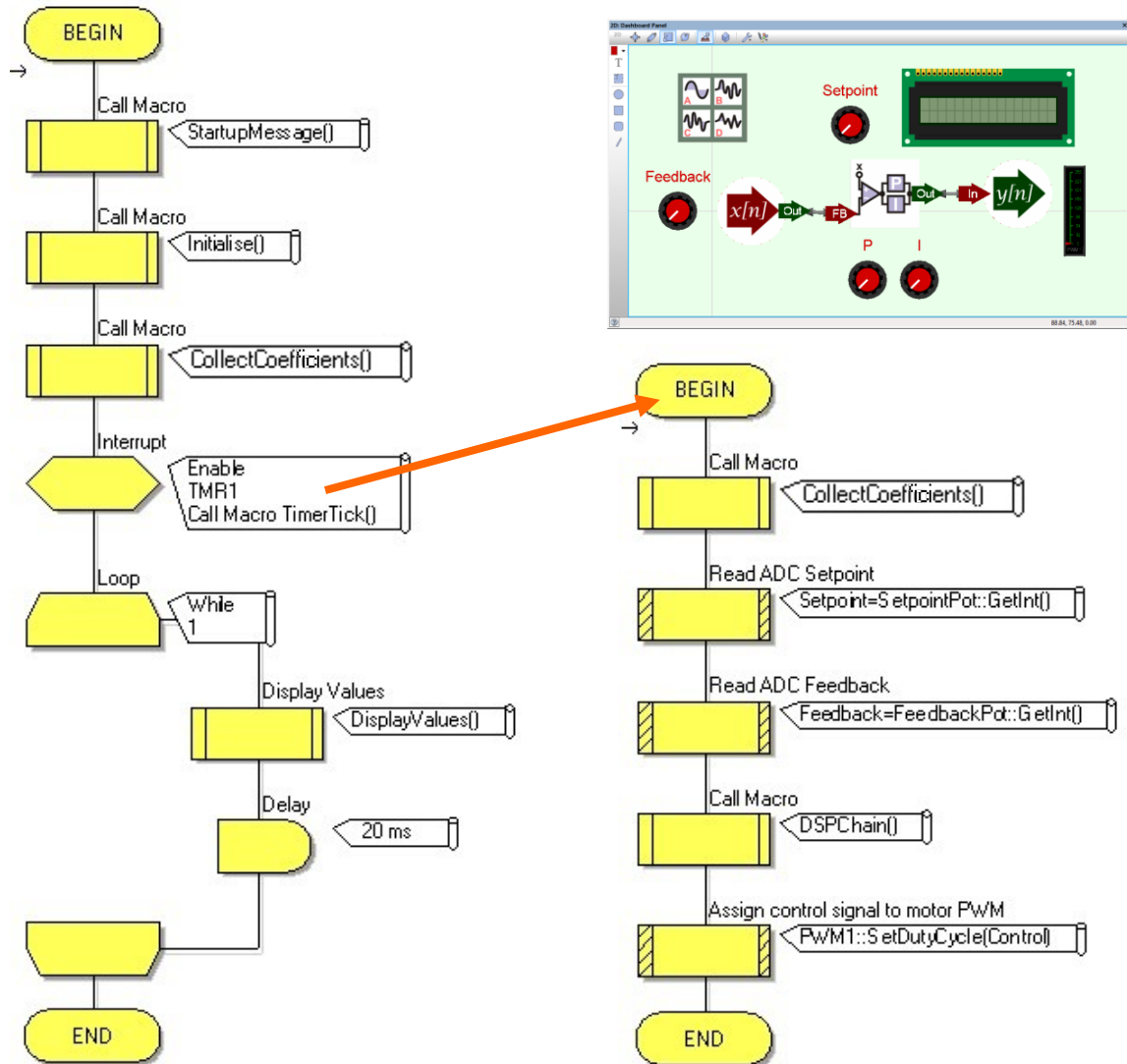
Exercise:

- Use Ghost to monitor the set point and the feedback signals.
- Adjust the **P** potentiometer and record the difference between the two values at various different settings of '**P**'.

5.1.4 Speed control - Proportional and Integral (PI) control

Program overview:

This program modifies the previous one by adding the integral component 'I' to reduce the steady-state error.



Program in detail:

- The 'StartupMessage' macro activates the LCD and displays the message "Speed Control 4 PI Control" for 2 seconds.
- The 'Initialise' macro does exactly the same as in the last program but also displays the value of the 'I' signal.
- The 'Collect coefficients' macro reads the voltage on the 'P' 'pot', as before, and copies it into the 'P' constant and then does the same for the 'I' 'pot'.

- Once again, the 'TimerTick macro' is triggered repeatedly, whenever the internal timer reaching its maximum value. It reads the current values of the signals and uses them to update the value of the 'control' signal, now using the relationship:

$$\mathbf{Control = Set\ point + (P * (Error - PrevError + (Error / I)))}$$

where **Error = Set point – Feedback**

and **PrevError = Error the last time around**

It uses the value of 'control' to modify the duty cycle of the PWM signal, as before.

- The loop repeatedly updates the LCD with current values of the '**P**', '**I**', '**SP**', '**FB**' and '**C**' signals.

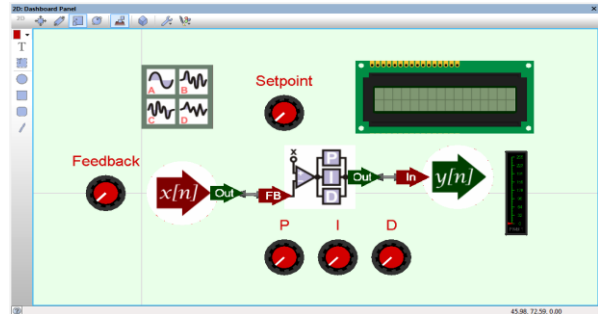
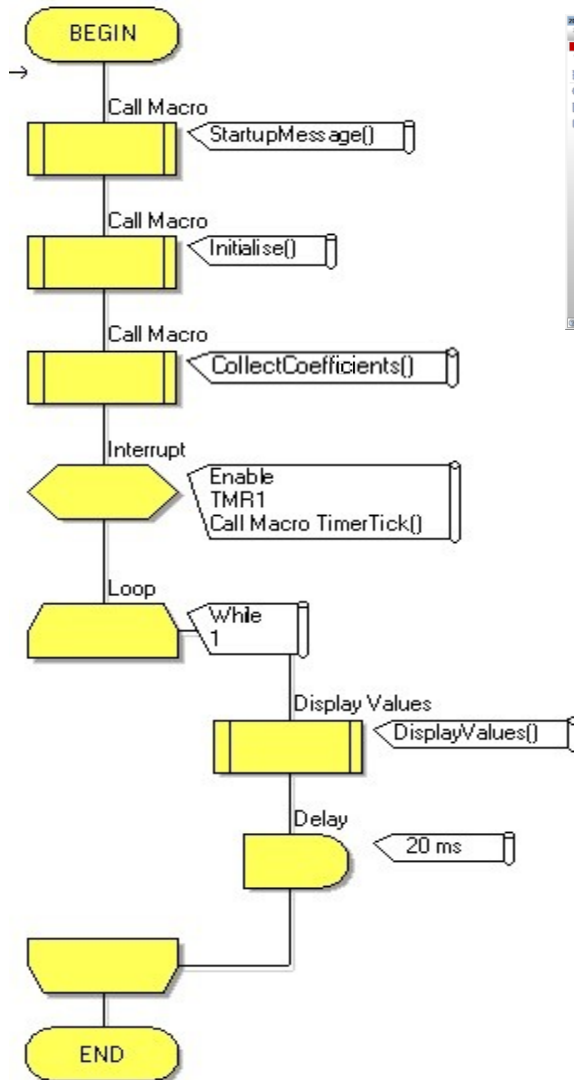
Exercise:

- Use Ghost to monitor the set point and the feedback signals.
- Adjust the '**I**' pot to see how this effects the incremental nudges aimed to eliminate the steady-state error.

5.1.5 Speed control - Proportional, Integral and Derivative (PID) control

Program overview:

This program continues the development by taking PI control and trying to further reduce errors by adding the Derivative component and trying to predict what will happen next.



Program in detail:

- The 'StartupMessage' macro activates the LCD and displays program title.
- The 'Initialise' macro does the same as before but adds the value of the 'D' signal to the LCD.
- The 'Collect coefficients' macro reads signals from the 'P', 'I' and 'D' 'pots', and copies the values into the 'P', 'I' and 'D' constants.
- Then the 'TimerTick macro' uses these to update the 'control' signal every time it is triggered. It now uses the relationship:

$$\text{Control} = \text{Set point} + (P * (\text{Error} - \text{Prev_Error} + (\text{Error} / I) + D * (\text{Error} - (2 * \text{Prev_Error}) + \text{Prev_Prev_Error})))$$

where **Error = Set point – Feedback**
and **PrevError = Error the last time around**
and **PrevPrevError = PrevError the last time around**

This value of 'control' then modifies the duty cycle of the PWM signal.

- The loop repeatedly updates the LCD with current values of '**P**', '**I**', '**SP**' '**FB**' and '**C**' signals, now with the addition of the '**D**' constant.

Exercise:

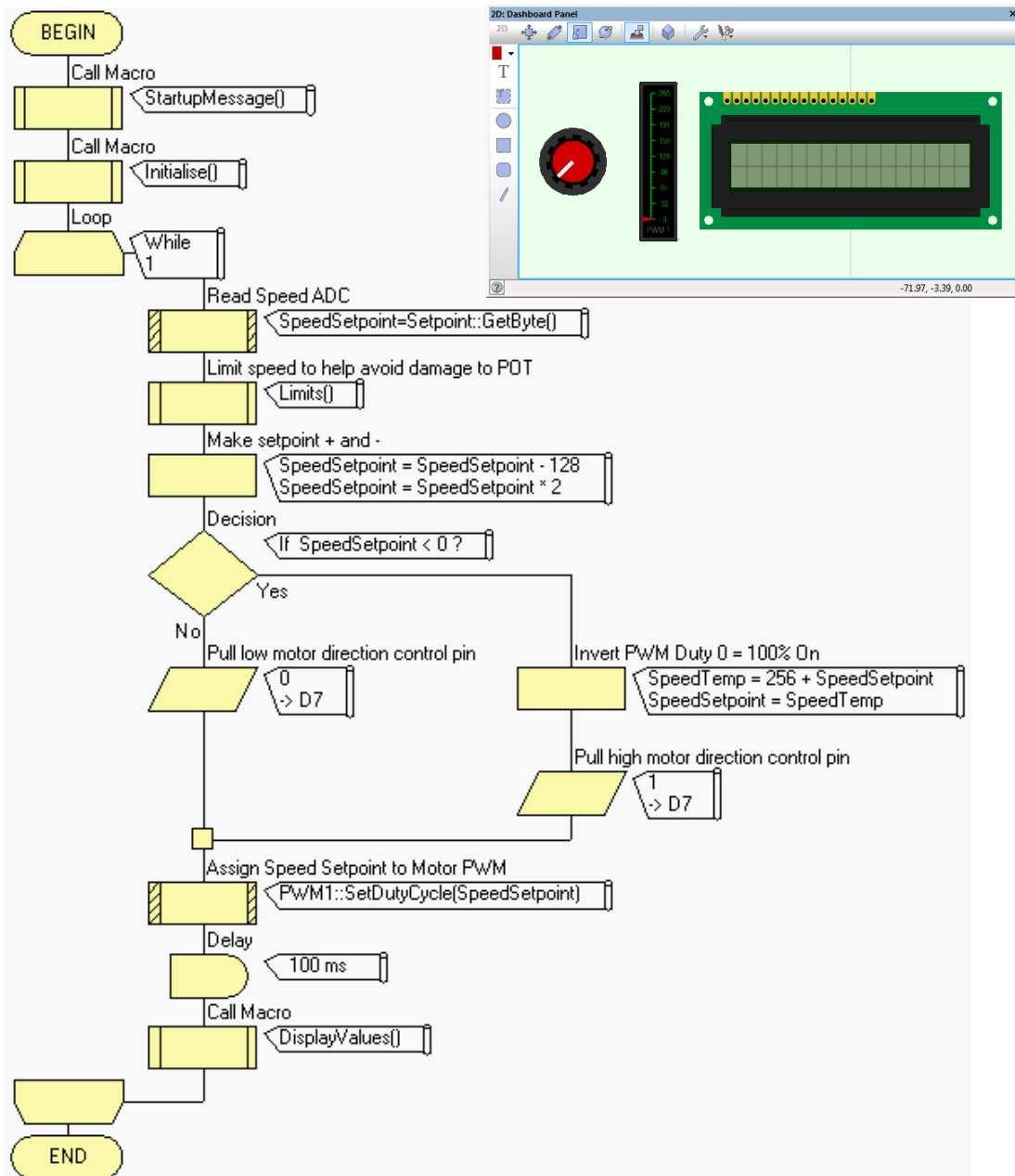
- Use Ghost to monitor the set point and the feedback signals.
- Adjust the **D** 'pot' to find the optimum value.

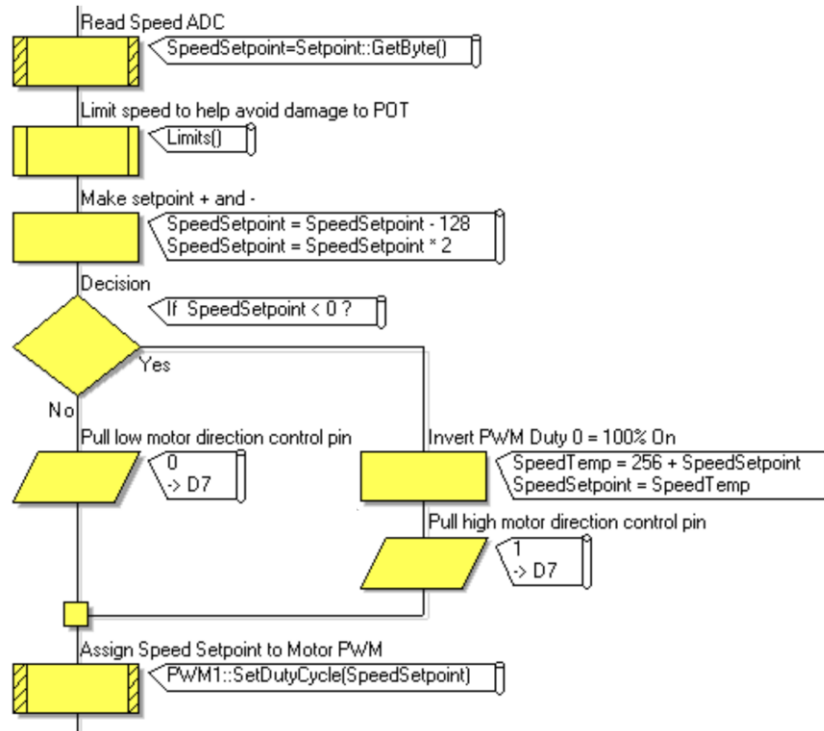
5.2.1 Position control - Open loop control

Program overview:

This program reads the set point 'pot', assigns the value to the motor speed PWM and direction control and displays it on the LCD. The duty cycle is specified as a byte, and so '0' ⇒ 100% in one direction, '128' ⇒ no movement, and '255' ⇒ 100% in the other direction.

The full Flowcode program is shown below, with a detailed look at part of the loop section on the following page.





Program in detail:

- The ‘StartupMessage’ macro activates the LCD and displays program title “Angle Control 1” for two seconds.
- The ‘Initialise’ macro enables the motor driver, the PWM component and the LCD. It gives initial values to the direction control and PWM duty cycle.
- The loop then repeatedly:
 - reads the value produced by the set point ‘pot’;
 - uses the ‘Limits’ macro to constrain set point values to the range 28 to 228, to protect the feedback ‘pot’;
 - uses calculations to weight the value and correct the direction control where necessary;
 - uses the result to set the duty cycle of the PWM signal;
 - displays ‘Speed’ on the LCD.

Exercise:

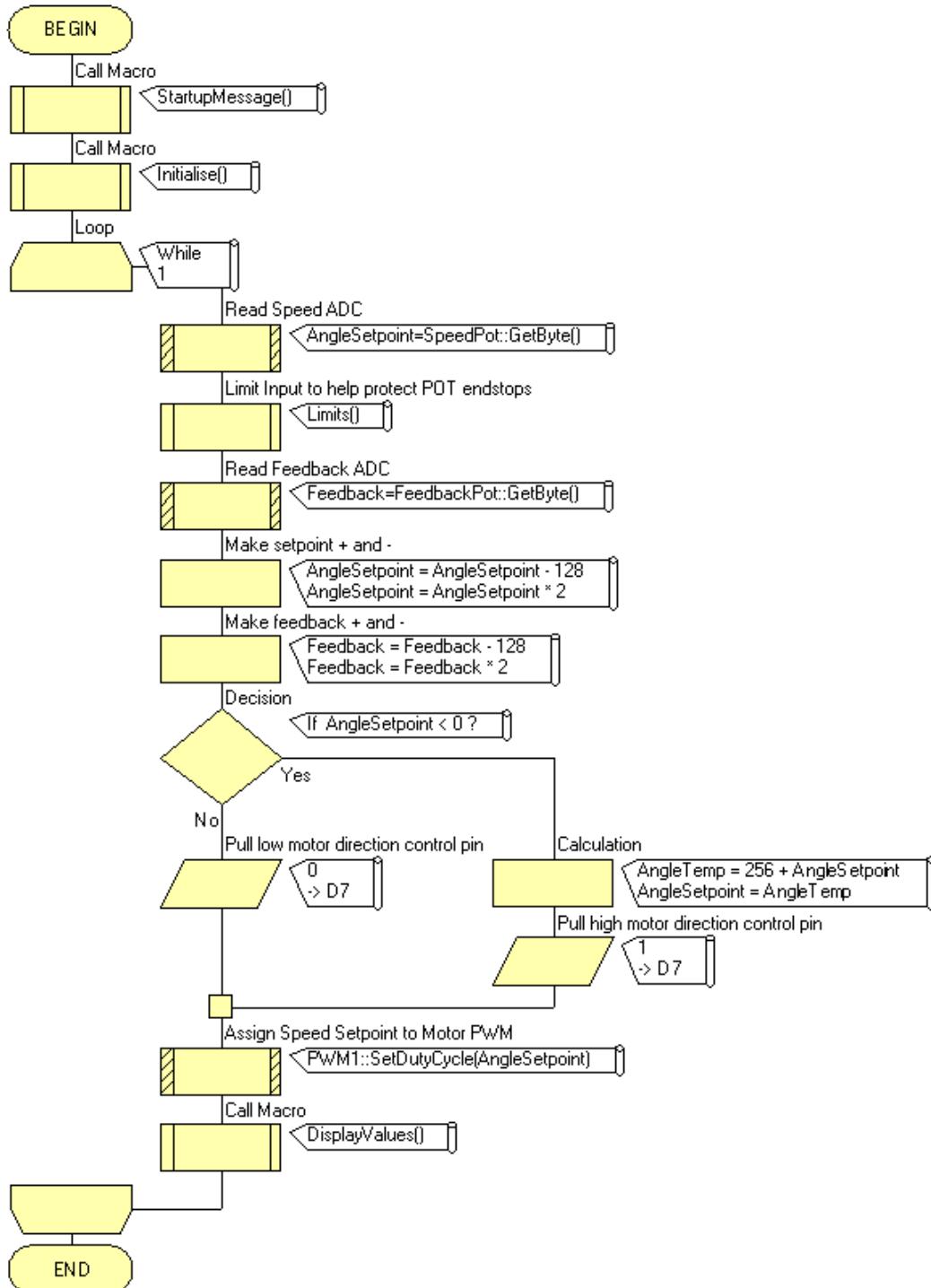
- Adjust the set point ‘pot’ to move the motor by the smallest amount possible. Is this repeatable adjustment?
- Select an angle on the disk and use the set point potentiometer to try and get there. How long does it take?

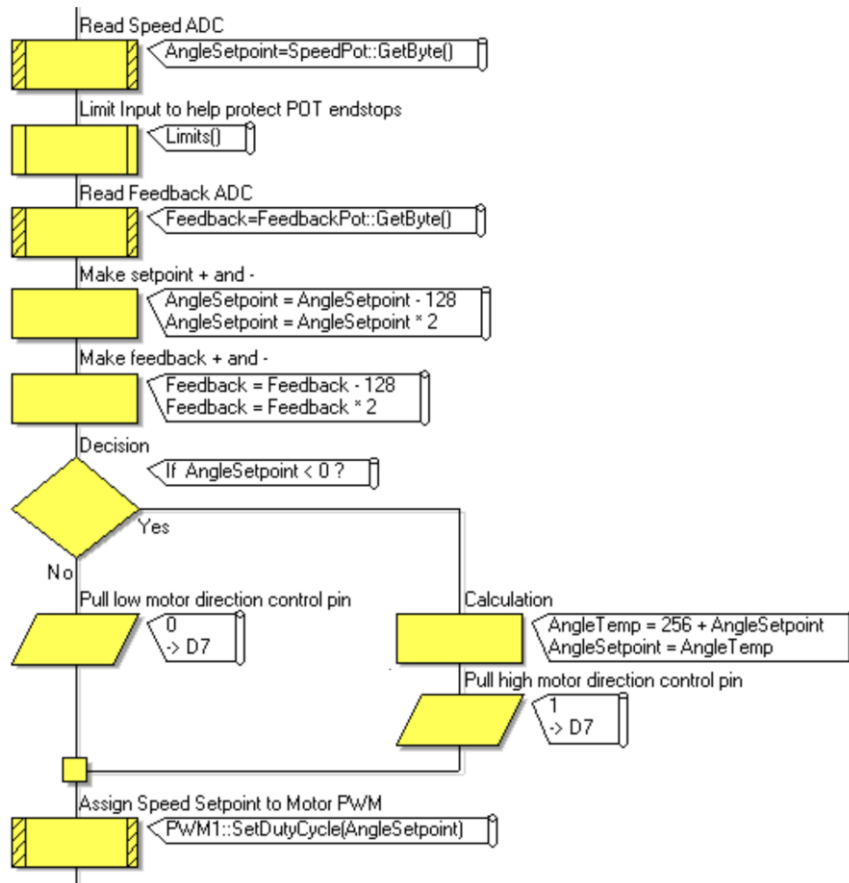
5.2.2 Position control with feedback - Closing the loop

Program overview:

This program adds feedback as a reading on the LCD to indicate the speed of the motor on the LCD.

The full Flowcode program is shown below, with a details of part of the loop on the following page.





Program in detail:

- The ‘StartupMessage’ macro activates the LCD and displays the title.
- The ‘Initialise’ macro enables the motor driver, the PWM component and the LCD and gives initial values to the direction control and PWM duty cycle.
- The loop then repeatedly:
 - reads the values produced by the set point and feedback ‘pots’;
 - conditions those values as in the previous program;
 - uses the result to set the duty cycle of the PWM signal;
 - displays ‘Speed’ and ‘Fback’ values on the LCD.

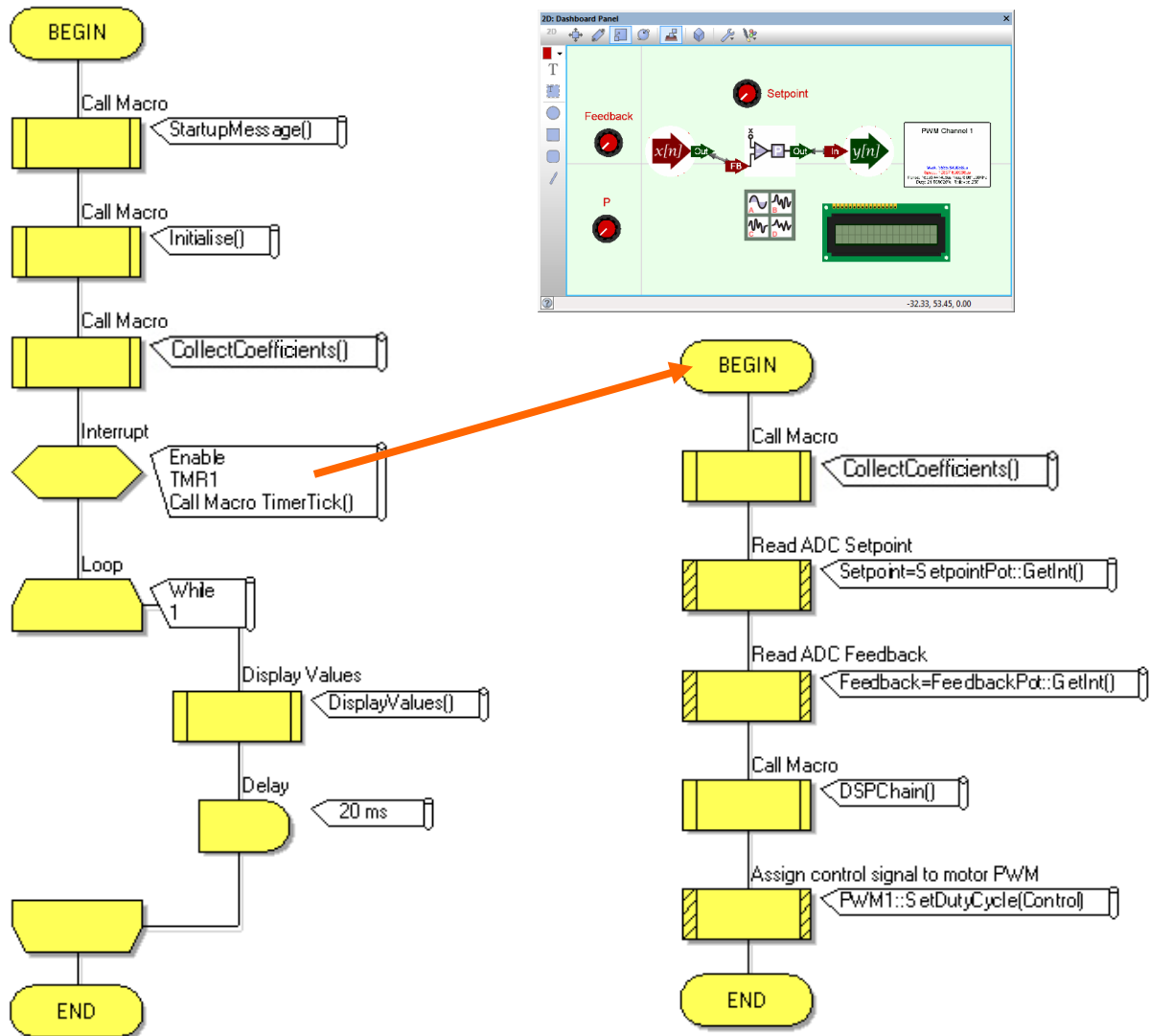
Exercise:

- Turn the motor shaft by hand and compare the angle reading on the scale with the feedback value.
- Does the effort needed to rotate the motor shaft change with angle?
- Use the set point control to vary the feedback value by 1. Is this easy to do?

5.2.3 Position control - Proportional (P) control

Program overview:

This program uses the feedback reading to adjust the PWM duty cycle to reduce the steady-state error. The timer interrupt macro allows regular timing between samples, unaffected by LCD updates etc.



Program in detail:

- The 'StartupMessage' macro activates the LCD and displays the title.
- The 'Initialise' macro does exactly the same as before. It also initialises the DSP system and displays the 'P' signal, the set point signal (as 'SP'), the feedback signal (as 'FB') and the control output signal (as 'C').
- The 'Collect coefficients' macro reads the voltage on the 'P' 'pot' and copies it into the 'P' constant.
- The 'TimerTick macro' is triggered repeatedly and uses the values of the set point and feedback signals to update the value of the 'control' signal and then modify the duty cycle of the PWM signal.

- In the meantime, the loop is repeatedly updating the LCD with current values of the 'P', 'SP', 'FB' and 'C' signals.

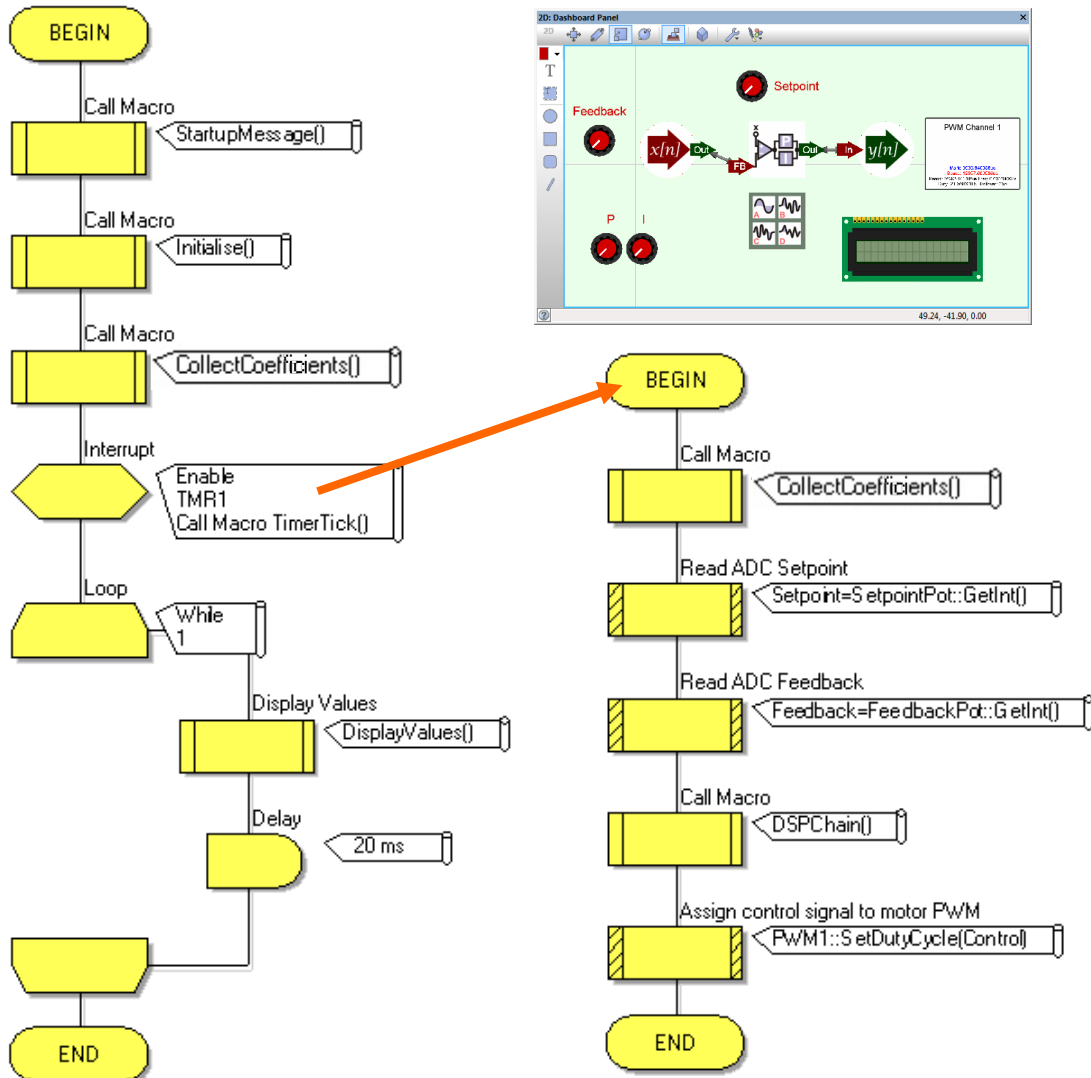
Exercise:

- Turn the motor shaft by hand again, does the amount of effort needed to rotate the motor shaft change now?
- Use Ghost to monitor the set point and the feedback signals. Adjust the P potentiometer and record the difference between the two values over a range of settings.

5.2.4 Position control - Proportional and Integral (PI) control

Program overview:

This program calculates the Integral component and uses it to reduce the steady-state error.



Program in detail:

- The 'StartupMessage' macro activates the LCD and displays the title.
- The 'Initialise' macro does exactly the same as before but also displays the value of the 'I' component.
- The 'Collect coefficients' macro reads the 'P' and 'I' signals and copies them into the appropriate components.
- The 'TimerTick macro' uses the values of the set point and feedback signals to modify the 'control' signal and the the duty cycle of the PWM signal.
- The loop repeatedly updates the LCD with current values of the 'P', 'I', 'SP', 'FB' and 'C' signals.

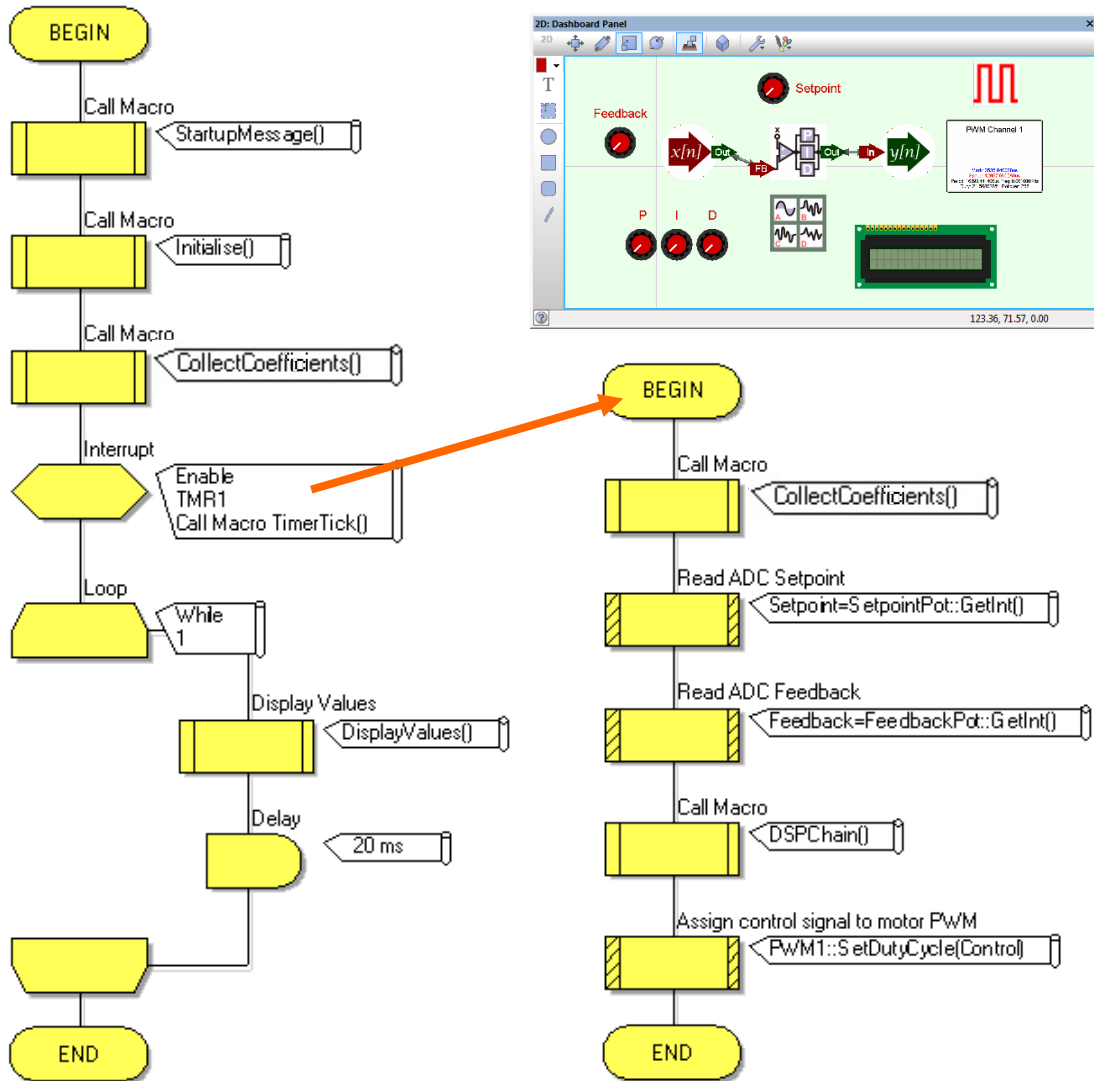
Exercise:

- Use Ghost to monitor the set point and the feedback signals.
- Adjust the 'I' 'pot' and try and eliminate the steady-state error.

5.2.5 Position control - Proportional, Integral and Derivative (PID) control

Program overview:

This program aims to reduce the error further by adding the Derivative component and using it to predict future changes.



Program in detail:

- The 'StartupMessage' macro activates the LCD and displays program title.
- The 'Initialise' macro does the same as before but adds the value of the 'D' signal to the LCD.
- The 'Collect coefficients' macro reads signals from the 'P', 'I' and 'D' 'pots', and copies the values into the 'P', 'I' and 'D' constants.
- Then the 'TimerTick macro' uses these to update the 'control' signal every time it is triggered and modifies the PWM duty cycle accordingly.
- The loop repeatedly updates the LCD with current values of 'P', 'I', 'SP', 'FB' and 'C' signals, now with the addition of the 'D' constant.

Exercise:

- Use Ghost to monitor the set point and the feedback signals.
- Adjust the 'D' 'pot' until you find the optimum value.

6 Further work:

Challenge 1:

Write a program to make a machine tool move to one position, by rotating the position motor clockwise through 45° , wait there for 10 seconds and then move to a second position, by rotating anticlockwise through 90° .

Challenge 2:

Write a program for a speed control system that displays the letter:

- 'U' if the speed is less than 90% of the target;
- 'O' if the speed overshoots the target by more than 10%;
- 'T' if the speed is within +/- 10% of the target.

Challenge 3:

This course has referred on occasions to 'optimal' conditions for the control system. However, in practice, the optimal values for the constants 'P', 'I' and 'D' may depend on conditions within the system being controlled.

For example, where an industrial vessel containing a liquid is being heated, the constants may depend on the volume of the liquid in the vessel at the time.

Write a program for a position control system where the values of the 'P', 'I' and 'D' constants increase by 25% each when the set point 'pot' is turned clockwise past the '12 o'clock' position.

Congratulations – you have just completed the course!